セグメント木と代数構造の理論

Kimiyuki Onaka　　Shiho Midorikawa

2024 年 2 月 12 日

概要

セグメント木は列に関するクエリを処理するための汎用的なデータ構造である. しかし, どのようなクエリがセグメント木を用いて処理できるかどうかは自明なものではない. どのようなクエリがセグメント木で処理可能かについてより正確に理解するために, この PDF では, クエリを代数構造として表現して理解する手法について体系的に紹介する. 特に, 新たに得られた結果として, 遅延伝播セグメント木に対応する代数構造の定義とその性質についての議論を紹介する. その主な内容は, 遅延伝播セグメント木に対応する代数構造は加群の一般化であることと, 複数種類の更新クエリの併用は ある積構造として理解できることである. 今回の議論の内容は主に整理と体系化であるため, その結果によりこれまで解けなかった問題が新たに解けるようになるというわけではない. しかし, セグメント木への理解を深めることには貢献するだろう.

目次

1 公開にあたっての前置き

まず, 本文書は 2021 年 6 月頃に`@kimiyuki_u` と`@elliptic_shiho` により書かれていた文書である. そして, この前置きを書いているのは`@elliptic_shiho` である. 彼は本文書を完成させないままに執筆から離れてしまい, 私もあまり研究に時間を割けなくなったことから, 約 3 年の間塩漬けとなっている (情報の古い部分が存在しうる) ことを最初に注意したい.

2024 年も 2 月に入った今公開することを選択した理由は 2 つある. 1 つは 100 ページ近くも書かれたこの文書を公開しないのは単純に勿体ないということ. もう一つは, 執筆中常々「セグメント木の最初の提案文書が unpublished manuscript なのはいけない 知見はちゃんと公開してほしい」と愚痴を言い合っていたことに反するためである. 彼との議論の結果生まれた未公開文書は他にも多数存在するが, 本文書はそのうち最大のものであり, 同時に彼の思想がよく反映されているものの一つだ.

2021 年当時と比較すればセグメント木の抽象化は随分と普及したように思う. しかし, その上の代数的な構造そのもの, また「データ構造とはなんであるか」という哲学的な側面に関しての動きは, 私の知る限りはあまり見受けられないようだ. 本文書はそこに焦点を当てたものであり, (本文中にも存在する通り) 競プロにおいて役立つことはほぼない結果が殆どを占める. そんな微妙な領域でも, ちょっと面白そうな議論のできる余白はまだ残されていそうだと思ってもらえれば, というのが本文書の公開意図である.

本文書は未完成である. 随所に TODO は残されているし, 証明の無い主張もいくらか

存在する. 問題を発見した場合, Twitter において`@elliptic_shiho` に連絡してもらってもよいが, 私はこの分野 (競技プログラミングを中心とした世界) においては非専門家であるため, 自ら記事を書いてもらったほうがよい可能性が高い. 寧ろ, そのような議論のネタにしてもらえるのであれば, 我々として幸甚である.

---

2 はじめに

この PDF は, 競技プログラミングの界隈におけるセグメント木に関する知見を体系化しまとめたものである. 想定読者はレートで言うと AtCoder 黄色以上である. 逆に, この PDF ではじめてセグメント木について触れようという人はおそらくこの PDF を読むべきでないだろう.

この PDF の構成について述べておく. 3 章から 5 章は準備である. 3 章では代数の言葉を用いずに非形式的にセグメント木を導入する. 4 章ではモノイドおよび半群という代数構造を導入する. 5 章では代数構造へのモデル化についての諸注意を述べる. 6 章から 8 章は本論である. 6 章では最も単純な形のセグメント木と半群の関係を議論する. 7 章では双対セグメント木と呼ばれる形のセグメント木とモノイドの関係を議論する. 8 章では遅延伝播セグメント木と呼ばれる形のセグメント木とこれに対応する代数構造について議論する.

3 計算機的な準備

TODO(kimiyuki): この章の完成度を上げる

3.1 セグメント木

この節ではセグメント木の基本的な用法について紹介する. ただしその定義は競技プログラミングのコミュニティにおける理解 [1] に基づく.

まずは具体例から初めよう．「データの列に対しその集計と更新をする」という処理を考える機会は多い．たとえばありそうな実問題として以下のようなものが考えられるだろう．[1]

- SNS を運営している．検索機能やマーケティングなどに利用する目的で「指定された期間の間になされたコメント数の合計」などを高速に計算できるようにしたい．ただしコメントはユーザによって後から削除されることがあり, 削除は集計結果に反映されるようにしたい．
- ある地点についての気象情報 (温度や湿度など) について数分刻みで記録したデータが数年分ある．これについて分析のため「ある時刻からある時刻の間の最低気温」などを計算したい．
- ゲームを開発している．バランス調整に利用するためにユーザのプレイ状況を匿名化して収集し統計ダッシュボードから閲覧できるようにしている．「ある期間においてある武器が使用された回数」や「ある期間における試合の総数」などが見れるようにしたい．ただしゲームはインターネット環境なしでもプレイ可能であり, オフラインの間になされたゲームプレイについての統計情報は端末に保存され次にオンラインになったときにまとめて送信される．

計算するだけなら定義通りに集計をすればよいだろう．しかし, その処理の速度が問題になり高速化が必要になる場合もあるだろう．そのようなときどうすればよいだろうか？おそらく「日ごとや月ごとの集計結果を前もって計算しておき, 集計が要求されたときはこれを利用して計算する．更新が要求されたときは該当するデータを含む日や月についての集計結果も同時に更新する」という手法を使うことになるだろう．この手法を洗練させていくとセグメント木につながる．

では, 具体例を離れて抽象化していこう．紹介したこれらの問題の構造を整理すれば, どれも以下のような構造をしている．

問題 3.1.1.

長さ n の数列 $a = (a_0, a_1, \ldots, a_{n-1})$ がある．この数列についての以下のような形の要求が時間とともにたくさん与えられる．すべて高速に処理せよ．

(a.) 自然数 l, r (ただし $0 \leq l \leq r \leq n$) が指定される．総和 $a_l + a_{l+1} + \cdots + a_{r-1}$ (あるいは最大値 $\max\{ a_l, a_{l+1}, \ldots, a_{r-1} \}$ など) を答えよ．

[1] ただし, 実際にこのような場面に遭遇した経験はなく想像で書いていることを注意しておく

(b.) 自然数 i (ただし $0 \leq i < n$) と数 b が指定される．項 a_i の値を b に更新せよ．以降の計算は更新された新しい数列について行うものとする．

(c.) 数 b が指定される．数列 a の末尾に新しい項として b を付け加えて更新する．n の値は 1 増える．以降の計算は更新された新しい数列について行うものとする．

◇

このように整理された問題を効率良く解くものとして, 以下のようなアルゴリズムが考えられる．

アルゴリズム 3.1.2.

葉の数が n の平衡二分探索木 T を用意する．まず次のようにして頂点に再帰的に区間を割り当てる．

- 葉について．左から $i+1$ 番目 (ただし $0 \leq i < n$) の葉には区間 $[i, i+1)$ を割り当てる．
- 葉でない頂点 v について．v の左の子に割り当てられた区間が $[l, m)$ でかつ v の右の子に割り当てられた区間が $[m', r)$ であるとする．このとき必ず $m = m'$ である．その頂点 v には区間 $[l, r)$ を割り当てる．

そして, それぞれの頂点 v に対し, その割り当てられた区間 $[l, r)$ について集計した結果 ($a_l + a_{l+1} + \cdots + a_{r-1}$ など) を持たせる．この集計の結果は, それぞれの子の持つ集計結果 ($a_l + a_{l+1} + \cdots + a_{m-1}$ および $a_m + a_{m+1} + \cdots + a_{r-1}$ など) から効率良く計算できることに注意する．

このような準備のもとで, それぞれの形の要求は以下のように処理できる．

(a.) 根から再帰的に降りていきながら計算する．すこし一般化して, 指定された区間 $[l, r)$ とそのとき見ている頂点 v に割り当てられた区間 $[l_v, r_v)$ の共通部分 $[l', r') = [l, r) \cap [l_v, r_v)$ についての集計結果 ($a_{l'} + a_{l'+1} + \cdots + a_{r'-1}$ を求める．頂点 v を見ているとき,

- v に割り当てられた区間が指定された区間に包含されていれば, 頂点 v が記録している値をそのまま使えばよい．
- v に割り当てられた区間と指定された区間に交わりがなければ, その頂点は無視して適当な値 (総和を求めているなら 0 など) を使う．
- v に割り当てられた区間は指定された区間と交わりはするが包含されてはいないならば, その左右の子について再帰的に求めてその結果を併せればよい．こ

のとき v は葉ではなく, 左右の子への再帰は可能である.

この再帰について, それぞれの深さについて訪ずれる頂点はたかだか 4 個であることが示せる. 平衡二分探索木という仮定から, 全体では訪ずれる頂点は $O(\log n)$ 個しかない.

(b.) 左から $i+1$ 番目の葉の持つ値を b に更新する. その葉の先祖の頂点もすべて更新する. 平衡二分探索木を仮定していたので, 更新が必要な頂点は合計で $O(\log n)$ 個しかない.

(c.) 木 T の最も右の葉として値 b を持つ頂点を挿入する. 挿入操作で変化した頂点については区間の割り当てなどを再計算する. 平衡二分探索木を仮定しているので, 再計算が必要な頂点の個数は $O(\log n)$ 個しかない.

◇

このアルゴリズムのように使われる平衡二分探索木 T をデータ構造と見たものが「セグメント木」と呼ばれるものである. (c.) の操作がなく数列の要素数 n が固定である場合には, 実装の簡単さと速度のため完全二分木が使われることが多い.

セグメント木の例を実装したものをプログラム 1 として示しておく. これは Library Checker の Point Add Range Sum という問題 [2] に対する解答という形で書かれており, 実際に提出した結果は `https://judge.yosupo.jp/submission/26359` にある. このセグメント木は, 長さ n の整数列 $a = (a_0, a_1, \ldots, a_{n-1})$ があるとしたとき, 以下の操作が可能である.

- 与えられた i (ただし $0 \leq i < n$) と整数 $b \in \mathbb{Z}$ に対し, $a_i \leftarrow b$ と $O(\log n)$ で更新する
- 与えられた l, r (ただし $0 \leq l \leq r \leq n$) に対し, 総和 $a_l + a_{l+1} + \cdots + a_{r-1}$ を $O(\log n)$ で計算する

その他の話題については説明を省略する[2].

プログラム 1　Point Add Range Sum に対する実装例

```
1 #include <cassert>
2 #include <cstdio>
```

[2] 再帰でなくループを用いて実装すると速いこと, セグメント木上での集計結果についての二分探索を $O((\log n)^2)$ でなく $O(\log n)$ で行うことができること, Fenwick tree や binary indexed tree と呼ばれる類似のデータ構造との関係性について, 列でなく行列に対する 2 次元のセグメント木について, 2 分木でなく 3 分木以上を考えた場合にどうなるか, などの話題がある

```
3  #include <memory>
4  #include <vector>
5  #define REP(i, n) for (int i = 0; (i) < (int)(n); ++ (i))
6  #define ALL(x) std::begin(x), std::end(x)
7
8  struct node {
9      long long sum;
10     int width;
11     std::unique_ptr<node> left, right;
12 };
13
14 template <class RandomAccessIterator>
15 std::unique_ptr<node> make_segment_tree(
       RandomAccessIterator first, RandomAccessIterator last)
       {
16     int width = std::distance(first, last);
17
18     if (width == 0) {
19         return nullptr;
20
21     } else if (width == 1) {
22         return std::make_unique<node>(node {
23             .sum = *first,
24             .width = width,
25             .left = nullptr,
26             .right = nullptr,
27         });
28
29     } else {
30         auto left = make_segment_tree(first, first + width
               / 2);
```

```
31          auto right = make_segment_tree(first + width / 2,
                last);
32          long long sum = left->sum + right->sum;
33          return std::make_unique<node>(node {
34              .sum = sum,
35              .width = width,
36              .left = std::move(left),
37              .right = std::move(right),
38          });
39      }
40  }
41
42  void set_value(const std::unique_ptr<node>& x, int i, long
        long value) {
43      assert (x != nullptr);
44      assert (0 <= i and i < x->width);
45
46      if (x->width == 1) {
47          // leaf
48          x->sum = value;
49
50      } else {
51          // non-leaf
52          if (i < x->left->width) {
53              set_value(x->left, i, value);
54          } else {
55              set_value(x->right, i - x->left->width, value)
                    ;
56          }
57          x->sum = x->left->sum + x->right->sum;
58      }
```

```
59 }
60
61 long long get_sum(const std::unique_ptr<node>& x, int l,
       int r) {
62     if (x == nullptr or x->width <= l or r <= 0) {
63         // disjoint
64         return 0;
65
66     } else if (l <= 0 and x->width <= r) {
67         // included
68         return x->sum;
69
70     } else {
71         // intersecting
72         long long sum_l = get_sum(x->left, l, r);
73         long long sum_r = get_sum(x->right, l - x->left->
               width, r - x->left->width);
74         return sum_l + sum_r;
75     }
76 }
77
78 int main() {
79     // initialize
80     int n, q;
81     scanf("%d%d", &n, &q);
82     std::vector<long long> a(n);
83     REP (i, n) {
84         scanf("%lld", &a[i]);
85     }
86     auto segtree = make_segment_tree(ALL(a));
87
```

```
88      // query
89      while (q --) {
90          int t;
91          scanf("%d", &t);
92
93          if (t == 0) {
94              // update query
95              int p; long long x;
96              scanf("%d%lld", &p, &x);
97              long long a_p = get_sum(segtree, p, p + 1);
98              set_value(segtree, p, a_p + x);
99
100         } else if (t == 1) {
101             // get query
102             int l, r;
103             scanf("%d%d", &l, &r);
104             long long sum = get_sum(segtree, l, r);
105             printf("%lld\n", sum);
106
107         } else {
108             assert (false);
109         }
110     }
111     return 0;
112 }
```

3.2 セグメント木という名前の由来

混乱を防ぐため, 競技プログラミングのコミュニティの外での「セグメント木」についても紹介しておく. これは特に計算幾何の分野においての語である [3]. この (計算幾何における) セグメント木は Jon Louis Bentley によって 1977 年に初めて発見され 1980 年

に発表され [4], 次のような問題を解くデータ構造であった.

問題 3.2.1.
数直線上の線分の集合 $S = \{ [l_i, r_i) \mid i < n \}$ がある. この集合についての以下のような形の要求が時間とともにたくさん与えられる. すべて高速に処理せよ.

(a.) 整数 x が指定される. S に含まれる線分 $[l, r) \in S$ であって x を含む (つまり $x \in [l, r)$ を満たす) ものをすべて列挙し報告せよ.
(b.) 線分 $[l, r]$ が指定される. これを集合 S に追加して更新せよ. 以降の処理は更新された新しい集合について行うものとする.

◇

代数構造を用いての分析もされている. Bernard Chazelle による 1988 年の論文には (競技プログラミングのコミュニティでの意味の) セグメント木について可換半群を用いての議論がある [5].

この (計算幾何における) セグメント木は 3.4 節で紹介し 7 節で議論する (競技プログラミングのコミュニティにおいて) 双対セグメント木と呼ばれるデータ構造に近い. これが現在の (競技プログラミングのコミュニティでの意味の) セグメント木に変化した歴史的過程は不明である.

3.3 遅延伝播セグメント木

セグメント木を発展させたものとして, 遅延伝播セグメント木と呼ばれる[3]データ構造がある. これは従来のセグメント木でできる操作に加え, 範囲を指定しての一括での更新操作ができるようなデータ構造である. たとえば次のような問題が解ける.

問題 3.3.1.
長さ n の数列 $a = (a_0, a_1, \ldots, a_{n-1})$ がある. この数列についての以下のような形の要求が時間とともにたくさん与えられる. すべて高速に処理せよ.

(a.) 自然数 l, r (ただし $0 \leq l \leq r \leq n$) が指定される. 最大値 $\max \{ a_l, a_{l+1}, \ldots, a_{r-1} \}$ を答えよ.
(b.) 自然数 l, r (ただし $0 \leq l \leq r \leq n$) と数 b が指定される. $i \in [l, r)$ のそれぞれにつ

[3] 過去には「遅延評価セグメント木」と呼ばれることも多かったが, この呼び方は推奨されない. 正格評価と対比される評価戦略であるところの「遅延評価」と紛らわしいためである

いて a_i を $a_i + b$ で置き換えて更新せよ. 以降の計算は更新された新しい数列について行うものとする.

◇

正確な定義や実装の詳細およびその他の話題については説明を省略する[4].

3.4 双対セグメント木

遅延伝播セグメント木を (セグメント木とは異なる方向に) 簡略化したものとして, 双対セグメント木と呼ばれるデータ構造がある [6][5]. 遅延伝播セグメント木でできる操作から範囲を指定しての集計操作を除いた操作が処理可能である. たとえば次のような問題が解ける.

問題 3.4.1.
長さ n の数列 $a = (a_0, a_1, \ldots, a_{n-1})$ がある. この数列についての以下のような形の要求が時間とともにたくさん与えられる. すべて高速に処理せよ.

(a.) 自然数 i (ただし $0 \leq i < n$) が指定される. a_i の値を答えよ.
(b.) 自然数 l, r (ただし $0 \leq l \leq r \leq n$) と数 b が指定される. $i \in [l, r)$ のそれぞれについて a_i を $\min\{ a_i, b \}$ で置き換えて更新せよ. 以降の計算は更新された新しい数列について行うものとする.

◇

正確な定義や実装の詳細およびその他の話題については説明を省略する.

[4] 再帰でなくループを用いて実装すること, 更新操作の順序に交換可能性があれば高速化できること, などの話題がある

[5] このデータ構造の「双対セグメント木」という名の由来は「通常のセグメント木」との関係によるもので. 自明ではあるがしかし自明であるがゆえに注目されず言語化もされてこなかったこの概念を広めるための, 覚えやすさを優先した命名である. 通常のセグメント木から双対セグメント木を作ることができ, 双対セグメント木から通常のセグメント木を作ることができ, これらの変換は (モノイドと半群の差を無視すれば) 一対一対応をなす. しかし圏論的な双対にあることが言えるわけでも最小値と最大値の一致などの双対定理と呼べるものがあるわけでもない.

4　数学的な準備

4.1　モノイドについての基本的な事項

定義 4.1.1 (半群)**.**

S を集合, $\cdot\colon S \times S \to S$ を二項演算とする. 次を満たすとき $(S, \cdot)$ を**半群**という.

$$\forall x, y, z \in S.\ x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

◇

注意 4.1.2.

集合 S を半群 $(S, \cdot)$ の台集合と呼ぶ. 文脈から明らかなとき, 半群と台集合の区別を省略することがある. たとえば $(S, \cdot)$ のことを単に S と書くことがある[6]. ◇

定義 4.1.3 (モノイド)**.**

M を集合, $e \in M$ を M の要素, $\cdot\colon M \times M \to M$ を二項演算とする. 以下のすべてを満たすとき $(M, \cdot, e)$ を**モノイド**という.

- $\forall x, y, z \in M.\ x \cdot (y \cdot z) = (x \cdot y) \cdot z$
- $\forall x \in M.\ x \cdot e = x$
- $\forall x \in M.\ e \cdot x = x$

◇

注意 4.1.4.

文脈から明らかなとき, モノイドと台集合の区別を省略することがある. たとえば $(M, \cdot, e)$ のことを単に M と書くことがある[7]. ◇

注意 4.1.5.

モノイドならば半群である[8]. ◇

記法 4.1.6 (総積)**.**

モノイド $(M, \cdot, e)$ の要素の列 $(a_0, a_1, \ldots, a_{n-1})$ (ただし $n \geq 0$) があるとする. この

[6] 記号の濫用

[7] 記号の濫用

[8] 半群は 2 つ組のことでありモノイドは 3 つ組のことであるが, 一般にそのような差は無視される

とき総積 $e \cdot a_0 \cdot a_1 \cdot \cdots \cdot a_{n-1}$ を $\prod_{i=0}^{n-1} a_i$ や $\prod_i a_i$ などと書く. ◇

注意 4.1.7.
可換性は仮定されておらず, 総積の順序には注意が必要である. $\prod_{i=0}^{n-1} a_i \neq \prod_{i=0}^{n-1} a_{n-i-1}$ である場合がある. ◇

注意 4.1.8.
$n = 0$ のとき $\prod_{i=0}^{0} a_i = e$ となる. ◇

定義 4.1.9 (部分半群)**.**
$(S, \cdot_S)$, $(T, \cdot_T)$ を半群とする. 集合として $T \subseteq S$ でありかつ任意の $a, b \in S$ に対し $a \cdot_T b = a \cdot_S b$ を満たすとする. このとき $(T, \cdot_T)$ は $(S, \cdot_S)$ の**部分半群**であるという. ◇

定義 4.1.10 (部分モノイド)**.**
$(M, \cdot_M, e_M)$, $(N, \cdot_N, e_N)$ であるとする. $(N, \cdot_N)$ が $(M, \cdot_M)$ の部分半群でありかつ $e_M = e_N$ を満たすとする. このとき $(N, \cdot_N, 1_N)$ は $(M, \cdot_M, 1_M)$ の**部分モノイド**であるという. ◇

定義 4.1.11 (半群準同型)**.**
$(S, \cdot_S)$ と $(T, \cdot_T)$ を半群とする. 関数 $\varphi\colon S \to T$ が任意の $x, y \in S$ に対し

$$\varphi(x \cdot_S y) = \varphi(x) \cdot_T \varphi(y)$$

を満たすとき, φ を S から T への**半群準同型**と呼ぶ. ◇

定義 4.1.12 (モノイド準同型)**.**
$(M, \cdot_M, e_M)$ と $(N, \cdot_N, e_N)$ をモノイドとする. 関数 $\varphi\colon M \to N$ が以下のすべてを満たすとき, φ を M から N への**モノイド準同型**と呼ぶ.

- $\varphi(e_M) = e_N$
- 任意の $x, y \in M$ について $\varphi(x \cdot_M y) = \varphi(x) \cdot_N \varphi(y)$

◇

注意 4.1.13.
モノイド準同型なら半群準同型である. ◇

定義 4.1.14 (モノイド同型)**.**
$\varphi\colon M \to N$ を M から N へのモノイド準同型とする. 逆関数 $\varphi^{-1}\colon N \to M$ が存在

しかつその φ^{-1} が N から M へのモノイド準同型であるとき, φ を M と N の間の**モノイド同型**であると呼ぶ. またこのようなとき M と N は同型であると呼ぶ. ◇

定義 4.1.15 (半群同型)**.**

省略 (モノイド同型と同様に定義する) ◇

注意 4.1.16.

同型なモノイドどうしや同型な半群どうしはほとんど同じものだと思える. 同型なものどうしの差は要素をどのような方法でラベル付けるかのみであり, 演算の構造については差がないためである. 同型性が自明なものは断わりなく同一視することがある. ◇

命題 4.1.17.

$\varphi\colon M \to N$ をモノイド $(M, \cdot_M, e_M)$ から $(N, \cdot_N, e_N)$ へのモノイド準同型でかつ全単射とする. このとき φ はモノイド同型である. ◇

証明

φ^{-1} がモノイド準同型であることを言えばよい.

- $\varphi(e_M) = e_N$ であるので $\varphi^{-1}(e_N) = e_M$ である.
- $x, y \in N$ とする. $x \cdot_N y = \varphi(\varphi^{-1}(x)) \cdot_N \varphi(\varphi^{-1}(y)) = \varphi(\varphi^{-1}(x) \cdot_M \varphi^{-1}(y))$ である. よって $\varphi^{-1}(x \cdot_N y) = \varphi^{-1}(x) \cdot_M \varphi^{-1}(y)$ である.

□

定理 4.1.18 (単位元の付与)**.**

半群 $(S, \cdot)$ と $* \notin S$ に対し, モノイド $(M, \cdot', *)$ であって半群として $M \setminus \{ * \} \simeq S$ なるものが存在する. 半群 S に対しこのようなモノイド M は単位元 $*$ の差による同型を除いて一意である. ◇

証明

$*$ に対し演算 $\cdot'$ を以下で定める.

$$x \cdot' y = \begin{cases} y & (x = *) \\ x & (y = *) \\ x \cdot y & (\text{otherwise}) \end{cases}$$

この演算により $(S \cup \{ * \}, \cdot', *)$ はモノイドをなす. 半群としての同型性 $M \setminus \{ * \} \simeq S$ は明らか.

いま, 半群 $(S, \cdot')$ と任意の $*_1, *_2 \notin S$ に対し $(M_1, \cdot'_1, *_1)$, $(M_2, \cdot'_2, *_2)$ を上記操作で得られるモノイドとする. $\varphi\colon M_1 \to M_2$ を以下で定める.

$$\varphi(x) = \begin{cases} x & (x \in S) \\ *_2 & (x = *_1) \end{cases}$$

φ は明らかにモノイド同型であるから, $M_1 \simeq M_2$ である. □

注意 4.1.19.

この定理により, 実用する場面において半群とモノイドはあまり区別する必要はない. 以下では基本的にモノイドに統一して議論を行う. ◇

例 4.1.20.

- 整数の全体は加法によってモノイド $(\mathbb{Z}, +, 0)$ をなす
- 自然数の全体は加法によってモノイド $(\mathbb{N}, +, 0)$ をなす
- 正整数の全体は加法によって半群 $(\{1, 2, 3, \dots\}, +)$ をなす. $x + e = x$ を満たすような正整数 e は存在しないため, モノイドにはならない
- 自然数の全体に $x \cdot y = 2x + y$ という二項演算を考えると, これは半群にはならない. $x \neq 0$ のとき $x \cdot (y \cdot z) = 2x + 2y + z \neq 4x + 2y + z = (x \cdot y) \cdot z$ であることが反例である
- 自然数の全体は乗法によってモノイド $(\mathbb{N}, \cdot, 1)$ をなす
- 自然数の全体は最小値 min によって半群 $(\mathbb{N}, \min)$ をなす. 無限大 ∞ は自然数ではないため, モノイドにはならない. 自然数の全体に無限大を付け加えてできる集合 $\mathbb{N} \cup \{\infty\}$ は最小値によってモノイド $(\mathbb{N} \cup \{\infty\}, \min, \infty)$ をなす
- 自然数の全体は最大値 max によってモノイド $(\mathbb{N}, \max, 0)$ をなす
- 自然数の全体は最小公倍数 gcd によってモノイド $(\mathbb{N}, \gcd, 0)$ をなす
- 自然数の全体は最大公約数 lcm によってモノイド $(\mathbb{N}, \mathrm{lcm}, 1)$ をなす
- 自然数の全体は bit ごとの論理和 $|$ によってモノイド $(\mathbb{N}, |, 0)$ をなす
- 自然数の全体は bit ごとの論理積 & によって半群 $(\mathbb{N}, \,\&\,)$ をなす. すべての bit が 1 であるような自然数はないため, モノイドにはならない
- 自然数の全体は bit ごとの排他的論理和 $\oplus$ によってモノイド $(\mathbb{N}, \oplus, 0)$ をなす
- 集合 Σ の要素の有限列の全体 Σ^* は連結 $\frown$ と空列 ϵ についてモノイド $(\Sigma^*, \frown, \epsilon)$ をなす

◇

注意 4.1.21.

モノイドや半群の演算の記号は省略されることがある. たとえばモノイド $(M, \cdot, e)$ の要素 $x, y \in M$ の積 $x \cdot y$ が単に xy と書かれたりする. ◇

記法 4.1.22 (λ 記法)**.**

$f(x) = y$ で定まるような関数 f のことを $\lambda x.y$ のように書く. 引数が複数の場合は $\lambda xy.z$ のように書く. ◇

例 4.1.23.

- 整数から整数への関数の全体 $\mathbb{Z}^{\mathbb{Z}}$ は関数合成 $\circ$ と恒等関数 $\mathrm{id} = \lambda x.x$ によってモノイド $(\mathbb{Z}^{\mathbb{Z}}, \circ, \mathrm{id})$ をなす
- 一般に, 集合 X から X それ自身への関数の全体 X^X は関数合成 $\circ$ と恒等関数 $\mathrm{id} = \lambda x.x$ によってモノイド $(X^X, \circ, \mathrm{id})$ をなす
- 1 次関数の全体 $F = \{ \lambda x.ax + b \mid a, b \in \mathbb{Z} \}$ は関数合成によってモノイド $(F, \circ, \mathrm{id})$ をなす
- 2 次関数の全体 $F = \left\{ \lambda x.ax^2 + bx + c \,\middle|\, a, b, c \in \mathbb{Z} \right\}$ は関数合成によって半群をなさない. 2 次関数と 2 次関数の合成が必ずしも 2 次関数ではないため
- ある整数との最小値を取る関数の全体に恒等関数 $\lambda x. \min(\infty, x)$ を加えたもの $F = \{ \lambda x. \min(a, x) \mid a \in \mathbb{Z} \cup \{ \infty \} \}$ は

  $$(\lambda x. \min(a, x)) \circ (\lambda x. \min(b, x)) = \lambda x. \min(a, \min(b, x)) = \lambda x. \min(\min(a, b), x)$$

  であることから関数合成によってモノイドをなす.
- 値を一定の範囲内に収めるような関数の全体

  $$F = \{ \lambda x. \max(l, \min(r, x)) \mid l, r \in \mathbb{Z} \cup \{ \pm\infty \}, l \leq r \}$$

  は関数合成によってモノイド $(F, \circ, \mathrm{id})$ をなす. ただし $\mathrm{id} = \lambda x. \max(-\infty, \min(+\infty, x))$ が示せるため $\mathrm{id} \in F$ である.

◇

定義 4.1.24 (左零半群)**.**

X を集合とする. $(X, \lambda xy.x)$ は半群であり, これを**左零半群**と呼ぶ. ◇

定義 4.1.25 (右零半群)**.**
X を集合とする. $(X, \lambda xy.y)$ は半群であり, これを**右零半群**と呼ぶ. ◇

定義 4.1.26 (自明半群)**.**
e を任意の要素とする. $(\{\, e \,\}, \lambda xy.e)$ は半群であり, これを**自明半群**と呼ぶ. ◇

定義 4.1.27 (自明モノイド)**.**
e を任意の要素とする. $(\{\, e \,\}, \lambda xy.e, e)$ はモノイドであり, これを**自明モノイド**と呼ぶ. ◇

定義 4.1.28 (直積半群)**.**
$(S, \cdot_S)$ と $(T, \cdot_T)$ を半群とする. 直積 $S \times T$ と二項演算 $(x, y) \cdot (x', y') = (x \cdot_S x', y \cdot_T y')$ は半群 $(S \times T, \cdot)$ をなす. これを S と T の**直積半群**と呼ぶ. ◇

定義 4.1.29 (直積モノイド)**.**
$(M, \cdot_M, e_M)$ と $(N, \cdot_N, e_N)$ をモノイドとする. 直積 $M \times N$ と二項演算 $(x, y) \cdot (x', y') = (x \cdot_M x', y \cdot_N y')$ はモノイド $(M \times N, \cdot, (e_M, e_N))$ をなす. これを M と N の**直積モノイド**と呼ぶ. ◇

例 4.1.30.

- 関数 $\varphi(n) = 2^n$ は, 自然数と最小値のなす半群 $(\mathbb{N}, \min)$ から自然数と最大公約数のなす半群 $(\mathbb{N}, \gcd)$ への半群準同型である
- 関数 $\varphi(n) = 3n$ は, 自然数と加法のなすモノイド $(\mathbb{N}, +, 0)$ から自然数と加法のなすモノイド $(\mathbb{N}, +, 0)$ それ自身へのモノイド準同型である
- 関数 $\varphi(x) = e_N$ は, モノイド $(M, \cdot_M, e_M)$ からモノイド $(N, \cdot_N, e_N)$ へのモノイド準同型である
- ある整数との最小値を取る関数の全体 $F = \{\, \lambda x. \min(a, x) \mid a \in \mathbb{Z} \,\}$ を考える. 関数 $\varphi(a) = \lambda x. \min(a, x)$ は半群 $(\mathbb{Z}, \min)$ から半群 $(F, \circ)$ への半群準同型である
- $(x, y) \cdot (x', y') = (xx', xy' + y)$ という二項演算を考えれば $(\mathbb{Z} \times \mathbb{Z}, \cdot, (1, 0))$ はモノイドをなす. このモノイドは, 1 次関数の全体 $F = \{\, \lambda x.ax + b \mid a, b \in \mathbb{Z} \,\}$ のなすモノイド $(F, \circ, \mathrm{id})$ との間にモノイド同型 $\varphi(a, b) = \lambda x.ax + b$ を持つ.

◇

定義 4.1.31 (同値関係)**.**

X を集合とする. 二項関係 $R \subseteq X \times X$ が以下のすべてを満たすとき, R は X 上の同値関係であると言う.

- 任意の $x \in X$ に対し, xRx
- 任意の $x, y \in X$ に対し, もし xRy ならば yRx
- 任意の $x, y, z \in X$ に対し, もし xRy かつ yRz ならば xRz

◇

定義 4.1.32 (商集合)**.**

X を集合とし $\sim$ を X 上の同値関係とする. X の要素 $x \in X$ ごとに定まる集合 $\{y \in X \mid y \sim x\}$ を $[x]$ と書き, $[x]$ を x の $\sim$ による**同値類**と呼ぶ. また逆に, 要素 x を同値類 $[x]$ の代表元と呼ぶ. 同値類をすべて集めてできる集合 $\{[x] \mid x \in X\}$ を $X/\sim$ と書き, X を $\sim$ で割ってできる**商集合**と呼ぶ. ◇

例 4.1.33.

整数上の $a \sim b \leftrightarrow a \equiv b \pmod{10^9+7}$ という二項関係を考えると, これは同値関係である. この二項関係 $\sim$ で整数の全体 $\mathbb{Z}$ を割ったもの $\mathbb{Z}/\sim$ を考える. $\mathbb{Z}/\sim$ の要素は $[0] = \{\ldots, -2\cdot(10^9+7), -(10^9+7), 0, 10^9+7, 2\cdot(10^9+7), \ldots\}$ や $[1] = \{\ldots, 1-2\cdot(10^9+7), 1-(10^9+7), 1, 1+10^9+7, 1+2\cdot(10^9+7), \ldots\}$ などである. $\mathbb{Z}/\sim$ は合計で 10^9+7 個の要素を持つ. また $\cdots = [-(10^9+7)] = [0] = [10^9+7] = [2\cdot(10^9+7)] = \ldots$ や $\cdots = [1-(10^9+7)] = [1] = [1+10^9+7] = [1+2\cdot(10^9+7)] = \ldots$ などが成り立つ. ◇

命題 4.1.34.

$(S, \cdot)$ を半群とする. $\sim$ を集合 S 上の同値関係であって, 任意の $x, x', y, y' \in S$ に対し $x \sim x'$ かつ $y \sim y'$ ならば $x \cdot y \sim x \cdot y'$ を満たすものとする. $S/\sim$ の上の二項演算 $\cdot' \colon S/\sim \times S/\sim \to S/\sim$ であって, 任意の $x, y \in S$ に対し $[x] \cdot' [y] = [x \cdot y]$ を満たすものが一意に存在する. ◇

証明

任意の $x, y, x', y' \in S$ に対し $[x] = [x']$ かつ $[y] = [y']$ ならば $[x \cdot y] = [x' \cdot y']$ であることを示せばよい. 同値類の定義より任意の z に対し $[z] = [z']$ ならば $z \sim z'$ であることと同値関係 $\sim$ についての仮定より明らか. □

定義 4.1.35 (商半群)**.**

$(S, \cdot)$ を半群とする. $\sim$ を集合 S 上の同値関係であって, 任意の $x, x', y, y' \in S$ に対し $x \sim x'$ かつ $y \sim y'$ ならば $x \cdot y \sim x \cdot y'$ を満たすものとする. 命題 で存在が保証される二項演算 $\cdot'$ は, 商集合 $S/\sim$ と共に半群 $(S/\sim, \cdot')$ をなす. これを S の $\sim$ による商半群と呼ぶ. ◇

定義 4.1.36 (商モノイド)**.**
省略 (商半群と同様に定義する. ただし単位元は $[e]$ である) ◇

例 4.1.37.
整数の加法のなすモノイド $(\mathbb{Z}, +, 0)$ および $a \sim b \leftrightarrow a \equiv b \pmod{10^9+7}$ という同値関係を考える. $a \equiv b \pmod{10^9+7}$ かつ $a' \equiv b' \pmod{10^9+7}$ ならば $a + a' \equiv b + b' \pmod{10^9+7}$ であることから $\mathbb{Z}/\sim$ はモノイドをなす. ◇

5 数学と計算機の接続についての準備

5.1 クエリの概念の形式化

省略する. クエリの概念を形式化しておくと複数のセグ木の間の帰着についての議論が数学的にできるようになる. この形式化は, 具体的には Haskell の State モナドのようにする. しかし, 形式化が可能であるという事実以上になにか面白い結果はなさそうである.

5.2 計算量についての暗黙の仮定

この PDF 中において, そのとき想定されている状況において半群 S について以下のふたつが成り立つことを, 半群 S は**小さい**と[9]言うこととする.

- 半群 S の要素をデータとして表現できる. 特に, その個々のデータの容量はある定数で抑えられる
- 半群 S の演算に対応する処理を実行できる. 特に, その計算時間はある定数で抑えられる

また, 小さくないことを**大きい**と呼ぶ. モノイドについても同様に定義する.

この用語法において, たとえば, 実数から実数への関数の全体と関数合成のなすモノイド $\mathbb{R}^{\mathbb{R}}$ は大きい. そのような関数は非常に多く存在するため, 何らかの制限を加えない限

[9] 独自の用語である

り, 計算機で扱えないのは明らかだろう. 一方で $\mathbb{F}_{10^9+7}$ や $\mathbb{Z}/2^{64}\mathbb{Z}$ とその加法や乗法のなすモノイドは小さい. それぞれ 4 バイトと 8 バイトだけあればそのどんな要素も表現できるし, 演算は値によらず数クロックで実行できる.

ただし 2 点の注意が必要である. 大きいか小さいかは想定されている状況に依存すること, また状況を固定しても実装方法に依存することである.

状況への依存の例としては, 自然数の全体のなすモノイド $\mathbb{Z}$ などが考えられる. 自然数は無限個存在するので, ほとんどすべての自然数は現実的にはデータとして表現できない. この意味で $\mathbb{Z}$ は大きい. 一方で, 我々が「データ構造」「含まれる要素の和」などと言うとき, 通常の 32 bit 整数や 64 bit 整数で表現してオーバーフローが起きない範囲の自然数のみを扱うことを暗黙に仮定している. この仮定は, たとえば「配列へのランダムアクセスは $O(1)$ である」のような仮定と同程度には妥当な仮定であろう. この意味で $\mathbb{Z}$ は小さいとも言える.

実装方法への依存の例としては, $\mathbb{F}_{10^9+7}$ 上の 1 次間数 $\lambda x.ax+b$ とその合成のなすモノイドを考えよう. もしこの関数を係数 a,b によって値として表現 (C++ での例: `struct { int a, b; }`) すれば, そのような表現は要件を満たすだろう. しかし, もしこの関数をそのまま関数オブジェクト (C++ での例: `std::function<int (int)>`) として表現して自然に実装したとすると, おそらく要件を満たさないだろう. つまり, 必要に応じて適切な実装方法を選ぶ必要がある.

この記事の以下では, 半群やモノイドが大きいか小さいかについては曖昧に扱い, これについての記述は基本的に省略する. 必要な場面では「小さい」という語を適宜補って読むようにせよ. これは, 大きいか小さいかの差が, 実際にプログラムとして実装した際の計算量にのみ影響し, 代数構造としての議論には影響しないためである[10].

6 通常のセグメント木

6.1 半群が乗る

どのような問題がセグメント木で扱えるかについてはモノイドを使って整理できることが知られている [7]. この事実は次の主張 6.1.1 として書かれる. この章ではこの主張からの帰結を見ていくことになる.

[10] ところで, セグメント木に関して, 大きい演算を小さい演算にうまく分割して計算量を落とす技法も存在している. しかしこれについては数学の枠組みの上に乗せにくいため省略する.

主張 6.1.1 (セグメント木には半群が乗る)**.**

半群 $(S, \cdot)$ の要素の列 $a = (a_0, a_1, \ldots, a_{n-1}) \in S^n$ があるとする. このときセグメント木を用いれば, 以下の操作が可能である:

- 点更新: 与えられた i (ただし $i < n$) と $b \in S$ に対し, $a_i \leftarrow b$ と $O(\log n)$ で更新する
- 区間取得: 与えられた l, r (ただし $0 \leq l < r \leq n$) に対し, 積 $a_l \cdot a_{l+1} \cdot \cdots \cdot a_{r-1}$ を $O(\log n)$ で計算する

ただし, これらの操作以外で直接に列 a にアクセスすることはできない. ◇

注意 6.1.2.

この主張が妥当であることは, なにかひとつセグメント木の具体例を用意し, そこで用いられている仮定を観察することで納得できるだろう[11].

たとえばプログラム 1 は $\mathbb{Z}$ の要素の列 $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$ を扱うセグメント木であった. そこでは $\mathbb{Z}$ の性質として「演算 $+$ を持つこと」「その演算が結合的であること」「その演算が単位元 0 を持つこと」のみしか用いられていないことが観察できる. 任意のモノイドはこれらの性質を持つ. また, モノイドと半群の区別は重要ではないのであった. よって, プログラム 1 のセグメント木は整数 $\mathbb{Z}$ を半群 S で置き換えることで S の要素の列 $(a_0, a_1, \ldots, a_{n-1}) \in S^n$ を扱うセグメント木になる. ◇

注意 6.1.3.

この主張はある種の「セグメント木の特徴付け」として機能する. つまり「あるデータ構造がセグメント木ではないこと」を主張する際の手段として使われることがある. [12] ◇

注意 6.1.4.

以下ではセグメント木についての操作のことを「クエリ」と呼ぶ. 点更新の操作と区間取得の操作をそれぞれ「更新クエリ」「取得クエリ」と呼び分ける.

単に「操作」や「処理」でなく「クエリ」と呼ぶことはそれが「システムの外部からの要求によって実行される処理」であることや「ある特定の操作のクラスあるいはそのインスタンス」であることを強調する響きがあるように思われる. しかしそれ以上の差はなさそうである. 実際, 以下を読むにあたっては出現する「クエリ」という語をすべて「操作」

[11] ただし, それが帰納的な推論でしかないことには注意したい

[12] もちろん, この主張自体はあくまで「セグメント木であってある条件を満たすものが存在する」ことを言うものである.

あるいは「処理」という語で置き換えて読んでも支障はない. ◇

例 6.1.5.

- 半群として $(\mathbb{N}, +)$ を選べば, 指定した区間中の要素の和を取得できるセグメント木になる
- 半群として $(\mathbb{N}, \max)$ を選べば, 指定した区間中の要素の最大値を取得できるセグメント木になる

◇

6.2 代数構造に依らず常に扱える追加のクエリ

セグメント木では, 主張 6.1.1 の形のクエリ以外にもいくつかの形のクエリを扱うことができる.

注意 6.2.1.
次の形の点更新クエリも常に処理可能である.

- 点関数適用更新: 与えられた i (ただし $i < n$) と関数 $f\colon S \to S$ に対し, $a_i \leftarrow f(a_i)$ と $O(\log n)$ で更新する

これには, 区間 $[i, i+1)$ に対する区間取得クエリにより値 a_i を取得し, セグメント木の外側で値 $b = f(a_i)$ を計算し, 通常の点更新クエリで a_i に b を代入すればよい[13]. ◇

注意 6.2.2.
次の形のクエリも常に処理可能である. ただし, セグメント木の実装のために利用する二分木が merge 操作や split 操作などが可能なものであることを仮定する.

- 列挿入: 与えられた i (ただし $i < n$) と自然数 $k \in \mathbb{N}$ と長さ k の列 $(b_0, b_1, \ldots, b_{k-1})$ に対し, 列 a を

$$a \leftarrow (a_0, a_1, \ldots, a_{i-1}, b_0, b_1, \ldots, b_{k-1}, a_i, a_{i+1}, \ldots, a_{n-1})$$

  と $O(\log n)$ で更新する. 列 a の長さは $n + k$ に変化する.

[13] セグメント木の側をこの形の点更新クエリのために特別に修正し対応させることもできる. そのような対応は, 必要な仮定や漸近的計算量は変えないが, 速度を定数倍だけ改善する.

- 列削除: 与えられた l, r (ただし $l < r$) に対し, 列 a を

$$a \leftarrow (a_0, a_1, \ldots, a_{l-1}, a_r, a_{r+1}, \ldots, a_{n-1})$$

  と $O(\log n)$ で更新する. 列 a の長さは $n - r + l$ に変化する.
- 列反転: 与えられた l, r (ただし $l < r$) に対し, 列 a を

$$a \leftarrow (a_0, a_1, \ldots, a_{l-1}, a_{r-1}, a_{r-2}, \ldots, a_l, a_r, a_{r+1}, \ldots, a_{n-1})$$

  と $O(\log n)$ で更新する. 列 a の長さは変化しない.

詳細については省略する. ◇

注意 6.2.3.
次の形のクエリも常に処理可能である.

- 二分探索: 与えられた l (ただし $l < n$) と述語 $p \subseteq S$ であって $\forall xy \in S.p(x) \to p(x \cdot y)$ を満たすものに対し, $\min \{ r \geq l \mid p(a_l \cdot a_{l+1} \cdot \cdots \cdot a_{r-1} \}$ を $O(\log n)$ で計算する

詳細については省略する. ◇

6.3 圏が乗る

セグメント木と関連するデータ構造としてより適切なものとしては圏がある [8]. たとえば, 大きさが一定とは限らない行列の積をセグメント木で計算することは可能である. しかし, 任意の大きさの行列の全体は行列積についてはモノイドや半群にはならず, 一方でこれは圏にはなる. たとえば, 数直線上の半開区間の全体の集合 $\{ [l, r) \mid l, r \in \mathbb{Z} \wedge l \leq r \}$ は端点を共有する区間同士の非交和 $[l, m) \cup [m, r) = [l, r)$ について閉じている. このような区間の非交和をセグメント木で計算することは可能である. しかし, この非交和の演算はモノイドや半群にはならず, 一方でこれは圏にはなる. ゆえに, モノイドや半群と比べて, 圏はセグメント木と関連する構造をより過不足なく捉えられていると言える.

詳細については省略する. 議論に用いられる用語を簡単なものにするため, 以下では半群とモノイドのみを議論する.

6.4 複数種類のクエリの併用

複数種類のクエリに対応したセグメント木を作ることはできるだろうか？ たとえば, 半群 $S_1 = (\mathbb{N}, +)$ によって得られるセグメント木と, 半群 $S_2 = (\mathbb{N}, \max)$ によって得られるセグメント木とを組み合わせて, 以下の 3 種のクエリを処理できるセグメント木は作れるだろうか？

- 点更新: 与えられた i, b に対し, $a_i \leftarrow b$ と $O(\log n)$ で更新する
- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $a_l + a_{l+1} + \cdots + a_{r-1}$ を $O(\log n)$ で計算する
- 区間最大値取得: 与えられた l, r (ただし $l < r$) に対し, 最大値 $\max\{ a_l, a_{l+1}, \ldots, a_{r-1} \}$ を $O(\log n)$ で計算する

自明な方法として, 2 本のセグメント木を並べるというものがある. 半群 S_1, S_2 のそれぞれに対応するセグメント木 A, B を用意し, 更新クエリについては A, B 両方に伝え, 取得クエリについては A もしくは B の適切な方に問い合わせればよい.

自明な方法以外に, 直積半群を使う方法がある. この方法では「2 本のセグメント木を使って 3 種のクエリを処理する」のではなく「3 種のクエリを処理できる 1 本のセグメント木を作る」ことができる. これには, 直積半群 $S = S_1 \times S_2$ をもとにセグメント木を作る. 更新クエリについては, 与えられた値 b を対 (b, b) に変換して更新すればよい. 取得クエリは, 与えられた区間 l, r についてそのまま問い合わせ, 結果の対から適切な要素を取り出して返せばよい.

6.5 添字の情報の利用

半群の演算の際に添字についての情報を用いたい場面は多い. これは半群を拡張することで達成できる.

例として, 最小値の位置の取得について見よう. つまり, 整数の列 $(a_0, a_1, \ldots, a_{n-1})$ についての以下の 2 種のクエリについて考える.

- 点更新: 与えられた i (ただし $i < n$) と $b \in \mathbb{Z}$ に対し, $a_i \leftarrow b$ と $O(\log n)$ で更新する
- 区間最小値位置取得: 与えられた l, r (ただし $l < r$) に対し, 最小値の位置 $\operatorname{argmin}_{i \in [l,r)} a_i$ (ただし最小値が複数ある場合は添字が最も小さいもの) を

$O(\log n)$ で計算する

これには整数の集合 $\mathbb{Z}$ と自然数の集合 $\mathbb{N}$ の直積 $\mathbb{Z} \times \mathbb{N}$ をとり, この直積に対し辞書順で全順序を入れてできる最小値についての半群 $S = (\mathbb{Z} \times \mathbb{N}, \min)$ を考えればよい. この S の要素についての積を取ったとき, その結果の対の第 2 要素は目的の最小値の位置になっている. なお, 最小値半群と最小値半群との直積半群を用いているのではないことに注意してほしい.

たとえば長さ $n = 6$ の整数列 $a = (50, 27, 43, 39, 39, 48)$ を考えるとき, 代わりに $S \times \mathbb{N}$ の列 $a' = ((50, 0), (27, 1), (43, 2), (39, 3), (39, 4), (48, 5))$ を使う. このとき区間 $[2, 6)$ についての a' での総和 $\min_{i \in [2,6)} = \{ a'_2, a'_3, a'_4, a'_5 \} = a'_3 = (39, 3)$ であり, その第 2 要素は目的の添字 $\operatorname{argmin}_{i \in [2,6)} a_i = 3$ と一致する.

注意 6.5.1.
添字の情報の利用はセグメント木の側を拡張することでも実現可能である. しかし, そのようにセグメント木を拡張することは, 半群を拡張することと実質的に同じである. ◇

7 双対セグメント木

7.1 モノイドが乗る

6 章と同様に, 次の主張を認めるところから初めよう.

主張 7.1.1 (双対セグメント木にはモノイドが乗る)**.**
モノイド $(M, \cdot, e)$ の要素の列 $a = (a_0, a_1, \ldots, a_{n-1}) \in M^n$ があるとする. このとき双対セグメント木を用いれば, 以下の操作が可能である:

- 区間更新: 与えられた l, r (ただし $l \leq r$) と $b \in M$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow b \cdot a_i$ と $O(\log n)$ で更新する
- 点取得: 与えられた i (ただし $i < n$) に対し, 要素 a_i を $O(\log n)$ で計算する

ただし, これらの操作以外で直接に列 a にアクセスすることはできない. ◇

例 7.1.2.

- モノイドとして $(\mathbb{N}, +, 0)$ を選べば, 指定した区間に一様加算できる双対セグメン

ト木になる

- モノイドとして $(\mathbb{N}, \max, 0)$ を選べば, 指定した区間中の要素を指定した値との最大値で更新できる双対セグメント木になる
- モノイドとして左零半群に単位元を付与してできるモノイド $(X \cup \{ e \}, \lambda xy.x, e)$ を選べば, 指定した区間中に一様代入できる双対セグメント木になる

◇

注意 7.1.3.

双対セグメント木の抽象化において, 作用を考える必要はない. 区間更新などが別の列への作用を意図している場合, 作用される集合の要素の列を別に持っておけば十分であるためである. たとえば自然数の列 $(x_0, x_1, \ldots, x_{n-1})$ を一次関数の適用によって更新していくことを考えよう. これには一次関数とその関数合成によるモノイド $F = (\{ \lambda x.ax + b \mid a, b \in \mathbb{N} \}, \circ, \mathrm{id})$ を考え, この F のみについての双対セグメント木を用意して関数の列 $(f_0, f_1, \ldots, f_{n-1})$ を管理する. ただし取得クエリの際に, f_i を返すのでなく $f_i(x_i)$ を返すようにする. これで目的の自然数の列を管理することと同じになる.

◇

7.2 代数構造に依らず常に扱える追加のクエリ

双対セグメント木では, 主張 7.1.1 の形のクエリ以外の形のクエリはほぼ扱えない.

注意 7.2.1.

注意 6.2.2 と同様にすれば, 列挿入クエリや列削除クエリや列反転クエリも扱うことができる. ◇

7.3 複数種類のクエリの併用

まずいくつか追加の準備をする.

定義 7.3.1 (自由モノイド)**.**

A を集合とする. A の元からなる有限列の全体を A^* で表すと, A^* は列の連結 $\frown$ を演算とし空列 ϵ を単位元としてモノイド $(A^*, \frown, \epsilon)$ をなす. これを A によって生成される**自由モノイド**と呼ぶ. ◇

定義 7.3.2 (自由積)**.**

$M_1 = (M_1, \cdot_1, e_1)$ と $M_2 = (M_2, \cdot_2, e_2)$ をモノイドとし, $M_1 \cap M_2 = \emptyset$ と仮定する. 集合 $M_1 \cup M_2$ によって生成される自由モノイドの上の同値関係 $\sim$ を, 以下のすべてを満たす最小の同値関係として定義する.

- α, β を $M_1 \cup M_2$ の要素の列とし $a_1, a_2 \in M_1$ とすると, $(\alpha \frown (a_1, a_2) \frown \beta) \sim (\alpha \frown (a_1 \cdot_1 a_2) \frown \beta)$ が成り立つ
- α, β を $M_1 \cup M_2$ の要素の列とし $b_1, b_2 \in M_2$ とすると, $(\alpha \frown (b_1, b_2) \frown \beta) \sim (\alpha \frown (b_1 \cdot_2 b_2) \frown \beta)$ が成り立つ
- α, β を $M_1 \cup M_2$ の要素の列とすると, $(\alpha \frown (e_1) \frown \beta) \sim (\alpha \frown \beta)$ が成り立つ
- α, β を $M_1 \cup M_2$ の要素の列とすると, $(\alpha \frown (e_2) \frown \beta) \sim (\alpha \frown \beta)$ が成り立つ

このとき, 集合 $M_1 \cup M_2$ の同値関係 $\sim$ による商 $(M_1 \cup M_2)/\sim$ は, 代表元の連結 $\frown'$ を演算とし空列 ϵ を単位元としてモノイド $(M_1 + M_2, \frown', \epsilon)$ をなす. このモノイド $M_1 + M_2$ を M_1 と M_2 の**自由積**と呼ぶ. ◇

例 7.3.3.

自然数の加法についてのモノイド $(\mathbb{N}, +, 0)$ と, 文字 a, b からなる文字列の連結についてのモノイド $(\{ a, b \}^*, \frown, \epsilon)$ との自由積を $M = \mathbb{N} + \{ a, b \}^*$ とする. このとき $(2, 3, ab)$ や $(ab, 0, 1, 2, ab, aba, 0)$ は M の要素[14]である. また $(2, 3, ab) = (5, ab)$ や $(ab, 0, 1, 2, ab, aba, 0) = (ab, 3, ababa)$ などが成り立つ. ◇

注意 7.3.4.

M_1, M_2 をそれらと同型なモノイドで取り替えて $M_1 \cap M_2 = \emptyset$ が成り立つようにすることが常にできる. このため $M_1 \cap M_2 = \emptyset$ という条件は基本的に忘れてよい. 以降ではこの事実を断わりなく使うことがある. ◇

例 7.3.5.

自然数の加法についてのモノイド $M_1 = (\mathbb{N}, +, 0)$ と 自然数の乗法についてのモノイド $M_2 = (\mathbb{N}, \cdot, 1)$ があるとする. $\mathbb{N} \cap \mathbb{N} = \mathbb{N} \neq \emptyset$ なので, 定義に厳密に従えば, M_1 と M_2 の自由積は作れない. しかしこのとき, 対の集合 $\{ 1 \} \times \mathbb{N} = \{ (1, n) \mid n \in \mathbb{N} \}$ とその上の演算 $(1, x) +' (1, y) = (1, x + y)$ およびある対 $(1, 0) \in \{ 1 \} \times \mathbb{N}$ を考える. これらはモノイド $M_1' = (\{ 1 \} \times \mathbb{N}, +', (1, 0))$ をなし, また $M_1 \simeq M_1'$ が示せる. 同様にし

[14] 正確には, M の要素の代表元

て $M_1 \simeq M_2'$ な $M_2' = (\{\,2\,\} \times \mathbb{N}, \cdot', (2,1))$ も得られる. $\{\,1\,\} \times \mathbb{N} \cap \{\,2\,\} \times \mathbb{N} = \emptyset$ なので M_1' と M_2' の自由積 $M_1' + M_2'$ は存在する. そしてほとんどの場合, この $M_1' + M_2'$ を M_1 と M_2 の自由積であるかのように扱ってしまってもかまわない. ◇

さて, 準備を終えたので, 次のような問題を考えていこう: 複数種類のクエリに対応した双対セグメント木を作ることはできるだろうか？ たとえば, 区間加算更新ができる双対セグメント木と, 区間最大値更新ができる双対セグメント木とがあるとき, それらの両方ができる双対セグメント木を作れるだろうか？ つまり, モノイド $M_1 = (\mathbb{N}, +, 0)$ とモノイド $M_2 = (\mathbb{N}, \max, 0)$ を使って, 自然数の列 $(x_0, x_1, \ldots, x_{n-1})$ についての以下の 3 種のクエリを処理できる双対セグメント木は作れるだろうか？

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $a \in \mathbb{N}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a + x_i$ と $O(\log n)$ で更新する
- 区間最大値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{N}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow \max(b, x_i)$ と $O(\log n)$ で更新する
- 点取得: 与えられた i に対し, x_i を $O(\log n)$ で計算する

計算量を無視すれば, これは自由積 $M_1 + M_2$ を考えることで可能となる. モノイド M_1, M_2 のそれぞれをそれらの用途に合致していてかつ同型なモノイドで取り替えて $M_1 = (\{\,\lambda x.a + x \mid a \in \mathbb{N}\,\}, \circ, \mathrm{id})$ および $M_2 = (\{\,\lambda x.\max(b, x) \mid b \in \mathbb{N}\,\}, \circ, \mathrm{id})$ と書くことにしよう. $M_1 + M_2$ の要素はたとえば $[(\lambda x.2+x, \lambda x.3+x, \lambda x.\max(5, x), \lambda x.1+x)]$ のようになる. この $M_1 + M_2$ についての双対セグメント木を用意する. 更新クエリにおいては, 与えられた g (具体的には $\lambda x.a + x$ か $\lambda x.\max(b, x)$) を自由積の要素 $[(g)]$ とみて更新するように変換する. 取得クエリにおいては, 与えられた i について取得した自由積の値 $f_i = [(g_0, g_1, \ldots, g_k)]$ を使って $g_0(g_1(\ldots g_k(x_i)\ldots))$ を計算し, これを返せばよい. このようにすれば, 少なくともクエリの処理結果については正しい双対セグメント木が得られる.

ただし, 自由積の要素は必ずしも小さいとは限らない. 自由積の要素は列であったが, この列の長さに上界がないためである. よって, そのままでは「$O(\log n)$ で」という要求は達成できない. $O(\log n)$ を達成するには「自由積の値をどのように利用する予定であるか」の情報を用いることが必須である. その上で, モノイド M_1 と M_2 の関係をよく見て専用の処理をする必要がある. 具体的には $(\lambda x.a + x) \circ (\lambda x.\max(b, x)) = (\lambda x.\max(a + b, x)) \circ (\lambda x.a + x)$ という等式を利用する. この等式があれば, 自由積 $M_1 + M_2$ の任意の要素を $\lambda x.\max(b, a + x)$ の形の関数と同一視できることが示せ, デー

タとしてはふたつの自然数 a, b だけを持てばよいことが分かる. つまり, 今回例に挙げた自由積は $M_1 + M_2$ は小さい. よってクエリは $O(\log n)$ で処理でき, これで所望の双対セグメント木が得られたと言える.

さて, クエリの併用について議論は双対セグメント木がどのような意味で「双対」であるのかの説明も与えてくれる. クエリの併用のためには, 通常のセグメント木においては直積半群を用い, 双対セグメント木においては自由積を用いるということであった. 詳細は省略するが, 直積モノイドはモノイドの圏における直積対象であり, モノイドの自由積はモノイドの圏における直和対象である. つまり, クエリの併用を考えたときに出現する構成物が圏論的な双対の関係にある[15]という意味において, 双対セグメント木は通常のセグメント木の双対である.

まとめると以下のようになる.

- 計算量を無視すれば, クエリの併用は自由積で説明できる
- 計算量を考慮するならば, 作用としての用途を前提としたクエリごとに個別の議論が必要になる
- 双対セグメント木に対して「双対」という語を用いることは, クエリの併用の観点から説明できる

なお, 自由積の表現については後でまた議論する.

7.4 添字の情報の利用

モノイドを作用させる際に作用している区間についての情報を用いたい場面は多い. これは添字の情報の利用を取得クエリの時点まで遅らせることで達成できる.

例として, 添字に依存した更新について見よう. つまり, 整数の列 $(c_0, c_1, \ldots, c_{n-1})$ についての以下の 2 種のクエリについて考える.

- 区間一次関数更新: 与えられた l, r (ただし $l \leq r$) と $a, b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $c_i \leftarrow c_i + ai + b$ と $O(\log n)$ で更新する
- 点取得: 与えられた i (ただし $i < n$) に対し, 値 x_i を $O(\log n)$ で計算する

これには一次関数とその関数合成によるモノイド $F = (\{\, \lambda x.ax + b \mid a, b \in \mathbb{N} \,\}, \circ, \mathrm{id})$ を考えればよい. 更新クエリは添字に依存しない関数合成として行い, 取得クエリの際に

[15] 今回の定義では通常のセグメント木と双対セグメント木にはそれぞれ半群とモノイドを使うという差があるが, これは無視して構わないだろう.

添字の情報を引数として注入して値を求める. 初期状態には数列の代わりに定数関数の列 $(\lambda x.c_0, \lambda x.c_1, \ldots, \lambda x.c_{n-1})$ を用いる.

7.5 一様代入クエリ

モノイドとして左零半群に単位元を付与してできるモノイド $(X \cup \{ * \}, \lambda xy.x, *)$ を選べば, 指定した区間中に一様代入できる双対セグメント木になる. つまり, 集合 X の要素の列 $(x_0, x_1, \ldots, x_{n-1})$ について, 以下の 2 種のクエリを処理できる双対セグメント木が得られる.

- 区間一様代入更新: 与えられた l, r (ただし $l < r$) と $a \in X$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a$ と $O(\log n)$ で更新する
- 点取得: 与えられた l, r (ただし $l < r$) に対し, 値 x_l を $O(\log n)$ で計算する

また, 一様代入クエリは, 計算量を悪化させずに他のクエリと併用可能である. この詳細は遅延伝播セグメント木の場合とまとめて 8.12 節で議論する.

8 遅延伝播セグメント木

8.1 追加の準備

定義 8.1.1 (モノイドについての自己準同型モノイド)**.**
M をモノイドとする. M から M への準同型のことを M の**自己準同型**と呼ぶ. M の自己準同型の全体を $\mathrm{End}(M)$ と書く. これは関数合成によってモノイド $(\mathrm{End}(M), \circ, \mathrm{id})$ をなす. これを M についての**自己準同型モノイド**と呼ぶ. ◇

定義 8.1.2 (半群についての自己準同型モノイド)**.**
S を半群とする. S から S への準同型のことを S の**自己準同型**と呼ぶ. S の自己準同型の全体を $\mathrm{End}(S)$ と書く. これは関数合成によってモノイド $(\mathrm{End}(S), \circ, \mathrm{id})$ をなす. これを S についての**自己準同型モノイド**と呼ぶ. ◇

例 8.1.3.
関数 $\varphi(n) = \lambda x.nx$ を考える. 整数と乗法のなすモノイド $M = (\mathbb{Z}, \cdot, 1)$ と, 整数と加法のなすモノイド $N = (\mathbb{Z}, +, 0)$ の自己準同型モノイド $\mathrm{End}(N)$ とを考えたとき, φ は M から $\mathrm{End}(N)$ へのモノイド準同型である. ◇

定義 8.1.4 (モノイド作用)**.**
$(M, \cdot, e)$ をモノイドとし, X を集合とする. 演算 $\star$: $M \times X \to X$ が以下のすべてを満たすとき, $\star$ を M による集合 X への**モノイド作用**と呼ぶ.

- 任意の $f, g \in M$ と $x \in X$ について, $(f \cdot g) \star x = f \star (g \star x)$
- 任意の $x \in X$ について, $e \star x = x$

◇

例 8.1.5.
自然数の乗法についてのモノイド $(\mathbb{N}, \cdot, 1)$ と自然数の全体の集合 $\mathbb{N}$ の間に $a \star x = x^a$ という演算を考えると, これは $\mathbb{N}$ へのモノイド作用である. ◇

命題 8.1.6.
$(M, \cdot, e)$ をモノイドとし, X を集合とする. このとき, 演算 $\star$: $M \times X \to X$ について, 以下のふたつの命題は同値.

(1.) $\star$: $M \times X \to X$ はモノイド作用である
(2.) $\varphi(a) = \lambda x.a \star x$ で定まる関数 φ: $M \to X^X$ はモノイド準同型である

ただし X^X は X から X への関数全体と関数合成からなるモノイドであった. ◇

証明

(1.) $\Rightarrow$ (2.) **について**　任意の $f, g \in M$ に対し

$$\begin{aligned}\varphi(fg) &= \lambda x.(fg) \star x \\ &= \lambda x.f \star (g \star x) \\ &= (\lambda x.f \star x) \circ (\lambda x.g \star x) \\ &= \varphi(f) \circ \varphi(g)\end{aligned}$$

がなりたつ. また

$$\begin{aligned}\varphi(e) &= \lambda x.e \star x \\ &= \mathrm{id}_X\end{aligned}$$

である. よって φ はモノイド準同型である.

(2.) ⇒ (1.) **について**　任意の $f, g \in M$ と任意の $x \in X$ に対し

$$\begin{aligned}(fg) \star x &= \varphi(fg)(x) \\ &= (\varphi(f) \circ \varphi(g))(x) \\ &= \varphi(f)(\varphi(g)(x)) \\ &= f \star (g \star x)\end{aligned}$$

がなりたつ. また

$$\begin{aligned}e \star x &= \varphi(e)(x) \\ &= \mathrm{id}_X(x) \\ &= x\end{aligned}$$

である. よって $\star$ はモノイド作用である.

□

命題 8.1.7.

$(M, \cdot_M, 1_M)$ をモノイドとし $(S, \cdot_S)$ を半群とする. このとき, 演算 $\star\colon M \times S \to S$ について, 以下のふたつの命題は同値.

(1.) $\star\colon M \times S \to S$ は集合 S へのモノイド作用である. また, 任意の $f \in M$ と $x, y \in S$ について, $f \star (x \cdot_S y) = (f \star x) \cdot_S (f \star y)$ が成り立つ

(2.) 任意の $f \in M$ について, $\lambda x.f \star x$ は半群準同型である. また, $\varphi(f) = \lambda x.f \star x$ で定まる関数 $\varphi\colon M \to \mathrm{End}(S)$ はモノイド準同型である

◇

証明

(1.) ⇒ (2.) **について**　仮定より明らかに, 任意の $f \in M$ に対し $\lambda x.f \star x$ は半群準同型である. 命題 8.1.6 により $\varphi(f)$ はモノイド準同型である.

(2.) ⇒ (1.) **について**　命題 8.1.6 により $\star$ はモノイド作用である. 任意の $f \in M$, $x, y \in S$ に対し

$$\begin{aligned}f \star (x \cdot_S y) &= \varphi(f)(x \cdot_S y) \\ &= \varphi(f)(x) \cdot_S \varphi(f)(y) \\ &= (f \star x) \cdot_S (f \star y).\end{aligned}$$

である.

□

8.2 半群についての自己準同型モノイドへのモノイド準同型を使った構造が乗る

これまでと同様に, 主張 8.2.4 を認めるものとする[16].

定義 8.2.1 (遅延伝播構造[17])**.**
モノイド M と半群 S とモノイド準同型 $\varphi\colon M \to \mathrm{End}(S)$ からなる 3 つ組 (M, S, φ) を**遅延伝播構造**と呼ぶ. ◇

注意 8.2.2.
遅延伝播構造は $f\star(x\cdot_S y) = (f\star x)\cdot_S (f\star y)$ を満たすようなモノイド作用 $\star\colon M\times S \to S$ を使う定義もできる. 命題 8.1.7 により, これらは互いに同値である. ◇

記法 8.2.3.
遅延伝播構造 (M, S, φ) があるとき $\star$ と書けば, これは $f \star x = \varphi(f)(x)$ で定まるモノイド作用 $\star\colon M \times S \to S$ のことであるとする. また, そのようなモノイド作用 $\star$ があるとき $(M, S, \star)$ と書けば, これは遅延伝播構造 (M, S, φ) のことであるとする. ◇

主張 8.2.4 (遅延伝播セグメント木には遅延伝播構造が乗る)**.**
モノイド $(M, \cdot_M, e)$ と半群 $(S, \cdot_S)$ からなる遅延伝播構造 $(M, S, \star)$ があるとする. S の要素の列 $x = (x_0, x_1, \ldots, x_{n-1}) \in S^n$ があるとする. このとき遅延伝播セグメント木を用いれば, 以下の操作が可能である:

- 区間更新: 与えられた l, r (ただし $0 \le l \le r \le n$) と $f \in M$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow f \star x_i$ と $O(\log n)$ で更新する
- 区間取得: 与えられた l, r (ただし $0 \le l < r \le n$) に対し, 積 $x_l \cdot_S x_{l+1} \cdot_S \cdots_S x_{r-1}$

[16] TODO(kimiyuki): この主張はこの PDF の中で最も重要な結果だし「認める」じゃだめでは？

[17] 新たに導入される独自の用語. 遅延伝播セグメント木の文脈における分かりやすさを優先してのとりあえずの命名である. これは用途のひとつに注目しての命名でしかなく, 代数構造そのものの性質を表していないために将来的には不適切なものとなるかもしれない. より良い名前がもしあればそれで置き換えられることを期待する. また, 後に「正規な遅延伝播構造」として定義するすこし異なる構造とこの「遅延伝播構造」とのどちらを基本的なものとして定義するかの選択の幅があるが, これは議論のしやすさを優先して選択した.

を $O(\log n)$ で計算する

ただし, これらの操作以外で直接に列 x にアクセスすることはできない. ◇

注意 8.2.5.

もし S に単位元 e_S が存在したとしても $f \star e_S = e_S$ である必要はない. つまり, S の側を半群でなくモノイドで取り替えても, その取り替えたモノイドの単位元についての追加の条件が作用に課されることはない[18]. ◇

注意 8.2.6.

遅延伝播セグメント木が要求する構造は, 遅延伝播構造を用いずに「半群 S および特定の条件を満たす関数のある集合 $F \subseteq \{\, f \mid f\colon S \to S \,\}$ の組」などとも表現できる. このような表現は, 数学を避けて平易に要件を説明したい場合には妥当であろう. しかしこのような表現は遅延伝播セグメント木が要求する構造についての分析を面倒にしてしまう. これはたとえば, 群を「関数の集合であって恒等関数を含みかつ関数合成と逆関数を得る操作について閉じているもの」と言い換えた場合と同様である. ◇

8.3 代数構造に依らず常に扱える追加のクエリ

遅延伝播セグメント木では, 主張 8.2.4 の形のクエリ以外にもいくつかの形のクエリを扱うことができる. これは 6.2 節で見たものとほぼ同様である.

注意 8.3.1.

遅延伝播構造の側でなく遅延伝播セグメント木の側を修正することで, 追加で次の操作も可能である.

- 点更新: 与えられた i (ただし $i < n$) と $y \in S$ に対し, $x_i \leftarrow y$ と $O(\log n)$ で更新する

遅延伝播セグメント木における一様代入クエリの対応は自明ではないために, この点更新クエリは有用である. 一様代入クエリについては後の 8.6 節で議論する. ◇

注意 8.3.2.

[18] このことを理由として, この PDF では通常のセグメント木に半群を用いている. また, 定義より $e_M \star x = x$ は要請されている. このことを理由として, この PDF では双対セグメント木にモノイドを用いている.

注意 6.2.1 と同様にすれば, 関数適用による点更新クエリも扱うことができる. ◇

注意 8.3.3.
注意 6.2.2 と同様にすれば, 列挿入クエリや列削除クエリや列反転クエリも扱うことができる. ◇

注意 8.3.4.
注意 6.2.3 と同様にすれば, 二分探索クエリも扱うことができる. ◇

8.4 加群を一般化して得られる構造が乗る

定義 8.4.1 (加群)**.**
$(R, +, \cdot, 0_R, 1_R)$ を (必ずしも可換とは限らない) 環とし, $(M, +, 0_M)$ を可換群とする. 演算 $\star$: $R \times M \to M$ が以下のすべてを満たすとき, 3 つ組 $(R, M, \star)$ を環 R 上の左 R-**加群**と呼ぶ.

- 任意の $r \in R$ と $x, y \in M$ について, $r \star (x + y) = (r \star x) + (r \star y)$
- 任意の $r, s \in R$ と $x \in M$ について, $(r + s) \star x = (r \star x) + (s \star x)$
- 任意の $r, s \in R$ と $x \in M$ について, $(r \cdot s) \star x = r \star (s \star x)$
- 任意の $x \in M$ について, $1_R \star x = x$

この R と M の間の演算 $\star$: $R \times M \to M$ はスカラー乗法と呼ばれる. ◇

定理 8.4.2 (遅延伝播構造は加群の一般化である)**.**
左 R-加群 M があり, そのスカラー乗法を $\star$ と書くとする. このとき $(R, M, \star)$ は遅延伝播構造である. ◇

注意 8.4.3.
このことから, 以下の主張が系として得られる.

- 加群ならば遅延伝播セグメント木に乗せられる
- ベクトル空間ならば遅延伝播セグメント木に乗せられる
- 半環ならば遅延伝播セグメント木に乗せられる
- 可換環ならば遅延伝播セグメント木に乗せられる
- 体ならば遅延伝播セグメント木に乗せられる

◇

例 8.4.4.

- 遅延伝播構造として整数環 $\mathbb{Z} = (\mathbb{Z}, +, \cdot, 0, 1)$ から得られるものを使えば, 区間一様乗算更新と区間和取得の遅延伝播セグメント木になる
- 遅延伝播構造としてトロピカル半環 $(\mathbb{R} \cup \{ \infty \}, \min, +, \infty, 0)$ から得られるものを使えば, 区間一様加算更新と区間最小値取得の遅延伝播セグメント木になる

◇

8.5 区間の情報の利用

モノイドを作用させる際に作用している区間についての情報を用いたい場面は多い. これは半群を拡張することで達成できる. この節は 6.5 節および 7.4 節の続きである.

例として, 整数の列 $(x_0, x_1, \ldots, x_{n-1})$ についての以下の 2 種のクエリについて考える.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $a \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a + x_i$ と $O(\log n)$ で更新する
- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $x_l + x_{l+1} + \cdots + x_{r-1}$ を $O(\log n)$ で計算する

これをセグメント木を用いて素朴に実装しようとすると, モノイドと半群の間の作用において区間の長さが必要となることに注意したい. これは $(a + x_l) + (a + x_{l+1}) + \cdots + (a + x_{r-1}) = a + (x_l + x_{l+1} + \cdots + x_{r-1})$ とは限らないためである. $(r - l) \cdot a$ を計算すればよいことを知るには区間 $[l, r)$ についての作用であることを知る必要がある.

これを扱える遅延伝播セグメント木は, 次の補題 8.5.2 で得られる遅延伝播構造から得られる.

補題 8.5.1.

整数についての加法モノイド $M = (\mathbb{Z}, +, 0)$ と整数についての加法半群 $S = (\mathbb{Z}, +)$ を考える. S と自然数の加法半群 $(\mathbb{N}, +)$ との直積半群 $S \times \mathbb{N}$ に関する関数 $\varphi \colon M \to \mathrm{End}(S \times \mathbb{N})$ を

$$\varphi(f)(x, k) = (kf + x, k)$$

で定義する. このとき $(M, S \times \mathbb{N}, \varphi)$ は遅延伝播構造である. ◇

証明

まず φ が well-defined であること, つまり $\varphi(f)\colon S \times \mathbb{N} \to S \times \mathbb{N}$ が半群準同型であることを示す. 任意の $f \in M$ と任意の $(x,k),(y,k') \in S \times \mathbb{N}$ に対し

$$\begin{aligned} & \varphi(f)((x,k)\cdot(y,k')) \\ =\ & \varphi(f)(x+y,k+k') \\ =\ & ((k+k')f+x+y,k+k') \\ =\ & (kf+x,k)\cdot(k'f+y,k') \\ =\ & \varphi(f)(x,k)\cdot\varphi(f)(y,k') \end{aligned}$$

であるので, $\varphi(f)$ は半群準同型である. よって φ は well-defined である.

あとは φ がモノイド準同型であることを示せばよい.

- 任意の $(x,k) \in S \times \mathbb{N}$ に対し $\varphi(0)(x,k) = (k \cdot 0 + x, k) = (x,k)$ である.
- 任意の $f,g \in M$ と任意の $(x,k) \in S \times \mathbb{N}$ に対し

  $$\begin{aligned} & \varphi(f+g)(x,k) \\ =\ & (k\cdot(f+g)+x,k) \\ =\ & (kf+(kg+x),k) \\ =\ & \varphi(f)(kg+x,k) \\ =\ & \varphi(f)(\varphi(g)(x,k)) \\ =\ & (\varphi(f)\circ\varphi(g))(x,k) \end{aligned}$$

  が成り立つ. つまり $\varphi(f+g) = \varphi(f) \circ \varphi(g)$ である.

よって φ はモノイド準同型である. □

この例をさらに強めて, 可換環 R の要素の列 $(x_0, x_1, \ldots, x_{n-1})$ についての以下の 2 種のクエリについて考える.

- 区間一次関数更新: 与えられた l, r (ただし $l < r$) と $a, b \in R$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a x_i + b$ と $O(\log n)$ で更新する
- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $x_l + x_{l+1} + \cdots + x_{r-1}$ を $O(\log n)$ で計算する

これは, 注意 8.5.3 を踏まえた上で, 次の補題 8.5.2 のような遅延伝播構造による遅延伝播セグメント木で処理できる.

補題 8.5.2.

可換環 R についての一次関数のなすモノイド $M = (\{\, \lambda x.ax + b \mid a, b \in R \,\}, \circ, \lambda x.1 \cdot x + 0)$ と加法半群 $S = (R, +)$ を考える. S と自然数の加法半群 $(\mathbb{N}, +)$ との直積半群 $S \times \mathbb{N}$ に関する関数 $\varphi\colon M \to \mathrm{End}(S \times \mathbb{N})$ を

$$\varphi(\lambda x.ax + b)(x, k) = (ax + kb, k)$$

で定義する. このとき $(M, S \times \mathbb{N}, \varphi)$ は遅延伝播構造である. ◇

証明

まず φ が well-defined であること, つまり $\varphi(f)\colon S \times \mathbb{N} \to S \times \mathbb{N}$ が半群準同型であることを示す. 任意の $f = \lambda x.ax + b \in M$ と任意の $(x, k), (y, k') \in S \times \mathbb{N}$ に対し

$$\begin{aligned} & \varphi(f)((x, k) \cdot (y, k')) \\ = \ & \varphi(f)(x + y, k + k') \\ = \ & (a(x + y) + (k + k')b, k + k') \\ = \ & ((ax + kb) + (ay + k'b), k + k') \\ = \ & \varphi(f)(x, k) \cdot \varphi(f)(y, k') \end{aligned}$$

であるので, $\varphi(f)$ は半群準同型である. よって φ は well-defined である.

あとは φ がモノイド準同型であることを示せばよい.

- 任意の $(x, k) \in S \times \mathbb{N}$ に対し $\varphi(\lambda x.1 \cdot x + 0)(x, k) = (1 \cdot x + 0, k) = (x, k)$ である.
- 任意の $f = \lambda x.ax + b \in M$ と $g = \lambda x.cx + d \in M$ と任意の $(x, k) \in S \times \mathbb{N}$ に対し

  $$\begin{aligned} & \varphi(f \circ g)(x, k) \\ = \ & \varphi(\lambda x.acx + ad + b)(x, k) \\ = \ & (acx + ad + b, k) \\ = \ & \varphi(f)(cx + d, k) \\ = \ & \varphi(f)(\varphi(g)(x, k)) \\ = \ & (\varphi(f) \circ \varphi(g))(x, k) \end{aligned}$$

  が成り立つ. つまり $\varphi(f \circ g) = \varphi(f) \circ \varphi(g)$ である.

よって φ はモノイド準同型である. □

注意 8.5.3.

補題 8.5.2 の遅延伝播構造を用いるには, 可換環 R の要素 $b \in R$ と自然数 $k \in \mathbb{N}$ に対

し自然数倍 kb を $O(1)$ で計算できる必要がある．そうでない場合は繰り返し二乗法を用いることになり, クエリの計算量は $O((\log n)^2)$ に悪化する． ◇

なお, 区間の情報は使わない場合は常に付け加えることができる．より一般的な補題として次を示しておこう．

補題 8.5.4.
(M, S, φ) を遅延伝播構造とする．T を半群とする．関数 $\varphi' \colon M \to \mathrm{End}(S \times T)$ を

$$\varphi'(f)(x,t) = (\varphi(f)(x), t)$$

で定義する．このとき $(M, S \times T, \varphi')$ は遅延伝播構造である． ◇

証明
まず φ' が well-defined であること, つまり $\varphi'(f) \colon S \times T \to S \times T$ が半群準同型であることを示す．任意の $f \in M$ と任意の $(x,t), (y,t') \in S \times T$ に対し

$$\begin{aligned}
& \varphi'(f)((x,t) \cdot (y,t')) \\
=\ & \varphi'(f)(x \cdot y, t \cdot t') \\
=\ & (\varphi(f)(x \cdot y), t \cdot t') \\
=\ & (\varphi(f)(x) \cdot \varphi(f)(y), t \cdot t') \\
=\ & (\varphi(f)(x), t) \cdot (\varphi(f)(y), t') \\
=\ & \varphi'(f)(x,t) \cdot \varphi'(f)(y,t')
\end{aligned}$$

であるので, $\varphi'(f)$ は半群準同型である．よって φ' は well-defined である．

あとは φ' がモノイド準同型であることを示せばよい．

- 任意の $(x,t) \in S \times T$ に対し $\varphi'(e)(x,t) = (\varphi(e)(x), t) = (x,t)$ である．
- 任意の $f, g \in M$ と任意の $(x,t) \in S \times T$ に対し

$$\begin{aligned}
& \varphi'(f \cdot g)(x,t) \\
=\ & (\varphi(f \cdot g)(x), t) \\
=\ & (\varphi(f)(\varphi(g)(x)), t) \\
=\ & \varphi'(f)(\varphi(g)(x), t) \\
=\ & \varphi'(f)(\varphi'(g)(x,t)) \\
=\ & (\varphi'(f) \circ \varphi'(g))(x,t)
\end{aligned}$$

が成り立つ．つまり $\varphi'(f \cdot g) = \varphi'(f) \circ \varphi'(g)$ である．

よって φ' はモノイド準同型である. □

注意 8.5.5.

区間の情報の利用はセグメント木の側を拡張することでも実現可能である. しかし, そのようにセグメント木を拡張することは, 半群を拡張することと実質的に同じである. ◇

8.6 一様代入クエリ

双対セグメント木の場合の一様代入クエリについては 7.5 節で議論した. この節では遅延伝播セグメント木の場合の一様代入クエリについて確認する. つまり, 半群 S の要素の列 $(x_0, x_1, \ldots, x_{n-1})$ についての以下の 2 種のクエリを処理できる遅延伝播セグメント木を考える.

- 区間一様代入更新: 与えられた l, r (ただし $l < r$) と $a \in S$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a$ と $O(\log n)$ で更新する
- 区間取得: 与えられた l, r (ただし $l < r$) に対し, 積 $x_l \cdot x_{l+1} \cdot \cdots \cdot x_{r-1}$ を $O(\log n)$ で計算する

このとき, そのまま左零半群に単位元を付与してできるモノイドを使うことはできない. 左零半群に単位元を付与してできるモノイドでは $\varphi(f)(x \cdot y) = f$ だが $\varphi(f)(x) \cdot \varphi(f)(y) = f \cdot f = f^2$ となり遅延伝播構造の要件を満たさないためである.

これには 8.5 節で見たようにして半群の側に区間の長さの情報を乗せればよい. 具体的には次の定理 8.6.1 で得られる遅延伝播構造を用いればよい.

定理 8.6.1.

$(S, \cdot_S)$ を半群とする. 自然数の加法についての半群 $\mathbb{N}$ との直積半群 $S \times \mathbb{N}$ を考える. 集合 S についての左零半群に単位元を付与してできるモノイドを $M = (S \cup \{ e \}, \cdot_M, e)$ とする. 関数 $\varphi\colon M \to \mathrm{End}(S \times \mathbb{N})$ を

$$\varphi(f)(x, k) = \begin{cases} (x, k) & (f = e) \\ (kf, k) & (f \neq e) \end{cases}$$

で定義する. このとき $(M, S \times \mathbb{N}, \varphi)$ は遅延伝播構造である. ◇

証明

まず φ が well-defined であること, つまり $\varphi(f)\colon S \times \mathbb{N} \to S \times \mathbb{N}$ が半群準同型であることを示す.

- 任意の $(x,k),(y,k') \in S \times \mathbb{N}$ に対し $\varphi(e)((x,k)\cdot(y,k')) = (x,k)\cdot(y,k') = \varphi(e)(x,k)\cdot\varphi(e)(y,k')$ である.
- 任意の $f \in M \setminus \{ e \}$ と任意の $(x,k),(y,k') \in S \times \mathbb{N}$ に対し

$$\begin{aligned} & \varphi(f)((x,k)\cdot(y,k')) \\ = \ & \varphi(f)(x+y,k+k') \\ = \ & ((k+k')f,k+k') \\ = \ & \varphi(f)(x,k)\cdot\varphi(f)(y,k') \end{aligned}$$

  である.

よって $\varphi(f)$ は半群準同型であり, φ は well-defined である.

あとは φ がモノイド準同型であることを示せばよい.

- 任意の $(x,k) \in S \times \mathbb{N}$ に対し $\varphi(e)(x,k) = (x,k)$ である.
- $f,g \in M$ と $(x,k) \in S \times \mathbb{N}$ を任意にとる. f が単位元かどうかで場合分け.
  - $f = e$ の場合:

$$\begin{aligned} & \varphi(f \cdot_M g)(x,k) \\ = \ & \varphi(g)(x,k) \\ = \ & \varphi(f)(\varphi(g)(x,k)) \\ = \ & (\varphi(f)\circ\varphi(g))(x,k) \end{aligned}$$

    が成り立つ.
  - $f \neq e$ の場合:

$$\begin{aligned} & \varphi(f \cdot_M g)(x,k) \\ = \ & \varphi(f)(x,k) \\ = \ & (kf,k) \\ = \ & \varphi(f)(\varphi(g)(x,k)) \\ = \ & (\varphi(f)\circ\varphi(g))(x,k) \end{aligned}$$

    が成り立つ.

  つまり $\varphi(f\cdot g) = \varphi(f)\circ\varphi(g)$ である.

よって φ はモノイド準同型である. □

注意 8.6.2.

自然数倍を計算するときの計算量については 8.5 節の注意 8.5.3 と同様の問題がある.

定理 8.6.1 の遅延伝播構造を用いるには, 半群 S の要素 $x \in S$ と自然数 $k \in \mathbb{N}$ に対し自然数倍 kx を $O(1)$ で計算できる必要がある. そうでない場合は繰り返し二乗法を用いることになり, クエリの計算量は $O((\log n)^2)$ に悪化する. ◇

注意 8.6.3.

注意 8.6.2 の制約は, 完全二分木の上に構築された遅延伝播セグメント木であれば, 演算の実装とセグメント木の実装の両方を工夫することで回避可能ではある. まず, 取得クエリの実行に再帰的なもの (そうでない場合は専用の前処理) を用い, それぞれの頂点で遅延している状態の作用素を子に伝播させて親の作用素を単位元で置き換えるようにする. これにより自然数倍の計算はちょうど 2 冪であるような $k = 2^i$ に対する kb のみに限られるようになる. そして, 自然数倍の繰り返し二乗法による計算をメモ化などの形でキャッシュするようにする. すると $b, 2b, 4b, \ldots, 2^{\lfloor \log n \rfloor} b$ をまとめて $O(\log n)$ で計算可能となり, 計算すべき自然数倍はこれですべてなので, 全体で $O(\log n)$ となる. ◇

8.7 通常のセグメント木は特殊な形の遅延伝播セグメント木である

遅延伝播セグメント木で用いる半群やモノイドとして自明なものを選択すれば, 通常のセグメント木や双対セグメント木と同等の能力の遅延伝播セグメント木が得られる.

より正確には次が言える. 与えられた半群 S に対し定理 8.6.1 によって得られる遅延伝播構造 $(M, S \times \mathbb{N}, \varphi)$ を考えれば, S を乗せた通常のセグメント木と同じクエリのみを処理できる遅延伝播セグメント木が得られる. ただし, 点更新クエリには長さ 1 の区間についての区間更新クエリを使う. また, 取得クエリについてはその結果の半群の第 1 要素を用いる.

注意 8.7.1.

ただし, 実用的には次の命題 8.7.2 のような (M, S, φ) を使えば十分である. これは長さ 2 以上の区間に対する更新クエリは扱えないが, そのような更新クエリは与えられないためである. ただし, 遅延伝播構造であるためには長さ 2 以上の区間に対する更新クエリを扱える必要があるため, この差によってこの (M, S, φ) は遅延伝播構造にはならない. ◇

命題 8.7.2.

$(S, \cdot_S)$ を半群とする. 集合としての S についての左零半群 $(S, \lambda xy.x)$ に単位元を付与してできるモノイドを $M = (S \cup \{ e \}, \cdot_M, e)$ とする. 関数 $\varphi \colon M \to S^S$ を $\varphi(f)(x) = f \cdot_M x$ で定義する. このとき次のふたつが言える.

(1.) $\varphi\colon M \to \mathrm{End}(S)$ であるとは限らない．また 3 つ組 (M, S, φ) は遅延伝播構造とは限らない．

(2.) φ はモノイド M から S 上の関数全体のなすモノイド S^S へのモノイド準同型ではある．

◇

証明

(1.) **について**　$S = (\mathbb{N}, +)$ の場合を例に見る．$f = 1 \in M$ かつ $x = y = 0 \in S$ とする．$\varphi(f)(x + y) = f = 1$ だが $\varphi(f)(x) + \varphi(f)(y) = f + f = 2$ である．よって $\varphi\colon M \to \mathrm{End}(S)$ ではない．また (M, S, φ) は遅延伝播構造ではない．

(2.) **について**

- 任意の $x \in S$ に対し $\varphi(e)(x) = e \cdot_M x = x$ である．よって $\varphi(e)$ は恒等関数であり, これは S^S の単位元である．
- $f, g \in M$ と $x \in S$ を任意にとる．$f = e$ ならば $\varphi(e \cdot_M g)(x) = \varphi(g)(x) = \varphi(e)(\varphi(g)(x))$ である．$f \neq g$ ならば $\varphi(f \cdot_M g)(x) = \varphi(f)(x) = f = \varphi(f)(\varphi(g)(x))$ である．よって $\varphi(f \cdot_M g) = \varphi(f) \circ \varphi(g)$ である．

よって φ はモノイド準同型である．

□

8.8　双対セグメント木は特殊な形の遅延伝播セグメント木である

遅延伝播セグメント木で用いる半群として自明なものを選択すれば, 双対セグメント木と同等の能力の遅延伝播セグメント木が得られる．

より正確には次が言える．与えられたモノイド M に対し次の定理 8.8.1 によって得られる遅延伝播構造 (M, S, φ) を考えれば, M を乗せた双対双対セグメント木と同じクエリのみを処理できる遅延伝播セグメント木が得られる．ただし, 点取得クエリには長さ 1 の区間についての区間取得クエリを使う．

定理 8.8.1.

$(M, \cdot_M, e)$ をモノイドとする．集合としての M についての左零半群を $S = (M, \lambda xy.x)$ とする．関数 $\varphi\colon M \to \mathrm{End}(S)$ を $\varphi(f)(x) = f \cdot_M x$ で定義する．このとき (M, S, φ) は

遅延伝播構造である. ◇

証明

半群 S の演算を $\cdot_S$ と書くこととする.

まず φ が well-defined であること, つまり $\varphi(f)\colon S \to S$ が半群準同型であることを示す. 任意の $f \in M$ と任意の $x, y \in S$ に対し $\varphi(f)(x \cdot_S y) = \varphi(f)(x) = \varphi(f)(x) \cdot_S \varphi(f)(y)$ である. よって $\varphi(f)$ は半群準同型であり, φ は well-defined である.

あとは φ がモノイド準同型であることを示せばよい.

- $\varphi(e)(x) = e \cdot_M x = x$ である. よって $\varphi(e)$ は恒等関数であり, これは $\mathrm{End}(S)$ の単位元である.
- 任意の $f, g \in M$ と任意の $x \in S$ に対し

$$\begin{aligned} & \varphi(f \cdot_M g)(x) \\ = \ & f \cdot_M g \cdot_M x \\ = \ & f \cdot_M \varphi(g)(x) \\ = \ & \varphi(f)(\varphi(g)(x)) \\ = \ & (\varphi(f) \circ \varphi(g))(x) \end{aligned}$$

である.

よって φ はモノイド準同型である. □

注意 8.8.2.

この定理 8.8.1 では半群 S として左零半群を用いているが, S の演算はあまり重要ではない. これは右零半群などでもよい. ◇

8.9 遅延伝播構造どうしの構造としての等しさ

補題 8.9.1.

M, N をモノイドとし $\varphi\colon M \to N$ をモノイド準同型とする. M 上の関係 $\sim$ を $x \sim y \leftrightarrow \varphi(x) \sim \varphi(y)$ で定めると, これは同値関係をなす. さらに, 任意の $x, y, x', y' \in M$ に対し $x \sim x'$ かつ $y \sim y'$ ならば $x \cdot y \sim x' \cdot y'$ を満たす. ◇

証明

- $\sim$ が同値関係であることは明らか.

- $x, y, x', y' \in M$ を任意にとり $x \sim x'$ かつ $y \sim y'$ を仮定する. $\varphi(x) = \varphi(x')$ かつ $\varphi(y) = \varphi(y')$ であるので $\varphi(x \cdot y) = \varphi(x) \cdot \varphi(y) = \varphi(x') \cdot \varphi(y') = \varphi(x' \cdot y')$ である. よって $x \cdot y \sim x' \cdot y'$ である.

□

記法 8.9.2.

モノイド準同型 $\varphi\colon M \to N$ があるとき, モノイド M を $x \sim y \leftrightarrow \varphi(x) = \varphi(y)$ という同値関係で割った商モノイド $M/\sim$ のことを単に M/φ と書く. また, φ を M から $M/\sim$ へ持ち上げてできるモノイド準同型 $\varphi'\colon M/\sim \to N$ を $\varphi'([x]) = \varphi(x)$ と定義できるが, この φ' を単に φ と書く. ◇

定義 8.9.3 (正規な遅延伝播構造)**.**

(M, S, φ) を遅延伝播構造とする. M を φ で割った商モノイド M/φ と元のモノイド M が同型であるとき, 遅延伝播構造 (M, S, φ) は**正規**であると呼ぶ. ◇

定義 8.9.4 (正規化)**.**

(M, S, φ) を遅延伝播構造とする. このとき $(M/\varphi, S, \varphi)$ は正規な遅延伝播構造である. この正規な遅延伝播構造 $(M/\varphi, S, \varphi)$ のことを, (M, S, φ) を**正規化**して得られる遅延伝播構造と呼ぶ. ◇

注意 8.9.5.

遅延伝播構造 (M, S, φ) において M の要素 $f \in M$ は, モノイド準同型 $\varphi(f) \in \mathrm{End}(S)$ を表現するために用いられている. 一方で要素 f 自体はあまり重要でなく $\varphi(f)$ を表現する手段でしかない. このとき「φ で写した先の等しさ $\sim$ で割る」ことは「実装方法の差を無視する」ということである. よって, セグメント木で処理できるクエリの数学的構造をより正確に捉えるたものとしては, 正規な遅延伝播構造を考えるのが適切である.

以下では基本的に正規な遅延伝播構造のみを考えていく. $M/\sim$ と M をあまり区別しないことがある. 特に, $\varphi\colon M \to \mathrm{End}(S)$ を $\varphi\colon (M/\sim) \to \mathrm{End}(S)$ でもあるかのように書くことがある. ◇

注意 8.9.6.

「加群ならば遅延伝播構造である」は言えた (定理 8.4.2) にも関わらず「加群ならば正規な遅延伝播構造である」は言えない. ◇

命題 8.9.7.

(M, S, φ) を遅延伝播構造とする. 次のふたつの命題は同値である.

- (M, S, φ) は正規
- φ は単射

◇

証明

それぞれの定義より明らか. ただし, φ が単射であることの定義は $\varphi(f) = \varphi(g) \to f = g$ であることであった. □

定義 8.9.8 (正規な遅延伝播構造の準同型)**.**

正規な遅延伝播構造 (M_1, S_1, φ_1) と (M_2, S_2, φ_2) があるとする. 関数 $\psi_M \colon M_1 \to M_2$ と半群準同型 $\psi_S \colon S_1 \to S_2$ が, 任意の $f \in M_1$, $x \in S_1$ に対し

$$\psi_S(\varphi_1(f)(x)) = \varphi_2(\psi_M(f))\,(\psi_S(x))$$

を満たすとき, これらの対 (ψ_M, ψ_S) を (M_1, S_1, φ_1) から (M_2, S_2, φ_2) への**準同型**という.

◇

定義 8.9.9 (正規な遅延伝播構造の同型)**.**

正規な遅延伝播構造 (M_1, S_1, φ_1) と (M_2, S_2, φ_2) があるとする. 関数 $\psi_M \colon M_1 \to M_2$ と半群準同型 $\psi_S \colon S_1 \to S_2$ が以下のみっつをすべて満たすとき, これらの対 (ψ_M, ψ_S) を (M_1, S_1, φ_1) と (M_2, S_2, φ_2) の間の**同型**という.

- (ψ_M, ψ_S) が (M_1, S_1, φ_1) から (M_2, S_2, φ_2) への準同型
- ψ_M が全単射
- ψ_S が全単射

また, 同型が存在するとき $(M_1, S_1, \varphi_1) \simeq (M_2, S_2, \varphi_2)$ と表す. ◇

注意 8.9.10.

正規とは限らない一般の遅延伝播構造についての準同型はここでは定義しない. 一般の遅延伝播構造についての準同型を定義しても以降の議論には役立たないためである[19].

◇

[19] なにが同型とされるかが異なるために, 一般の「遅延伝播構造」と「正規な遅延伝播構造」は類似するが異なるふたつの代数構造であると認識した方がよいかもしれない.

注意 8.9.11.

この準同型の定義は「それぞれを遅延伝播セグメント木の上に乗せたとき, 一方へのクエリをもう一方へのクエリに帰着させられる」を意図している. また, この同型の定義は「それぞれを遅延伝播セグメント木の上に乗せたとき, 外部からクエリを発行してその結果を観察する限りにおいて区別できない」を意図している.

このことは次の命題 8.9.12 によって確認される. 命題 8.9.12 の (2.) は「どのような初期値 $(x_0, x_1, \ldots, x_{w-1})$ とどのような更新クエリ $(f_{0,0}, f_{0,1}, \ldots, f_{0,h_0-1})$, $(f_{1,0}, f_{1,1}, \ldots, f_{1,h_1-1})$, $\ldots$, $(f_{w-1,0}, f_{w-1,1}, \ldots, f_{w-1,h_{w-1}-1})$ を考えても, $(M_1, S_1, \star_1)$ の側での取得クエリの結果 $\psi_S\left(\prod_{i=0}^{w-1}\left(\left(\prod_{j=0}^{h_i-1} f_{i,j}\right) \star_1 x_i\right)\right)$ と $(M_2, S_2, \star_2)$ の側での取得クエリの結果 $\prod_{i=0}^{w-1}\left(\left(\prod_{j=0}^{h_i-1} \psi_M(f_{i,j})\right) \star_2 \psi_S(x_i)\right)$ が等しい」ということを言っている. ◇

命題 8.9.12.

$(M_1, S_1, \star_1)$ と $(M_2, S_2, \star_2)$ を正規な遅延伝播構造とする. 関数 $\psi_M \colon M_1 \to M_2$ と関数 $\psi_S \colon S_1 \to S_2$ に対し, 以下のふたつは同値である.

(1.) (ψ_M, ψ_S) は正規な遅延伝播構造の準同型である.

(2.) 任意の自然数 $w \geq 0$, 任意の自然数列 $(h_0, h_1, \ldots, h_{w-1})$, 任意の M_1 の要素の列たち $(f_{0,0}, f_{0,1}, \ldots, f_{0,h_0-1})$, $(f_{1,0}, f_{1,1}, \ldots, f_{1,h_1-1})$, $\ldots$, $(f_{w-1,0}, f_{w-1,1}, \ldots, f_{w-1,h_{w-1}-1})$, および任意の S_1 の要素の列 $(x_0, x_1, \ldots, x_{w-1})$ について,

$$\psi_S\left(\prod_{i=0}^{w-1}\left(\left(\prod_{j=0}^{h_i-1} f_{i,j}\right) \star_1 x_i\right)\right) = \prod_{i=0}^{w-1}\left(\left(\prod_{j=0}^{h_i-1} \psi_M(f_{i,j})\right) \star_2 \psi_S(x_i)\right)$$

が成り立つ.

◇

証明

(1.) $\Rightarrow$ (2.) **について** まず φ_1 がモノイド準同型であることと ψ_S が半群準同型である

ことを使って, 以下のように変形する.

$$
\begin{aligned}
& \psi_S \left(\prod_{i=0}^{w-1} \left(\left(\prod_{j=0}^{h_i-1} f_{i,j} \right) \star_1 x_i \right) \right) \\
= \ & \psi_S \left(\prod_{i=0}^{w-1} \left(\varphi_1 \left(\prod_{j=0}^{h_i-1} f_{i,j} \right) (x_i) \right) \right) \\
= \ & \psi_S \left(\prod_{i=0}^{w-1} \left(\left(\prod_{j=0}^{h_i-1} \varphi_1 \left(f_{i,j} \right) \right) (x_i) \right) \right) \\
= \ & \prod_{i=0}^{w-1} \psi_S \left(\left(\prod_{j=0}^{h_i-1} \varphi_1 \left(f_{i,j} \right) \right) (x_i) \right)
\end{aligned}
$$

さらに $\varphi_1(f_{i,j})$ が半群準同型であることと (ψ_M, ψ_S) が正規な遅延伝播構造の準

同型であることを使って, 任意の i について,

$$
\begin{aligned}
& \psi_S\left(\left(\prod_{j=0}^{h_i-1}\varphi_1\left(f_{i,j}\right)\right)\left(x_i\right)\right) \\
= \ & \psi_S\left(\left(\varphi_1\left(f_{i,0}\right)\circ\varphi_1\left(f_{i,1}\right)\circ\cdots\circ\varphi_1\left(f_{i,h_i-1}\right)\right)\left(x_i\right)\right) \\
= \ & \psi_S\left(\varphi_1\left(f_{i,0}\right)\left(\varphi_1\left(f_{i,1}\right)\left(\ldots\varphi_1\left(f_{i,h_i-1}\right)\left(x_i\right)\ldots\right)\right)\right) \\
= \ & \varphi_2\Big(\psi_M\left(f_{i,0}\right)\Big)\left(\varphi_1\left(f_{i,1}\right)\left(\ldots\varphi_1\left(f_{i,h_i-1}\right)\left(x_i\right)\ldots\right)\right) \\
= \ & \varphi_2\Big(\psi_M\left(f_{i,0}\right)\Big)\left(\varphi_2\Big(\psi_M\left(f_{i,1}\right)\Big)\left(\ldots\varphi_1\left(f_{i,h_i-1}\right)\left(x_i\right)\ldots\right)\right) \\
& \vdots \\
= \ & \varphi_2\Big(\psi_M\left(f_{i,0}\right)\Big)\left(\varphi_2\Big(\psi_M\left(f_{i,1}\right)\Big)\left(\ldots\varphi_2\Big(\psi_M\left(f_{i,h_i-1}\right)\Big)\Big(\psi_S\left(x_i\right)\Big)\ldots\right)\right) \\
= \ & \left(\varphi_2\Big(\psi_M\left(f_{i,0}\right)\Big)\circ\varphi_2\Big(\psi_M\left(f_{i,1}\right)\Big)\circ\cdots\circ\varphi_2\Big(\psi_M\left(f_{i,h_i-1}\right)\Big)\right)\Big(\psi_S\left(x_i\right)\Big) \\
= \ & \left(\prod_{j=0}^{h_i-1}\varphi_2\Big(\psi_M\left(f_{i,j}\right)\Big)\right)\Big(\psi_S\left(x_i\right)\Big)
\end{aligned}
$$

が成り立つ. このことと φ_2 がモノイド準同型であることを使って, 続きから,

$$
\begin{aligned}
& \prod_{i=0}^{w-1}\psi_S\left(\left(\prod_{j=0}^{h_i-1}\varphi_1\left(f_{i,j}\right)\right)\left(x_i\right)\right) \\
= \ & \prod_{i=0}^{w-1}\left(\left(\prod_{j=0}^{h_i-1}\varphi_2\Big(\psi_M\left(f_{i,j}\right)\Big)\right)\Big(\psi_S\left(x_i\right)\Big)\right) \\
= \ & \prod_{i=0}^{w-1}\left(\varphi_2\left(\prod_{j=0}^{h_i-1}\psi_M\left(f_{i,j}\right)\right)\Big(\psi_S\left(x_i\right)\Big)\right) \\
= \ & \prod_{i=0}^{w-1}\left(\left(\prod_{j=0}^{h_i-1}\psi_M(f_{i,j})\right)\star_2\psi_S(x_i)\right)
\end{aligned}
$$

と変形できる. よって (2.) が成り立つ.

(2.) ⇒ (1.) **について**　モノイド M_1, M_2 の単位元を e_1, e_2 とする.

- $w = 2$ かつ $h_0 = h_1 = 0$ の場合を考える. (2.) の式は $\psi_S((e_1 \star_1 x_0) \cdot (e_1 \star_1 x_1)) = \psi_S(e_2 \star_2 x_0) \cdot \psi_S(e_2 \star_2 x_1)$ となる. これは $\psi_S(x_0 \cdot x_1) = \psi_S(x_0) \cdot \psi_S(x_1)$ と等しく, ψ_S が半群準同型であることを意味する.
- $w = 1$ かつ $h_0 = 1$ の場合を考える. (2.) の式は $\psi_S(f_{0,0} \star_1 x_0) = \psi_M(f_{0,0}) \star_2 \psi_S(x_0)$ となる.

よって (ψ_M, ψ_S) は遅延伝播構造の準同型である.

□

注意 8.9.13.

準同型の定義において「関数 $\psi_M \colon M_1 \to M_2$ がモノイド準同型であること」は要求されていない. これが要求されないのは, クエリを使うのみでは遅延伝播セグメント木の内部にある M_1 の要素の様子を直接観察することができないことに由来する. 例として, モノイド $M_1 = (\mathbb{N}, +, 0)$ および $M_2 = (\mathbb{N}, \max, 0)$ を考えよう. もしこれらが $a \star_1 x = a + x$ および $b \star_2 x = \max(b, x)$ という作用と共に用いられるならば, 自由積 $M_1 + M_2$ とある意味で同じと見なせるモノイド $M = (M_1 \times M_2, \cdot, (e_1, e_2))$ が構成できるのだった. ここで言う「ある意味で同じ」ということこそが, 遅延伝播構造の同型性で捉えたい等しさである. 一方で, モノイド $M_1 + M_2$ とモノイド M は明らかに同型でない. よって, そのような同型性を要求することはできないことが分かる.

ただし ψ_M もある程度の「モノイド準同型っぽさ」は備えている. このことは次の命題 8.9.14 として表現される. ◇

命題 8.9.14.

対 (ψ_M, ψ_S) を正規な遅延伝播構造 (M_1, S_1, φ_1) から (M_2, S_2, φ_2) への準同型とする. M_1, M_2 の単位元をそれぞれ e_1, e_2 とする. $f, g \in M_2$ に対する同値関係 $\sim \subseteq M_2 \times M_2$ を $f \sim g \leftrightarrow \forall x \in S_1.\ \varphi_2(f)(\psi_S(x)) = \varphi_2(g)(\psi_S(x))$ で定める. このとき以下のふたつが成り立つ.

(1.) $\psi_M(e_1) \sim e_2$
(2.) 任意の $f, g \in M_1$ に対し $\psi_M(f \cdot g) \sim \psi_M(f) \cdot \psi_M(g)$

◇

証明

(1.) **について** $x \in M_1$ を任意にとる．正規な遅延伝播構造の準同型の定義および $\varphi_1(e_1), \varphi_2(e_2)$ が恒等関数であることを使って $\varphi_2(\psi_M(e_1))(\psi_S(x)) = \psi_S(\varphi_1(e_1)(x)) = \psi_S(x) = \varphi_2(e_2)(\psi_S(x))$ が成り立つ．これは $\psi_M(e_1) \sim e_2$ である．

(2.) **について** $f, g \in M_1$ と $x \in M_1$ を任意にとる．正規な遅延伝播構造の準同型の定義および φ_1, φ_2 が $\mathrm{End}(S_1), \mathrm{End}(S_2)$ への準同型であることを使って

$$
\begin{aligned}
& \varphi_2\Big(\psi_M(f \cdot g)\Big)(\psi_S(x)) \\
=\ & \psi_S\Big(\varphi_1(f \cdot g)(x)\Big) \\
=\ & \psi_S\Big(\Big(\varphi_1(f) \circ \varphi_1(g)\Big)(x)\Big) \\
=\ & \psi_S\Big(\varphi_1(f)\Big(\varphi_1(g)(x)\Big)\Big) \\
=\ & \varphi_2(\psi_M(f))\Big(\psi_S(\varphi_1(g)(x))\Big) \\
=\ & \varphi_2(\psi_M(f))\Big(\varphi_2(\psi_M(g)(\psi_S(x)))\Big) \\
=\ & \Big(\varphi_2(\psi_M(f)) \circ \varphi_2(\psi_M(g))\Big)(\psi_S(x)) \\
=\ & \varphi_2\Big(\psi_M(f) \cdot \psi_M(g)\Big)(\psi_S(x))
\end{aligned}
$$

が成り立つ．これは $\psi_M(f \cdot g) \sim \psi_M(f) \cdot \psi_M(g)$ である．

□

8.10 複数種類のクエリの併用

6.4 節と 7.3 節ではそれぞれ通常のセグメント木と双対セグメント木における複数種類のクエリの併用について確認した．この節では遅延伝播セグメント木のクエリの併用について確認する．遅延伝播セグメント木のクエリには「取得クエリ」と「更新クエリ」の 2 種類があるためそれぞれ見ていく．

まず取得クエリについて．これは通常のセグメント木の場合と同様である．ある集合 X の要素の列に対するひとつの更新クエリとその集合 X 上の異なるふたつの演算により定まるふたつの取得クエリをすべて同時に処理したいとき，「遅延伝播セグメント木をふたつ用意する」「直積半群を用いる」というふたつの選択肢がある．前者は 6.4 節で紹介し

たものとまったく同様である. 後者もほぼ同様であるが, 対応する命題は新しいものとなるのでこれを紹介しておく.

命題 8.10.1.

(M, S_1, φ_1) と (M, S_2, φ_2) を遅延伝播構造とする. 直積半群 $S_1 \times S_2$ についての関数 $\varphi\colon M \to \mathrm{End}(S_1 \times S_2)$ を

$$\varphi(f)(x_1, x_2) = \Big(\varphi_1(f)(x_1), \varphi_2(f)(x_2)\Big)$$

で定義すると $(M, S_1 \times S_2, \varphi)$ は遅延伝播構造である. ◇

証明

まず φ が well-defined であること, つまり $\varphi(f)\colon S_1 \times S_2 \to S_1 \times S_2$ が半群準同型であることを示す. 任意の $f \in M$ と任意の $(x_1, x_2), (y_1, y_2) \in S_1 \times S_2$ に対し

$$\begin{aligned}
& \varphi(f)\Big((x_1, x_2) \cdot (y_1, y_2)\Big) \\
= \ & \varphi(f)(x_1 \cdot y_1, x_2 \cdot y_2) \\
= \ & \Big(\varphi_1(f)(x_1 \cdot y_1), \varphi_2(f)(x_2 \cdot y_2)\Big) \\
= \ & \Big(\varphi_1(f)(x_1) \cdot \varphi_1(f)(y_1), \varphi_2(f)(x_2) \cdot \varphi_2(f)(y_2)\Big) \\
= \ & \Big(\varphi_1(f)(x_1), \varphi_2(f)(x_2)) \cdot (\varphi_1(f)(y_1), \varphi_2(f)(y_2)\Big) \\
= \ & \varphi_1(f)(x_1, x_2) \cdot \varphi_1(f)(y_1, y_2)
\end{aligned}$$

である. よって $\varphi(f)$ は半群準同型であり, φ は well-defined である.

あとは φ がモノイド準同型であることを示せばよい.

- $e \in M$ を M の単位元とする. 任意の $(x_1, x_2) \in S_1 \times S_2$ に対し $\varphi(e)(x_1, x_2) = (\varphi_1(e)(x_1), \varphi_2(e)(x_2)) = (x_1, x_2)$ である. よって $\varphi(e)$ は恒等関数であり, つまり $\mathrm{End}(S_1 \times S_2)$ の単位元である.

- 任意の $f, g \in M$ と任意の $(x, y) \in S_1 \times S_2$ に対し

$$\begin{aligned}
& \varphi(f \cdot g)(x, y) \\
= \ & \Big(\varphi_1(f \cdot g)(x), \varphi_2(f \cdot g)(y)\Big) \\
= \ & \Big(\varphi_1(f)\big(\varphi_1(g)(x)\big), \varphi_2(f)\big(\varphi_2(g)(y)\big)\Big) \\
= \ & \varphi(f)\Big(\varphi_2(g)(x), \varphi_2(g)(y)\Big) \\
= \ & \varphi(f)\Big(\varphi(g)(x, y)\Big) \\
= \ & \Big(\varphi(f) \circ \varphi(g)\Big)(x, y)
\end{aligned}$$

  である.

よって φ はモノイド準同型である. □

次に更新クエリについて. こちらは双対セグメント木の場合と同様である. つまり, ある集合 X の要素の列に対するふたつの更新クエリとひとつの取得クエリをすべて同時に処理したいときに「自由積を用いる」ことができる. これについても遅延伝播構造についての命題の形で示しておく.

命題 8.10.2.

M_1, M_2 を $M_1 \cap M_2 = \emptyset$ を満たすモノイドとし (M_1, S, φ_1) と (M_2, S, φ_2) を遅延伝播構造とする. 和集合 $M_1 \cup M_2$ からの関数 $\varphi' \colon (M_1 \cup M_2) \to \mathrm{End}(S)$ を

$$\varphi'(f)(x) = \begin{cases} \varphi_1(f)(x) & (f \in M_1) \\ \varphi_2(f)(x) & (f \in M_2) \end{cases}$$

で定義する. これを用いて, 自由積 $M_1 + M_2$ からの関数 $\varphi \colon (M_1 + M_2) \times S \to S$ を

$$\varphi\Big((f_0, f_1, \ldots, f_{k-1})\Big)(x) = \varphi'(f_0)\Big(\varphi'(f_1)\Big(\ldots\Big(\varphi'(f_{k-1})(x)\Big)\ldots\Big)\Big)$$

で定義する. $(M_1 + M_2, S, \varphi)$ は遅延伝播構造である. ◇

証明

まず関数 φ' について確認する. φ' は well-defined である. これにはその終域が $\mathrm{End}(S)$ であることを言う必要があるが, これは φ_1, φ_2 の終域が $\mathrm{End}(S)$ であることから成り立つ. φ' がモノイド準同型であるとは言えない. その始域がモノイドとは限らないためである. しかし, φ_1, φ_2 がモノイド準同型であることから, 任意の $f \in M_1 \cup M_2$ と $x, y \in S$ に対し $\varphi'(f)(x \cdot y) = \varphi'(f)(x) \cdot \varphi'(f)(y)$ が成り立つ.

次に関数 φ の well-definedness を確認する必要がある. φ の定義が $M_1 + M_2$ の要素そのものに対してでなく代表元に対しての定義であるためである. 完全な証明については省略するが, 基本的には, $i = 1$ と $i = 2$ のどちらについても $f, g \in M_i$ ならば $\varphi'(f \cdot g)(x) = \varphi'(f)(\varphi'(g)(x))$ であることから言える.

後は φ が自由積モノイド $M_1 + M_2$ から自己準同型モノイド $\mathrm{End}(S)$ へのモノイド準同型であることを確認すればよい.

- 空列 $\epsilon \in M_1 + M_2$ は自由積モノイドの単位元であった. $\varphi(\epsilon)(x) = x$ である. よって $\varphi(\epsilon)$ は恒等関数であり, つまり $\mathrm{End}(S)$ の単位元である.
- さて, $[(f_0, f_1, \ldots, f_{n-1})], [(g_0, g_1, \ldots, g_{m-1})] \in M_1 + M_2$ と $x \in S$ を任意にとる. このとき以下のように計算できる.

$$
\begin{aligned}
& \varphi\Big([(f_0, f_1, \ldots, f_{n-1})] \cdot [(g_0, g_1, \ldots, g_{m-1})]\Big)(x) \\
=\ & \varphi\Big([(f_0, f_1, \ldots, f_{n-1}, g_0, g_1, \ldots, g_{m-1})]\Big)(x) \\
=\ & \varphi'(f_0)\Big(\varphi'(f_1)\Big(\ldots\Big(\varphi'(f_{n-1})\Big(\varphi'(g_0)\Big(\varphi'(g_1)\Big(\ldots\Big(\varphi'(g_{m-1})(x)\Big)\ldots\Big)\Big)\Big)\ldots\Big)\Big) \\
=\ & \varphi([(f_0, f_1, \ldots, f_{n-1})])\Big(\varphi'(g_0)\Big(\varphi'(g_1)\Big(\ldots\Big(\varphi'(g_{m-1})(x)\Big)\ldots\Big)\Big)\Big) \\
=\ & \varphi([(f_0, f_1, \ldots, f_{n-1})])\Big(\varphi([(g_0, g_1, \ldots, g_{m-1})])(x)\Big) \\
=\ & \Big(\varphi([(f_0, f_1, \ldots, f_{n-1})]) \circ \varphi([(g_0, g_1, \ldots, g_{m-1})])\Big)(x)
\end{aligned}
$$

よって φ はモノイド準同型であり $(M_1 + M_2, S, \varphi)$ は遅延伝播構造である. □

8.11 自由積の小さな表現

7.3 節の後半では, 双対セグメント木における更新クエリの併用を実際に利用するにあたって, 自由積の計算量的に扱いやすい表現を探す必要があることを見た. 自由積の作用としての用途 (つまり, 遅延伝播セグメント木という形での利用) を前提とすれば, そのような小さな表現の存在には使いやすい十分条件が存在する. 本節ではこれを紹介する.

補題 8.11.1.

$(M_1, \cdot_1, e_1), (M_2, \cdot_2, e_2)$ をモノイドとし S を半群とする. $(M_1, S, \varphi_1), (M_2, S, \varphi_2)$ を遅延伝播構造とする. 任意の $f \in M_1$ と $g \in M_2$ に対しある $f' \in M_1$ と $g' \in M_2$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ を満たすとする. 直積集合 $M_1 \times M_2$ からの関数

$\varphi_{\bowtie}\colon M_1 \times M_2 \to \mathrm{End}(S)$ を

$$\varphi_{\bowtie}(f,g) = \varphi_1(f) \circ \varphi_2(g)$$

で定義する. この関数で直積集合 $M_1 \times M_2$ を割って得られる商集合 $(M_1 \times M_2)/\varphi_{\bowtie}$ を考える. このとき, 以下のよっつが成り立つ.

(1.) 任意の $[(f_1,f_2)],[(g_1,g_2)] \in (M_1 \times M_2)/\varphi_{\bowtie}$ に対し, ある $[(h_1,h_2)] \in (M_1 \times M_2)/\varphi_{\bowtie}$ が一意に存在して, 任意の $(g_1',f_2') \in M_1 \times M_2$ に対し $\varphi_2(f_2)\circ\varphi_1(g_1) = \varphi_1(g_1')\circ\varphi_2(f_2')$ ならば $[(h_1,h_2)] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)]$ が成り立つ.

(2.) 演算 $\odot\colon (M_1\times M_2)/\varphi_{\bowtie}\times(M_1\times M_2)/\varphi_{\bowtie} \to (M_1\times M_2)/\varphi_{\bowtie}$ を, $\varphi_2(f_2)\circ\varphi_1(g_1) = \varphi_1(g_1')\circ\varphi_2(f_2')$ な $(g_1',f_2') \in M_1 \times M_2$ を使って

$$[(f_1,f_2)] \odot [(g_1,g_2)] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)]$$

で定義する. これは well-defined である.

(3.) 任意の $f,g,h \in M_1 \times M_2$ に対し $[f] \odot [g] = [h]$ であることと $\varphi_{\bowtie}(f)\circ\varphi_{\bowtie}(g) = \varphi_{\bowtie}(h)$ であることは同値である.

(4.) $((M_1 \times M_2)/\varphi_{\bowtie}, \odot, [(e_1,e_2)])$ はモノイドである.

◇

証明

(1.) から (4.) を順番に示す. 商集合 $(M_1 \times M_2)/\varphi_{\bowtie}$ の定義から, 任意の $f,g \in M_1 \times M_2$ に対し, $[f] = [g]$ であることと $\varphi_{\bowtie}(f) = \varphi_{\bowtie}(g)$ であることは同値である, ということを注意しておく.

(1.) について $[(f_1,f_2)],[(g_1,g_2)] \in (M_1 \times M_2)/\varphi_{\bowtie}$ を任意にとる. 条件を満たす $[(h_1,h_2)] \in (M_1 \times M_2)/\varphi_{\bowtie}$ の存在性と一意性をそれぞれ示す.

- 存在性について. 補題の仮定により $\varphi_2(f_2)\circ\varphi_1(g_1) = \varphi_1(g_1')\circ\varphi_2(f_2')$ を満たす $(g_1',f_2') \in M_1 \times M_2$ がとれる. $h_1 = f_1 \cdot_1 g_1'$ とおき $h_2 = f_2' \cdot_2 g_2$ とおく. $\varphi_{\bowtie}(h_1,h_2) = \varphi_1(f_1)\circ\varphi_1(g_1')\circ\varphi_2(f_2')\circ\varphi_2(g_2) = \varphi_1(f_1)\circ\varphi_2(f_2)\circ\varphi_1(g_1)\circ\varphi_2(g_2)$ である. さて, $(g_1'',f_2'') \in M_1 \times M_2$ を任意にとり, $\varphi_2(f_2)\circ\varphi_1(g_1) = \varphi_1(g_1'')\circ\varphi_2(f_2'')$ を満たすと仮定する. $\varphi_{\bowtie}(f_1 \cdot_1 g_2'', f_1'' \cdot_2 g_2) = \varphi_1(f_1)\circ\varphi_1(g_1')\circ\varphi_2(f_2')\circ\varphi_2(g_2) = \varphi_1(f_1)\circ\varphi_2(f_2)\circ\varphi_1(g_1)\circ\varphi_2(g_2)$ である. よって $[(h_1,h_2)] = [(f_1 \cdot_1 g_2'', f_1'' \cdot_2 g_2)]$ が成り立つ. つまり, 条件を満たす $[(h_1,h_2)]$ が存在する.

- 一意性について. $[(h_1, h_2)] \in (M_1 \times M_2)/\varphi_{\bowtie}$ をとり, 任意の $(g_1', f_2') \in M_1 \times M_2$ に対し $\varphi_2(f_2) \circ \varphi_1(g_1) = \varphi_1(g_1') \circ \varphi_2(f_2')$ ならば $[(h_1, h_2)] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)]$ と仮定する. $[(h_1', h_2')] \in (M_1 \times M_2)/\varphi_{\bowtie}$ をとり, これについても同様の条件を仮定する. 補題の仮定により $\varphi_2(f_2) \circ \varphi_1(g_1) = \varphi_1(g_1') \circ \varphi_2(f_2')$ を満たす $(g_1', f_2') \in M_1 \times M_2$ がとれる. このとき $[(h_1, h_2)] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)]$ かつ $[(h_1', h_2')] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)]$ である. よって $[(h_1, h_2)] = [(h_1', h_2')]$ が成り立つ. つまり, 条件を満たす $[(h_1, h_2)]$ は一意である.

(2.) **について** (1.) から明らか.

(3.) **について** $f = (f_1, f_2) \in M_1 \times M_2$ と $g = (g_1, g_2) \in M_1 \times M_2$ を任意にとる. $(g_1', f_2') \in M_1 \times M_2$ を $\varphi_2(f_2) \circ \varphi_1(g_1) = \varphi_1(g_1') \circ \varphi_2(f_2')$ を満たすものとする. $\odot$ の定義より $[f] \odot [g] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)]$ である. また, 明らかに $\varphi_1(f_1) \circ \varphi_2(f_2) \circ \varphi_1(g_1) \circ \varphi_2(g_2) = \varphi_1(f_1) \circ \varphi_1(g_1') \circ \varphi_2(f_2') \circ \varphi_2(g_2)$ であり, これは書き直すと $\varphi_{\bowtie}(f) \circ \varphi_{\bowtie}(g) = \varphi_{\bowtie}(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)$ である. このとき,

- $h \in M_1 \times M_2$ を $[f] \odot [g] = [h]$ であるようにとる. $[h] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)]$ である. よって $\varphi_{\bowtie}(h) = \varphi_{\bowtie}(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2) = \varphi_{\bowtie}(f) \circ \varphi_{\bowtie}(g)$ である.
- $h \in M_1 \times M_2$ を $\varphi_{\bowtie}(f) \circ \varphi_{\bowtie}(g) = \varphi_{\bowtie}(h)$ であるようにとる. $\varphi_{\bowtie}(h) = \varphi_{\bowtie}(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)$ である. よって $[h] = [(f_1 \cdot_1 g_1', f_2' \cdot_2 g_2)] = [f] \odot [g]$ である.

よって (3.) が成り立つ.

(4.) **について** $\odot$ が結合的であることと $\odot$ の単位元が $[(e_1, e_2)]$ であることを示せばよい.

- $f, g, h \in M_1 \times M_2$ を任意にとる. $([f] \odot [g]) \odot [h] = [f] \odot ([g] \odot [h])$ を示せばよい. これには, (3.) により, $(\varphi_{\bowtie}(f) \circ \varphi_{\bowtie}(g)) \circ \varphi_{\bowtie}(h) = \varphi_{\bowtie}(f) \circ (\varphi_{\bowtie}(g) \circ \varphi_{\bowtie}(h))$ を示せばよい. これは関数合成が結合的であることから明らか. よって $\odot$ は結合的である.
- $e = (e_1, e_2) \in M_1 \times M_2$ とおく. $f \in M_1 \times M_2$ を任意にとる. $[e] \odot [f] = [f]$ と $[f] \odot [e] = [f]$ を示せばよい. これには, (3.) により, $\varphi_{\bowtie}(e) \circ \varphi_{\bowtie}(f) = \varphi_{\bowtie}(f)$ と $\varphi_{\bowtie}(f) \circ \varphi_{\bowtie}(e) = \varphi_{\bowtie}(f)$ を示せばよい. これは $\varphi_{\bowtie}(e) = \varphi_1(e_1) \circ \varphi_2(e_2) = \mathrm{id}$ であることから明らか. よって $\odot$ の単位元は $[(e_1, e_2)]$ である.

□

定義 8.11.2 (遅延伝播構造から得られる積構造)**.**

$(M_1, S, \varphi_1), (M_2, S, \varphi_2)$ を遅延伝播構造とする. 任意の $f \in M_1$ と $g \in M_2$ に対しある $f' \in M_1$ と $g' \in M_2$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ を満たすとする. このとき補題 8.11.1 により得られるモノイド $((M_1 \times M_2)/\varphi_{\bowtie}, \odot, (e_1, e_2))$ のことを, M_1 と M_2 の $\varphi_1, \varphi_2, \mathrm{End}(S)$ についての **遅延伝播積**と呼び $M_1 \bowtie M_2$ と書く. ◇

注意 8.11.3.

任意の g, f' に対しある f'', g'' であって $\varphi_2(g) \circ \varphi_1(f') = \varphi_1(f'') \circ \varphi_2(g'')$ を満たすものが存在するが, かといって $(f, g) \cdot (f', g') = (f \cdot f'', g'' \cdot g')$ という演算を定義することはできないことに注意したい. なぜならそのような f'', g'' が一意に存在するとは限らず, 従って $(f \cdot f'', g'' \cdot g')$ という値も一意に定まるとは限らないからである. このため, 遅延伝播積 $M_1 \bowtie M_2$ の台集合に直積モノイド $M_1 \times M_2$ を直接使うことはできない[20]. ◇

命題 8.11.4.

$(M_1, S, \varphi_1), (M_2, S, \varphi_2)$ を遅延伝播構造とする. 任意の $f \in M_1$ と $g \in M_2$ に対しある $f' \in M_1$ と $g' \in M_2$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ を満たすとする. $(M_1 \bowtie M_2, S, \varphi_{\bowtie})$ は正規な遅延伝播構造である. ◇

証明

遅延伝播積の定義から明らか. □

ここからは遅延伝播積による遅延伝播構造と自由積による遅延伝播構造の関係を見ていく. 最終的に示されるのは次の定理 8.11.5 である.

定理 8.11.5.

$(M_1, S, \varphi_1), (M_2, S, \varphi_2)$ を遅延伝播構造とする. 任意の $f \in M_1$ と $g \in M_2$ に対しある $f' \in M_1$ と $g' \in M_2$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ を満たすとする. 命題 8.10.2 によって得られる遅延伝播構造を $(M_1 + M_2, S, \varphi_+)$ とする. 命題 8.11.4 によって得られる正規な遅延伝播構造を $(M_1 \bowtie M_2, S, \varphi_{\bowtie})$ とする. このとき, 正規な遅延伝播構造の間の同型性

$$((M_1 + M_2)/\varphi_+, S, \varphi_+) \simeq (M_1 \bowtie M_2, S, \varphi_{\bowtie})$$

が成り立つ. ◇

[20] しかし, 計算機上への実装の際には代表元のみを持つことになるだろう. つまり, M_1 の要素と M_2 の要素の対を持つだろう

この節の残りはすべてこの定理 8.11.5 の証明である. この定理の証明は, S が共通であることによって $\varphi_+ = \varphi_{\bowtie} \circ \psi$ を満たす同型 $\psi\colon (M_1 + M_2)/\varphi_+ \to M_1 \bowtie M_2$ のみを得ればよいことを言い, ひとまず可換図式

$$\begin{array}{ccc} M_1 \cup M_2 & \xrightarrow{\mu_{\bowtie}} & M_1 \bowtie M_2 \\ \downarrow{\scriptstyle \mu_+} & \nearrow{\scriptstyle \psi} & \downarrow{\scriptstyle \varphi_{\bowtie}} \\ M_1 + M_2 & \xrightarrow{\varphi_+} & \mathrm{End}(S) \end{array}$$

の成立を言い, そしてこの右下の三角形を $M_1 + M_2$ でなく $(M_1 + M_2)/\varphi_+$ のものとして解釈することで必要な同型 ψ を得る, という流れである.

まず, 次の補題により目標を整理する.

補題 8.11.6.

$(M_1, S_1, \varphi_1), (M_2, S_2, \varphi_2)$ を正規な遅延伝播構造とし $S_1 = S_2$ であるとする. モノイド準同型 $\psi\colon M_1 \to M_2$ が $\varphi_1 = \varphi_2 \circ \psi$ を満たすとする. id を恒等関数とする. このとき (ψ, id) は正規な遅延伝播構造 $(M_1, S_1, \varphi_1), (M_2, S_2, \varphi_2)$ の準同型である. ◇

証明

$\mathrm{id}(\varphi_1(f)(x)) = \varphi_2(\psi_M(f))(\mathrm{id}(x))$ が成り立てばよい. これは仮定から明らか. □

注意 8.11.7.

この節のこれ以降では, 以下を常に仮定する.

> $(M_1, S, \varphi_1), (M_2, S, \varphi_2)$ を遅延伝播構造とする. 自由積による遅延伝播構造 $(M_1 + M_2, S, \varphi_+)$ および遅延伝播積による正規な遅延伝播構造 $(M_1 \bowtie M_2, S, \varphi_{\bowtie})$ が存在するとする. モノイド M_1, M_2 の単位元を e_1, e_2 とし, モノイド $M_1 \bowtie M_2$ の演算を $\odot$ とする.

◇

さて $\mu_+, \mu_{\bowtie}, \psi$ の定義は以下のようになる.

定義 8.11.8.

関数 $\mu_+\colon M_1 \cup M_2 \to M_1 + M_2$ を

$$\mu_+(f) = [(f)]$$

で定義する. ◇

定義 8.11.9.

関数 $\mu_{\bowtie}\colon M_1 \cup M_2 \to M_1 \bowtie M_2$ を

$$\mu_{\bowtie}(f) = \begin{cases} [(f, e_2)] & (f \in M_1) \\ [(e_1, f)] & (f \in M_2) \end{cases}$$

で定義する. ◇

補題 8.11.10.

$\varphi_+ \circ \mu_+ = \varphi_{\bowtie} \circ \mu_{\bowtie}$ が成り立つ. ◇

証明

$f \in M_1 \cup M_2$ を任意にとる. このとき ψ の定義より

$$\begin{aligned} & \psi(\mu_+(f)) \\ = \ & \psi((f)) \\ = \ & \psi(\epsilon) \odot \mu_{\bowtie}(f) \\ = \ & [(e_1, e_2)] \odot \mu_{\bowtie}(f) \\ = \ & \mu_{\bowtie}(f) \end{aligned}$$

が成り立つ. □

補題 8.11.11.

$[f], [g] \in M_1 + M_2$ とする. 代表元 $f = (f_0, f_1, \ldots, f_{n-1}), g = (g_0, g_1, \ldots, g_{m-1}) \in (M_1 \cup M_2)^*$ は $M_1 \cup M_2$ の要素の有限列であった. このとき

$$[f] = [g] \to \prod_i \mu_{\bowtie}(f_i) = \prod_j \mu_{\bowtie}(g_j)$$

が成り立つ. ◇

証明

$f = (f_0, f_1, \ldots, f_{n-1}), g = (g_0, g_1, \ldots, g_{m-1})$ を $M_1 \cup M_2$ の要素の有限列とする. $f_i \in M_{p_i}$ かつ $g_j \in M_{q_j}$ (ただし $i < n$ と $j < m$) であるように $p_0, p_1, \ldots, p_{n-1}, q_0, q_1, \ldots, q_{m-1} \in \{1, 2\}$ をとる. このとき, 任意の $i \in n$ に対し $\varphi_1(f_i) = \varphi_+(\mu_+(f_i))$ であることと, $\prod_i \mu_+(f_i) = [(f_0)] \cdot [(f_1)] \cdot \cdots \cdot [(f_{n-1})] =$

$[(f_0, f_1, \ldots, f_{n-1})] = [f]$ であることから,

$$\begin{aligned}
&\prod_i \varphi_{p_i}(f_i) \\
=\ &\prod_i \varphi_+(\mu_+(f_i)) \\
=\ &\varphi_+\left(\prod_i \mu_+(f_i)\right) \\
=\ &\varphi_+([f])
\end{aligned}$$

が成り立つ. 同様に $\prod_j \varphi_{q_j}(g_j) = \varphi_+([g])$ が成り立つ. よって $[f] = [g]$ ならば $\prod_i \varphi_{p_i}(f_i) = \varphi_+([f]) = \varphi_+([g]) = \prod_j \varphi_{q_j}(g_j)$ が成り立つ. □

この補題 8.11.11 により次の定義 8.11.12 の ψ が well-defined に定義できる.

定義 8.11.12.

関数 $\psi\colon M_1 + M_2 \to M_1 \bowtie M_2$ を

$$\psi([(f_0, f_1, \ldots, f_{n-1})]) = \prod_{i=0}^{n-1} \mu_\bowtie(f_i)$$

で定義する. ◇

補題 8.11.13.

$\psi\colon M_1 + M_2 \to M_1 \bowtie M_2$ はモノイド準同型である. ◇

証明

$\psi([\epsilon]) = [(e_1, e_2)]$ であることは明らか. $[f], [g] \in M_1 + M_2$ とし $f = (f_0, f_1, \ldots, f_{n-1}), g = (g_0, g_1, \ldots, g_{m-1})$ とする. ψ の定義より

$$\begin{aligned}
&\psi([f] \cdot [g]) \\
=\ &\psi([(f_0, f_1, \ldots, f_{n-1}, g_0, g_1, \ldots, g_{m-1})]) \\
=\ &\mu_\bowtie(f_0) \odot \mu_\bowtie(f_1) \odot \cdots \odot \mu_\bowtie(f_{n-1}) \odot \mu_\bowtie(g_0) \odot \mu_\bowtie(g_1) \odot \cdots \odot \mu_\bowtie(g_{m-1}) \\
=\ &\prod_i \mu_\bowtie(f_i) \cdot \prod_j \mu_\bowtie(g_j) \\
=\ &\psi([f]) \odot \psi([g])
\end{aligned}$$

が成り立つ. □

これら $\mu_+, \mu_\bowtie, \psi$ および $\varphi_+, \varphi_\bowtie$ の間の関係を見ていく.

補題 8.11.14.

$\psi \circ \mu_+ = \mu_{\bowtie}$ が成り立つ. ◇

証明

$f \in M_1 \cup M_2$ とする. μ_+, ψ の定義より $\psi(\mu_+(f)) = \psi([(f)]) = \mu_{\bowtie}(f)$ である. □

補題 8.11.15.

$\varphi_+ = \varphi_{\bowtie} \circ \psi$ が成り立つ. ◇

証明

$[f] \in M_1 + M_2$ とし $f = (f_0, f_1, \ldots, f_{n-1})$ とする. このとき ψ, μ_+ の定義と $\varphi_{\bowtie}, \varphi_+$ の準同型性と補題 8.11.10 から

$$\begin{aligned} & \varphi_{\bowtie}(\psi([f])) \\ = \ & \varphi_{\bowtie}\left(\prod_i \mu_{\bowtie}(f_i)\right) \\ = \ & \prod_i \varphi_{\bowtie}(\mu_{\bowtie}(f_i)) \\ = \ & \prod_i \varphi_+(\mu_+(f_i)) \\ = \ & \varphi_+\left(\prod_i \mu_+(f_i)\right) \\ = \ & \varphi_+([f]) \end{aligned}$$

が成り立つ. □

補題 8.11.16.

$f, g \in M_1 + M_2$ とする.

$$\varphi_+(f) = \varphi_+(g) \to \psi(f) = \psi(g)$$

が成り立つ. ◇

証明

$f, g \in M_1 + M_2$ とし $\varphi_+(f) = \varphi_+(g)$ と仮定する. 補題 8.11.15 により $\varphi_{\bowtie}(\psi(f)) = \varphi_+(f) = \varphi_+(g) = \varphi_{\bowtie}(\psi(g))$ である. $\varphi_{\bowtie}\colon M_1 \bowtie M_2 \to \mathrm{End}(S)$ は定義より単射であるので $\psi(f) = \psi(g)$ である. □

注意 8.11.17.

補題 8.11.16 により $\psi\colon M_1 + M_2 \to M_1 \bowtie M_2$ は $\psi\colon (M_1 + M_2)/\varphi_+ \to M_1 \bowtie M_2$ と見ることができる. ◇

補題 8.11.18.

$\psi\colon (M_1 + M_2)/\varphi_+ \to M_1 \bowtie M_2$ について以下のすべてが成り立つ.

(1.) ψ はモノイド準同型である.
(2.) ψ は全射である.
(3.) ψ は単射である.

◇

証明

(1.) **について** $\psi([[\epsilon]]) = [(e_1, e_2)]$ である. $[[f]], [[g]] \in (M_1 + M_2)/\varphi_+$ を任意にとる. $\psi\colon M_1 + M_2 \to M_1 \bowtie M_2$ を同時に考えて, 補題 8.11.13 を使って

$$\begin{aligned} & \psi([[f]] \cdot [[g]]) \\ = \ & \psi([[f \frown g]]) \\ = \ & \psi([f \frown g]) \\ = \ & \psi([f] \cdot [g]) \\ = \ & \psi([f]) \odot ([g]) \\ = \ & \psi([[f]]) \odot ([[g]]) \end{aligned}$$

が成り立つ. よって ψ はモノイド準同型である.

(2.) **について** $[(f, g)] \in M_1 \bowtie M_2$ を任意にとる. $f \in M_1$ かつ $g \in M_2$ である. $[[(f, g)]] \in (M_1 + M_2)/\varphi_+$ を考えれば, $\psi([[(f, g)]]) = [(f, e_2)] \odot [(e_1, g)] = [(f, g)]$ である. よって ψ は全射である.

(3.) **について** $f, g \in (M_1 + M_2)/\varphi_+$ を任意にとる. このとき補題 8.11.15 および定義から φ_+ が単射であることを使って

$$\begin{aligned} & \psi(f) = \psi(g) \\ \Longrightarrow \ & \varphi_{\bowtie}(\psi(f)) = \varphi_{\bowtie}(\psi(g)) \\ \Longleftrightarrow \ & \varphi_+(f) = \varphi_+(g) \\ \Longleftrightarrow \ & f = g \end{aligned}$$

である. よって ψ は単射である.

□

補題 8.11.19.

モノイド同型 $(M_1 + M_2)/\varphi_+ \simeq M_1 \bowtie M_2$ が成り立つ. ◇

証明

補題 8.11.18 と命題 4.1.17 により成り立つ. □

証明 (定理 8.11.5)

補題 8.11.6 と補題 8.11.19 より成り立つ. □

8.12 一様代入クエリの併用

8.11 節での遅延伝播積を用いた議論の応用例のひとつとして, 一様代入クエリが任意の更新クエリと併用可能であることを示す. 一様代入クエリ単体については 8.6 節で見たが, これの続きである.

モノイド M とモノイド作用 $\star\colon M \times S \to S$ があるときに, 半群 S の要素の列 $(x_0, x_1, \ldots, x_{n-1})$ についての以下の 2 種のクエリを処理できる遅延伝播セグメント木を考える.

- 区間一様代入更新: 与えられた l, r (ただし $l < r$) と $a \in S$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a$ と $O(\log n)$ で更新する
- 区間更新: 与えられた l, r (ただし $l < r$) と $f \in M$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow f \star x$ と $O(\log n)$ で更新する
- 区間取得: 与えられた l, r (ただし $l < r$) に対し, 積 $x_l \cdot x_{l+1} \cdot \cdots \cdot x_{r-1}$ を $O(\log n)$ で計算する

このような遅延伝播セグメント木は, 次の定理 8.12.1 により得られる遅延伝播構造によって得られる.

定理 8.12.1.

(M, S, φ) を遅延伝播構造と仮定する. 集合 S についての左零半群に単位元を付与してできるモノイドを $L = (S \cup \{ e_L \}, \cdot_L, e_L)$ とおく. S と自然数の加法についての半群 $\mathbb{N}$ との直積半群 $S \times \mathbb{N}$ を考える. 関数 $\varphi_1\colon M \to \mathrm{End}(S \times \mathbb{N})$ を $\varphi_1(f)(x, k) = (\varphi_1(f), k)$

で定義し, 関数 $\varphi_2 \colon L \to \mathrm{End}(S \times \mathbb{N})$ を $\varphi_2(g)(x,k) = \begin{cases} (x,k) & (g = e) \\ (kg,k) & (g \neq e) \end{cases}$ で定義する.
このとき $(M, S \times \mathbb{N}, \varphi_1)$ と $(L, S \times \mathbb{N}, \varphi_2)$ は遅延伝播構造である. また, これらに対し遅延伝播積 $M \bowtie L$ が存在し, 自然に定義される関数 φ' によって $(M \bowtie L, S \times \mathbb{N}, \varphi')$ は遅延伝播構造である. ◇

証明

以下のみっつを示す必要がある.

(1.) $(M, S \times \mathbb{N}, \varphi_1)$ が遅延伝播構造である.
(2.) $(L, S \times \mathbb{N}, \varphi_2)$ が遅延伝播構造である.
(3.) $(M \bowtie L, S \times \mathbb{N}, \varphi')$ が遅延伝播構造である.

(1.) については補題 8.5.4 から明らか. (2.) については定理 8.6.1 から明らか. (3.) については命題 8.11.4 を使うが, このためには次の (3'.) を示せばよい.

(3'.) 任意の $f \in M$ と $g \in L$ に対しある $f' \in M$ と $g' \in L$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ を満たす.

この (3'.) を示していく. $f \in M$ と $g \in L$ を任意にとる. g が単位元かどうかで場合分け.

- $g = e_L$ の場合: $f' = f$ かつ $g' = g = e_L$ とすればよい. $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f) = \varphi_1(f) \circ \varphi_2(g)$ である.
- $g \neq e_L$ の場合: $f' = e_M$ かつ $g' = g$ とすればよい. 任意の $(x,k) \in S \times \mathbb{N}$ に対し

$$\begin{aligned} & (\varphi_2(g) \circ \varphi_1(f))(x,k) \\ = \ & \varphi_2(g)(\varphi_1(f)(x,k)) \\ = \ & (kg,k) \\ = \ & \varphi_2(g)(x,k) \\ = \ & \varphi_1(e_M)(\varphi_2(g)(x,k)) \end{aligned}$$

である.

よって (3'.) が成り立つ. □

注意 8.12.2.

注意 8.6.2 の問題は依然として存在する. ◇

注意 8.12.3.

一様代入クエリの併用はセグメント木の側を拡張することでも実現可能である. しかし, そのようにセグメント木の側を修正することは, 遅延伝播構造の側を拡張することと実質的に同じである. ◇

8.13 Segment Tree Beats の背後の構造

8.11 節での遅延伝播積を用いた議論の応用例のひとつとして, Segment Tree Beats [9] と呼ばれるセグメント木の変種について, その更新クエリの代数構造について見よう. ただしその計算量解析については複雑でかつ今回の議論と関連が薄いので紹介しない. つまり, 取得クエリについては無視する.

Segment Tree Beats と呼ばれるセグメント木にも多くの種類があるが, ここでは整数の列 $(x_0, x_1, \ldots, x_{n-1})$ についての以下の 4 種のクエリを処理できるセグメント木の変種のこととする.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $a \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a + x_i$ と $O((\log n)^2)$ で更新する
- 区間最大値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow \max(b, x_i)$ と $O((\log n)^2)$ で更新する
- 区間最小値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow \min(b, x_i)$ と $O((\log n)^2)$ で更新する
- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $x_l + x_{l+1} + \cdots + x_{r-1}$ を $O((\log n)^2)$ で計算する

さて, 区間和の取得については実装も計算量も解析が難しい. 基本は 8.5 節で見たようにするのだが, それだけでは計算が不可能であるため, 遅延伝播セグメント木の内部に踏み込んだ修正を使ってこれを回避する必要がある. その回避の手法などについては, 今回は紹介しない.

更新クエリの代数構造のみを見るため単純化して, 以下の 4 種のクエリを処理できるセグメント木を考えよう.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $a \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a + x_i$ と $O(\log n)$ で更新する
- 区間最大値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$

について $x_i \leftarrow \max(b, x_i)$ と $O(\log n)$ で更新する
- 区間最小値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow \min(b, x_i)$ と $O(\log n)$ で更新する
- 点取得: 与えられた i に対し, x_i を $O(\log n)$ で計算する

これは双対セグメント木で十分であるが, 作用が重要となる都合から遅延伝播セグメント木を用いて議論するのが楽である. 次の定理によって得られる遅延伝播構造を考えればよい. ただし, 遅延伝播積の結合性などについての議論は省略する.

定理 8.13.1.

整数の集合上の左零半群を $S = (\mathbb{Z}, \lambda xy.x)$ とする. 以下のよっつが成り立つ.

(1.) 整数の加法モノイドを $M_1 = (\mathbb{Z}, +, 0)$ とする. 関数 $\varphi_1 \colon M_1 \to \mathrm{End}(S)$ を $\varphi_1(f)(x) = f + x$ で定義する. このとき (M_1, S, φ_1) は遅延伝播構造である.

(2.) 整数の最大値によるモノイドを $M_2 = (\mathbb{Z} \cup \{ -\infty \}, \max, -\infty)$ とする. 関数 $\varphi_2 \colon M_2 \to \mathrm{End}(S)$ を $\varphi_2(f)(x) = \max\{ f, x \}$ で定義する. このとき (M_2, S, φ_2) は遅延伝播構造である.

(3.) 整数の最小値によるモノイドを $M_3 = (\mathbb{Z} \cup \{ \infty \}, \min, \infty)$ とする. 関数 $\varphi_3 \colon M_3 \to \mathrm{End}(S)$ を $\varphi_3(f)(x) = \min\{ f, x \}$ で定義する. このとき (M_3, S, φ_3) は遅延伝播構造である.

(4.) これらの遅延伝播構造に対しての遅延伝播積 $(M_1 \bowtie M_2 \bowtie M_3, S, \varphi)$ が存在する.

◇

証明

(1.) について (1.) は定理 8.8.1 から明らかである.

(2.), (3.) について (2.), (3.) も定理 8.8.1 と同様にして示せる.

(4.) について 次のみっつを示し, 命題 8.11.4 を用いればよい.

(a.) 任意の $f \in M_1$ と $g \in M_2$ に対しある $f' \in M_1$ と $g' \in M_2$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ を満たす.

(c.) 任意の $f \in M_1$ と $g \in M_3$ に対しある $f' \in M_1$ と $g' \in M_3$ が存在し $\varphi_3(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_3(g')$ を満たす.

(b.) 任意の $f \in M_2$ と $g \in M_3$ に対しある $f' \in M_2$ と $g' \in M_3$ が存在し $\varphi_3(g) \circ \varphi_2(f) = \varphi_2(f') \circ \varphi_3(g')$ を満たす.

これらを順に示していく.

(a.) **について**　$f \in M_1$ と $g \in M_2$ を任意にとる. g が単位元かどうかで場合分け:

- $g = -\infty$ の場合: $f' = f$ かつ $g' = -\infty$ とすればよい. $-\infty$ は M_2 の単位元であるので $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f) = \varphi_1(f) \circ \varphi_2(g)$ である.
- $g \neq -\infty$ の場合: $f' = f$ かつ $g' = g - f$ とすればよい. 任意の $x \in \mathbb{Z}$ に対し

$$\begin{aligned}(\varphi_2(g) \circ \varphi_1(f))(x) & \\ &= \max(g, f + x) \\ &= f + \max(g - f, x) \\ &= (\varphi_1(f) \circ \varphi_2(g - f))(x)\end{aligned}$$

が成り立つ.

(b.) **について**　(a.) と同様にすればよい.

(c.) **について**　$f \in M_2$ と $g \in M_3$ を任意にとる. g が単位元かどうかで場合分け:

- $g \leq f$ の場合: $f' = g' = g$ とすればよい. 任意の $x \in \mathbb{Z}$ に対し

$$\begin{aligned}&(\varphi_3(g) \circ \varphi_2(f))(x) \\ =\ &\min(g, \max(f, x)) \\ =\ &g \\ =\ &\max(g, \min(g, x)) \\ =\ &(\varphi_2(g) \circ \varphi_3(g))(x)\end{aligned}$$

が成り立つ.

- $g > f$ の場合: $f' = f$ かつ $g' = g$ とすればよい. 任意の $x \in \mathbb{Z}$ に対し

$$\begin{aligned}(\varphi_2(g) \circ \varphi_1(f))(x) & \\ &= \min(g, \max(f, x)) \\ &= \max(f, \min(g, x)) \\ &= (\varphi_1(f) \circ \varphi_2(g))(x)\end{aligned}$$

が成り立つ.

(a.), (b.), (c.) が言えた. よって (4.) が成り立つ.

□

8.14 復元クエリの併用

この節では, 復元クエリと呼ばれるクエリについて紹介し, これが任意のクエリと併用可能であることを示す.

まず復元クエリを導入しよう. 半群 S の要素の列 $x = (x_0, x_1, \ldots, x_{n-1}) \in S^n$ があるとする. x とは別に列 $(a_0, a_1, \ldots, a_{n-1}) \in S^n$ があらかじめ固定されているとする. このとき以下の 2 種のクエリを処理できる遅延伝播セグメント木が存在する.

- 区間復元更新: 与えられた l, r (ただし $l < r$) に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a_i$ と $O(\log n)$ で更新する
- 区間取得: 与えられた l, r (ただし $l < r$) に対し, 積 $x_l \cdot x_{l+1} \cdot \cdots \cdot x_{r-1}$ を $O(\log n)$ で計算する

そのような遅延伝播セグメント木は, 次の補題 8.14.1 により得られる遅延伝播構造によって得られる. ただし, その遅延伝播セグメント木は, 半群 $S \times S$ の要素の列 $\big((x_0, a_0), (x_1, a_1), \ldots, (x_{n-1}, a_{n-1})\big)$ を管理する.

補題 8.14.1.

$M = (\{ 0, 1 \}, +, 0)$ を自明な半群 $\{ 1 \}$ に単位元 0 を付け加えてできるモノイドとする. S を半群とする. S と S の直積半群を考える. 関数 $\varphi\colon M \to \mathrm{End}(S \times S)$ を

$$\varphi(f)(x, y) = \begin{cases} (x, y) & (f = 0) \\ (y, y) & (f = 1) \end{cases}$$

で定義する. このとき $(M, S \times S, \varphi)$ は遅延伝播構造である. ◇

証明

まず φ が well-defined であること, つまり $\varphi(f)\colon S \times S \to S \times S$ が半群準同型であることを示す. $f \in M$ の値で場合分け.

- $f = 0$ の場合: $\varphi(0)$ は恒等関数である.

- $f = 1$ の場合: 任意の $(x_1, y_1), (x_2, y_2) \in S \times S$ に対し

$$\begin{aligned} & \varphi(1)((x_1, y_1) \cdot (x_2, y_2)) \\ = \ & \varphi(1)(x_1 \cdot x_2, y_1 \cdot y_2) \\ = \ & (y_1 \cdot y_2, y_1 \cdot y_2) \\ = \ & (y_1, y_1) \cdot (y_2, y_2) \\ = \ & \varphi(1)(x_1, y_1) \cdot \varphi(1)(x_2, y_2) \end{aligned}$$

  が成り立つ.

よって $\varphi(f)$ は半群準同型であり, φ は well-defined である.

あとは φ がモノイド準同型であることを示せばよい.

- $\varphi(0)$ は恒等関数である.
- $f, g \in M$ を任意にとる. $f, g \in M$ の値で場合分け.
  - $f = 0$ かつ $g = 0$ の場合: $\varphi(f + g), \varphi(f), \varphi(g), \varphi(f) \circ \varphi(g)$ はすべて恒等関数である.
  - $f = 0$ かつ $g = 1$ の場合: $\varphi(f)$ は恒等関数でかつ $\varphi(f + g) = \varphi(g)$ である. よって $\varphi(f + g) = \varphi(f) \circ \varphi(g)$ である.
  - $f = 1$ かつ $g = 0$ の場合: $f = 0$ かつ $g = 1$ の場合と同様にすればよい.
  - $f = 1$ かつ $g = 1$ の場合: $(x, y) \in S \times S$ を任意にとる.

$$\begin{aligned} & (\varphi(f) \circ \varphi(1))(x, y) \\ = \ & \varphi(1)(\varphi(1)(x, y)) \\ = \ & \varphi(1)(y, y) \\ = \ & (y, y) \\ = \ & \varphi(1)(x, y) \end{aligned}$$

    が成り立つ.

  つまり $\varphi(f + g) = \varphi(f) \circ \varphi(g)$ である.

よって φ はモノイド準同型である. □

注意 8.14.2.

ここでこのクエリを「復元クエリ」と呼んでいるのは固定された列 a が管理対象の列 x の初期状態に等しい場合を意識してのことである[21]. ◇

[21] TODO(kimiyuki): これ名前変えた方がいいか？

注意 8.14.3.
復元クエリは双対セグメント木の上でも考えることができる. しかし双対セグメント木の場合は一様代入クエリで代用できてしまう. ◇

注意 8.14.4.
任意の列を代入できるような列代入クエリは, 注意 6.2.2 の列挿入クエリと列削除クエリを組み合わせることで実現できる. これは復元クエリより強力なものであるが, 遅延伝播セグメント木のために利用する二分木に対する仮定も増加する. ◇

この復元クエリは任意のクエリと併用可能である. つまり, モノイド M とモノイド作用 $\star\colon M \times S \to S$ があり, 列 $(a_0, a_1, \ldots, a_{n-1})$ があらかじめ固定されているとき, 半群 S の要素の列 $(x_0, x_1, \ldots, x_{n-1})$ についての以下の 3 種のクエリを処理できる遅延伝播セグメント木が存在する.

- 区間更新: 与えられた l, r (ただし $l < r$) と $f \in M$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow f \star x$ と $O(\log n)$ で更新する
- 区間復元更新: 与えられた l, r (ただし $l < r$) に対し, すべての $i \in [l, r)$ について $x_i \leftarrow a_i$ と $O(\log n)$ で更新する
- 区間取得: 与えられた l, r (ただし $l < r$) に対し, 積 $x_l \cdot x_{l+1} \cdot \cdots \cdot x_{r-1}$ を $O(\log n)$ で計算する

そのような遅延伝播セグメント木は, 次の定理 8.14.5 の (4.) により得られる遅延伝播構造によって得られる. ただし, その遅延伝播セグメント木は, 半群 $S \times S$ の要素の列 $\big((x_0, a_0), (x_1, a_1), \ldots, (x_{n-1}, a_{n-1})\big)$ を管理する.

定理 8.14.5.
(M, S, φ) を遅延伝播構造とする. 以下のよっつが成り立つ.

(1.) S と S の直積半群 $S \times S$ を考える. 関数 $\varphi_1\colon M \to \mathrm{End}(S \times S)$ を

$$\varphi_1(f)(x, y) = (\varphi(f)(x), y)$$

で定義する. このとき $(M, S \times S, \varphi_\times)$ は遅延伝播構造である.

(2.) $N = (\{0, 1\}, +, 0)$ を自明な半群 $\{1\}$ に単位元 0 を付け加えてできるモノイドとする. 関数 $\varphi_2\colon N \to \mathrm{End}(S \times S)$ を

$$\varphi_2(f)(x, y) = \begin{cases} (x, y) & (f = 0) \\ (y, y) & (f = 1) \end{cases}$$

で定義する. このとき $(N, S \times S, \varphi_2)$ は遅延伝播構造である.

(3.) 任意の $f \in M$ と $g \in N$ に対しある $f' \in M$ と $g' \in N$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ である.

(4.) $(M, S \times S, \varphi_\times)$ と $(N, S \times S, \varphi_N)$ に対する遅延伝播積 $M \bowtie N$ が存在し, 自然に定義される関数 φ' によって $(M \bowtie N, S \times S, \varphi')$ は遅延伝播構造である.

◇

証明

(1.) **について**　補題 8.5.4 から明らか.

(2.) **について**　補題 8.14.1 として示した.

(3.) **について**　$f \in M$ と $g \in N$ を任意にとる. $g \in N$ の値で場合分け.

- $g = 0$ の場合: $f' = f$ かつ $g' = 0$ とすればよい.

$$\begin{aligned} & \varphi_2(g) \circ \varphi_1(f) \\ = \ & \mathrm{id} \circ \varphi_1(f) \\ = \ & \varphi_1(f) \circ \mathrm{id} \\ = \ & \varphi_1(f') \circ \varphi_2(g') \end{aligned}$$

  が成り立つ.

- $g = 1$ の場合: M の単位元を e とする. $f' = e$ かつ $g' = 1$ とすればよい. 任意の $(x, y) \in S \times S$ に対し

$$\begin{aligned} & \varphi_2(g)(\varphi_1(f)(x, y)) \\ = \ & \varphi_2(g)(\varphi(f)(x), y) \\ = \ & (y, y) \\ = \ & \varphi_2(g)(x, y) \\ = \ & \mathrm{id}(\varphi_2(g)(x, y)) \\ = \ & \varphi_1(f')(\varphi_2(g')(x, y)) \end{aligned}$$

  が成り立つ.

よって, ある $f' \in M$ と $g' \in N$ が存在し $\varphi_2(g) \circ \varphi_1(f) = \varphi_1(f') \circ \varphi_2(g')$ が成り立つ.

(4.) **について**　(3.) と定理 8.11.5 により成り立つ.

□

注意 8.14.6.

複数種類の復元クエリを同時に扱うことができる. k 種類の復元クエリを扱える遅延伝播セグメント木を得るには, 復元クエリを追加する操作を単に k 回行えばよい. このとき遅延伝播構造は $(M \bowtie \underbrace{\{0,1\} \bowtie \{0,1\} \bowtie \ldots \bowtie \{0,1\}}_{k \text{ times}}, S \times \underbrace{S \times S \times \cdots \times S}_{k \text{ times}}, \varphi)$ のようになる.

また, 定理 8.14.5 中の $N = \{0,1\}$ を $\{0,1,2,\ldots,k\}$ で置き換えかつ直積半群 $S \times S$ を $S \times \underbrace{S \times S \times \cdots \times S}_{k \text{ times}}$ で置き換えるようにしても k 種類の復元クエリを扱える. このとき遅延伝播構造は $(M \bowtie \{0,1,2,\ldots,k\}, S \times \underbrace{S \times S \times \cdots \times S}_{k \text{ times}}, \varphi)$ のようになる. これは実装した際により効率がよい. ◇

8.15 半群準同型を調べる

この節では, 半群 S を固定し, S を使う遅延伝播構造にはどのようなものがあるかを考える. まず一般論として, S を使う遅延伝播構造で最大のものは $(\mathrm{End}(S), S, \mathrm{id})$ であることを示す. その後に各論として, いくつかの代表的な半群 S についてその自己準同型モノイド $\mathrm{End}(S)$ を調べる.

次の定理 8.15.1 は正規な遅延伝播構造と $\mathrm{End}(S)$ の部分モノイドを利用した遅延伝播構造の同型性を示す. この定理により, どのようなクエリが利用可能かについて考えるとき, モノイド M としては $\mathrm{End}(S)$ の部分モノイドのみを考えればよいことが分かる. 言い換えれば, 用いる半群 S を固定した場合において, どのようなクエリが (計算量を無視して, 直接に) 実現可能でありかつどのようなクエリが (直接に) 実現不可能であるかについての, 完全な条件を与える. なお, クエリについて考えるときは正規な遅延伝播構造のみ考えてよいのであったため, この定理が正規な遅延伝播構造のみについての定理であることは問題にならないことを注意しておく.

定理 8.15.1 (正規な遅延伝播構造の表現定理)**.**

(M, S, φ) を正規な遅延伝播構造とする. $M' = (\{\varphi(f) \mid f \in M\}, \circ, \mathrm{id})$ とおく. このとき以下のよっつが成り立つ.

(1.) M' はモノイドである.
(2.) M' は M とモノイドとして同型 $M \simeq M'$ である.
(3.) M' は $\mathrm{End}(S)$ の部分モノイド $M' \subseteq \mathrm{End}(S)$ である.

(4.) (M, S, φ) は (M', S, id) と正規な遅延伝播構造として同型である.

◇

証明

(1.), (2.), (3.) はそれぞれ明らか.

(4.) について. 補題 8.11.6 により, モノイド同型 $\psi : M \to M'$ であって $\varphi = \mathrm{id} \circ \psi$ であるようなものがあればよい. $\psi = \varphi$ とすればよい. □

さて, 具体的な半群を見ていこう.

まずは整数と加法からなる半群 $(\mathbb{Z}, +)$ を考えよう. この半群の準同型は整数倍のみである. このことは補題 8.15.2 として示される.

補題 8.15.2.

半群 $S = (\mathbb{Z}, +)$ を考える. $\mathrm{End}(S) = \{ \lambda x.kx \mid k \in \mathbb{Z} \}$ が成り立つ. ◇

証明

$\mathrm{End}(S) \subseteq \{ \lambda x.kx \mid k \in \mathbb{Z} \}$ と $\{ \lambda x.kx \mid k \in \mathbb{Z} \} \subseteq \mathrm{End}(S)$ を示せばよい.

$f \in \mathrm{End}(S)$ とする. $k = f(1)$ とおく. f は準同型であるので, 任意の $x, y \in S$ に対し $f(x + y) = f(x) + f(y)$ である. $f(0) = f(0 + 0) = f(0) + f(0) = 2f(0)$ であるので $f(0) = 0$ である. $0 = f(0) = f(1 + (-1)) = f(1) + f(-1) = k + f(-1)$ であるので $f(-1) = -k$ である. つまり $x \in \{ -1, 0, 1 \}$ については $f(x) = kx$ である. $x \geq 0$ について $f(x) = kx$ が成り立つと仮定すると $f(x+1) = f(x)+f(1) = kx+k = k(x+1)$ が成り立つ. よって数学的帰納法により, 任意の $x \geq 0$ に対し $f(x) = kx$ である. 同様にして, 任意の $x \geq 0$ に対し $f(-x) = -kx$ であることも示せる. よって $f \in \{ \lambda x.kx \mid k \in \mathbb{Z} \}$ である.

$f = \lambda x.kx$ とする. $x, y \in \mathbb{Z}$ を任意にとる. $f(x+y) = k(x+y) = kx+ky = f(x)+f(y)$ である. よって f は準同型であり $f \in \mathrm{End}(S)$ である. □

この補題を使って「何が更新クエリとして使えないか」を判断できる. たとえば, 整数列 $(x_0, x_1, \ldots, x_{n-1})$ に対して以下のようなふたつのクエリを処理することは (少なくとも半群として $(\mathbb{Z}, +)$ を使う限りは) 不可能であることが分かる. $\lambda x.x + k$ (ただし $k \neq 0$) は $(\mathbb{Z}, +)$ の準同型ではないためである.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $k \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow x_i + k$ と $O(\log n)$ で更新する

- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $x_l + x_{l+1} + \cdots + x_{r-1}$ を $O(\log n)$ で計算する

補題 8.15.2 をより一般化した補題 8.15.3 を紹介しておく. これはつまり「ベクトル空間の準同型は線型写像であり, 線形写像は行列として書ける」ということである. 証明については省略する. この補題は, $S = (\mathbb{Z}, +)$ に対する $S \times S$ や $S \times S \times S$ などを半群として用いたときに, どのようなクエリが処理できてどのようなクエリが処理できないかの判定に役立てられる.

補題 8.15.3.

R を可換環とする. n を正整数とする. 数ベクトル空間 R^n の加法についての半群 $(R^n, +)$ を S とする. n 次正方行列の全体のなす環 $M_n(R)$ の乗法についてのモノイド $(M_n(R), \cdot, I)$ を M とする. このとき $\mathrm{End}(S) \simeq M$ である. ◇

半群として整数と最大値からなる半群 $(\mathbb{Z}, \max)$ についても見ておこう. この半群の準同型は単調関数である. このことは補題 8.15.4 として示される. 最小値の場合も同様である.

補題 8.15.4.

半群 $S = (\mathbb{Z}, \max)$ を考える. $\mathrm{End}(S) = \left\{ f \colon \mathbb{Z} \to \mathbb{Z} \;\middle|\; f \text{ は単調} \right\}$ が成り立つ. ただし, 関数 $f \colon \mathbb{Z} \to \mathbb{Z}$ が単調であるとは $\forall xy \in \mathbb{Z}. x \leq y \to f(x) \leq f(y)$ を満たすことを言う. ◇

証明

$\mathrm{End}(S) \subseteq \{ f \colon \mathbb{Z} \to \mathbb{Z} \mid \forall xy \in \mathbb{Z}. x \leq y \to f(x) \leq f(y) \}$ と
$\{ f \colon \mathbb{Z} \to \mathbb{Z} \mid \forall xy \in \mathbb{Z}. x \leq y \to f(x) \leq f(y) \} \subseteq \mathrm{End}(S)$ を示せばよい.

$f \in \mathrm{End}(S)$ とする. f は準同型であるので, 任意の $x, y \in S$ に対し $f(\max\{x, y\}) = \max\{f(x), f(y)\}$ である. $x, y \in \mathbb{Z}$ を任意にとる. 一般性を失わず $x \leq y$ と仮定してよい. $f(y) = f(\max\{x, y\}) = \max\{f(x), f(y)\}$ である. このことは $f(x) \leq f(y)$ を意味する.

関数 $f \colon \mathbb{Z} \to \mathbb{Z}$ であって $\forall xy \in \mathbb{Z}. x \leq y \to f(x) \leq f(y)$ を満たすものをとる. $x, y \in \mathbb{Z}$ を任意にとる. 一般性を失わず $x \leq y$ と仮定してよい. 仮定より $f(x) \leq f(y)$ であり $\max\{f(x), f(y)\} = f(y)$ が成り立つ. $f(\max\{x, y\}) = f(y) = \max\{f(x), f(y)\}$ である. よって f は準同型であり $f \in \mathrm{End}(S)$ である. □

応用として, 整数と加法からなる半群 $S_+ = (\mathbb{Z}, +)$ と整数と最大値からなる半群

$S_M = (\mathbb{Z}, \max)$ の直積半群 $S = S_+ \times S_M$ について見ておこう．この半群 S の準同型は, 直積の成分であるそれぞれの半群 S_+ と S_M の準同型の対として書ける．これは補題 8.15.5 として示される[22]．このことからは, これらの半群 S_+ と S_M について単純に直積をとっただけでは処理可能なクエリはまったく増えないことが言える．

補題 8.15.5.

整数と加法からなる半群 $S_+ = (\mathbb{Z}, +)$ と整数と最大値からなる半群 $S_M = (\mathbb{Z}, \max)$ の直積半群 $S = S_+ \times S_M$ を考える．このとき, 任意の $f\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} \times \mathbb{Z}$ に対し次のふたつは同値である．

(1.) f は半群 S の準同型である．

(2.) 半群 S_+ のある準同型 $f_+\colon \mathbb{Z} \to \mathbb{Z}$ と半群 S_M のある準同型 $f_M\colon \mathbb{Z} \to \mathbb{Z}$ が存在し, 任意の $(x, y) \in \mathbb{Z} \times \mathbb{Z}$ に対し $f(x, y) = (f_+(x), f_M(y))$ が成り立つ．

◇

証明

(1.) ⇒ (2.) **について**　f を S の準同型とする．$f(x, y) = (f_+(x, y), f_M(x, y))$ となるような関数 $f_+\colon S \to \mathbb{Z}$ と $f_M\colon S \to \mathbb{Z}$ がある．このとき, 補題 8.15.2 と補題 8.15.4 により, ある $k \in \mathbb{Z}$ が存在し $f_+(x, y) = kx$ であることと, ある単調関数 g が存在し $f_M(x, y) = g(y)$ であることをそれぞれ示せばよい．

整数 k の存在について　$y \in \mathbb{Z}$ を固定する．関数 $f_{+,y}(x) = f_+(x, y)$ であるような関数 $f_{+,y}\colon \mathbb{Z} \to \mathbb{Z}$ を考える．f の準同型性から, 任意の $x, x' \in \mathbb{Z}$ に対し $f_+(x + x', y) = f_+(x + x', \max\{ y, y \}) = f_+(x, y) + f_+(x', y)$ が成り立つ．よって $f_{+,y}$ は $(\mathbb{Z}, +)$ の準同型である．補題 8.15.2 により, ある $k' \in \mathbb{Z}$ が存在して $f_{+,y}(x) = k'x$ である．

y を固定するごとに得られるこの整数 k' を y の関数と見て $k(y) = k'$ とおく．このとき $f_+(x, y) = k(y)x$ である．この関数 $k\colon \mathbb{Z} \to \mathbb{Z}$ が定数関数であることを言えばよい．

$y, y' \in \mathbb{Z}$ を任意にとり $y \leq y'$ と仮定する．$0 = f_+(0, y') = f_+(1 + (-1), \max\{ y, y' \}) = f_+(1, y) + f_+(-1, y') = k(y) - k(y')$ が成り立つ．$k(y) = k(y')$ である．よって k は定数関数である．

[22] TODO: これもう少し一般化できない？

単調関数 g の存在について　$y \in \mathbb{Z}$ を固定する. f の準同型性から, 任意の $x \in \mathbb{Z}$ に対し $f_M(x,y) = f_M(0+x, \max\{y,y\}) = \max\{f_M(0,y), f_M(x,y)\}$ が成り立つ. よって, 任意の $x \in \mathbb{Z}$ に対し $f_M(0,y) \leq f_M(x,y)$ である. 再び f の準同型性から, 任意の $x \in \mathbb{Z}$ に対し $f_M(0,y) = f_M(x+(-x), \max\{y,y\}) = \max\{f_M(x,y), f_M(-x,y)\}$ が成り立つ. よって, 任意の $x \in \mathbb{Z}$ に対し $f_M(x,y) \leq f_M(0,y)$ である. これらふたつにより, 任意の $x \in \mathbb{Z}$ に対し $f_M(x,y) = f_M(0,y)$ である.
さて関数 g を $g(y) = f_M(0,y)$ で定める. このとき $f_M(x,y) = g(y)$ が成り立つ. また f の準同型性から g は単調である.

(2.) ⇒ (1.) **について**　明らか.

□

また補題 8.15.5 からは, $S_+ = (\mathbb{Z}, +)$ と $S_M = (\mathbb{Z}, \max)$ を組み合わせてそれら単体では処理できなかったクエリを処理しているように見える半群は, 単純な直積半群ではないということも言える. このような単純な直積ではない半群の例として, 最大値とその個数を数えるクエリに関しての半群がある. 結果となる条件のきれいさのために個数を正に制限すれば, その自己準同型モノイドの形は補題 8.15.6 のようになる.

補題 8.15.6.
演算 $\cdot\colon (\mathbb{Z} \times (\mathbb{N} \setminus \{0\})) \times (\mathbb{Z} \times (\mathbb{N} \setminus \{0\})) \to \mathbb{Z} \times (\mathbb{N} \setminus \{0\})$ を

$$(x,k) \cdot (y,k') = \begin{cases} (x,k) & (x > y) \\ (y,k') & (x < y) \\ (x,k+k') & (x = y) \end{cases}$$

で定義する. $S = (\mathbb{Z} \times (\mathbb{N} \setminus \{0\}), \cdot)$ は半群である. このとき, 任意の $f\colon \mathbb{Z} \times (\mathbb{N} \setminus \{0\}) \to \mathbb{Z} \times (\mathbb{N} \setminus \{0\})$ に対し次のふたつは同値である.

(1.) f は半群 S の準同型である.
(2.) ある狭義単調関数 $g\colon \mathbb{Z} \to \mathbb{Z}$ と関数 $h\colon \mathbb{Z} \to \mathbb{N} \setminus \{0\}$ が存在して, 任意の $(x,k) \in \mathbb{Z} \times (\mathbb{N} \setminus \{0\})$ に対し, $f(x,k) = (g(x), kh(x))$ が成り立つ.

◇

証明

(1.) ⇒ (2.) **について**　f を半群 S の準同型とする. $f(x,k) = (f_M(x,k), f_+(x,k))$ であ

るような関数 $f_M\colon \mathbb{Z}\times(\mathbb{N}\setminus\{0\})\to\mathbb{Z}$ および関数 $f_+\colon \mathbb{Z}\times(\mathbb{N}\setminus\{0\})\to\mathbb{N}$ をとる.

任意の $x\in\mathbb{Z}$ と $k\geq 1$ に対して $f(x,k+1)=f((x,k)\cdot(x,1))=f(x,k)\cdot f(x,1)$ である. $k\geq 1$ についての数学的帰納法により, 任意の $x\in ZZ$ と $k\geq 1$ に対して $f(x,k)=\underbrace{f(x,1)\cdot f(x,1)\cdot\cdots\cdot f(x,1)}_{k\,\text{times}}$ が成り立つ. これにより, 任意の $x\in\mathbb{Z}$ と $k\geq 1$ に対して $f(x,k)=(f_M(x,1),kf_+(x,1))$ である. よって $g(x)=f_M(x,1)$ および $h(x)=f_+(x,1)$ とおけば $f(x)=(g(x),kh(x))$ である.

任意の x,y で $x<y$ なものおよび k,k' に対し $f(y,k')=f((x,k)\cdot(y,k'))=f(x,k)\cdot f(y,k')$ である. つまり任意の x,y,k,k' に対し, $x<y$ ならば $h(x)\leq h(y)$ である. このとき $kh(x)\geq 1$ であることから $h(x)\neq h(y)$ も成り立つ.. よって h は狭義単調増加である.

(2.) ⇒ (1.) **について**　$g\colon\mathbb{Z}\to\mathbb{Z}$ を狭義単調関数とし $h\colon\mathbb{Z}\to\mathbb{N}\setminus\{0\}$ を関数とする. 関数 $f\colon\mathbb{Z}\times(\mathbb{N}\setminus\{0\})\to\mathbb{Z}\times(\mathbb{N}\setminus\{0\})$ を $f(x,k)=(g(x),kh(x))$ で定める. $(x,k),(y,k')\in\mathbb{Z}\times(\mathbb{N}\setminus\{0\})$ をとる. x,y の大小関係で場合分け.

$x=y$ **の場合**　以下のように計算できる.

$$\begin{aligned} & f((x,k)\cdot(y,k')) \\ =\ & f(x,k+k') \\ =\ & (g(x),(k+k')h(x)) \\ =\ & (g(x),kh(x))\cdot(g(y),k'h(y)) \\ =\ & f(x,k)\cdot(y,k') \end{aligned}$$

$x<y$ **の場合**　以下のように計算できる.

$$\begin{aligned} & f((x,k)\cdot(y,k')) \\ =\ & f(y,k') \\ =\ & (g(y),k'h(x)) \\ =\ & (g(x),kh(x))\cdot(g(y),k'h(y)) \\ =\ & f(x,k)\cdot(y,k') \end{aligned}$$

$x>y$ **の場合**　$x<y$ の場合と同様.

よって f は半群 S の準同型である.

□

9 おわりに

9.1 まとめと今後の課題

各種セグメント木と代数構造との関係を見た. 3 種類のセグメント木と代数構造の関係について整理でき, またクエリの併用についての一般的な理論も得られたと言ってよいだろう.

今後の課題としては, 遅延伝播構造と名付けた代数構造を一般化あるいは特殊化した場合を調べることが挙げられる. つまり,

- 遅延伝播構造の議論においてモノイドや半群を利用していた部分を圏で置き換えた場合を調べる. セグメント木に関する代数構造としてより過不足のないものが得られるだろう.
- 遅延伝播構造についての議論を環や加群で置き換えた場合を調べる. 今回のものよりも良い性質や使いやすい定理が得られる可能性がある.

9.2 謝辞

この PDF を公開前に読み有益な指摘や提案をくれた熨斗袋さん[23], 光成滋生さん[24]に感謝いたします. 特に 8.14 節と 8.15 節は熨斗袋さんとの議論の結果として追加されました.

[23] https://twitter.com/noshi91

[24] https://twitter.com/herumi

TODO(eshiho): ここから下を確認して修正する.

付録 A　対応可能なクエリの事例集

利用頻度の高いであろう整数についてのクエリについてのみ書く.

A.1　通常のセグメント木で対応可能

以下では整数列 $a = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$ についてのクエリを考える. 更新クエリは共通であるので略記するが, 正確に書くと次のようなものである.

- 点更新: 与えられた i (ただし $i < n$) と $b \in S$ に対し, $a_i \leftarrow b$ と $O(\log n)$ で更新する

注意 A.1.1.
次の形の点更新クエリも常に処理可能である.

- 点関数適用更新: 与えられた i (ただし $i < n$) と関数 $f\colon S \to S$ に対し, $a_i \leftarrow f(a_i)$ と $O(\log n)$ で更新する

◇

例 A.1.2.
次のふたつのクエリは整数の加法についての半群 $(\mathbb{Z}, +)$ で実現可能である.

- 点更新
- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $a_l + a_{l+1} + \cdots + a_{r-1}$ を $O(\log n)$ で計算する

整数でなく有限体やベクトルなどでも同様である. ◇

例 A.1.3.
次のふたつのクエリは整数の乗法についての半群 $(\mathbb{Z}, \cdot)$ で実現可能である.

- 点更新
- 区間積取得: 与えられた l, r (ただし $l < r$) に対し, 積 $a_l \cdot a_{l+1} \cdot \cdots \cdot a_{r-1}$ を $O(\log n)$ で計算する

整数でなく有限体や行列などでも同様である. 実装の際はオーバーフローに注意すること.

◇

例 A.1.4.
次のふたつのクエリは整数と最小値についての半群 $(\mathbb{Z}, \min)$ で実現可能である.

- 点更新
- 区間最小値取得: 与えられた l, r (ただし $l < r$) に対し, 最小値 $\min\{ a_l, a_{l+1}, \ldots, a_{r-1} \}$ を $O(\log n)$ で計算する

最大値についても同様である. ◇

例 A.1.5.
次のふたつのクエリは整数の集合と自然数の集合の直積についての辞書順での全順序の最小値による半群 $(\mathbb{Z} \times \mathbb{N}, \min)$ で実現可能である.

- 点更新
- 区間最小値位置取得: 与えられた l, r (ただし $l < r$) に対し, 最小値の位置 $\operatorname{argmin}_{i \in [l,r)} a_i$ (ただし最小値が複数ある場合は添字が最も小さいもの) を $O(\log n)$ で計算する

ただし更新クエリは与えられた整数 b を添字との対 (b, i) に変換してから使う. 取得クエリはその結果の第 2 要素のみを見る.
最大値の位置についても同様である. ◇

例 A.1.6.
次のふたつのクエリは整数と最小値についての半群 $(\mathbb{Z}, \min)$ で実現可能である.

- 点更新
- 区間傾斜付き最小値取得: 与えられた l, r (ただし $l < r$) に対し, 最小値 $\min\{ a_l + l, a_{l+1} + l + 1, \ldots, a_{r-1} + r - 1 \}$ を $O(\log n)$ で計算する

ただし更新クエリは与えられた整数 b を傾斜の付けられたもの $b + i$ に変換してから使う.
最大値の場合, その位置の場合, 傾斜が異なるものの場合も同様である. 一般に添字 i のみに依存する任意の関数を傾斜とできる. ◇

A.2 双対セグメント木

以下では整数列 $a = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$ についてのクエリを考える. 取得クエリは共通であるので略記するが, 正確に書くと次のようなものである.

- 点取得: 与えられた i (ただし $i < n$) に対し, 要素 a_i を $O(\log n)$ で計算する

注意 A.2.1.
次の形の点更新クエリも常に処理可能である.

- 点更新: 与えられた i (ただし $i < n$) と整数 $b \in \mathbb{Z}$ に対し, $a_i \leftarrow b$ と $O(\log n)$ で更新する
- 点関数適用更新: 与えられた i (ただし $i < n$) と関数 $f\colon \mathbb{Z} \to \mathbb{Z}$ に対し, $a_i \leftarrow f(a_i)$ と $O(\log n)$ で更新する

◇

例 A.2.2.
次のふたつのクエリは整数と加法についてのモノイド $(\mathbb{Z}, +, 0)$ で実現可能である.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow a_i + b$ と $O(\log n)$ で更新する
- 点取得

整数でなく有限体やベクトルなどでも同様である. ◇

例 A.2.3.
次のふたつのクエリは整数と乗法についてのモノイド $(\mathbb{Z}, \cdot, 1)$ で実現可能である.

- 区間乗算更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow b \cdot a_i$ と $O(\log n)$ で更新する
- 点取得

整数でなく有限体や行列などでも同様である. ◇

例 A.2.4.
次のふたつのクエリは整数と最小値についてのモノイド $(\mathbb{Z} \cup \{ \infty \}, \min, \infty)$ で実現可

能である.

- 区間最小値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow \min\{ a_i, b \}$ と $O(\log n)$ で更新する
- 点取得

最大値についても同様である. ◇

例 A.2.5.

次のふたつのクエリは整数についての左零半群に単位元を付与してできるモノイド $(\mathbb{Z} \cup \{ * \}, \cdot, *)$ で実現可能である.

- 区間一様代入更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow b$ と $O(\log n)$ で更新する
- 点取得

◇

例 A.2.6.

次のよっつのクエリは 8.13 節の定理 8.13.1 で得られるモノイド $M_1 \bowtie M_2 \bowtie M_3$ で実現可能である.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow a_i + b$ と $O(\log n)$ で更新する
- 区間最小値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow \min\{ a_i, b \}$ と $O(\log n)$ で更新する
- 区間最大値更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow \max\{ a_i, b \}$ と $O(\log n)$ で更新する
- 区間一様代入更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $a_i \leftarrow b$ と $O(\log n)$ で更新する
- 点取得

◇

A.3 遅延伝播セグメント木

以下では整数列 $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{Z}^n$ についてのクエリを考える.

注意 A.3.1.
次の形の点更新クエリも常に処理可能である.

- 点更新: 与えられた i (ただし $i < n$) と整数 $y \in \mathbb{Z}$ に対し, $x_i \leftarrow y$ と $O(\log n)$ で更新する
- 点関数適用更新: 与えられた i (ただし $i < n$) と関数 $f\colon \mathbb{Z} \to \mathbb{Z}$ に対し, $x_i \leftarrow f(x_i)$ と $O(\log n)$ で更新する

◇

例 A.3.2.
次のふたつのクエリは整数と加法についてのモノイド $M = (\mathbb{Z}, +, 0)$ および半群 $S = (\mathbb{Z}, +)$ からなる遅延伝播構造 (M, S, φ) で実現可能である.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow x_i + b$ と $O(\log n)$ で更新する
- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $x_l + x_{l+1} + \cdots + x_{r-1}$ を $O(\log n)$ で計算する

◇

例 A.3.3.
次のふたつのクエリは整数と加法についてのモノイド $M = (\mathbb{Z}, +, 0)$ および整数と最小値についての半群 $S = (\mathbb{Z}, \min)$ からなる遅延伝播構造 (M, S, φ) で実現可能である.

- 区間加算更新: 与えられた l, r (ただし $l < r$) と $b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow x_i + b$ と $O(\log n)$ で更新する
- 区間最小値取得: 与えられた l, r (ただし $l < r$) に対し, 最小値 $\min\{ x_l, x_{l+1}, \ldots, x_{r-1} \}$ を $O(\log n)$ で計算する

最大値の場合も同様である. ◇

例 A.3.4.

次のふたつのクエリは一次関数と関数合成についてのモノイド $M = (\{ \lambda x.ax + b \mid a, b \in \mathbb{Z} \}, \circ, \mathrm{id})$ および整数と加法についての半群 $S = (\mathbb{Z}, +)$ と自然数と加法についての半群 $(\mathbb{N}, +)$ の直積半群からなる遅延伝播構造 $(M, S \times \mathbb{N}, \varphi)$ で実現可能である.

- 区間一次関数更新: 与えられた l, r (ただし $l < r$) と $a, b \in \mathbb{Z}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow ax_i + b$ と $O(\log n)$ で更新する
- 区間和取得: 与えられた l, r (ただし $l < r$) に対し, 和 $x_l + x_{l+1} + \cdots + x_{r-1}$ を $O(\log n)$ で計算する

ただし, 関数 $f = \lambda x.ax + b \in M$ を総和 x かつ長さ k の状態 $(x, k) \in S \times \mathbb{N}$ に作用させるには $\varphi(f)(x, k) = (ax + bk, k)$ とする.

これは例 A.3.2 の一般化である. また, $a = 0$ に固定して $\lambda x.b$ という関数を考えることで, 区間一様代入クエリとしても利用可能である. ◇

例 A.3.5.

次のふたつのクエリは一次関数と関数合成についてのモノイド $M = (\{ \lambda x.ax + b \mid a, b \in \mathbb{N} \}, \circ, \mathrm{id})$ および自然数と最小値についての半群 $S = (\mathbb{N}, \min)$ からなる遅延伝播構造 (M, S, φ) で実現可能である.

- 区間一次関数更新: 与えられた l, r (ただし $l < r$) と $a, b \in \mathbb{N}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow ax_i + b$ と $O(\log n)$ で更新する
- 区間最小値取得: 与えられた l, r (ただし $l < r$) に対し, 最小値 $\min \{ x_l, x_{l+1}, \ldots, x_{r-1} \}$ を $O(\log n)$ で計算する

負数を許すように拡張することも可能である. 最大値の場合も同様である.

これは例 A.3.3 の一般化である. また, $a = 0$ に固定して $\lambda x.b$ という関数を考えることで, 区間一様代入クエリとしても利用可能である. ◇

付録 B　競技プログラミングの問題による具体例

TODO(kimiyuki): 他にもよさげな例題を探す

B.1 ADD DIV MAX RESTORE

8.11 節における遅延伝播積を用いた議論の例題として, Japan Alumni Group Summer Camp 2018 Day 2 の I 問題である ADD DIV MAX RESTORE [10] を考えよう. これは次のような問題である.

問題 B.1.1.

長さ n の自然数列 $x = (x_0, x_1, \ldots, x_{n-1})$ が与えられる. この数列 x の初期状態の値を $(x_0^*, x_1^*, \ldots, x_{n-1}^*)$ として記憶しておく. この数列に対し以下の 4 種のクエリが合計 q 個与えられる. すべて順番に処理せよ.

0. 区間加算更新: 与えられた l, r (ただし $l < r$) と自然数 $a \in \mathbb{N}$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow x_i + a$ と更新する
1. 区間 (切り捨て) 除算更新: 与えられた l, r (ただし $l < r$) と正整数 $b \geq 1$ に対し, すべての $i \in [l, r)$ について $x_i \leftarrow \lfloor x_i / b \rfloor$ と更新する
2. 区間最大値取得: 与えられた l, r (ただし $l < r$) に対し, 最大値 $\max \{ x_l, x_{l+1}, \ldots, x_{r-1} \}$ を計算する
3. 区間復元更新: 与えられた l, r (ただし $l < r$) に対し, すべての $i \in [l, r)$ について $x_i \leftarrow x_i^*$ と更新する

◇

さて, これら 4 種のクエリは遅延伝播セグメント木で同時にそれぞれ $O(\log n)$ で扱える. これを見ていこう.

その議論は次の 3 点へと整理できる.

(1.) 区間加算更新クエリ, 区間除算更新クエリ, 区間最大値取得クエリは併用可能である
(2.) クエリの持ち方を工夫することでオーバーフローを回避できる
(3.) 区間復元更新クエリは $\lambda(x, y).((\text{if } r \text{ then } y \text{ else } x), y)$ と書ける. これも併用可能である

まず (1.) について. ふたつの更新クエリを $\mathrm{End}(\mathbb{N})$ の中で考える. 区間加算更新クエリに対応するモノイドは $M_+ = (\{ \lambda x.x + a \mid a \in \mathbb{N} \}, \circ, \mathrm{id})$ であり, 区間除算更新クエリに対応するモノイドは $M_/ = (\{ \lambda x.\lfloor x/b \rfloor \mid b \geq 1 \}, \circ, \mathrm{id})$ である. これらは自然数の加

法モノイド $(\mathbb{N}, +, 0)$ や乗法モノイド $(\mathbb{N}^{\times}, \cdot, 1)$ と見ることもできる．また, 区間最大値取得クエリに対応する半群は $S = (\mathbb{N}, \max)$ である．また, 以下のみっつが言える．

- 任意の $a \in \mathbb{N}$ と $x, y \in \mathbb{N}$ に対し, $\max\{ x, y \} + a = \max\{ x + a, y + a \}$ が成り立つ．
- 任意の $b \geq 1$ と $x, y \in \mathbb{N}$ に対し, $\lfloor \max\{ x, y \} / b \rfloor = \max\{ \lfloor x/b \rfloor, \lfloor y/b \rfloor \}$ が成り立つ．
- 任意の $a \in \mathbb{N}$ と $b \geq 1$ に対し, ある $a' \in \mathbb{N}$ と $b' \geq 1$ が存在して $(\lambda x.x + a) \circ (\lambda x.\lfloor x/b \rfloor) = (\lambda x.\lfloor x/b' \rfloor) \circ (\lambda x.x + a')$ が成り立つ．$b' = b$ かつ $a' = ab$ とすればよい．

これらと 8.2 節や 8.11 節での議論を併せると, 区間加算更新クエリと区間除算更新クエリと区間最大値取得クエリは併用可能であることが分かる．特に, モノイドの要素は $(\lambda x.\lfloor x/b \rfloor) \circ (\lambda x.x + a) = \lambda x.\lfloor (x + a)/b \rfloor$ という形になる．

次に (2.) について．区間除算更新クエリどうしの合成は $(\lambda x.\lfloor x/b_1 \rfloor) \circ (\lambda x.\lfloor x/b_2 \rfloor) = \lambda x.\lfloor x/(b_1 \cdot b_2) \rfloor$ という式によって可能であるが, これは除数の値が指数的に増える．また, モノイドの合成方法の影響で区間加算更新クエリの値も指数的に増えてしまう．この問題の解決のためには以下のふたつを行なえばよい．

(2.1.) モノイドの要素を $\lambda x.\lfloor (x + a)/b \rfloor$ でなく $a'b + c = a$ な a', c を使って $\lambda x.\lfloor (x + a')/b \rfloor + c$ とする

(2.2.) ある十分大きな定数 B を固定しておく．b の値が B より大きくなったとき b を B で置き換えて修正する

(2.1.) の結果の関数は $(\lambda x.x + c) \circ (\lambda x.\lfloor x/b \rfloor) \circ (\lambda x.x + a)$ とも書ける．(2.2.) は, 正確には $b \geq B$ な $\lambda x.\lfloor (x + a)/b \rfloor + c$ を $a' = B - (b - a)$ とおいて $\lambda x.\lfloor (x + a')/B \rfloor + c$ に修正する必要がある．これは b の値が B を越えたとき a の値も B を越えているなどの可能性があるためである．

最後に (3.) について．これは 8.14 節で議論したものである．区間復元更新クエリは素朴に書くと $r \in \{ 0, 1 \}$ とおいて $\lambda x.(\text{if } r \neq 0 \text{ then } x_i^* \text{ else } x)$ となるが, 関数の中に添字が含まれてしまうのが問題である．これは引数を増やし $\lambda(x, y).((\text{if } r \text{ then } y \text{ else } x), y)$ とすることで解決できる．ただし, 同時に半群の要素を単一の自然数 x から対 (x, x_i^*) へと修正する．

また, 以下のみっつが言える．

(3.1.) 任意の r と a, b, c に対し，ある r' と a', b', c' が存在して，$(\lambda(x, y).((\text{if } r \neq 0 \text{ then } y \text{ else } x), y)) \circ (\lfloor (x+a)/b \rfloor + c, y) = (\lfloor (x+a')/b' \rfloor + c', y) \circ (\lambda(x, y).((\text{if } r' \neq 0 \text{ then } y \text{ else } x), y))$ が成り立つ

(3.2.) $f = \lambda x.\lfloor (x+a)/b \rfloor + c$ とする．$y = \max_{i \in [l,r)} x_i^*$ ならば $f(y) = \max_{i \in [l,r)} f(x_i^*)$ である

(3.1.) は更新クエリとの併用可能性を言うものである．$r = 0$ ならば $\lambda(x, y).((\text{if } r \neq 0 \text{ then } y \text{ else } x), y) = \mathrm{id}$ であること，および，$r \neq 0$ ならば任意の f に対し $(\lambda(x, y).((\text{if } r \neq 0 \text{ then } y \text{ else } x), y)) \circ (\lambda(x, y).(f(x), y)) = \lambda(x, y).(y, y) = \mathrm{id} \circ (\lambda(x, y).((\text{if } r \neq 0 \text{ then } y \text{ else } x), y))$ であることによる．(3.2.) は取得クエリとの併用可能性を言うものである．詳細については省略する．

そしてこれら (1.), (2.), (3.) を合わせれば，この問題を解く遅延伝播セグメント木が得られる．具体的な実装例などについては省略する．なお，まとめてひとつの遅延伝播構造として見ると，$(\{\, \lambda(x, y).(\lfloor (\text{if } r \text{ then } y \text{ else } x) + a)/b \rfloor + c, y) \mid a, b, c, r \,\}, \mathbb{N}_{\max} \times \mathbb{N}_{\max}, \mathrm{id})$ という遅延伝播構造であると言える．ただし $\mathbb{N}_{\max}$ は自然数の最大値についてのモノイドである．あるいは $(\{\, \lambda(x, y).(x + c, y) \mid \dots \,\} \bowtie \{\, \lambda(x, y).(\lfloor x/b \rfloor, y) \mid \dots \,\} \bowtie \{\, \lambda(x, y).(x + a, y) \mid \dots \,\} \bowtie \{\, \lambda(x, y).(x, y), \lambda(x, y).(y, y) \,\}, \mathbb{N}_{\max} \times \mathbb{N}_{\max}, \mathrm{id})$ や $(\mathbb{N} \bowtie \mathbb{N}^{\times} \bowtie \mathbb{N} \bowtie \{\, 0, 1 \,\}, \mathbb{N}_{\max} \times \mathbb{N}_{\max}, \varphi)$ と書くこともできるだろう．

これを AC Library(v1.2)[11] を用いて実装するとプログラム 2 のようになる．実際の提出結果は `https://atcoder.jp/contests/jag2018summer-day2/submissions/17258754` から見られる．

プログラム 2　ADD DIV MAX RESTORE に対する AC Library を用いた解答例

```
1  #include <algorithm>
2  #include <cstdio>
3  #include <vector>
4  #include <atcoder/lazysegtree>
5  #define REP(i, n) for (int i = 0; (i) < (int)(n); ++ (i))
6
7  struct M {
8      bool restore;
9      long long rem;
10     long long div;
```

```
11      long long quo;
12  };
13
14  M cdot_M(M f, M g) {
15      if (f.restore) return f;
16      constexpr long long LIMIT = 1e9;
17      long long rem = (g.quo + f.rem) % f.div * g.div + g.
            rem;
18      long long div = f.div * g.div;
19      long long quo = (g.quo + f.rem) / f.div + f.quo;
20      quo += rem / div;
21      rem %= div;
22      if (div >= LIMIT) {
23          rem = std::max(0ll, LIMIT - (div - rem));
24          div = LIMIT;
25      }
26      assert (0 <= quo and quo <= 1e9);
27      assert (0 <= rem and rem < div);
28      assert (1 <= div and div <= LIMIT);
29      return { g.restore, rem, div, quo };
30  }
31
32  M e_M() {
33      return { false, 0, 1, 0 };
34  }
35
36  M make_add_query(long long x) {
37      return { false, 0, 1, x };
38  }
39
40  M make_div_query(long long x) {
```

```
41      return { false, 0, x, 0 };
42  }
43
44  M make_restore_query() {
45      return { true, 0, 1, 0 };
46  }
47
48  struct S {
49      long long fx;
50      long long x;
51  };
52
53  S cdot_S(S x, S y) {
54      return { std::max(x.fx, y.fx), std::max(x.x, y.x) };
55  }
56
57  S e_S() {
58      return { 0, 0 };
59  }
60
61  S make_value(long long a_i) {
62      return { a_i, a_i };
63  }
64
65  long long get_result(S x) {
66      return x.fx;
67  }
68
69  S varphi(M f, S x) {
70      if (f.restore) {
71          x.fx = x.x;
```

```
72      }
73      x.fx = (x.fx + f.rem) / f.div + f.quo;
74      return x;
75  }
76
77  int main() {
78      // input
79      int n, q;
80      scanf("%d%d", &n, &q);
81      std::vector<long long> a(n);
82      REP (i, n) {
83          scanf("%lld", &a[i]);
84      }
85
86      // initialize
87      std::vector<S> b(n);
88      REP (i, n) {
89          b[i] = make_value(a[i]);
90      }
91      atcoder::lazy_segtree<S, cdot_S, e_S, M, varphi,
            cdot_M, e_M> seg(b);
92
93      // query
94      while (q --) {
95          int t, l, r;
96          long long x;
97          scanf("%d%d%d%lld", &t, &l, &r, &x);
98          ++ r;
99          if (t == 0) {
100             seg.apply(l, r, make_add_query(x));
101         } else if (t == 1) {
```

```
102                 seg.apply(l, r, make_div_query(x));
103             } else if (t == 2) {
104                 long long ans = get_result(seg.prod(l, r));
105                 printf("%lld\n", ans);
106             } else if (t == 3) {
107                 seg.apply(l, r, make_restore_query());
108             } else {
109                 assert (false);
110             }
111         }
112         return 0;
113 }
```

参考文献

[1] 秋葉拓哉, 岩田陽一, 北川宜稔. プログラミングコンテストチャレンジブック [第 2 版] ～問題解決のアルゴリズム活用力とコーディングテクニックを鍛える～. マイナビ出版, 2012.

[2] Point Add Range Sum - Library Checker. https://judge.yosupo.jp/problem/point_add_range_sum, 2019.

[3] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *More Geometric Data Structures*, pp. 209–231. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[4] Bentley and Wood. An optimal worst case algorithm for reporting intersections of rectangles. *IEEE Transactions on Computers*, Vol. C-29, No. 7, pp. 571–577, 1980.

[5] Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, Vol. 17, No. 3, pp. 427–462, 1988.

[6] 双対セグメント木という概念について - うさぎ小屋. https://kimiyuki.net/blog/2019/02/22/dual-segment-tree/, 2019.

[7] データ構造と代数構造 - koba-e964 の日記. https://koba-e964.hatenablog.

com/entry/2016/12/14/214132, 2016.

[8] セグメント木の上に乗る構造はモノイドではなく圏である - うさぎ小屋. https://kimiyuki.net/blog/2018/12/06/categories-on-segment-tree/, 2018.

[9] jiry_2. A simple introduction to S̈egment tree beats¨ Codeforces. https://codeforces.com/blog/entry/57319, 2018.

[10] Japan Alumni Group Summer Camp 2018 Day 2. I - ADD DIV MAX RESTORE. https://atcoder.jp/contests/jag2018summer-day2/tasks/jag2018summer_day2_i, 2018.

[11] atcoder/ac-library: Atcoder library. https://github.com/atcoder/ac-library, 2020.