JOI 春合宿 Day 1
たのしいたのしい家庭菜園 解説

チューター(IOI2011,タイ) 城下慎也(@phidnight)

東京大学 理学部情報科学科 3 年

問題概要

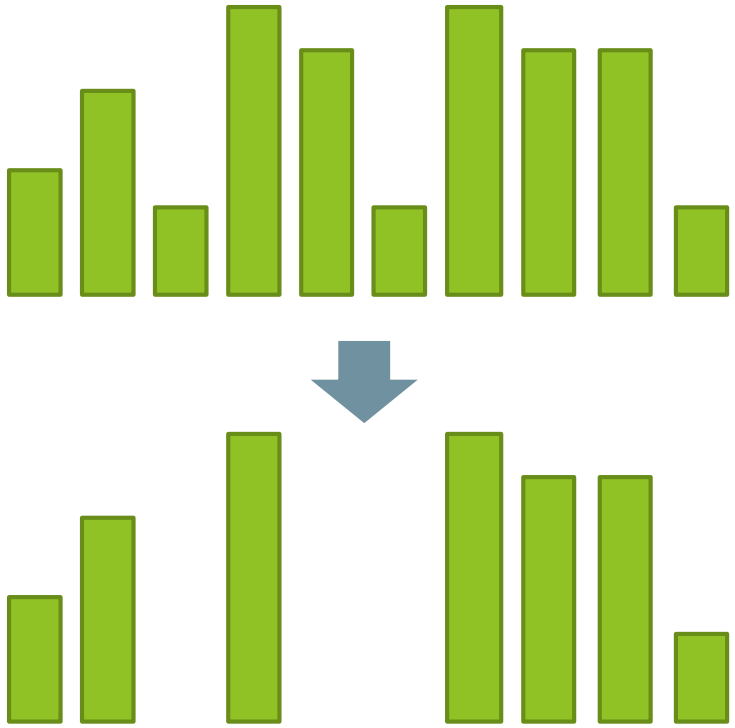
- N 本の IOI 草が一列に並んでいます。
- いくつかの IOI 草を抜くことによって、実を付けた IOI 草の合計価値から抜いた費用の合計を引いた値を最大化させてください。
- $3 \leq N \leq 100{,}000$
- その他：でかい

部分点解法 1

- どの IOI 草を抜くか、という組み合わせをすべて試します。
- 全部で 2^N 通り考えられるので、各組み合わせについて、どの IOI 草が実を付けるか、を探索します。
- 計算量は $O(N * 2^N)$ となり、小課題 1 が通ります。

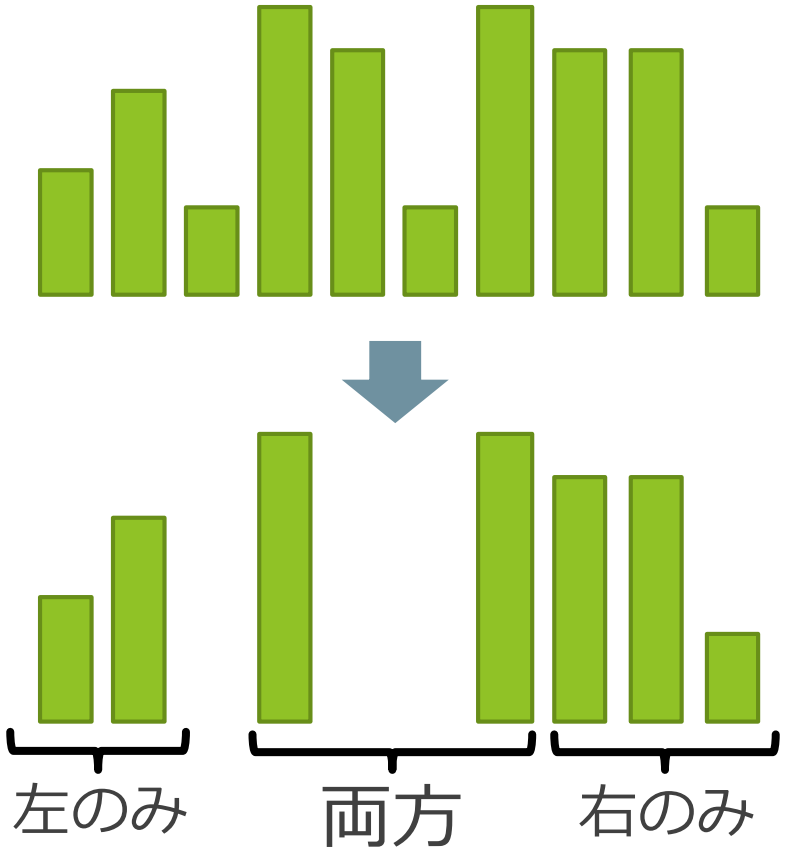
部分点解法 2,3

- 実をつける IOI 草について考えてみます。



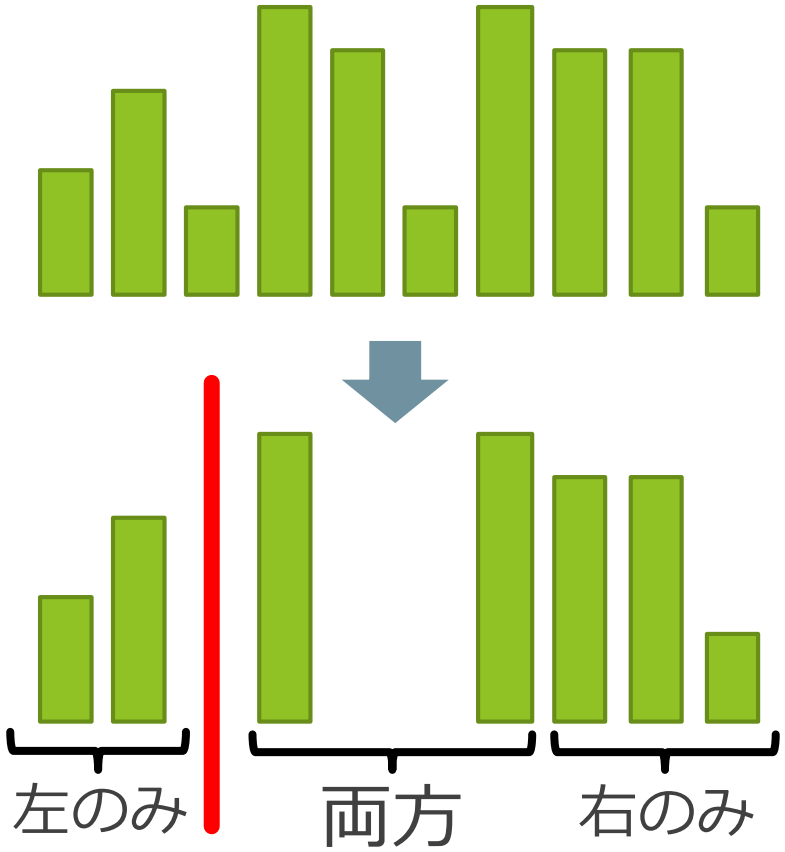
部分点解法 2,3

- 実をつける IOI 草について考えてみます。
  - 左図のように、どちらからの光が当たるかで分類できます。



部分点解法 2,3

- 実をつける IOI 草について考えてみます。



  - 左図のように、どちらからの光が当たるかで分類できます。
  - 「左のみ」と「その他」で分けて考えることにします。

部分点解法 2,3

- ある IOI 草 i について、「 IOI 草 i までで採用したものの高さが最大で j であるものの利益の最大値」を dpL[i][j] とおきます。
- 「左」を「右」に変えたものを dpR[i][j] と置きます。

部分点解法 2,3

- このとき、

dpL[i][j]=max(dpL[i-1][l]+P[i] (l ≦ j, H[i]＝j),

dpL[i-1][j] (H[i]≦j),

P[i]-CSUM(1,i-1) (i より前を抜く, H[i]＝j),

dpL[i-1][j]-C[i] (i を抜く))

となります (境界部分については省略)。

- これはO(N^2)で計算できます。

部分点解法 2,3

- dpR[i][j] も同様の方針で作ります。
- maxL[i]=max(dpL[i][j]) , maxR[i]=max(dpR[i][j])

とおくと、以下のうちの最大が答えとなります。

1. maxL[i] ($1 \leqq i \leqq N$)
2. maxR[i] ($1 \leqq i \leqq N$)
3. maxL[i]+maxR[i+1] ($1 \leqq i \leqq N-1$)

満点解法に向けて

- 先ほどの部分点解法 3 では、dp の状態数が最大 N^2 個あるので、すべての dp を 1 つ 1 つ求めていると O(N^2) よりは良くなりません。
- 最終的な答えは、maxL, maxR さえ求まっていれば計算できます。
- maxL, maxR を、dp の 1 つ 1 つを算出せずに(O(N^2) 費やさずに)求めたいです。

満点解法に向けて

- 配列 {dp[i][0],dp[i][1], … , dp[i][M]} (M は高さの上限) から、 {dp[i+1][0],dp[i+1][1], … , dp[i+1][M]} を求める方法について考えてみます。dp[i][j] (0 ≦ j ≦ M) を求めたいとき、

※dpL[i][j]=max(dpL[i-1][l]+P[i] (l ≦ j, H[i]＝j),
dpL[i-1][j] (H[i]≦j),
P[i]-CSUM(1,i-1) (i より前を抜く, H[i]＝j),
dpL[i-1][j]-C[i] (i を抜く))

満点解法に向けて

- dp[i][j] (0 ≦ j ≦ M) を求めたいとき、

第 2 式、第 4 式について、

- dpL[i-1][j] (H[i]≦j)→そのまま取ってくる。
- dpL[i-1][j]-C[i] → H[i]＞j で一律に C[i] 引く。

※dpL[i][j]=max(dpL[i-1][l]+P[i] (l ≦ j, H[i]＝j),
dpL[i-1][j] (H[i]≦j),
P[i]-CSUM(1,i-1) (i より前を抜く, H[i]＝j),
dpL[i-1][j]-C[i] (i を抜く))

満点解法に向けて

- dp[i][j] (0 ≦ j ≦ M) を求めたいとき、

第 1 式、第 3 式については、H[i]=j となるケースのみ考えればよいので、これらの計算は高々 N 回しかありません。

※dpL[i][j]=max(dpL[i-1][l]+P[i] (l ≦ j, H[i]＝j),
dpL[i-1][j] (H[i]≦j),
P[i]-CSUM(1,i-1) (i より前を抜く, H[i]＝j),
dpL[i-1][j]-C[i] (i を抜く))

満点解法に向けて

- 先ほどの計算では、

・ある値を変更する (第 1 式、第 3 式に登場)

・ある範囲を一律増減する (第 4 式に登場)

・ある範囲の最大値を求める

(第 1 式, maxL, maxR に登場)

が登場します。

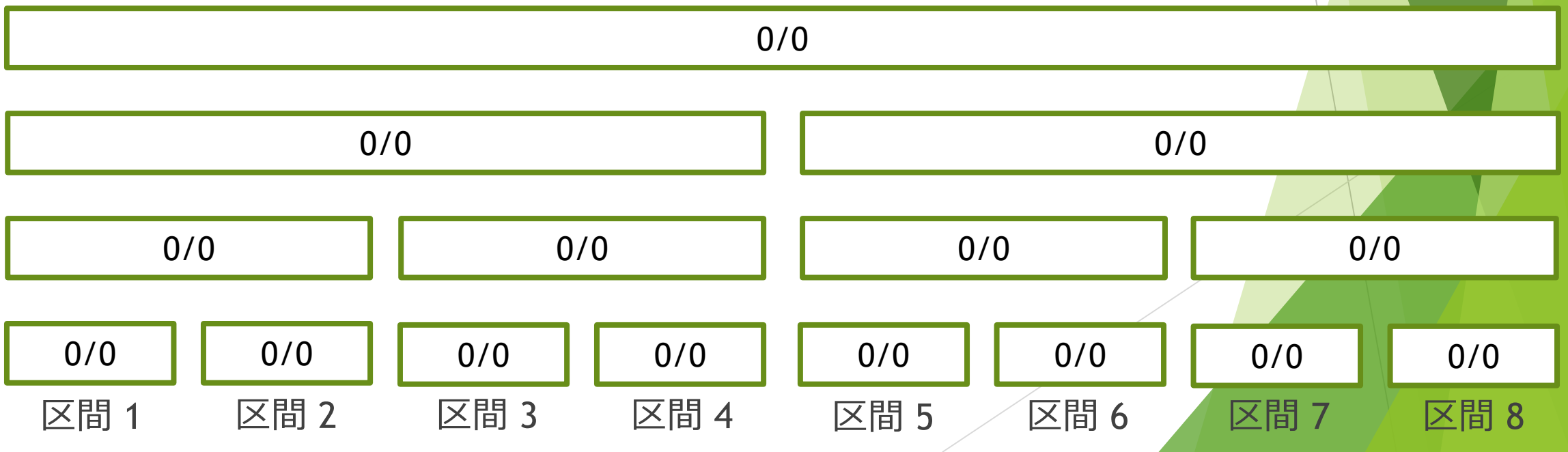
- これらを高速に実行するためには？

Segment Tree

- Segment Tree を使えば計算することができます。
- RMQ を計算する Segment Tree をベースにして計算することができます。
- 各ノードには、「その範囲の値 (その範囲全体に足された値については考慮しない)の最大値」「その範囲全体に加算されている値」を覚えておきます。
- まずは「範囲の増減」を例とともに考えます。

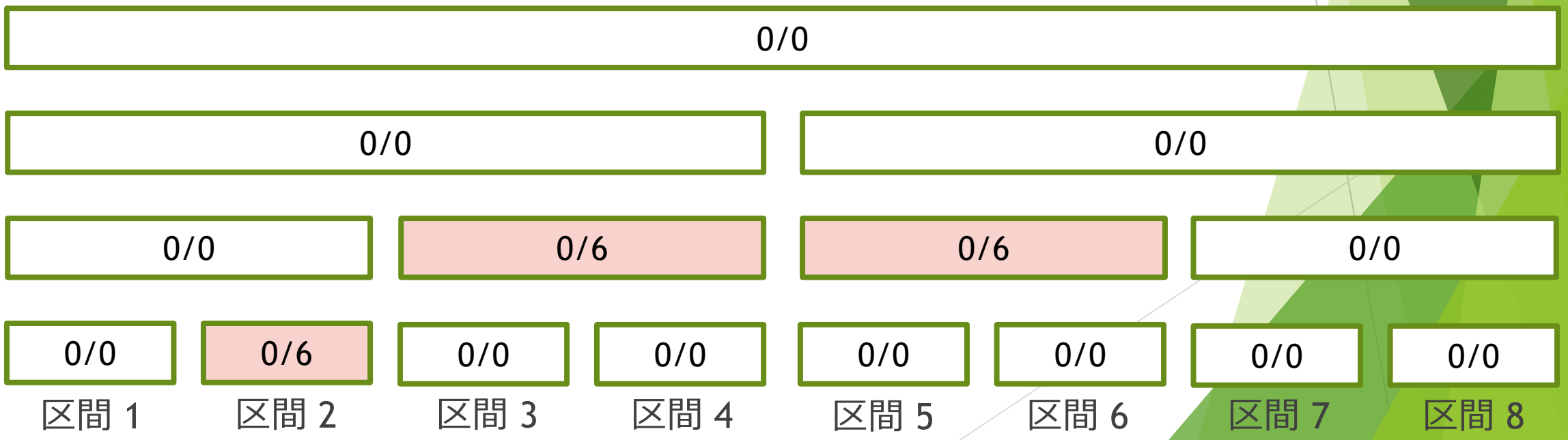
Segment Tree (例)

- 左辺が区間の最大値、右辺が区間全体に足された値です。
- 初期状態



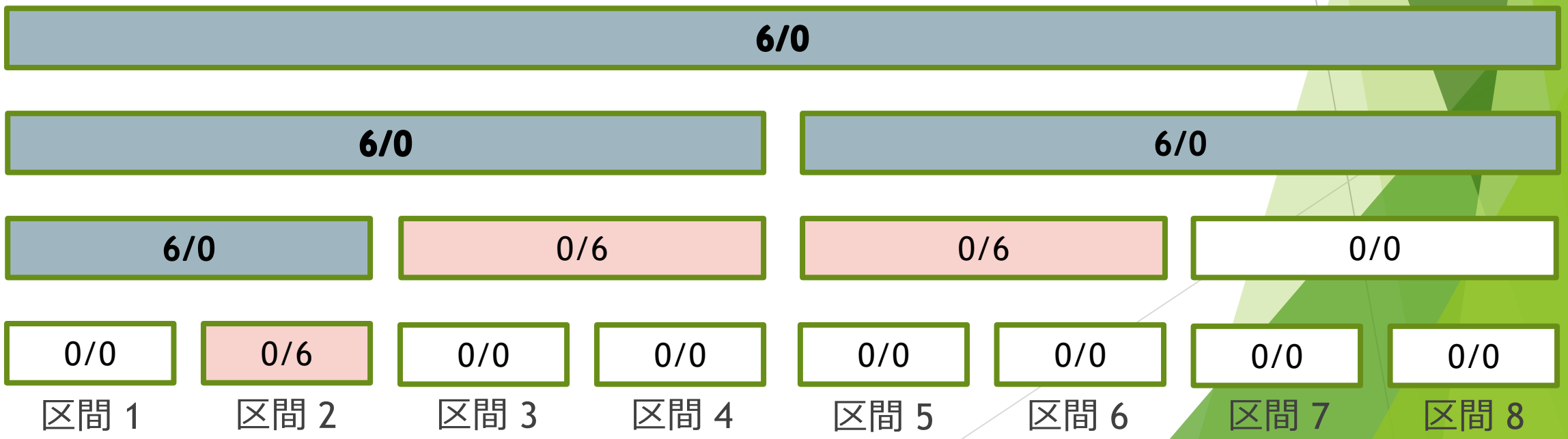
Segment Tree (例)

- 左辺が区間の最大値、右辺が区間全体に足された値です。
- 区間 [2,6] に定数 6 を足した後 (左辺未更新時)



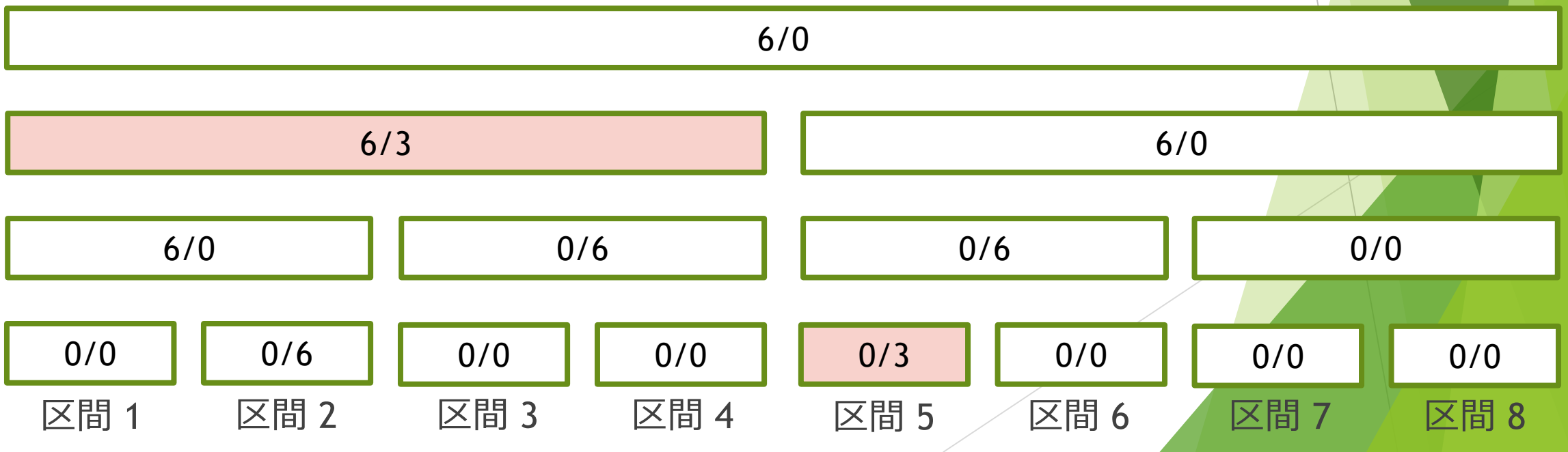
Segment Tree (例)

- 左辺が区間の最大値、右辺が区間全体に足された値です。
- 区間 [2,6] に定数 6 を足した後 (左辺更新後)



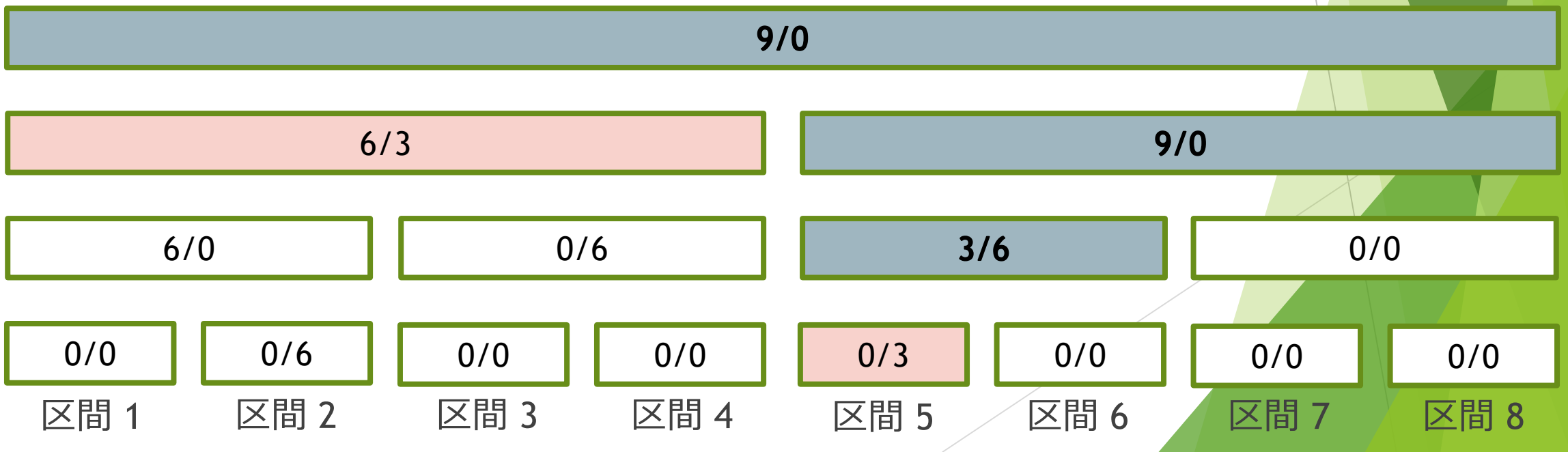
Segment Tree (例)

- 左辺が区間の最大値、右辺が区間全体に足された値です。
- さらに、区間 [1,5] に定数 3 を足した後(左辺未更新時)



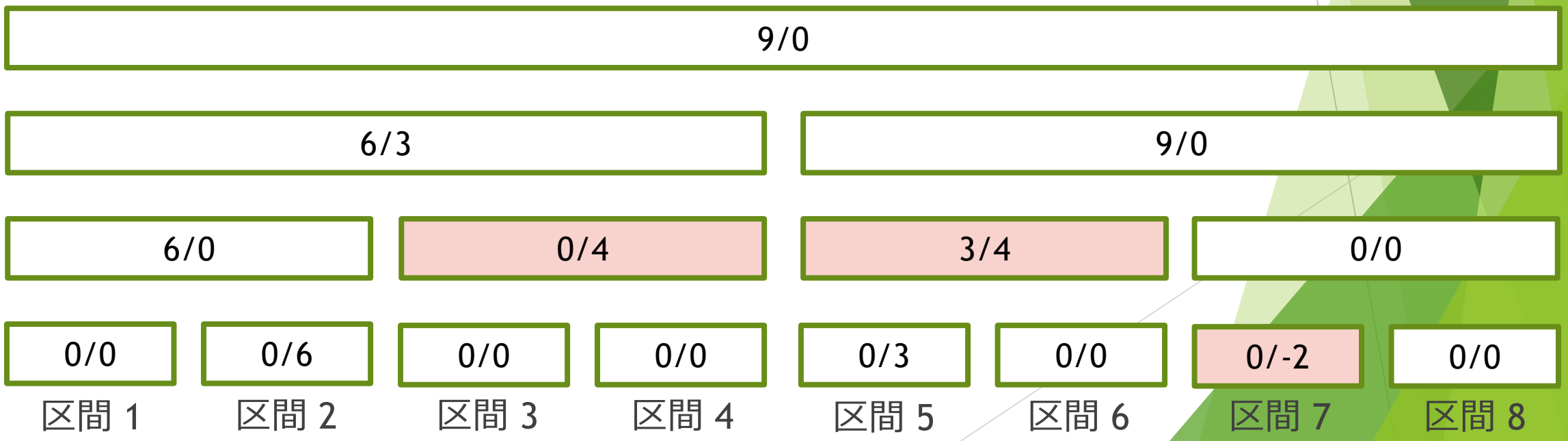
Segment Tree (例)

- 左辺が区間の最大値、右辺が区間全体に足された値です。
- さらに、区間 [1,5] に定数 3 を足した後(左辺更新後)



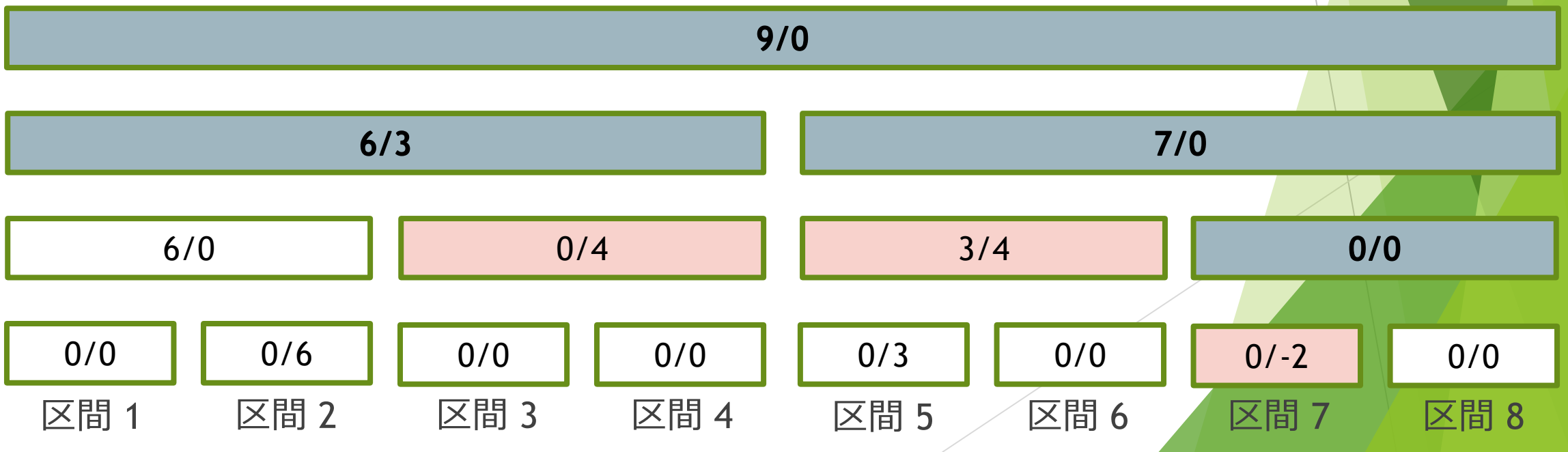
Segment Tree (例)

- 左辺が区間の最大値、右辺が区間全体に足された値です。
- さらに、区間 [3,7] に定数 -2 を足した後(左辺未更新時)



Segment Tree (例)

- 左辺が区間の最大値、右辺が区間全体に足された値です。
- さらに、区間 [3,7] に定数 -2 を足した後(左辺未更新時)



Segment Tree

- 例のように Segment Tree に対して、ある範囲に加減算させたいなら、

1. 「その範囲全体に加算されている値」を増減させる。
2. その後、変更した値よりも大きな範囲を囲むノードについて、 「その範囲の値の最大値」は、max(左の子のmax+左の子全体に足された数,
   右の子のmax+右の子全体に足された数)

で実現できます。

Segment Tree

- 次にある範囲の最大値を求める方法について考えます。
- 先ほどの加算のように、いくつかの区間に分割した後、

1. 小さい区間で最大値(計算済)を取り出す。
2. その区間以上の(覆っている)区間全体に足されている値の合計を求め、それと 1. の値を足す。
3. 2. で求めた値たちの最大値が答え。

Segment Tree の計算量

- 値に足す操作では、$O(\log N)$個のノードについて、直接加算が行われ、それ以上のノード(計算すると$O(\log N)$個しかないことが分かる)について、更新する必要があります。
- 値を求める場合、上から再帰関数を実装すれば $O(\log N)$ のノードアクセスで実現できます。
- 以上より dp[i][] から dp[i+1][] の配列を算出するのに $O(\log N)$ のコストで実現できます。
- maxL, maxR は全体の max を取ります。

おまけ

- 満点解法には、H をソートした順にアクセスするようにするという方針があります。
- この場合、先ほどの立式の高さに関する部分を気にしないで実装することができます。
- ただし、この方針だと同じ高さが複数ある場合にバグる可能性があります。
- 小課題 4 のみ通った場合は、この辺りが怪しいなどの情報が得られます。

得点分布