JOI/JOIG 2024/2025 春季トレーニング
Day 4 「ういろう」

解説担当：西本将樹（maspy）

問題概要

- 変数 x が $x = 0$ で初期化されている.
- $i = 1, 2, \ldots, N$ の順に，$a = A[i]$ として次のどちらかを行う.
  - $x := x + a$.
  - $x \geq a$ のとき，$x := x - a$ として 1 点を獲得.
- 最大何点獲得できますか？
- これを Q 個の区間で解いてください.

小課題 1 ($N \leq 20, Q \leq 20, A \leq 100$)

- 操作列は高々 2^N 通り．これらを全探索する.
- 実装方針の例：
  - 2^N 通りすべてに対してシミュレーション.
  - 1 手ずつ dfs.
- 計算量：$O(Q \times N \cdot 2^N)$ や $O(Q + N \cdot 2^N)$.
  - $\sum_{L,R}(R-L)2^{R-L} = O(N \cdot 2^N)$
- 小課題 1：JOI 3 点，JOIG 7 点.

小課題 2 ($N \leq 300, Q \leq 20, A \leq 100$)

- 愚直なシミュレーションにおいて，必要な情報だけ残す.
- 操作列の履歴は必要ない，x と獲得スコアだけが問題.
- **$\mathrm{dp}[i]$：ある時点で $x = i$ である場合の獲得スコア最大値**
  というテーブルを 1 手ずつ更新していく.
  - $\mathrm{dp}[i][j]$：i 枚終了時点で $x = j$ である場合の獲得スコア最大値.
    というテーブルを作っても構いません． 時間・空間が増えるし扱うべき添字が増えるので，個人的には推奨しません.
- 計算量：$O(Q \times N^2 A)$.
- 小課題 $1 + 2$：JOI 8 点，JOIG 18 点.

小課題 3 ($N \leq 5000, Q \leq 20, A \leq 100$)

- 愚直なシミュレーションにおいて，必要な情報だけ残す.
- 操作列の履歴は必要ない，x と獲得スコアだけが問題.
- x の範囲：$[0, NA]$.　獲得スコアの範囲：$[0, N]$.

小課題 3 ($N \leq 5000, Q \leq 20, A \leq 100$)

- 愚直なシミュレーションにおいて，必要な情報だけ残す.
- 操作列の履歴は必要ない，x と獲得スコアだけが問題.
- x の範囲：$[0, NA]$.　獲得スコアの範囲：$[0, N]$
- $\mathrm{dp}[i]$：**ある時点で獲得スコア i である場合の x としてありうる最大値**
- 計算量：$O(Q \times N^2)$.
- 小課題 $1 + 2 + 3$：JOI 15 点，JOIG 33 点.

小課題 4 ($N \leq 2 \times 10^5, Q \leq 20, A \leq 100$)

- 全探索的な方法（DP を含む）では難しいと思います.
- 適切な貪欲法を考えます.
- $i = 1, 2, 3, \ldots$ の順に進めながら, 「その時点での最適な状態」を持ちます.

小課題 4 ($N \leq 2 \times 10^5, Q \leq 20, A \leq 100$)

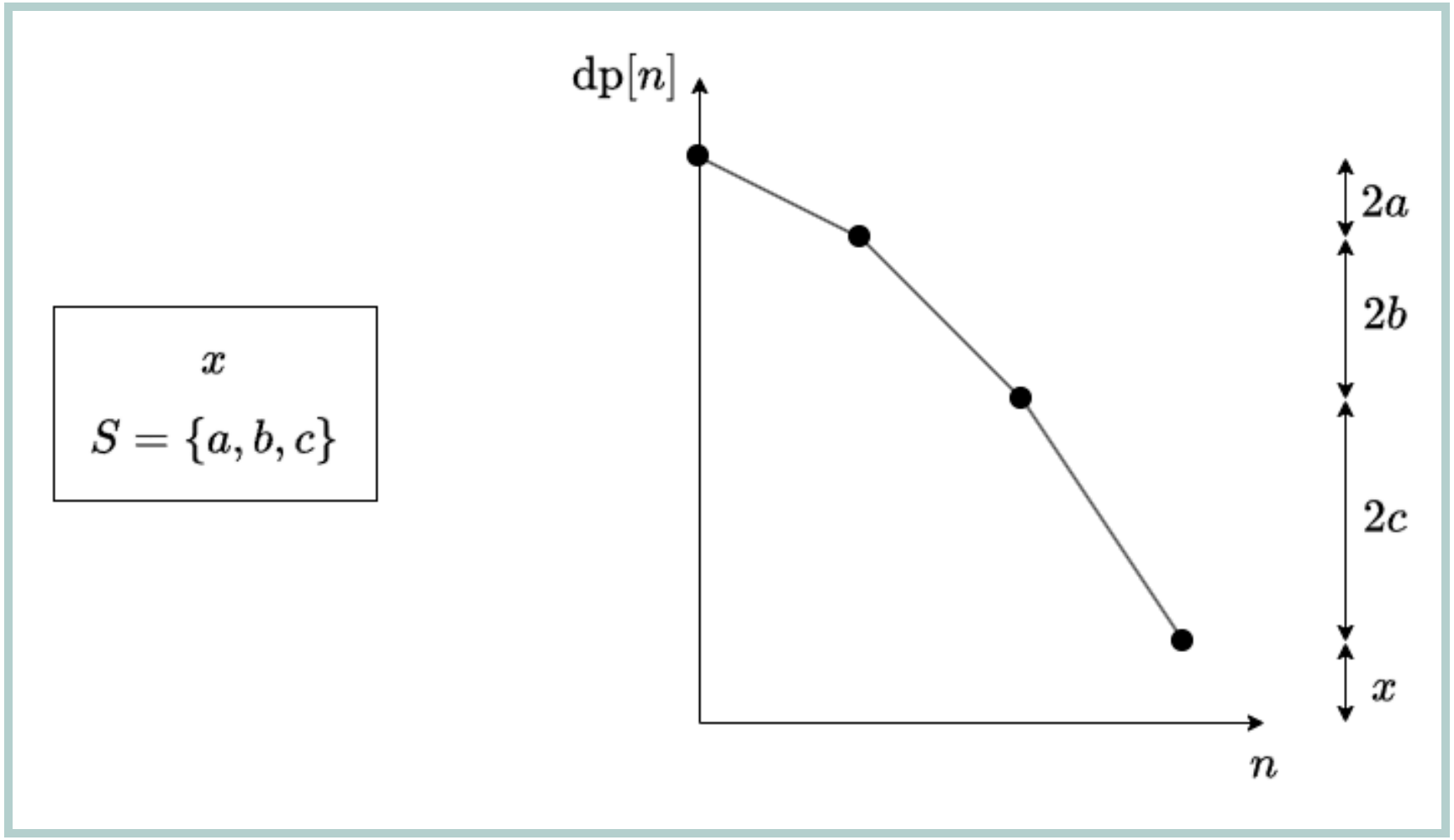
- 状態として
  - x の値
  - 多重集合 S：今まで減算で使った値の多重集合
- 「最適な状態」とはそれ以降何が来ても最適解が得られる状態のことだとします.
- ただしその際に，**「今まで減算で使った値をやっぱり加算にします」という操作を許します.**

小課題 4 ($N \leq 2 \times 10^5, Q \leq 20, A \leq 100$)

- $a = A[i]$ として，次の貪欲アルゴリズムが正当です.
  - $a \leq x$ ならば $x := x - a, S := S \cup \{a\}$.
  - $a > x$ かつ $b = \max S > a$ ならば $x := x + 2 \times b - a$, $S := S - \{b\} \cup \{a\}$.
  - $a > x$ かつ $b \leq a$ ならば $x := x + a$.
- $\max S$ を取り出すために priority queue を使って実装すると $O(N \log N)$ 時間.

小課題 4 ($N \leq 2 \times 10^5, Q \leq 20, A \leq 100$)

- なお, この貪欲法は小課題 3 で説明した dp テーブルと対応があります.



小課題 4 ($N \leq 2 \times 10^5, Q \leq 20, A \leq 100$)

- 小課題 3 の dp 更新は次の形.
  - $g(x) = \max(f(x) + a, f(x-1) - a)$.
  - ただし $g(x) < 0$ になりそうなら補正.
- 前者の操作は max plus convolution.
  - $g(x) = \max_{y+z=x}(f(y) + h(z))$
  - $h(x) := \begin{cases} a & (x = 0) \\ -a & (x = 1) \\ -\infty & (\text{otherwise}) \end{cases}$

小課題 4 ($N \leq 2 \times 10^5, Q \leq 20, A \leq 100$)

- dp テーブルは常に上凸になる.
- 上凸関数の max plus convolution は傾き集合のマージ（この場合ある傾きをひとつ挿入）.
- 「dp テーブルは実は上凸です」という問題では，その理由が max plus convolution で説明できることが多い.
- 凸関数の扱いが上手いと，小課題 3 の dp 高速化として直接小課題 4 の解法を得ることもできる.
- 小課題 $1 + 2 + 3 + 4$：JOI 30 点，JOIG 57 点

小課題 5 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 2$)

- ここから Q が増えます.
- 小課題 5 は $A \leq 2$ で，やや特殊な解法が可能です.

小課題 5 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 2$)

- 小課題 4 の貪欲を考えると，priority queue の中身としては，2 の個数 y だけを持てばよいです.
- 状態は，(x, y) $(0 \leq x < 4)$ という 2 つの整数組で書けて，各要素はこれに対する関数です.
- **関数合成をセグメント木にのせましょう.**
- 合成として現れる関数を $O(1)$ サイズのデータとして書ければ OK.

小課題 5 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 2$)

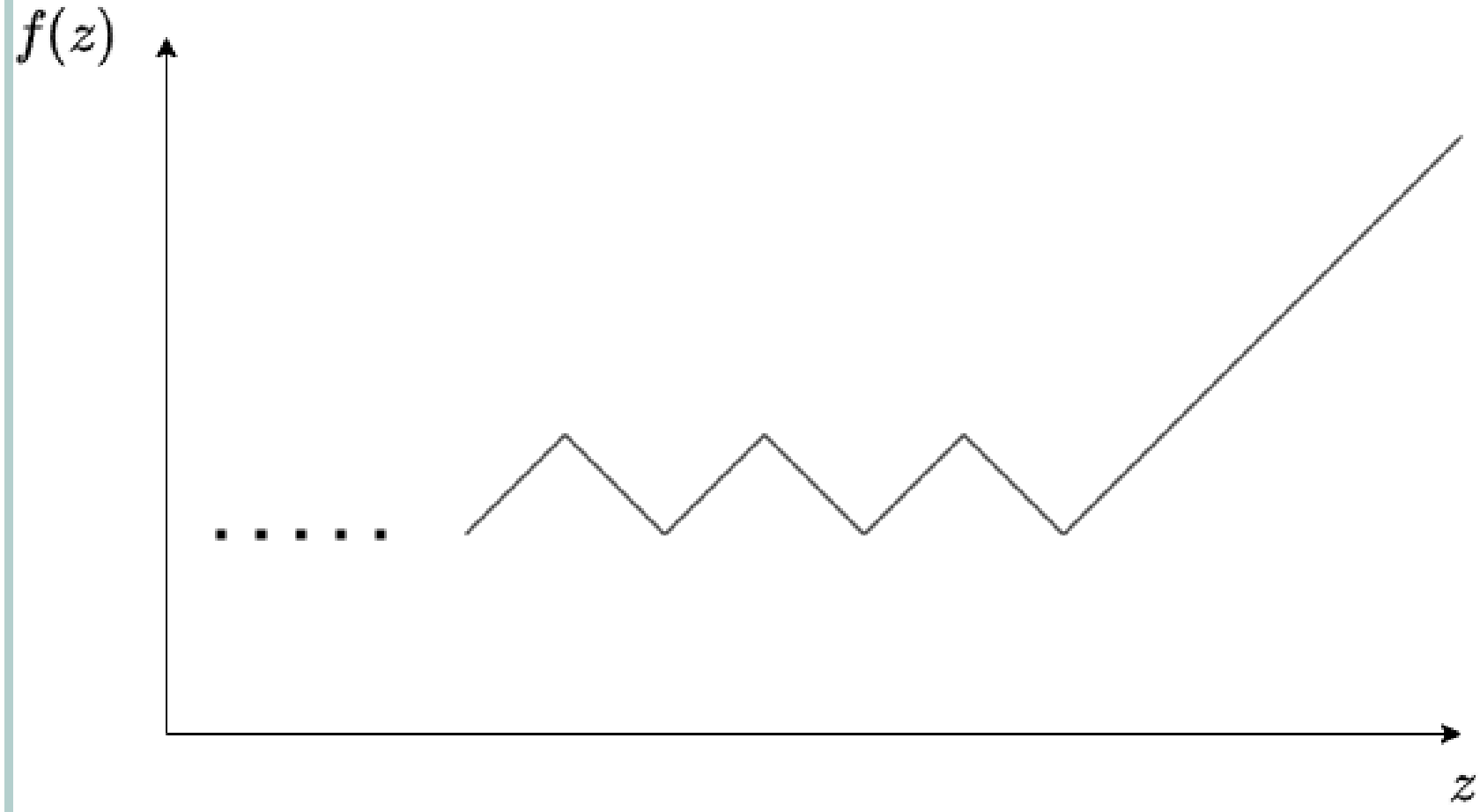
- 関数は $A[i] = 1, 2$ に応じて次の形です.
- $A[i] = 1$:
  - $1 \leq x < 4$ ならば $(x, y) \longmapsto (x - 1, y)$.
  - $x = 0$ かつ $y \geq 1$ ならば $(x, y) \longmapsto (3, y - 1)$.
  - $x = 0$ かつ $y = 0$ ならば $(x, y) \longmapsto (1, 0)$.
- $A[i] = 2$:
  - $0 \leq x \leq 1$ ならば $(x, y) \longmapsto (x + 2, y)$.
  - $x \geq 2$ ならば $(x, y) \longmapsto (x - 2, y + 1)$

小課題 5 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 2$)

- $z := x + 4y$ とするとかなり簡単になります.
- $A[i] = 1$:
  - $1 \leq z$ ならば $z \longmapsto z - 1$.
  - $z = 0$ ならば $z \longmapsto 1$.
- $A[i] = 2$:
  - $z \longmapsto z + 2$.

小課題 5 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 2$)

- 結局，こんな感じの関数になります.



小課題 5 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 2$)

- これをセグメント木にのせると解けます．$O(N + Q \log N)$ 時間.
- $x = 0, 1, 2, 3$ それぞれの場合の関数を持つことも考えられますが，上の方法よりは実装が複雑になると思います.
- 小課題 $1 + 2 + 3 + 4 + 5$：JOI 51 点，JOIG 78 点.

満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

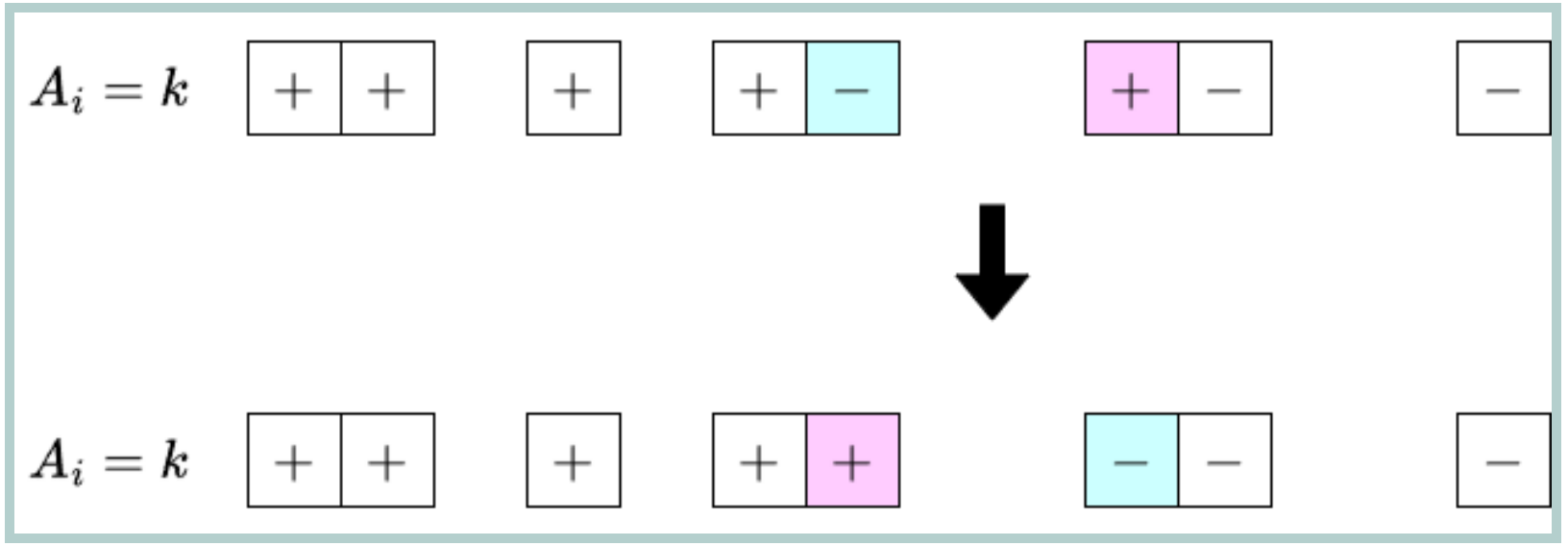
- 小課題 $6, 7$ の制約に大きな違いはないので，まとめて解説します.
- 小課題 4 の貪欲法では途中の状態のパターンが多すぎて難しいと思います.
- 全く別の貪欲法を考えます.
- A の種類数が少ないことに注目します.
- $A = 1, 2, \ldots, 100$ の順に何かを計算するという 100 ステップの計算で答えを求めることを考えます.

満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

- k を固定したとき，$A[i] = k$ であるような i 全体に対する戦略を決められそうか？
- 次を満たす p が存在するとしてよいです.
  - **$i < p$ ならば $A[i] = k$ を加算で使う.**
  - **$p \leq i$ ならば $A[i] = k$ を減算で使う.**
- 証明は簡単な exchange argument.

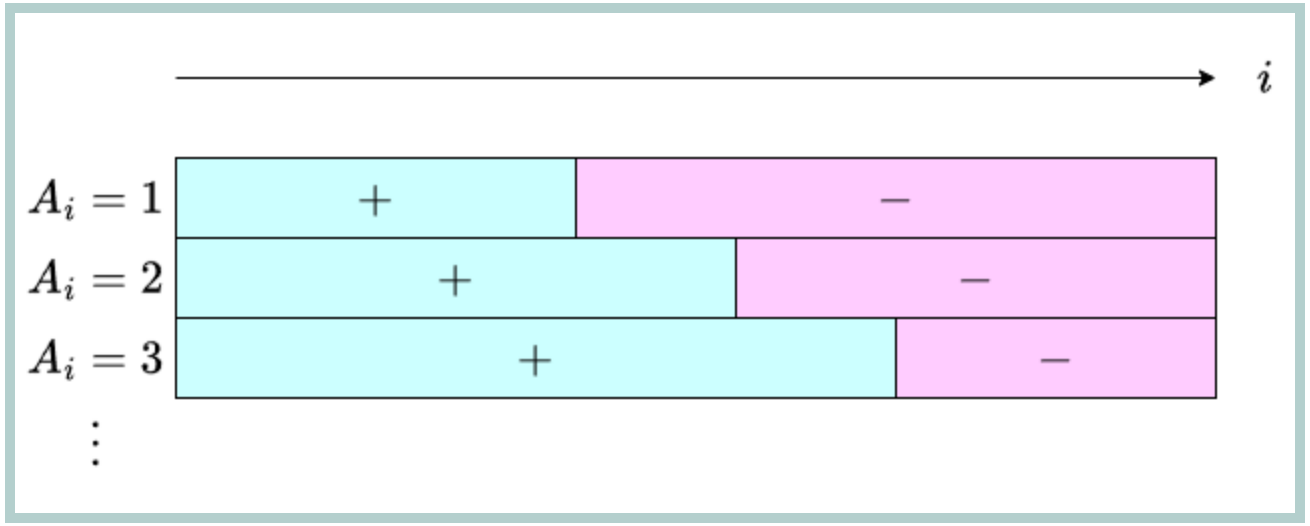
満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

- **exchange argument**：（性質）を満たさない解があったとき,
  こういう入れ替えを行っても解は悪くならない. これを可能な限り繰り返したら（性質）を満たす解に到達する.
  よって最適解であって（性質）を満たすものが存在する.



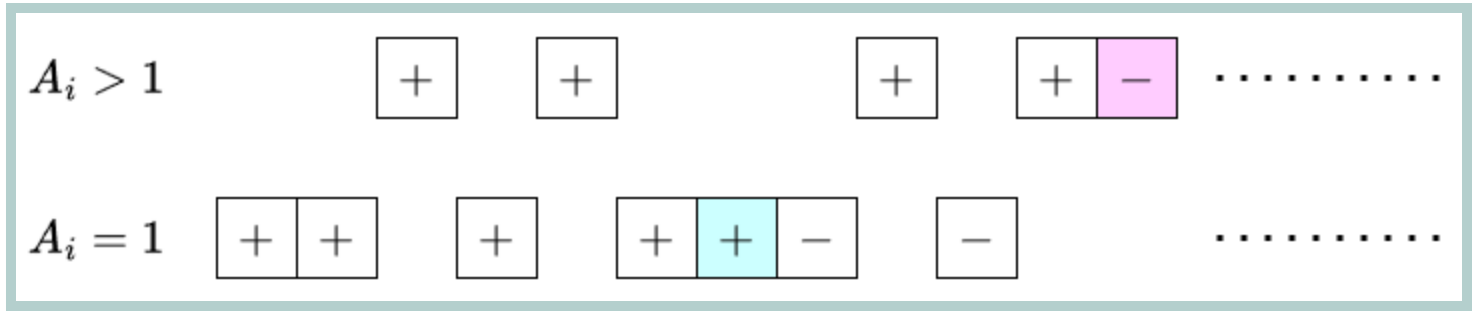
満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

- 上と全く同様に，最適解に次を仮定できることが分かります：
  - $a > b, A[i] = a, A[j] = b$ のとき，
    $A[j]$ を加算で使うならば $A[i]$ も加算で使う．
- したがって，**境界の位置は単調になります．** これらの境界を
  $k = 1, 2, \ldots, 100$ の順にすべて求めるというのが解法の概要です．



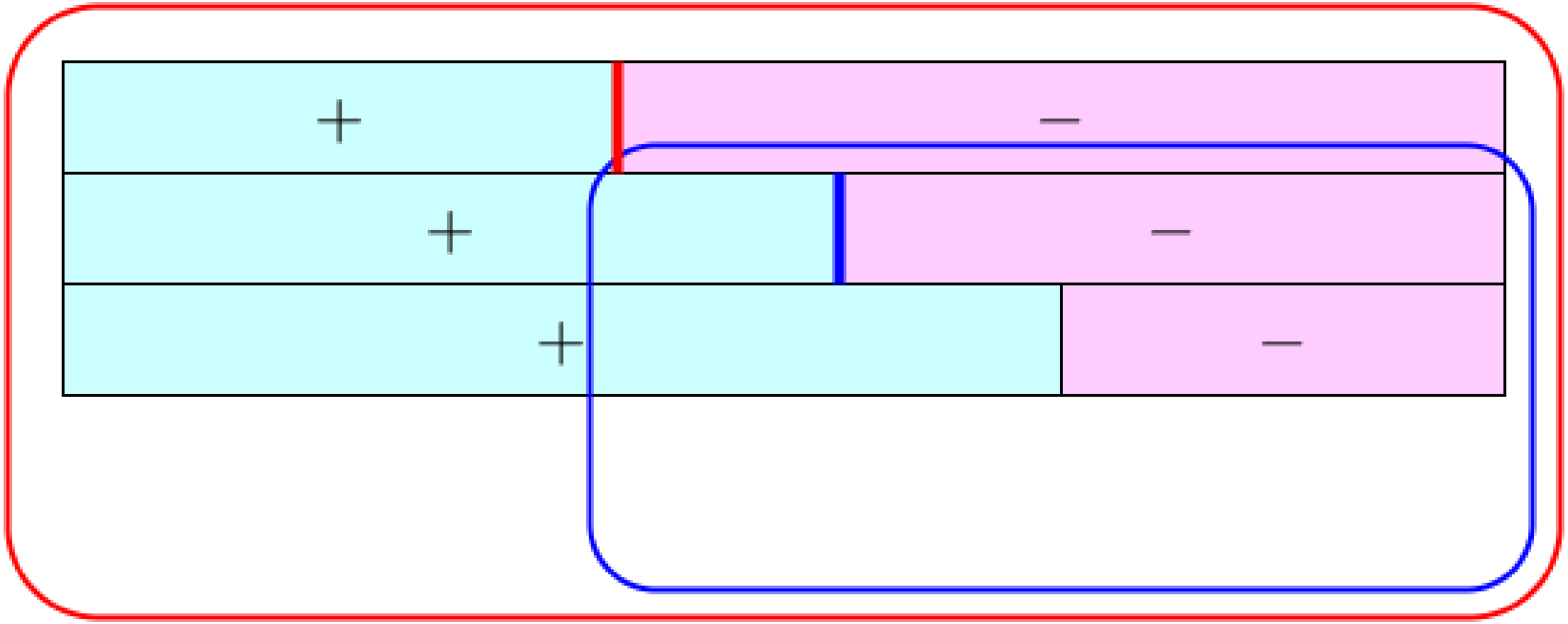
満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

- 境界を $k = 1, 2, 3, \ldots$ の順に決めていきます.
- $k = 1$ の場合，次が証明可能です：
  - **$A[i] = 1$ を加算で使う回数は最小化してよい.** つまり，
    $A[i] > 1$ がすべて加算である場合に 1 の最小加算回数が n 回ならば，最適解であって 1 の加算回数 n 回のものが存在.
- やっぱり exchange argument で証明可能です.
  （最小値の加算と最小値以外の最初の減算の入れ替え）



満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

- 解法をまとめると次のようになります.
- $A[i] = 1$ である i で最後に加算するものの場所 p を求める.
  - ある場所以降をすべて減算にすると矛盾するような p の最大値.
- 場所 p までは（$A[i] \geq 2$ を含めて）すべて加算で確定します.
  場所 p 以降は $A[i] = 1$ についてはすべて減算で確定します.
- 場所 p 以降で，$A[i] \geq 2$ に対する戦略を決める問題を解きます.
  $A[i] = 2$ であるような場所の境界を同様に決めて，再帰的に処理します.



満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

- 各段階では，「k 未満はすべて減算，$k+1$ 以上はすべて加算」という前提で，k をいくつ減算できるかという問題を解きます.
- 各 k に対してそれを行うためのデータ構造をひとつ持ちます.
- 必要な計算を整理すると，静的な列に対する range minimum query があればよいことが分かります.
- 例えばセグメント木で $O(AN + QA \log N)$ 時間で解けます.
- 小課題 $1+2+3+4+5+6$：JOI 80 点，JOIG 92 点.
- 小課題 $1+2+3+4+5+6+7$：JOI 100 点，JOIG 100 点.

満点 ($N \leq 2 \times 10^5, Q \leq 2 \times 10^5, A \leq 100$)

どうしても TLE してしまう場合，以下のような手段が考えられます.

- すべてのクエリを並列に処理すれば，同時に持つべきセグメント木は 1 つでよい.
- 静的な列に対する offline range minimum query をセグメント木よりも高速に行う.
  - 例えば，j を増やしながら $ans[i] = \min\{a_i, \ldots, a_j\}$ を管理すれば UnionFind で処理できる.
- いずれも C++ で満点をとるには不要だとは思います.

統計情報 (JOI)

得点	人数	累積
100	2	2
80	1	3
51	2	5
30, 36	6	11
3, 8, 15	12	23
0	4	27

統計情報 (JOIG)

得点	人数	累積
33	5	5
18	1	6
7	5	11
0	2	13