解説: 平木 康傑

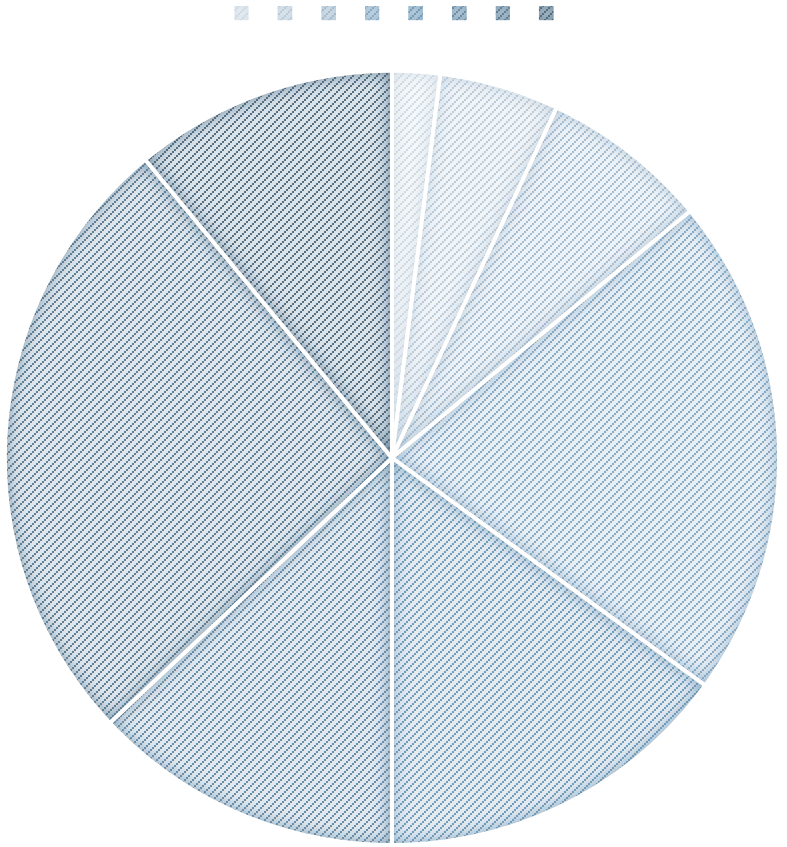
JOI 2021 春合宿 Day1 – Food Court

問題概要

- N 個の待機列 (queue) に整数を出し入れするクエリが時系列順に Q 個ある
- 以下の 3 種類
  - 「加入」：待機列 L~R に整数 C を K 個ずつ push
  - 「脱退」：待機列 L~R から K 個ずつ pop (K 個に満たなければ空になる)
  - 「サービス」：待機列 A の要素数が B 個以上か判定し，そうなら B 番目を出力

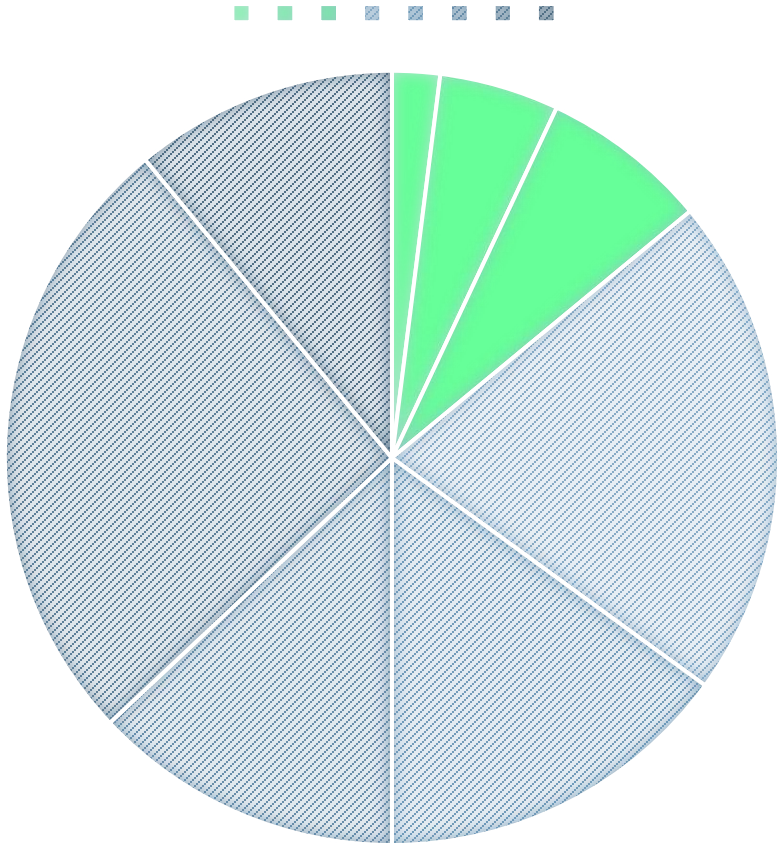
小課題

- (2 点) $O(NQ), K_i = 1$
- (5 点) $O(NQ)$
- (7 点) push クエリは狭い範囲に 1 個ずつ
- (21 点) $M = 1$
- (15 点) $K_i = 1$
- (13 点) pop なし
- (26 点) $\log^2$
- (11 点) $\log^1$

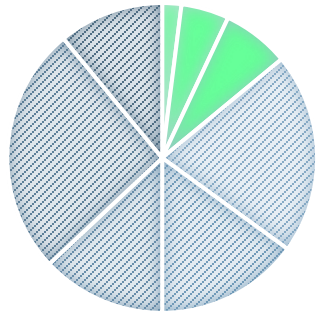


小課題

- ~~(2 点) $O(NQ), K_i = 1$~~ ✓
- ~~(5 点) $O(NQ)$~~ ✓
- ~~(7 点) push クエリは狭い範囲に 1 個ずつ~~ ✓
- (21 点) $M = 1$
- (15 点) $K_i = 1$
- (13 点) pop なし
- (26 点) $\log^2$
- (11 点) $\log^1$

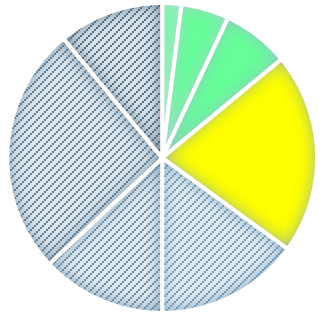


解法の方針



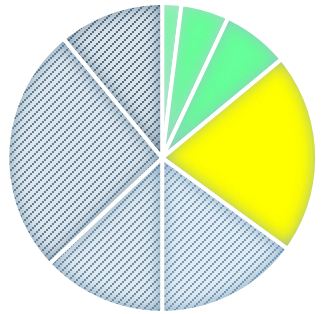
- どの小課題も糸口として使えるし，最終的な解法も多様
- 予告: 2 通りの解法を示す

小課題 4 の考察: $M = 1$



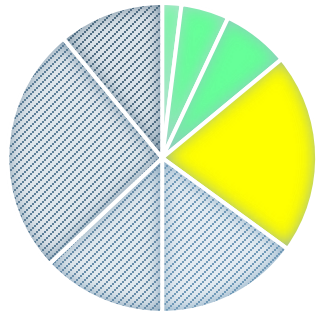
- 出し入れする整数は全て 1
- 「サービス」クエリにおいて,
  待機列 A の要素数が B 以上なら 1, そうでなければ 0 が答え
- → クエリ時点での待機列 A の要素数がわかればよい

小課題 4 の考察: 要素数を管理する



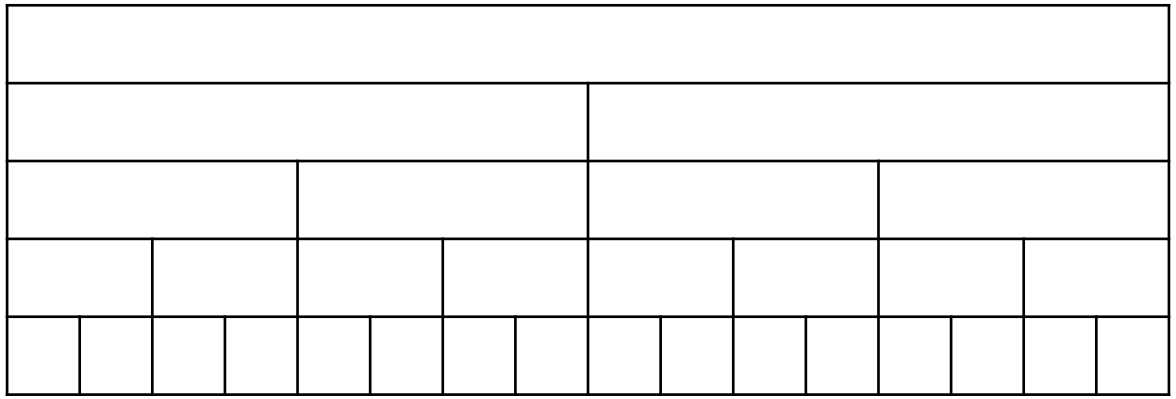
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - 「脱退’」：区間[L, R] から K を減算し，0 との max をとる
  - 「サービス’」：位置[A] の値を求める

小課題 4 の考察: 要素数を管理する

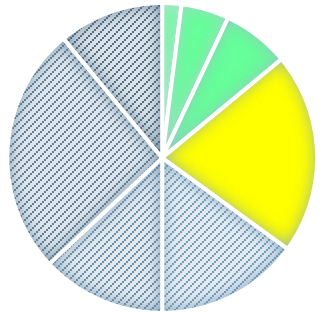


- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」: 区間[L, R] に K を加算
  - ~~「脱退’」: 区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」: 位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」

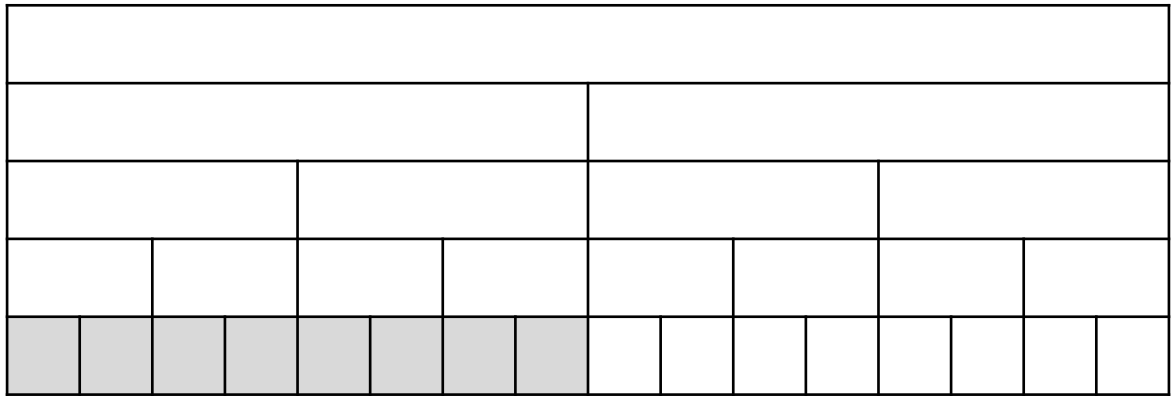


小課題 4 の考察: 要素数を管理する



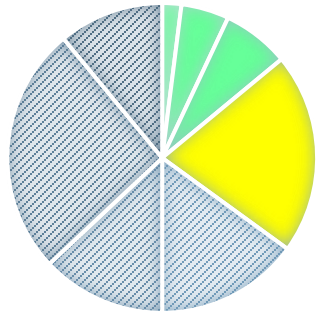
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - ~~「脱退’」：区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」：位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



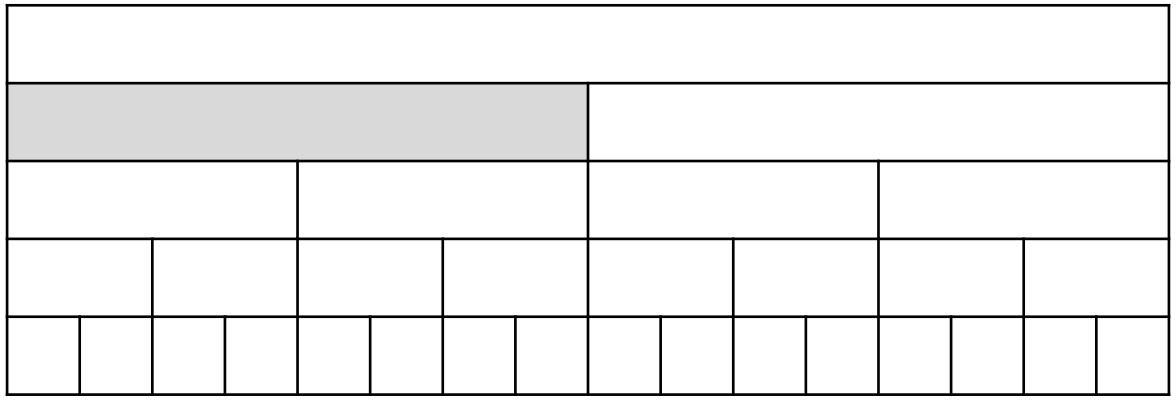
区間[1, 8] に 4 を加算

小課題 4 の考察: 要素数を管理する



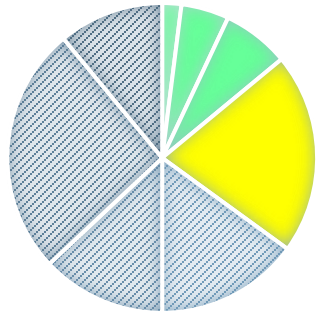
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - ~~「脱退’」：区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」：位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



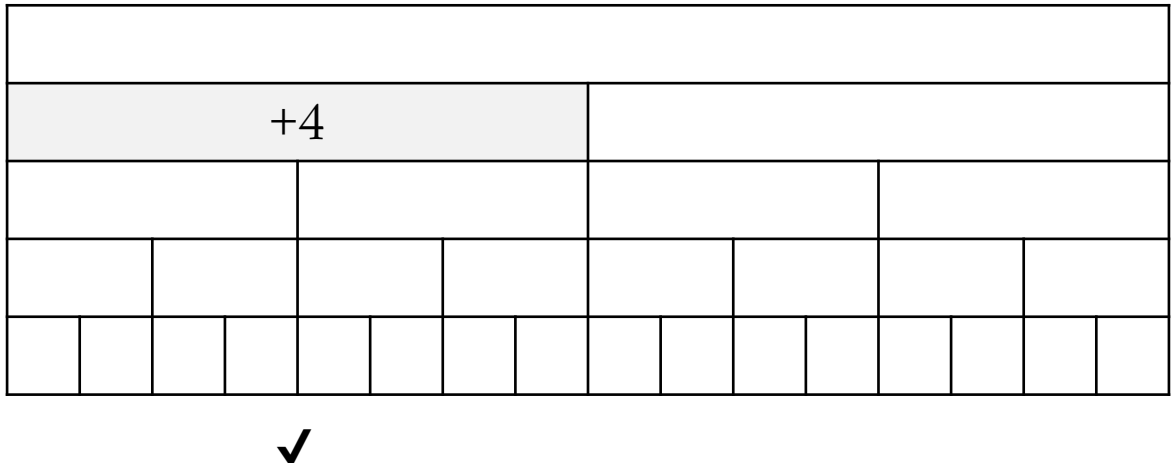
区間[1, 8] に 4 を加算

小課題 4 の考察: 要素数を管理する

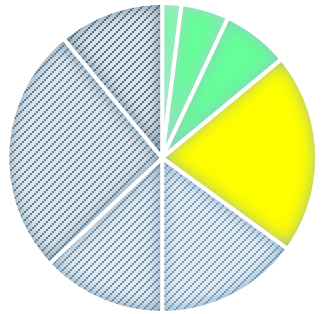


- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - ~~「脱退’」：区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」：位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」

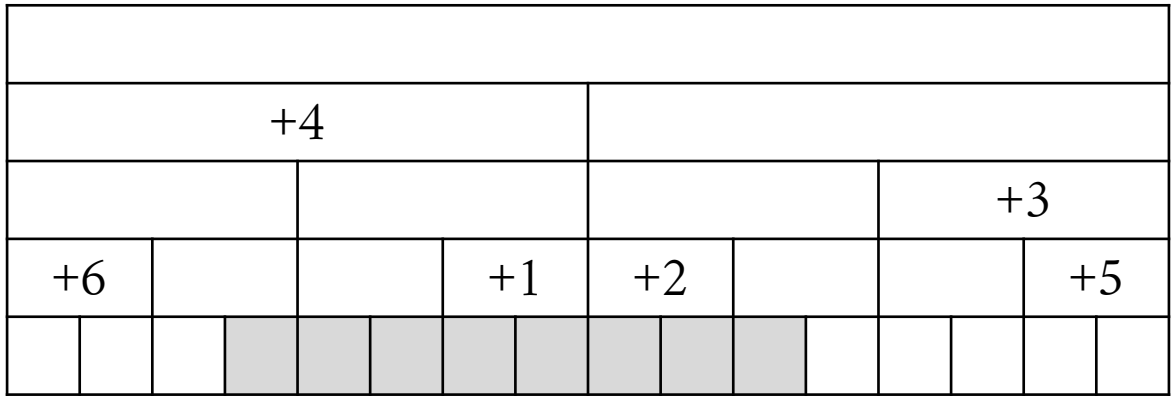


小課題 4 の考察: 要素数を管理する



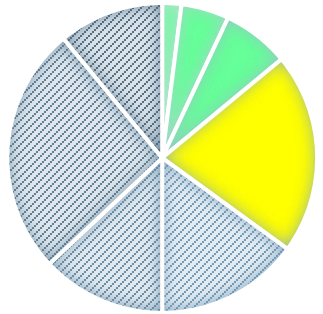
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - ~~「脱退’」：区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」：位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



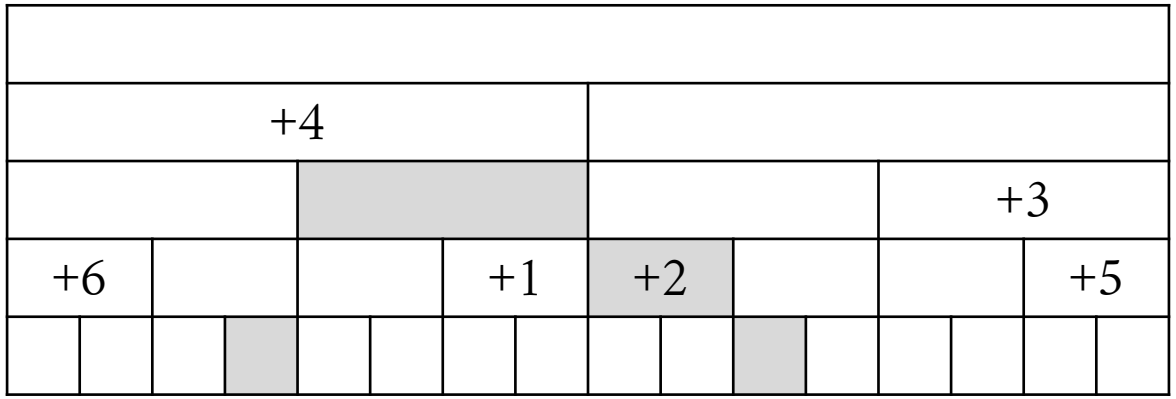
区間[4, 11] に 3 を加算

小課題 4 の考察: 要素数を管理する



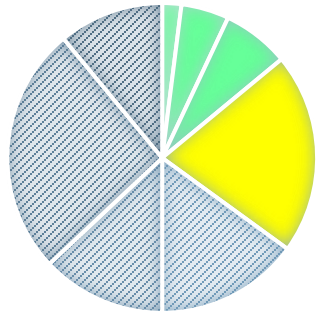
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - ~~「脱退’」：区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」：位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



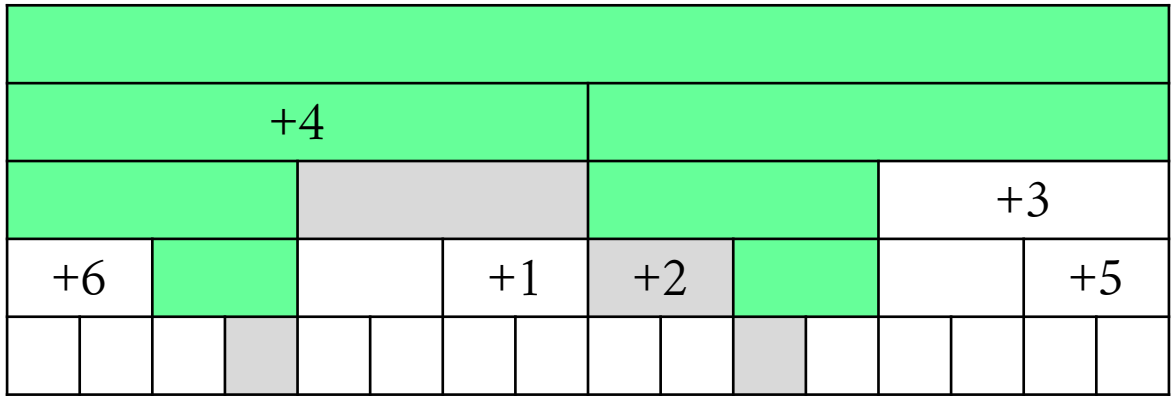
区間[4, 11] に 3 を加算

小課題 4 の考察: 要素数を管理する



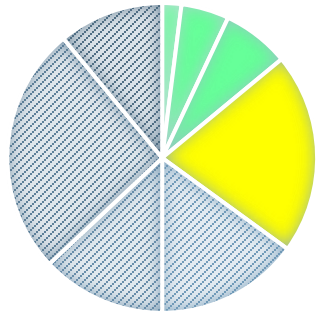
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」: 区間[L, R] に K を加算
  - ~~「脱退’」: 区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」: 位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



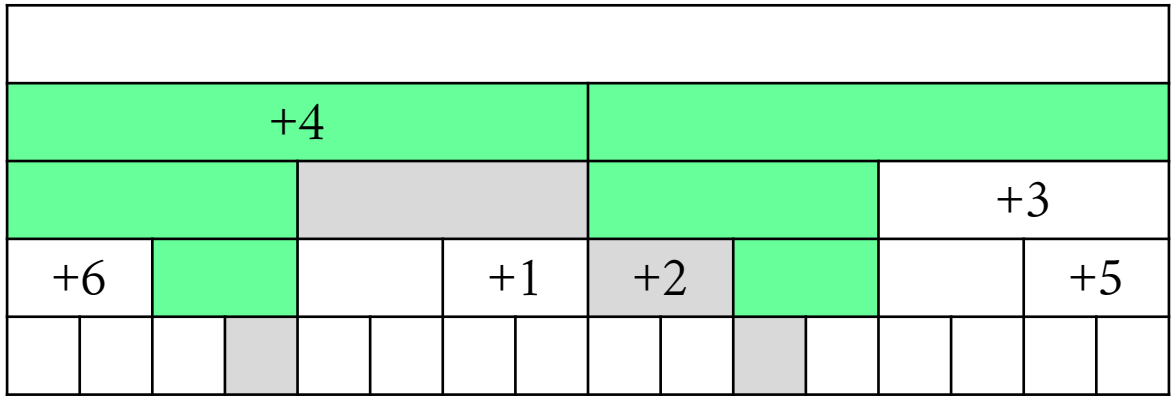
区間[4, 11] に 3 を加算

小課題 4 の考察: 要素数を管理する



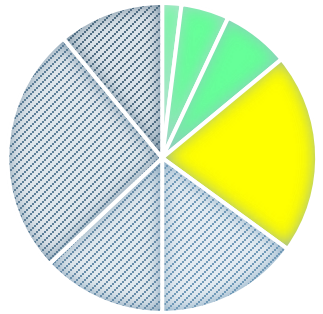
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」: 区間[L, R] に K を加算
  - ~~「脱退’」: 区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」: 位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



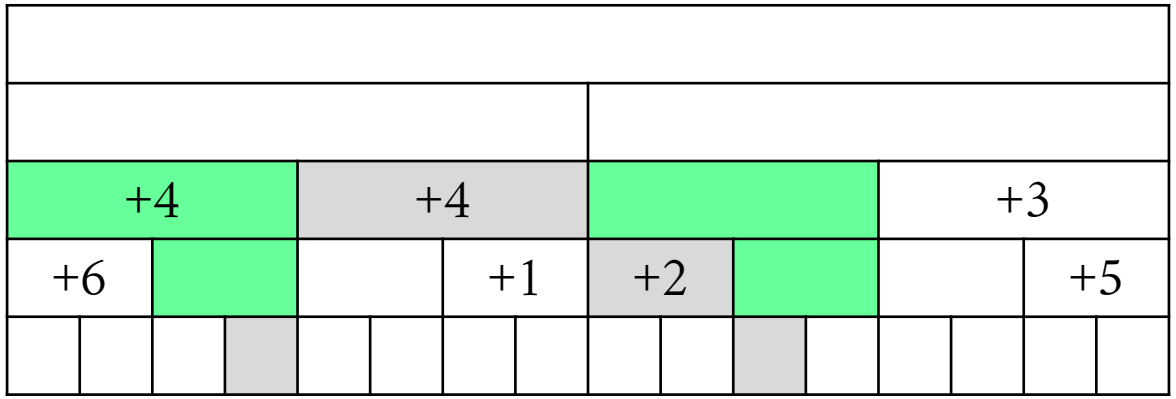
区間[4, 11] に 3 を加算

小課題 4 の考察: 要素数を管理する



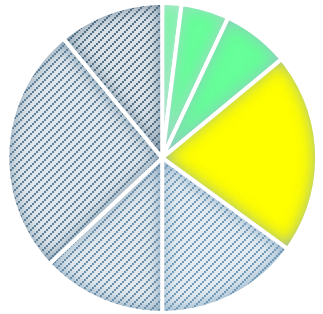
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」: 区間[L, R] に K を加算
  - ~~「脱退’」: 区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」: 位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



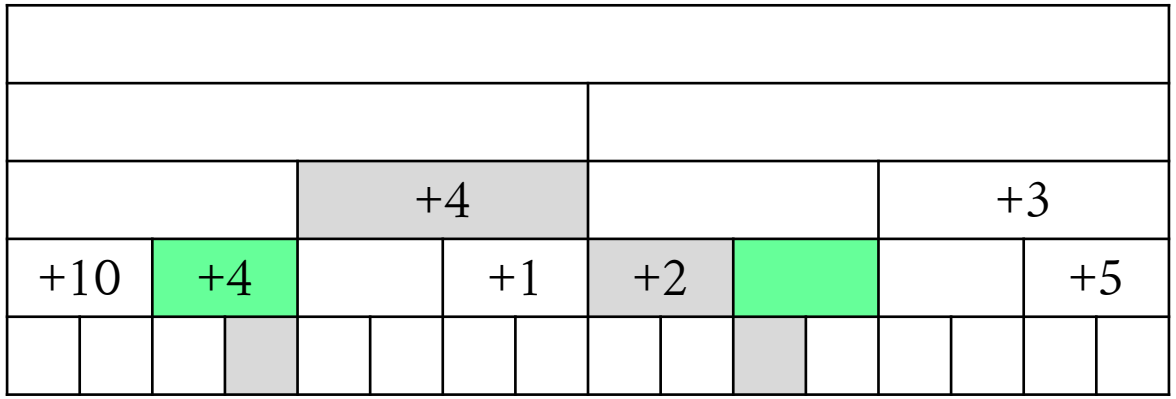
区間[4, 11] に 3 を加算

小課題 4 の考察: 要素数を管理する



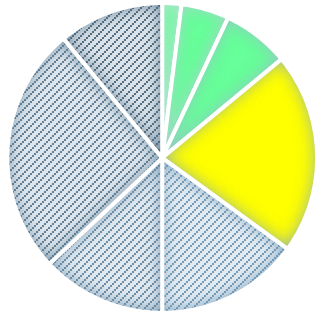
- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」: 区間[L, R] に K を加算
  - ~~「脱退’」: 区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」: 位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



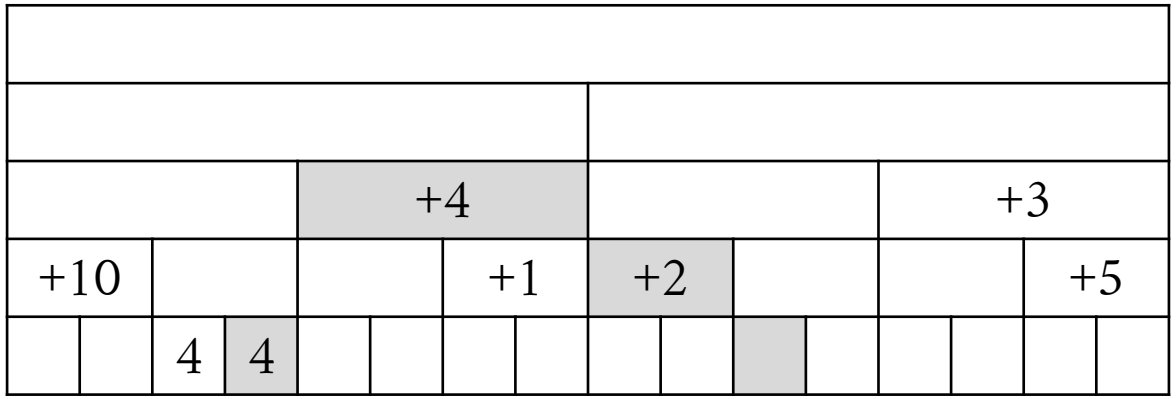
区間[4, 11] に 3 を加算

小課題 4 の考察: 要素数を管理する

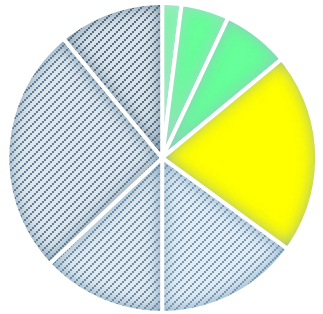


- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - ~~「脱退’」：区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」：位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



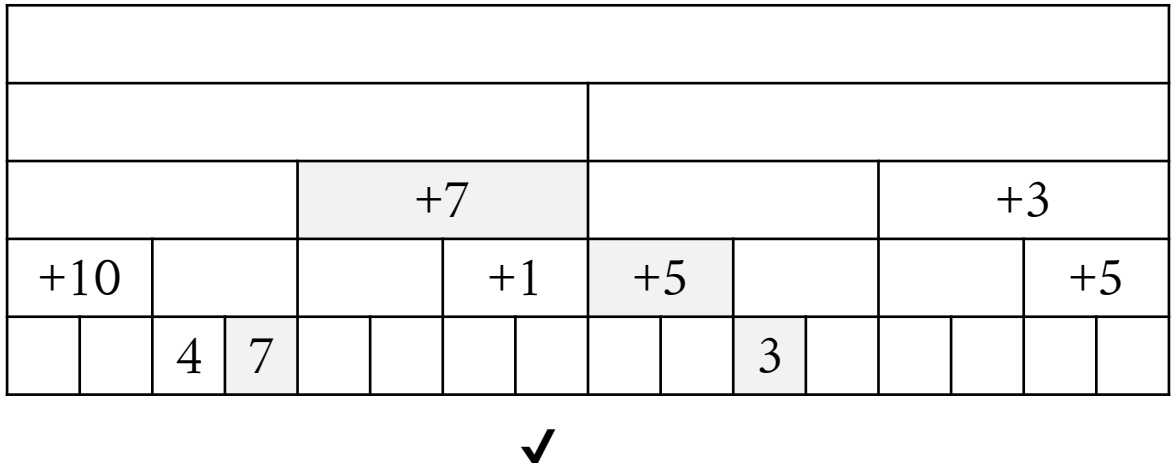
区間[4, 11] に 3 を加算



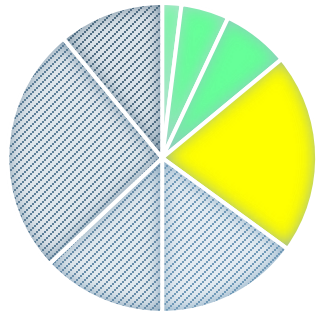
小課題 4 の考察: 要素数を管理する

- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - ~~「脱退’」：区間[L, R] から K を減算し，0 との max をとる~~
  - 「サービス’」：位置[A] の値を求める

- こうだったなら有名問題: 区間加算・一点取得
- いわゆる「双対セグメント木」



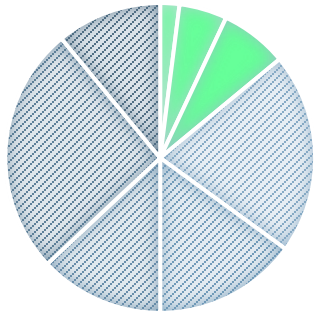
小課題 4 の考察: 要素数を管理する



- 各クエリを要素数の観点から捉え直すとこうなる
  - 「加入’」：区間[L, R] に K を加算
  - 「脱退’」：区間[L, R] から K を減算し，0 との max をとる
  - 「サービス’」：位置[A] の値を求める

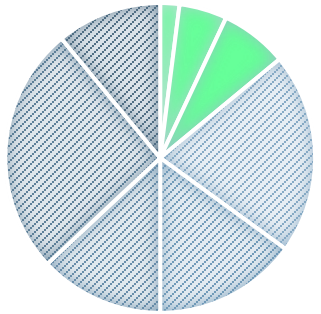
- これが元の問題の「前半」にあたり，小課題 4 が対応する

前半が解けた後の考察



- クエリ時点での待機列 A の先頭から B 番目の要素が知りたいとする
- クエリ時点での待機列 A の要素数 $s\ (\leq B)$ がわかったものとする
  - (つまり，前半が解けた)

前半が解けた後の考察

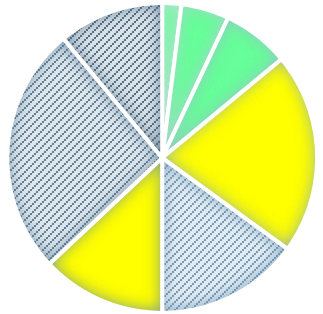


- クエリ時点での待機列 A の先頭から B 番目の要素が知りたいとする
- クエリ時点での待機列 A の要素数 $s\ (\leq B)$ がわかったものとする
  - (つまり，前半が解けた)
- クエリ時点までに待機列 A に加わった要素数 t は双対セグメント木でわかる
  - 2ページ前に挙げた通り，区間加算・一点取得

前半が解けたら，pop なしに帰着

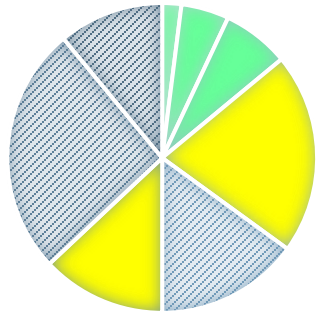
- クエリ時点での待機列 A の先頭から B 番目の要素が知りたいとする
- クエリ時点での待機列 A の要素数 $s\ (\leq B)$ がわかったものとする
  - (つまり，前半が解けた)
- クエリ時点までに待機列 A に加わった要素数 t は双対セグメント木でわかる
  - 2ページ前に挙げた通り，区間加算・一点取得
- すると目的の要素は，初めから数えて $t-s+B$ 番目に待機列 A に加わったということがわかる
- したがって pop の無い場合 (小課題 6) に帰着された(これが「後半」)

方針の整理



- クエリ時点の要素数を求めよう (前半; 小課題 4)
- すると，pop のない場合に帰着できる
- それを解こう (後半; 小課題 6)

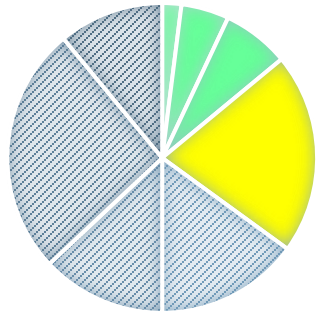
方針の整理



- クエリ時点の要素数を求めよう (前半; 小課題 4)
- すると，pop のない場合に帰着できる
- それを解こう (後半; 小課題 6)

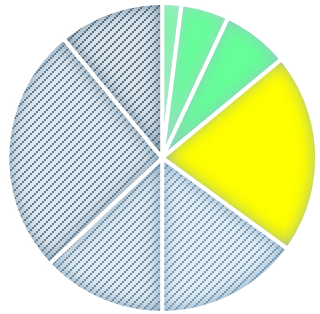
- ここからは多様な解法がある

前半(小課題 4)を解こう



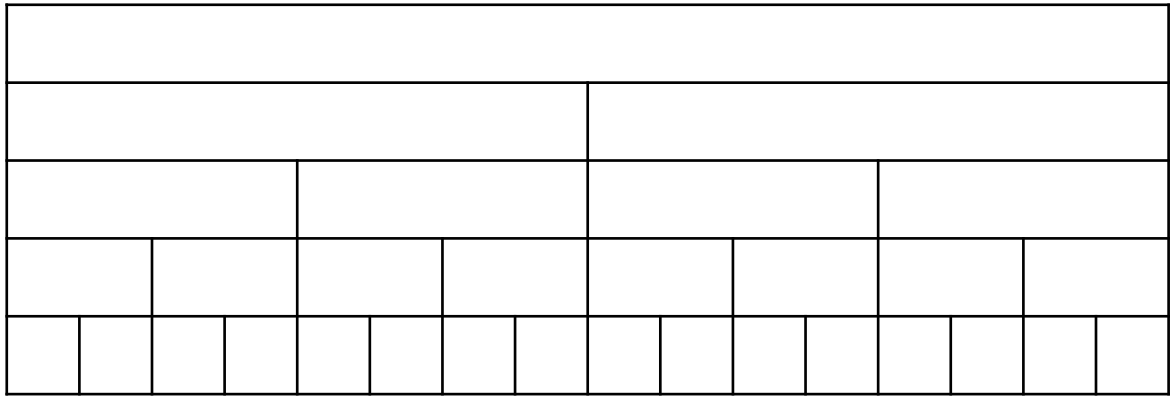
- 「加入’」：区間[L, R] に K を加算
- 「脱退’」：区間[L, R] から K を減算し，0 との max をとる
- 「サービス’」：位置[A] の値を求める

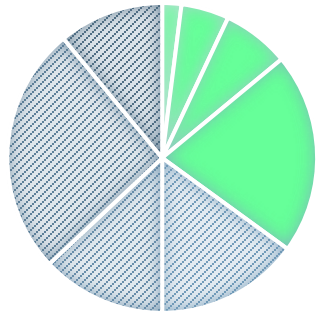
前半(小課題 4)を解こう



- 「加入’」：区間[L, R] に K を加算
- 「脱退’」：区間[L, R] から K を減算し，0 との max をとる
- 「サービス’」：位置[A] の値を求める

- 双対セグメント木で解くことを考える
- 単純な区間加算のみの場合，作用素は $x \leftarrow x + A$



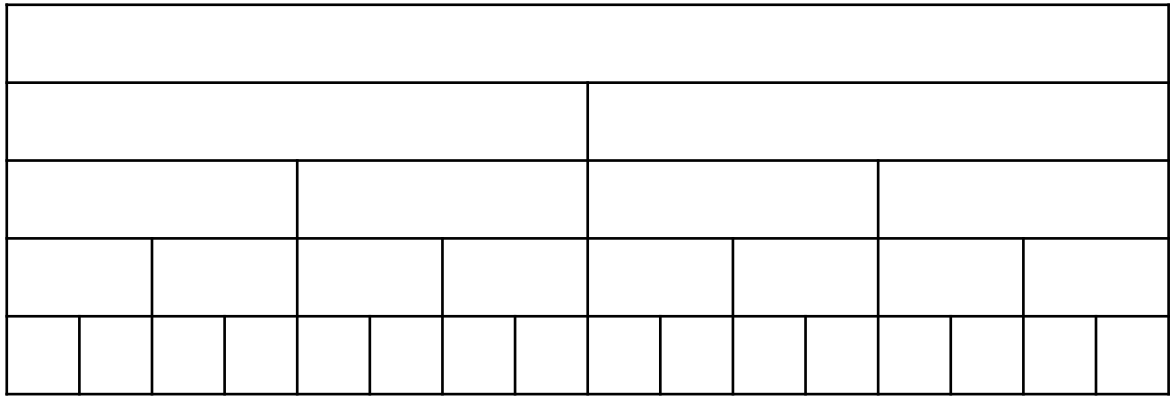


前半(小課題 4)を解こう

- 「加入’」：区間[L, R] に $x \leftarrow \max(x + K, -\infty)$ を適用
- 「脱退’」：区間[L, R] に $x \leftarrow \max(x - K, 0)$ を適用
- 「サービス’」：位置[A] の値を求める

- 双対セグメント木で解くことを考える
- 作用素を $x \leftarrow \max(x + A, B)$ とすることで，上のように適用できる
  - 作用素 2 つの合成が出来る
  - もちろん結合法則も成り立つ

よって解けた



後半(小課題 6)を解こう

- 簡単のため，クエリで聞かれる位置はすべて異なるものとする

後半(小課題 6)を解こう

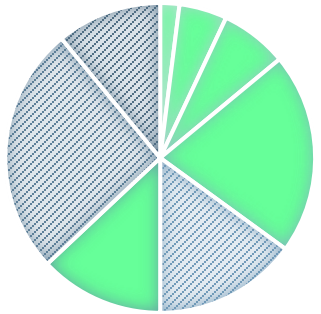
- 簡単のため，クエリで聞かれる位置はすべて異なるものとする
- 次のように初期化した配列を用意する
  - 位置 A を聞くクエリで B 番目が問われる場合，配列の位置 A の値は $-B$
  - クエリで聞かれない位置は $-\infty$
- そして，時系列順に「加入」クエリを見て，次のように処理する
  - 「加入」される位置に K を加える
  - 値が 0 以上の位置があるあいだ，
    その位置にクエリ番号を記録してからその位置の値を $-\infty$ にする
    ことを繰り返す

後半(小課題 6)を解こう

- 簡単のため，クエリで聞かれる位置はすべて異なるものとする
- 次のように初期化した配列を用意する
  - 「B 番目」を聞かれる位置の値は $-B$
  - クエリで聞かれない位置の値は $-\infty$
- そして，時系列順に「加入」クエリを見て，次のように処理する
  - 「加入」される位置に K を加える
  - 値が 0 以上の位置があるあいだ，
    その位置にクエリ番号を記録してからその位置の値を $-\infty$ にする
    ことを繰り返す

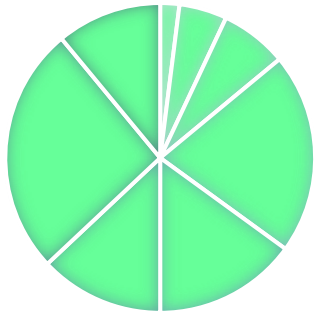
- B 番目の要素を含む push をした「加入」クエリではじめてその位置が 0 以上になるので，
  ある位置に記録した番号の「加入」クエリは，
  その位置の「サービス」クエリにおける目的の要素を含むものである

後半(小課題 6)を解けた



- したがって，「サービス」クエリの答えとなる要素がどの「加入」クエリ由来かを求めることが出来たので，後半も解けた

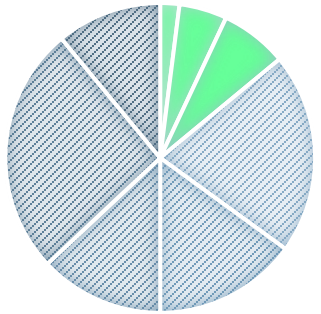
前半＋後半＝満点



- したがって，「サービス」クエリの答えとなる要素がどの「加入」クエリ由来かを求めることが出来たので，後半も解けた

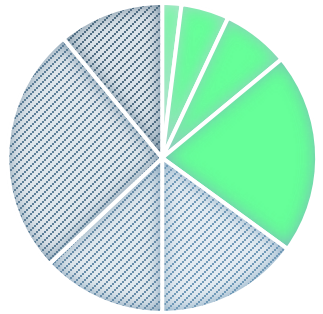
- 前半と後半を組み合わせて $O((N+Q)\log(N+Q))$ 時間の解法が得られ，満点をもぎとれる

別解(平面走査)



- あらかじめ全てのクエリが与えられている
- そこで，待機列の番号を時間軸とした順番で処理することを考える
- 前半と後半に分割するところまでは同じ

前半(小課題 4)



- 位置 [Q] に作用素 $x \leftarrow x + K$ を置く/取り除く
- 位置 [Q] に作用素 $x \leftarrow \min(x - K, 0)$ を置く/取り除く
- 区間 [0, Q] の作用素を順に合成した結果を求める

- 双対セグメント木(区間更新・一点取得)ではなく,
  本来のセグメント木(一点更新・区間取得)の形になった

- 同様に解ける

後半(小課題 6)

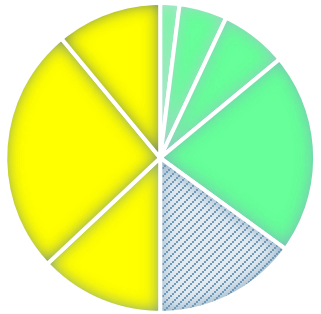
- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

後半(小課題 6)

- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

- セグメント木上で二分探索をすればよい

後半(小課題 6)



- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

- セグメント木上で二分探索をすればよい
- 前半と一緒にそのまま実装して…

後半(小課題 6)

- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

- セグメント木上で二分探索をすればよい
- 前半と一緒にそのまま実装して…TLE

後半(小課題 6)

- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

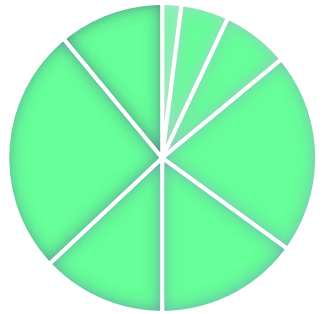
- セグメント木上で二分探索を $O(\log N)$ 時間で行いたい

後半(小課題 6)

- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

- セグメント木上で二分探索を $O(\log N)$ 時間で行いたい
- ノードを上から見て,
  左の子が x 以上なら左へ
  左の子が x 未満なら x から左の子の値を減じてから右へ
- 移動することを，葉に行きあたるまで繰り返す
- これで $O(\log N)$ 時間になり,

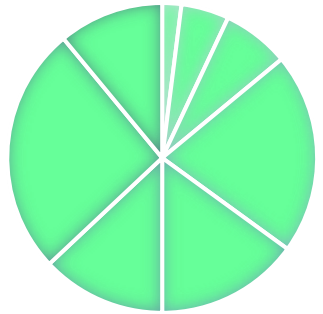
後半(小課題 6)



- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

- セグメント木上で二分探索を $O(\log N)$ 時間で行いたい
- ノードを上から見て,
  左の子が x 以上なら左へ
  左の子が x 未満なら x から左の子の値を減じてから右へ
- 移動することを，葉に行きあたるまで繰り返す
- これで $O(\log N)$ 時間になり，AC

後半(小課題 6)



- 位置 [Q] に整数 K を置く/取り除く
- 区間 $[0, x]$ の総和が B 以上になる最小の x を求める

- 平面走査をしなくともパラレル二分探索によって解けるが,
  結局 log が 2 つつくため満点は得られない(と思う)

得点分布

得点分布

- 2 点: 1 人
- 7 点: 4 人
- 14 点: 3 人
- 21 点: 1 人
- 27 点: 1 人
- 35 点: 1 人
- 89 点: 2 人
- 100 点: 1 人