KUPC2016 H

問題 drafear
解答 drafear, takise
解説 drafear

考察

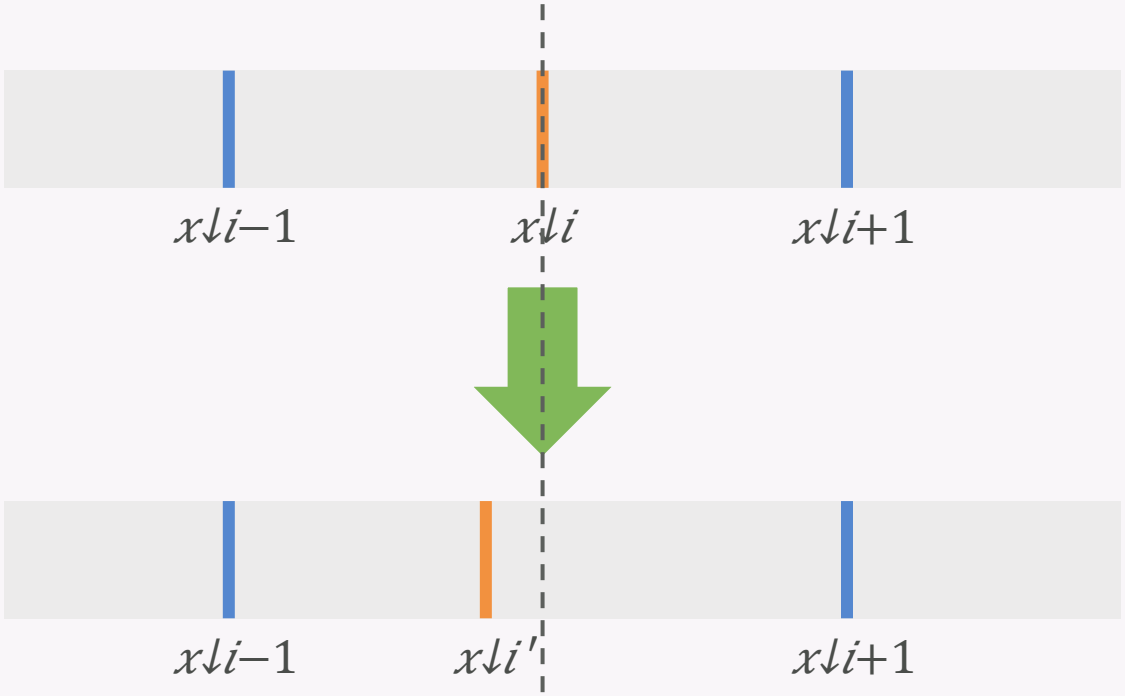
考察

- 補強材の合計は変わらない
  - $L = \sum_{i=1}^{N} A_i$ とする

- 補強材を移動させる操作は, i から j の代わりに次の2種類だけと考えても良い
  a. コスト1で i から $i+1$ に1個移動
  b. コスト1で $i+1$ から i に1個移動

考察

- 累積和 $x_n = \sum_{i=1}^{n} A_i$ を考えると
  x_n は操作aまたは操作bを 1回 行うごとに 1 ずれる
  (他の操作では変化しない)

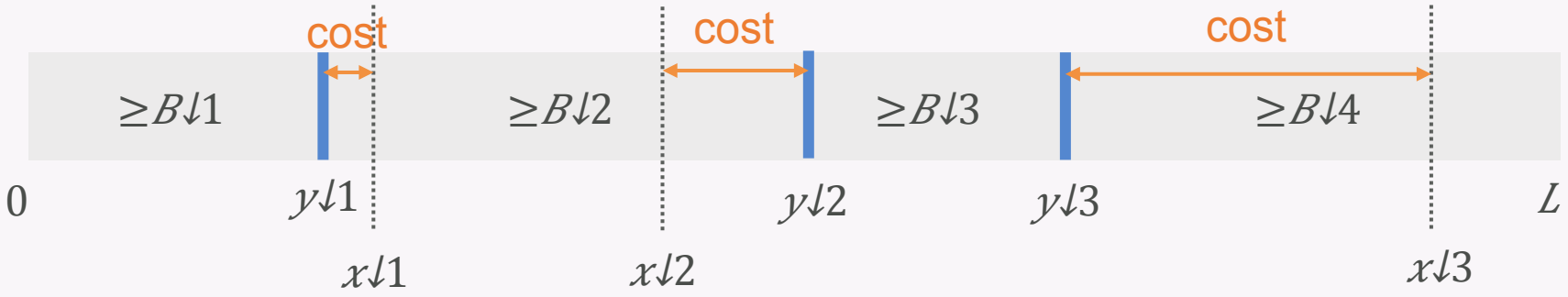


a. コスト1で i から $i+1$ に1個移動
b. コスト1で $i+1$ から i に1個移動

考察

- 無駄な操作(aもbも1回以上するような操作)はしないので
  y_i のように置くと, 合計コストは $\sum_{i=1}^{N} |x_i - y_i|$ となる

- つまり, 全ての補強材を取り去って, 配置しなおしたときに
  配置後の累積和と元の累積和との差の合計の最小値を求めれば良い



DP解 (部分点1)

DP解

$dp[i][j]$:= i番目の補強位置までで
$j(=y_i)$個の補強材を使ったときの合計コストの最小値

- 初期値
  - $dp[0][0]=0$
  - $dp[0][j]=\infty$ $(1 \leq j \leq L)$
- 遷移
  - $dp[i][j]=min\{dp[i-1][k] \mid 0 \leq k \leq j-B_i\}+|j-x_i|$
  - $x_k=\sum_{i=1}^{k} A_i$
- 出力
  - $dp[N][L=\sum_{i=1}^{N} A_i]$

DP解

- 計算量
  - $O(NL^2)$
- 部分点1の制約なら間に合う
  - $N \leq 100$
  - $L \leq 400$

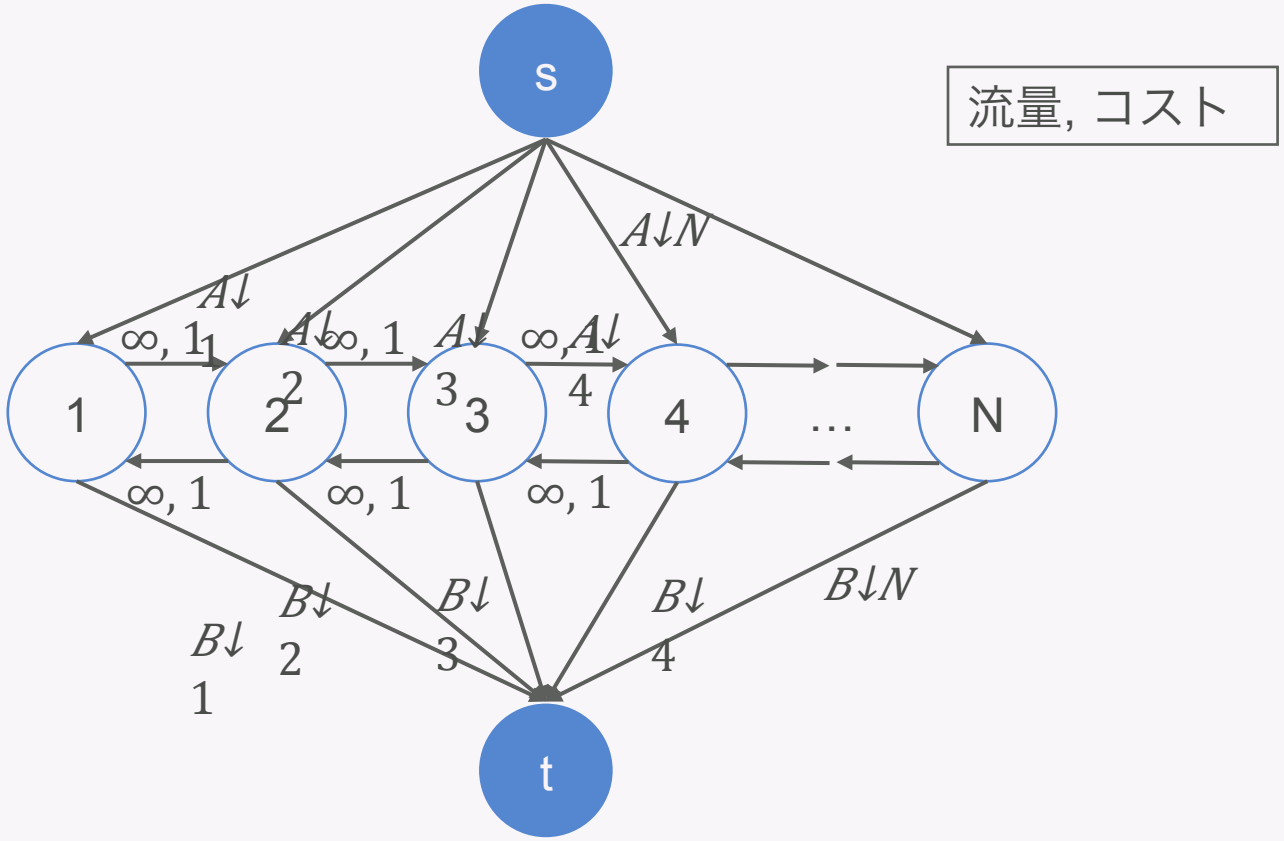
フロー解1 (部分点2)

フロー解1

- 補強材を移動させる操作は, i から j の代わりに次の2種類と考えることができる (再掲)
  a. コスト1で i から $i+1$ に1個移動
  b. コスト1で $i+1$ から i に1個移動

フロー解1

- 以下のグラフでs-t間流量 $L=\sum_{i=1}^{N} A_i$ の最小費用流を求める



フロー解1

- 出力
  - $min\_cost\_flow(s,t,\sum i=1\uparrow N\blacksquare A\downarrow i\ )$
- 計算量
  - 増大パスを見つけるのに$dijkstra$で $O(NlogN)$
  - フローを流す回数は $O(L)$???
    - 一度押し戻したところにまた流れることはなく
      押し戻すときにはそのパス上でどこか1つの辺が
      目一杯満たされるか全て押し戻されるので $O(N)$
    - なぜなら各$v-t$辺に流れるものは, vに近い所からもらってくるから (クロスしない)
      (vより左側の最も近い頂点から順に合計x, 右側からy というように)
  - あわせて $O(N\uparrow 2\ logN)$ となり部分点2を取れる
    - $N\leq 10\uparrow 3$
    - $L\leq 10\uparrow 12$

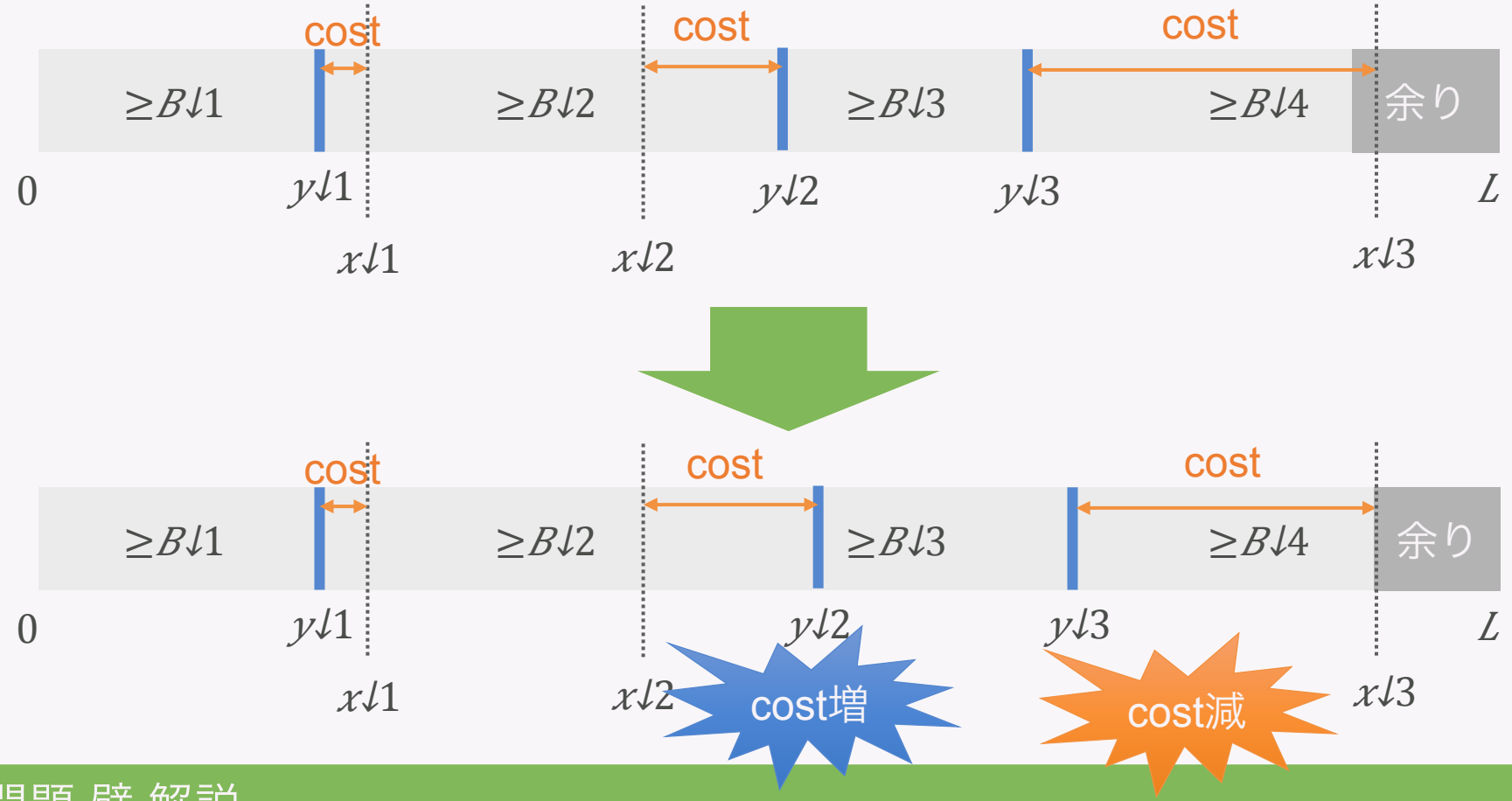
フロー解2 (部分点2)

フロー解2

- これまでの解法は,初め,各位置に置かれる補強材の数を$A↓i$として制約を満たすように変形してきた
- 考えを変えて,実行可能解である$B↓i$から始めることにする
- つまり,初めは各位置に$B↓i$個ずつ置かれていて,
  後から残りの $∑↓{i=1}↑N▒A↓i −∑↓{i=1}↑N▒B↓i$ 個を配ることにする

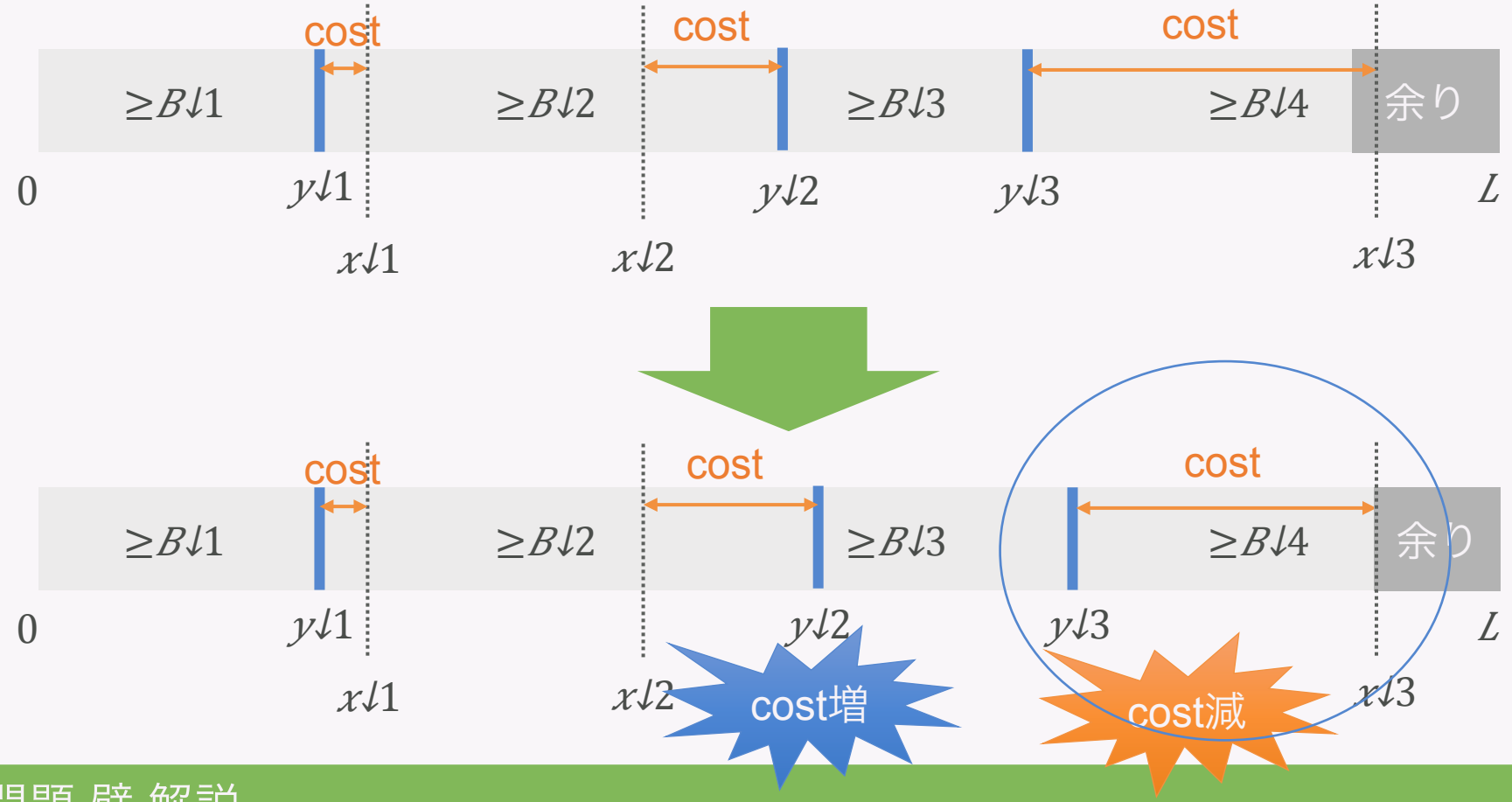
フロー解2

- ある位置iに1配ると,$k \geq i$での累積和y_kが1増える (以下の例ではi=2)



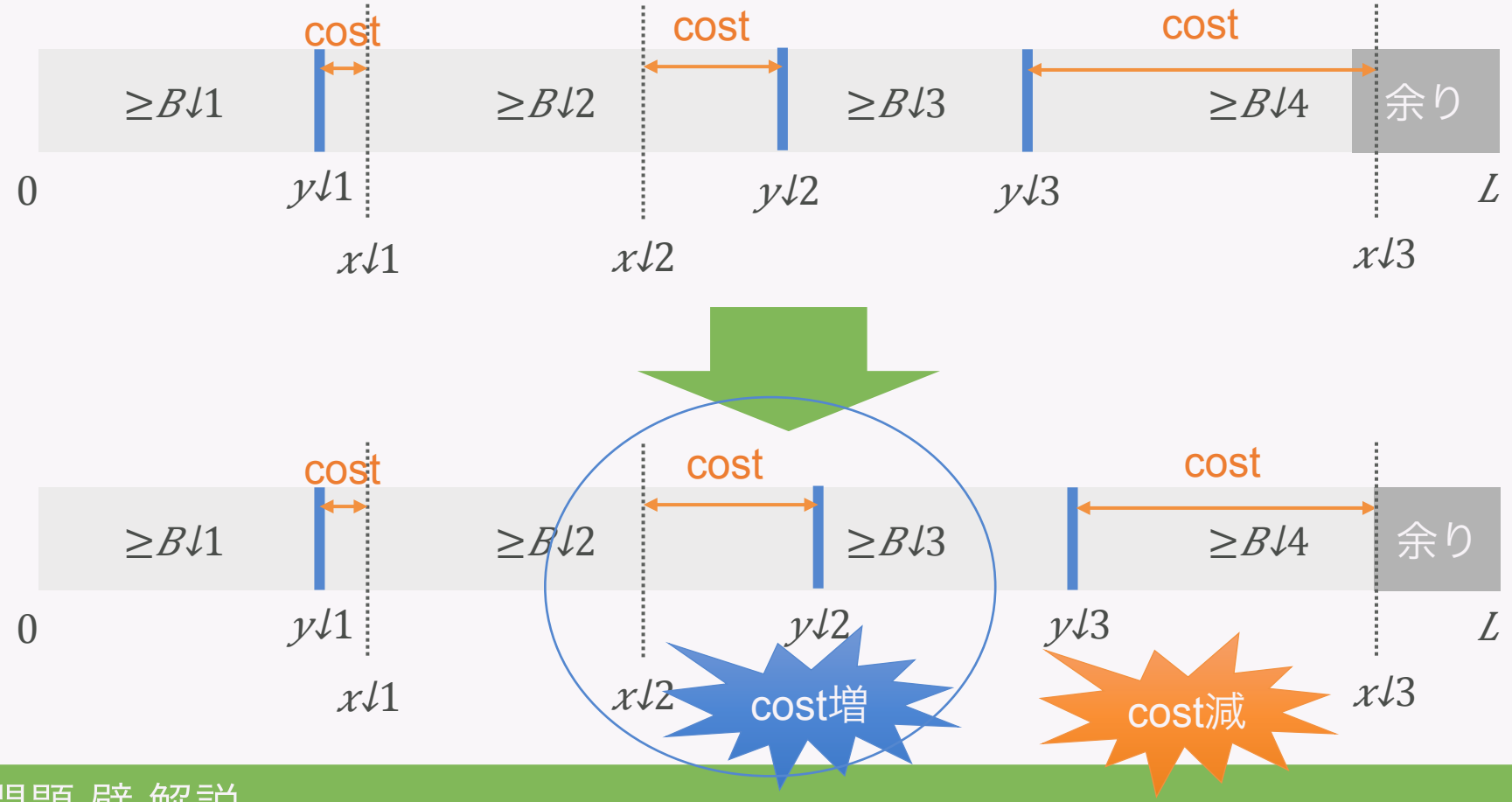
フロー解2

- $k \geq i$で$y_k < x_k$なるkについてはコスト-1



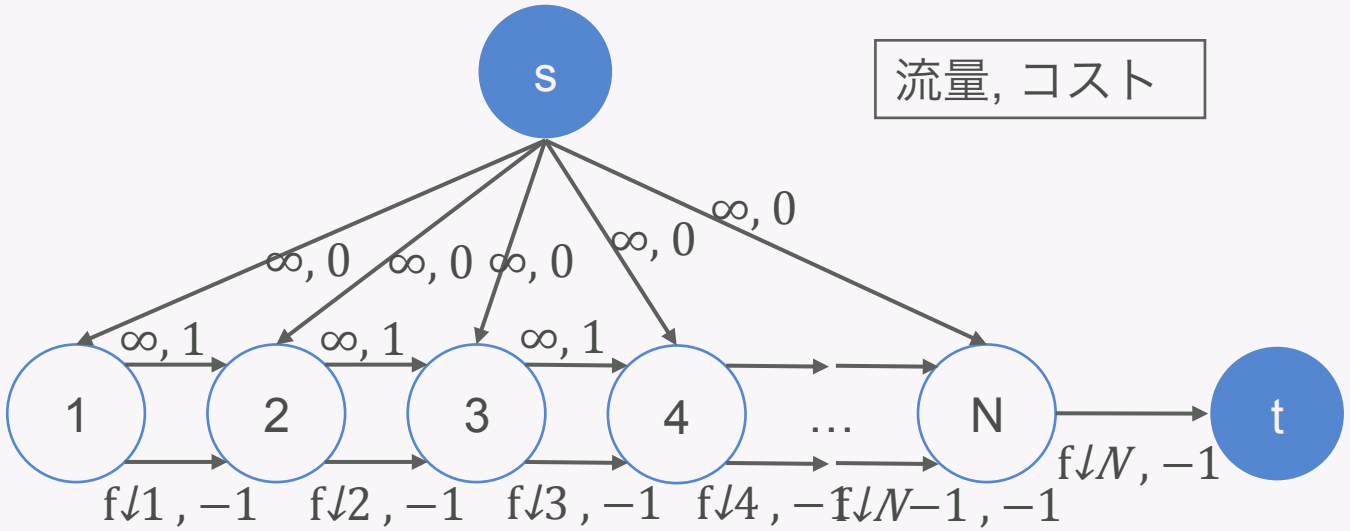
フロー解2

- $k \geq i$で$y_k \geq x_k$なるkについてはコスト+1



フロー解2

- よって, 以下のグラフで流量 $f_N = L - \sum_{i=1}^{N} B_i$ の最小費用流を求めれば良い



$$f_i = max(\sum_{j=1}^{i} A_j - \sum_{j=1}^{i} B_j, 0)$$

フロー解2

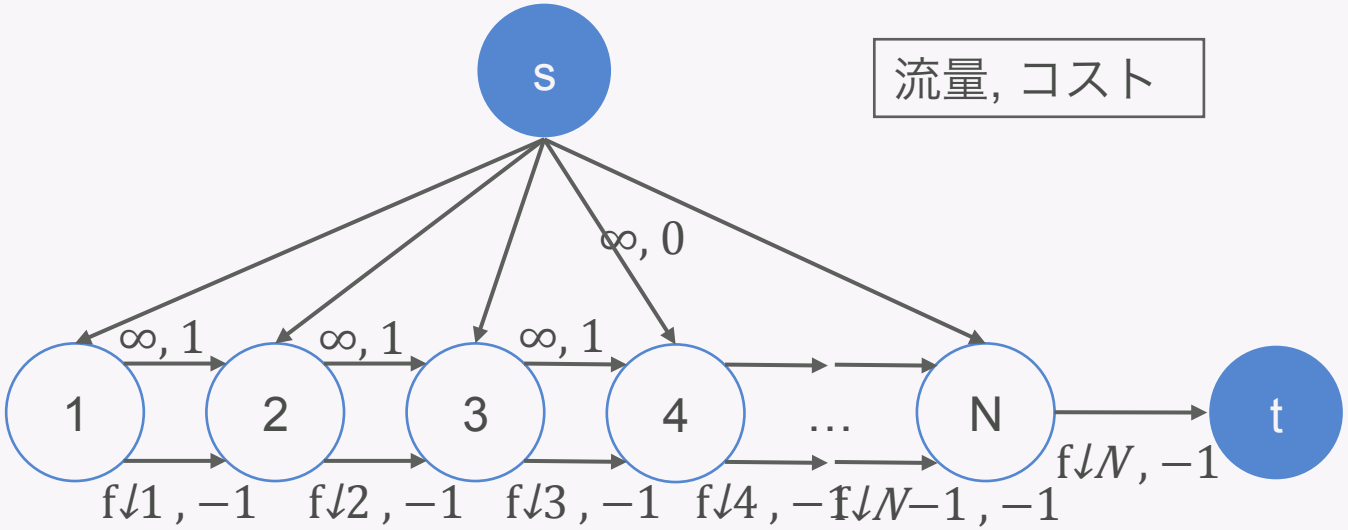
- 出力
  - $\sum_{i=1}^{N} |\sum_{j=1}^{i} A_j - \sum_{j=1}^{i} B_j| + min\_cost\_flow(s,t,f_N)$

    初期値　　差分
- 計算量
  - 明らかに押し戻すことはないので, $O(N^2 \log N)$
  - つまり, 辺に流すときに逆辺のcapを増やす(逆辺を張る)必要がない
  - これでも部分点2を取れる

満点解法

満点解法

- フロー解2を高速化する
- 一度流したものを押し戻すことはないので
  鎖状のDAGとなり, *dijkstra*の代わりに簡単なDPでできる
- 具体的には, 単にtから順に累積和を取れば$s-t$最短路が求まる



満点解法

- つまり以下を行う (後でじっくり見て下さい)
  1. $ans=\sum_{i=1}^{N}|\sum_{j=1}^{i}A_j-\sum_{j=1}^{i}B_j|$ とする
  2. $F=\sum_{i=1}^{N}A_i-\sum_{i=1}^{N}B_i$ とする (流す合計量)
  3. $v_i=\begin{cases}+1 & (f_i=0)\\ -1 & (f_i>0)\end{cases}$ として $dist_i=\sum_{j=i}^{N}v_j=dist_{i+1}+v_i$ を計算する
  4. $c=min\{dist_i\},\ u=argmin\{dist_i\}$ を求める
  5. $d=min(F,minf_i\ i\geq u,\ f_i>0\ )$ を求める
  6. $f_i-=d\ (u\leq i\leq N,\ f_i>0)$ とする
  7. $ans+=c*d,\ F-=d$ とする
  8. $d>0$ なら3に戻る
  9. ansを出力する

満点解法

- このアルゴリズムだと $O(N^2)$ で間に合わない
- StarrySkyTreeを用いて高速化する
- StarrySkyTreeとは,
  以下の各クエリに対し$O(logN)$で処理できるデータ構造
  i. $[l,r)$ に val を加算する (区間加算)
  ii. $[l,r)$ の最小値を求める (区間min)

満点解法

- もう一度見てみる
  1. $ans=\sum_{i=1}^{N}|\sum_{j=1}^{i}A_j-\sum_{j=1}^{i}B_j|$ とする … $O(N)$
  2. $F=\sum_{i=1}^{N}A_i-\sum_{i=1}^{N}B_i$ とする … $O(N)$
  3. $v_i=\begin{cases}+1 & (f_i=0)\\ -1 & (f_i>0)\end{cases}$ として $dist_i=\sum_{j=i}^{N}v_j$ を計算する … $O(N)$
  4. $c=min\{dist_i\}$, $u=argmin\{dist_i\}$ を求める … $O(N)$
  5. $d=min(F, minf_i\ i\geq u, f_i>0\ )$ を求める … $O(N)$
  6. $f_i-=d\ (u\leq i\leq N, f_i>0)$ とする … $O(N)$
  7. $ans+=c*d$, $F-=d$ とする … $O(1)$
  8. $d>0$ なら3に戻る … $O(1)$
  9. ansを出力する … $O(1)$

満点解法

- ボトルネック(繰り返し回数$O(N)$)
  - $v_i = \begin{cases} +1 & (f_i = 0) \\ -1 & (f_i > 0) \end{cases}$ として $dist_i = \sum_{j=i}^{N} v_j$ を計算する
  - $c = min\{dist_i\},\ u = argmin\{dist_i\}$ を求める
  - $d = min(F, minf_i\ i \geq u,\ f_i > 0\ )$ を求める

満点解法

- $v↓i = \{■ +1 \quad (f↓i = 0) -1 \quad (f↓i > 0)$ として $dist↓i = \sum_{j=i↑N▒} v↓j$ を計算する
  - $dist↓i$ のStarrySkyTree($sst↓1$)を持つ
- $c = min\{dist↓i\}, u = argmin\{dist↓i\}$ を求める
  - $sst↓1$ から求める
- $d = min(F, minf↓i \; i \geq u, \; f↓i > 0 \;)$ を求める
  - $f↓i$ のStarrySkyTree($sst↓2$)を持つ
  - ただし, $f↓i = 0$ のところは ∞ とする

満点解法

- $f[i] -= d\ (u \leq i \leq N, f[i] > 0)$ とする
  - $sst[2]$ に対する処理でできる
  - これによって $v[i] = \begin{cases} +1 & (f[i] = 0) \\ -1 & (f[i] > 0) \end{cases}$ が変化するため $dist[i]$ も変化するので次の処理も行う

- $f[i] = 0$ なる全ての iについて($sst[1]$ の minクエリで探す)
  $f[i] = \infty$ に更新し, $dist[j] += 2\ (0 \leq j \leq i)$ を行う

- 以上を行えば, $O(NlogN)$ で答えを導ける

統計

統計

- First Accept
  - ALL　　　yokozuna57 (167:10)
  - Onsite(東京)　සහ‿kල (173:45)
  - Onsite(京都)　--
- Accept　6 / 86 (7%)