Worst Reporter 4

解説：岸田陸玖

問題概要

- 頂点数 N の functional graph（各頂点の出次数が 1 の有向グラフ） G と長さ N の数列 H, C が与えられる

  操作：コスト C_i かけて、 H_i を任意の値を変える

  条件： $\forall (u, v) \in E(G)$, $H_u \geq H_v$

- コストの総和の最小値は？

考察

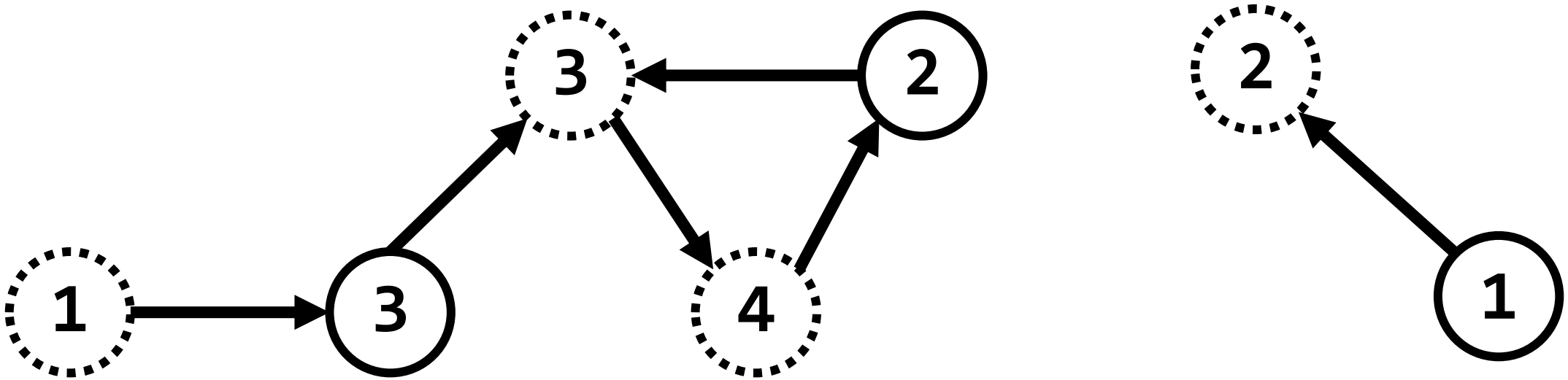
- S を H の値を**変えない**頂点集合とすると
  条件を満たすためには (#) を満たす必要がある

  $u, v \in S$ かつ $u \to v$ のパスが存在するならば $H_u \geq H_v$ …(#)

- 逆に (#) を満たすような S は H の値を適切に変えることで条件を満たすことが可能

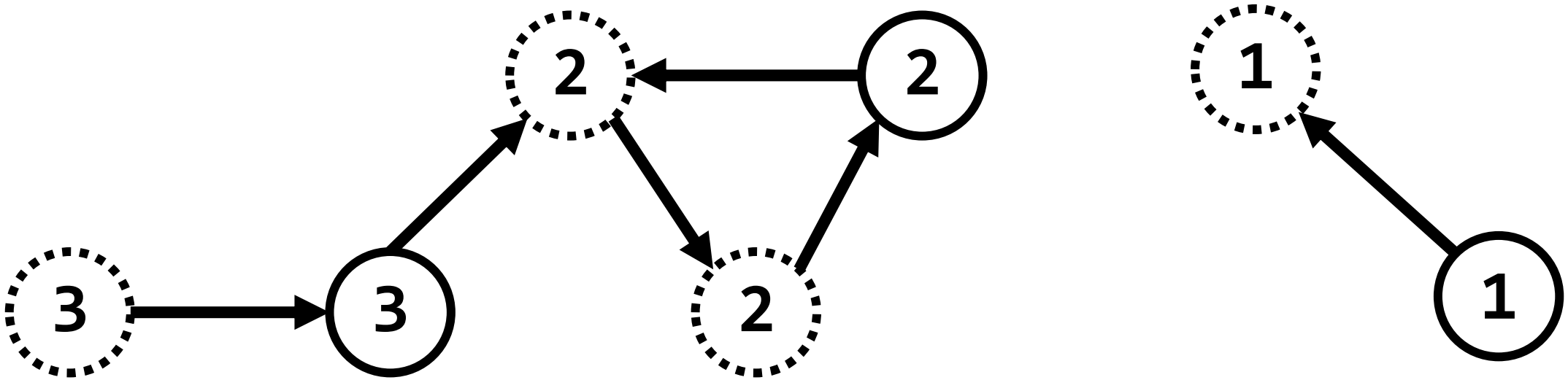
考察

- 辺にそって進んだとき、最初に到達する S の頂点の値に変えればよい
- ただし、到達しない場合は 1 に変える



考察

- 辺にそって進んだとき、最初に到達する S の頂点の値に変えればよい
- ただし、到達しない場合は 1 に変える



考察

$u, v \in S$ かつ $u \to v$ のパスが存在するならば $H_u \geq H_v$ …(#)

- (#) を満たす S の集合を $\mathcal{S}$ とすると

$\sum_{v \in \{1,\ldots,N\}} C_v - \max\{ \sum_{v \in S} C_v \mid S \in \mathcal{S}\}$ が答え

- 座標圧縮をすることで H は 1 以上 N 以下の値に変換可能

小課題 1

- グラフが木になっている
- $\mathrm{dp}[v][i] :=$ (#) に加え以下を満たす S の中で $\sum_{v \in S} C_v$ の最大値

  $S \subseteq V$(頂点 v の部分木)　and　$\forall v \in S, H_v \geq i$
- $\mathrm{dp}[root][1] = \max\{\sum_{v \in S} C_v \mid S \in \mathcal{S}\}$ が成り立つ

小課題 1

- $N_v \coloneqq (v$ の子の頂点集合$)$ とする
- $H_v < i$ の場合、 $v \notin S$ のみ考慮する

  $\mathrm{dp}[v][i] = \sum_{u \in N_v} \mathrm{dp}[u][i]$
- $H_v \geq i$ の場合、 $v \notin S$ と $v \in S$ の2通り考慮する

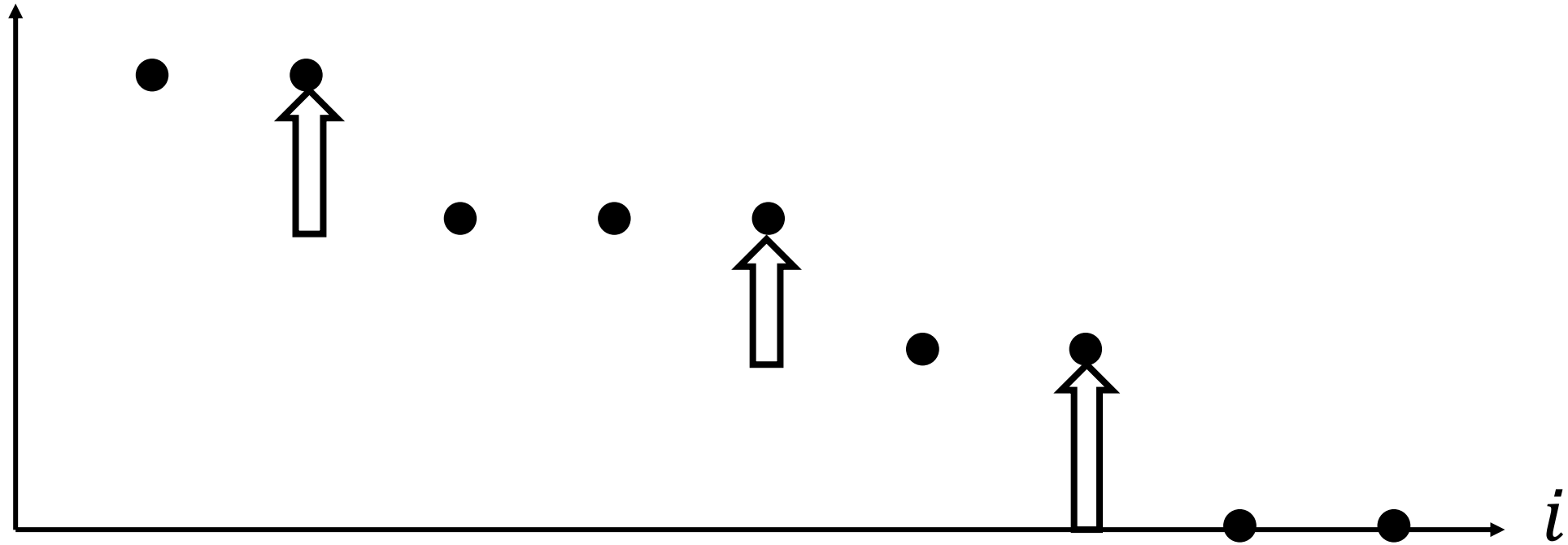
  $\mathrm{dp}[v][i] = \max\left(\sum_{u \in N_v} \mathrm{dp}[u][i], C_v + \sum_{u \in N_v} \mathrm{dp}[u][H_v]\right)$
- $O(N^2)$ で計算可能

小課題２

- $N \leq 2 \times 10^5$
- $\mathrm{dp}[v][i] \coloneqq (\#)$ に加え以下を満たす S の中で $\sum_{v \in S} C_v$ の最大値

  $S \subseteq V$(頂点 v の部分木)　and　$\forall v \in S\,, H_v \geq i$
- 各頂点 v に対し $\mathrm{dp}[v]$ を直接記録するのは避けたい
- $\mathrm{dp}[v][i]$ $(i \in \{1, \ldots, N\})$ の値の種類数は
  頂点 v の部分木の頂点数以下であり、広義単調減少している

小課題２



- 増加する箇所のみ $(i\,, \mathrm{dp}[v][i] - \mathrm{dp}[v][i+1])$ を記録すればよい

小課題２

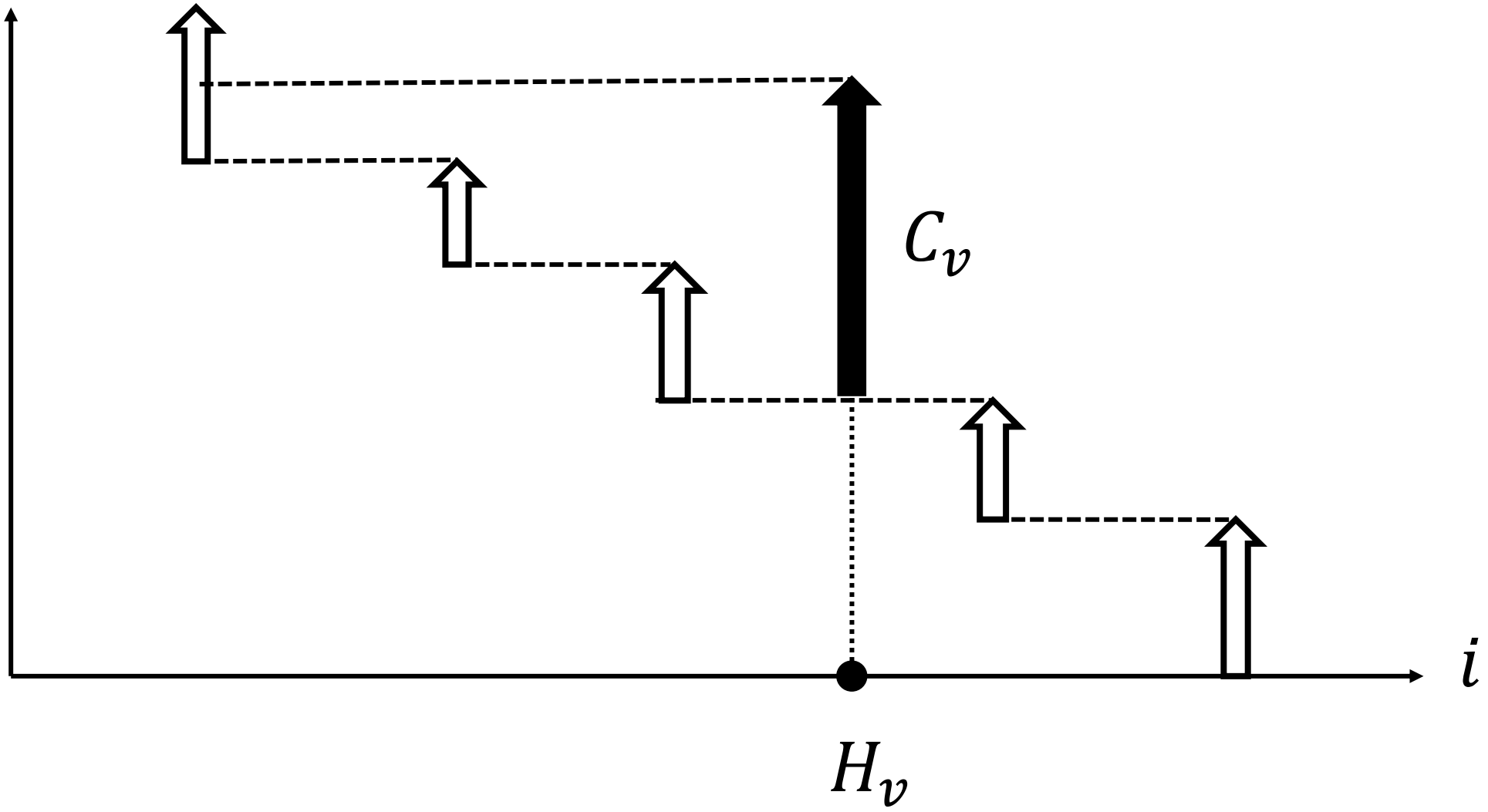
- $\mathrm{dp}[v]$ が $\left(i\,, d_v(i)\right)_{i \in X_v}$ $(X_v\colon \text{distinct}\,, d_i \geq 1)$ で表されたとする
- この情報だけで dp を計算したい
- まず dp の遷移における $v \notin S$ の部分 $\sum_{u \in N_v} \mathrm{dp}[u][i]$ 計算したい
- $X_v = \mathrm{U}_{u \in N_v} X_u$ として $\left(i\,, \sum_{u \in N_v\,,\, i \in X_u} d_u(i)\right)_{i \in X_v}$ で表される

小課題 2

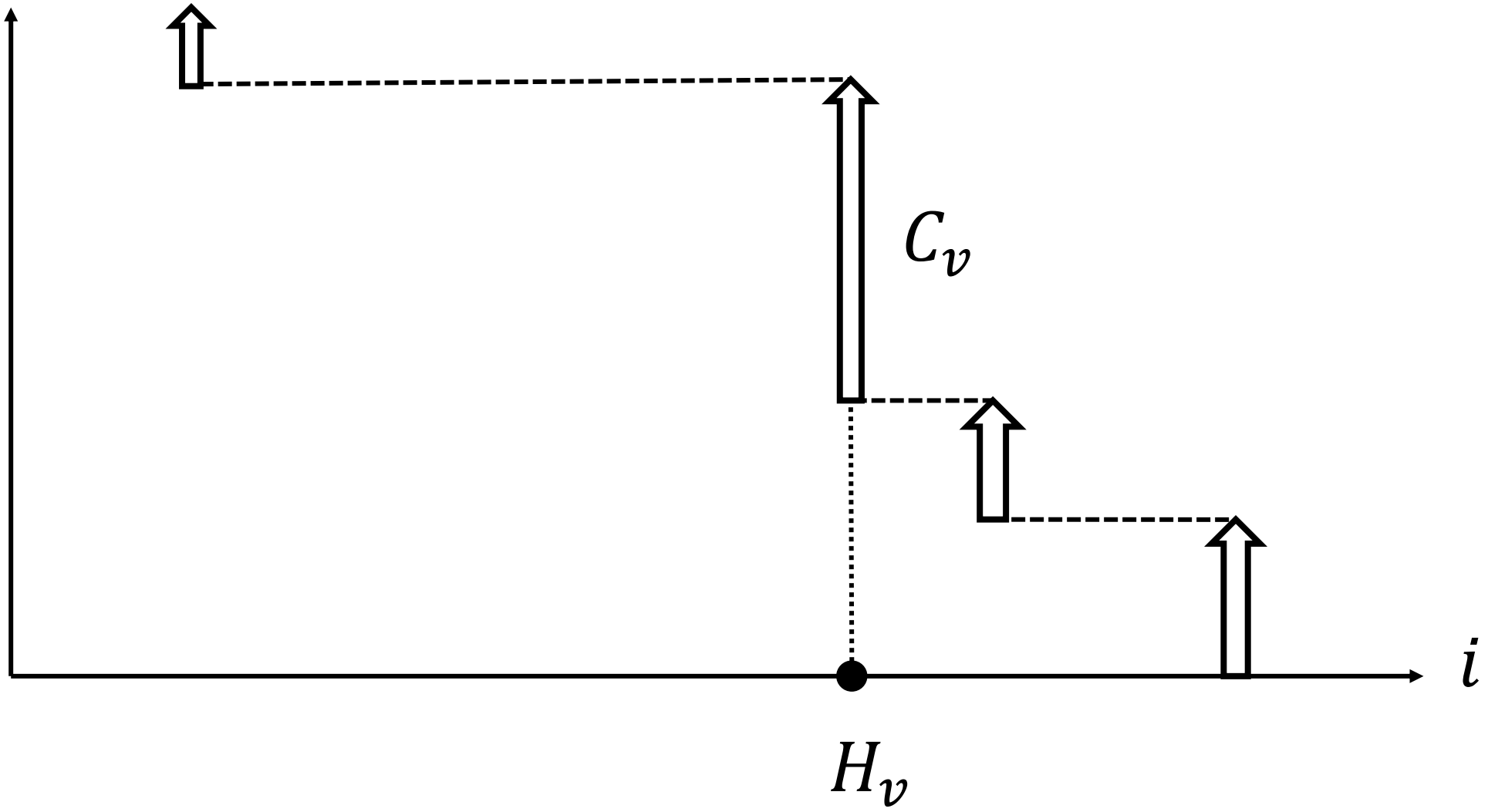
- 次に dp の遷移における $v \in S$ の部分を計算したい

$$\mathrm{dp}[v][i] = \max\left(\sum_{u \in N_v} \mathrm{dp}[u][i],\ C_v + \sum_{u \in N_v} \mathrm{dp}[u][H_v]\right)\ (i \leq H_v)$$

小課題２



小課題２



小課題２

- H_v 未満であるような i で大きいほうから順に $\left(i\,,d_v(i)\right)$ が消滅または d_i の値が減少する
- 以上の計算は `std::map` で高速に計算可能
  `std::map::erase , std::map::lower_bound` を利用すれば容易
- しかしこのままでは $O(N^2\log N)$ になり計算量が悪化する
- $\left(i\,,\sum_{u\in N_v\,,\,i\in X_u} d_u(i)\right)_{i\in X_v}$ の計算、すなわち `map` の `merge` を高速化したい

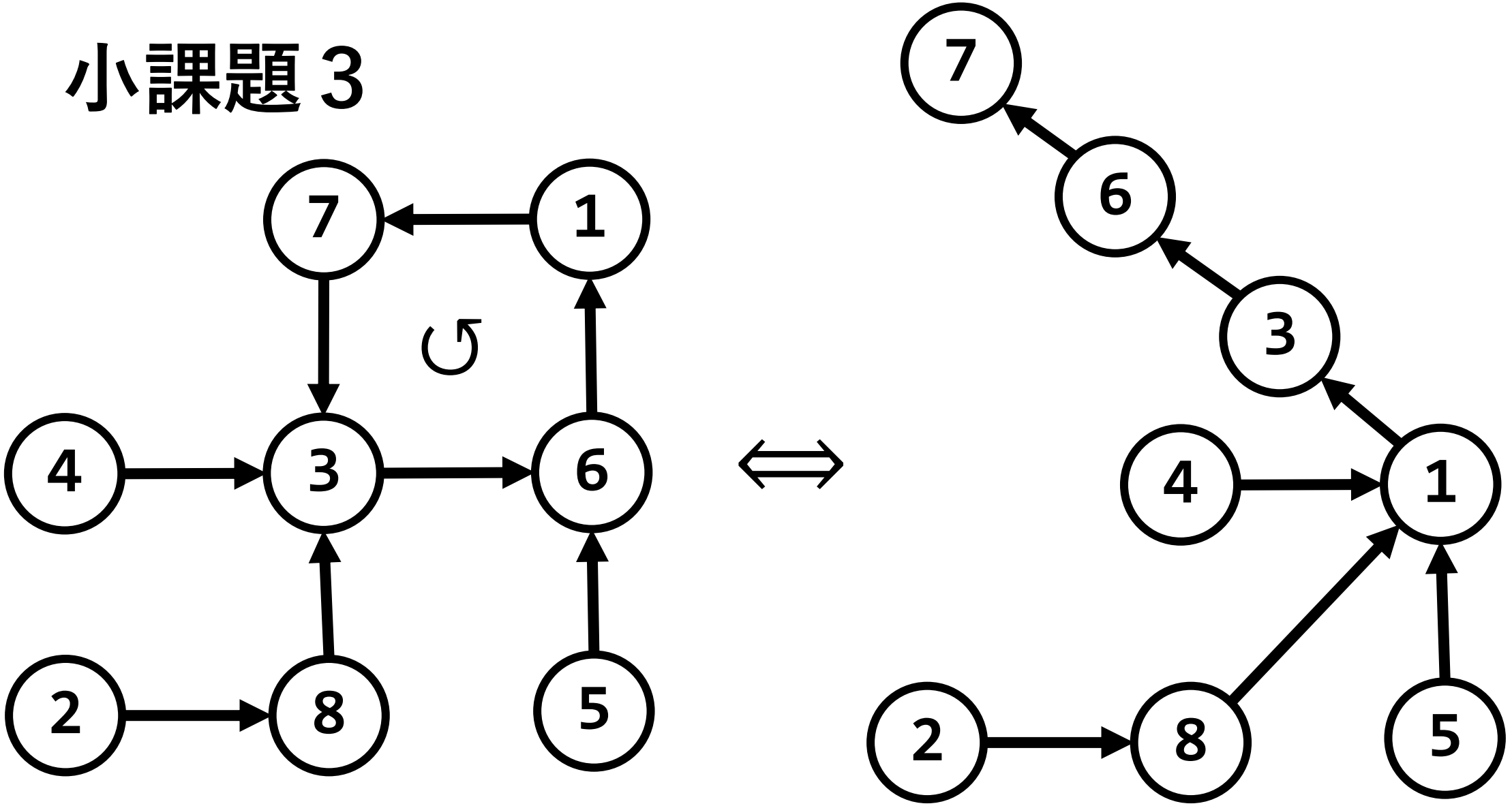
小課題２

- `map` の `size` が小さい方から大きい方に `merge` を行うようにする
- この処理を用いると計算量は $O(N(\log N)^2)$ に改善される
- 証明は省略
- 「**データ構造をマージする一般的なテク**」を調べてみましょう

小課題 3

- グラフが木ではなくなった
- 一般的な `functional graph` も木の場合に帰着させたい

小課題３



小課題３

- 一般的な `functional graph` も木の場合に帰着させたい
- サイクル部分を降順に並べ、木の部分の `root` の行先をサイクル部分の H が最も小さい頂点にする
- 線形時間で木の場合に帰着可能
- `100`点

別解法

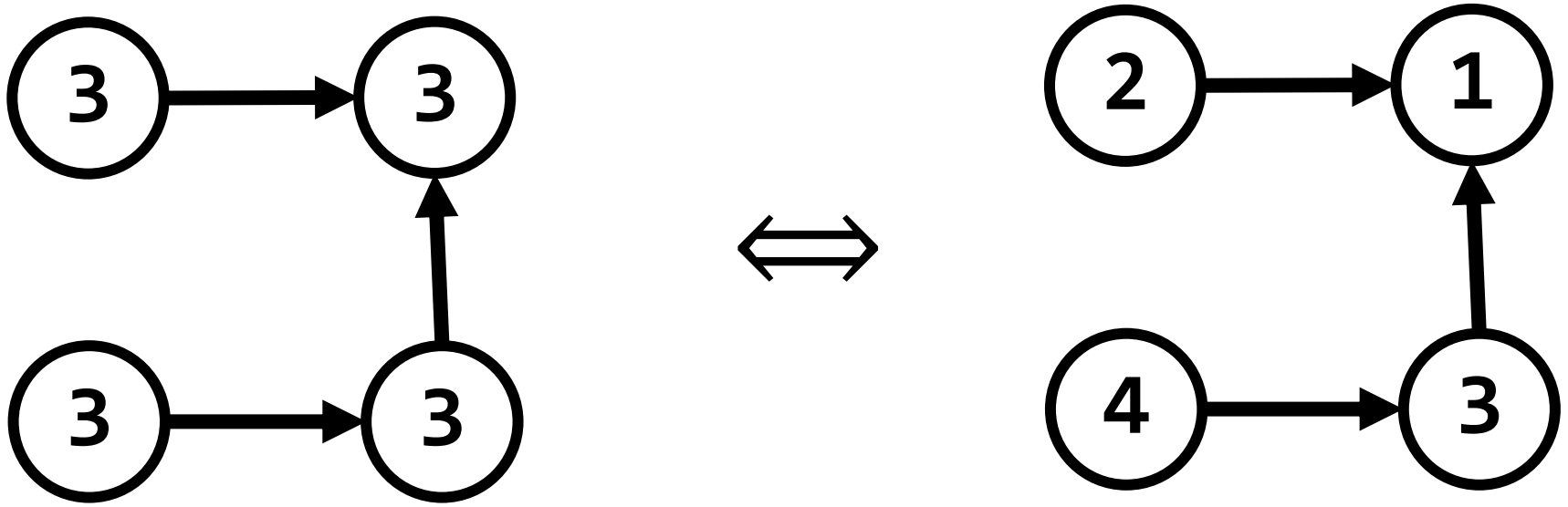
- $O(N\log N)$ で解くことも可能
- $\mathrm{dp}[v][i] :=$ (#) に加え以下を満たす S の中で $\sum_{v\in S} C_v$ の最大値

  $S \subseteq V$(頂点 v の部分木)　and　$\forall v \in S, H_v \geq i$
- $i = N, \ldots, 1$ として $\mathrm{dp}[v][i]$ の値がどのように変化するかを着目してみる

別解法

- H の値は $(1, \ldots, N)$ の permutation の場合に帰着できる

- 同じ値を比較するとき、深い方を大きいとみなす
  深さが一致する場合はどちらでもよい



別解法

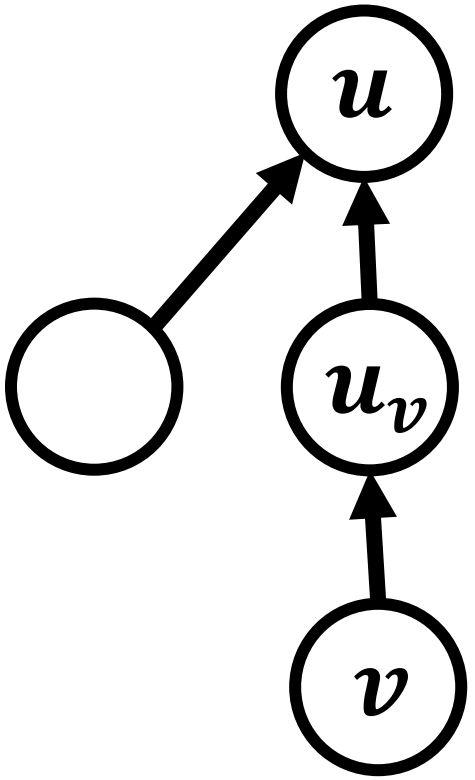
- v を $H_v = i$ を満たす頂点とすると

$$\mathrm{dp}[v][i] = \mathrm{dp}[v][i+1] + C_v$$

$$\mathrm{dp}[u][i] = \mathrm{dp}[u][i+1]\ (u \text{ は } v \text{ の祖先でない})$$

$$\mathrm{dp}[u][i] = \max\left(\mathrm{dp}[u][i+1], \sum_{w \in N_u} \mathrm{dp}[w][i]\right)\ (u \text{ は } v \text{ の祖先})$$

別解法



- v の祖先のみ着目すればよい

$$\mathrm{dp}[u][i] = \max\left(\mathrm{dp}[u][i+1], \sum_{w \in N_u} \mathrm{dp}[w][i]\right)$$

- $b[u][i] \coloneqq \mathrm{dp}[u][i] - \sum_{w \in N_u} \mathrm{dp}[w][i]$ と定義すると

$$b[u][i] = \max\left(\mathrm{dp}[u][i+1] - \sum_{w \in N_u} \mathrm{dp}[w][i], 0\right)$$

$$= \max\left(b[u][i+1] - \sum_{w \in N_u} (\mathrm{dp}[w][i] - \mathrm{dp}[w][i+1]), 0\right)$$

$$= \max(b[u][i+1] - (\mathrm{dp}[u_v][i] - \mathrm{dp}[u_v][i+1]), 0)$$

別解法

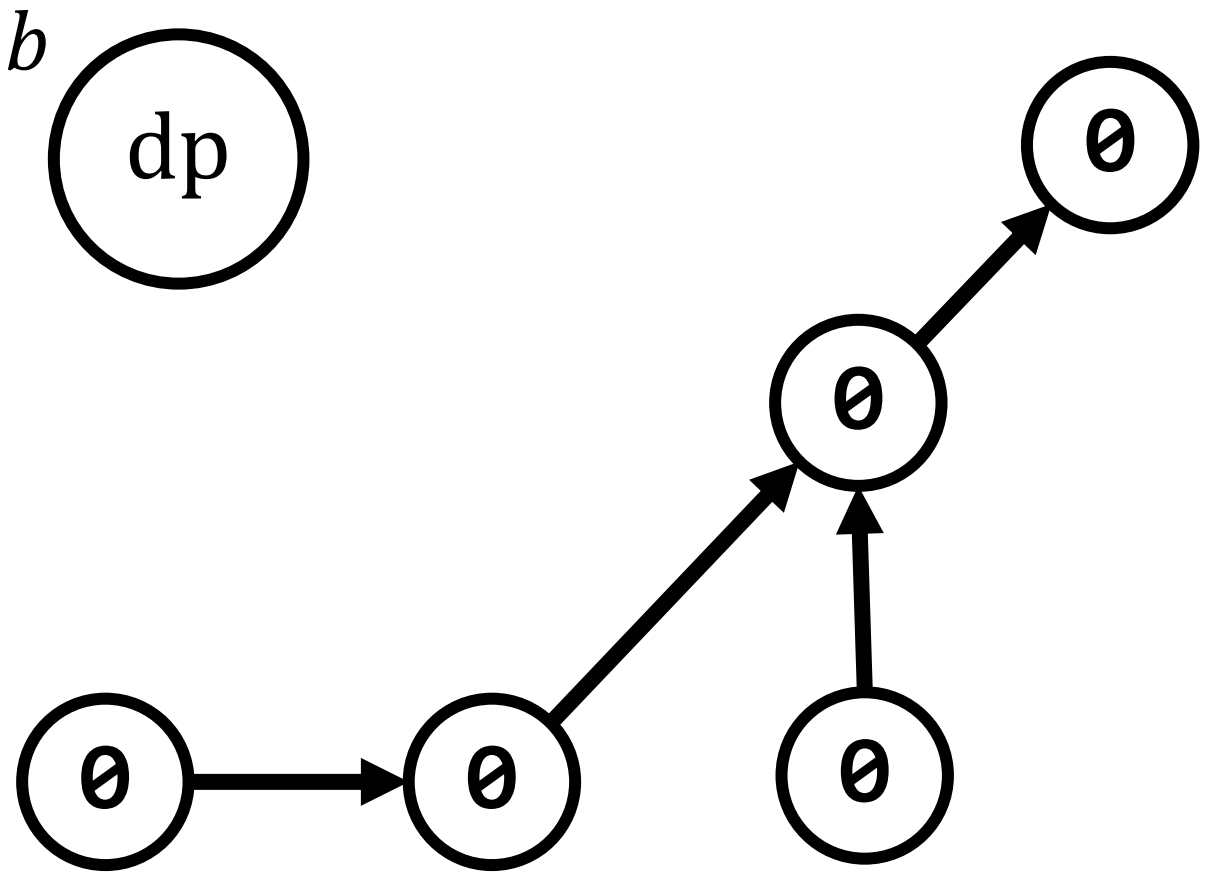
- $s[u][i] \coloneqq \mathrm{dp}[u][i] - \mathrm{dp}[u][i+1]$ と定義すると

$$b[u][i] = \max(b[u][i+1] - s[u_v][i], 0)$$

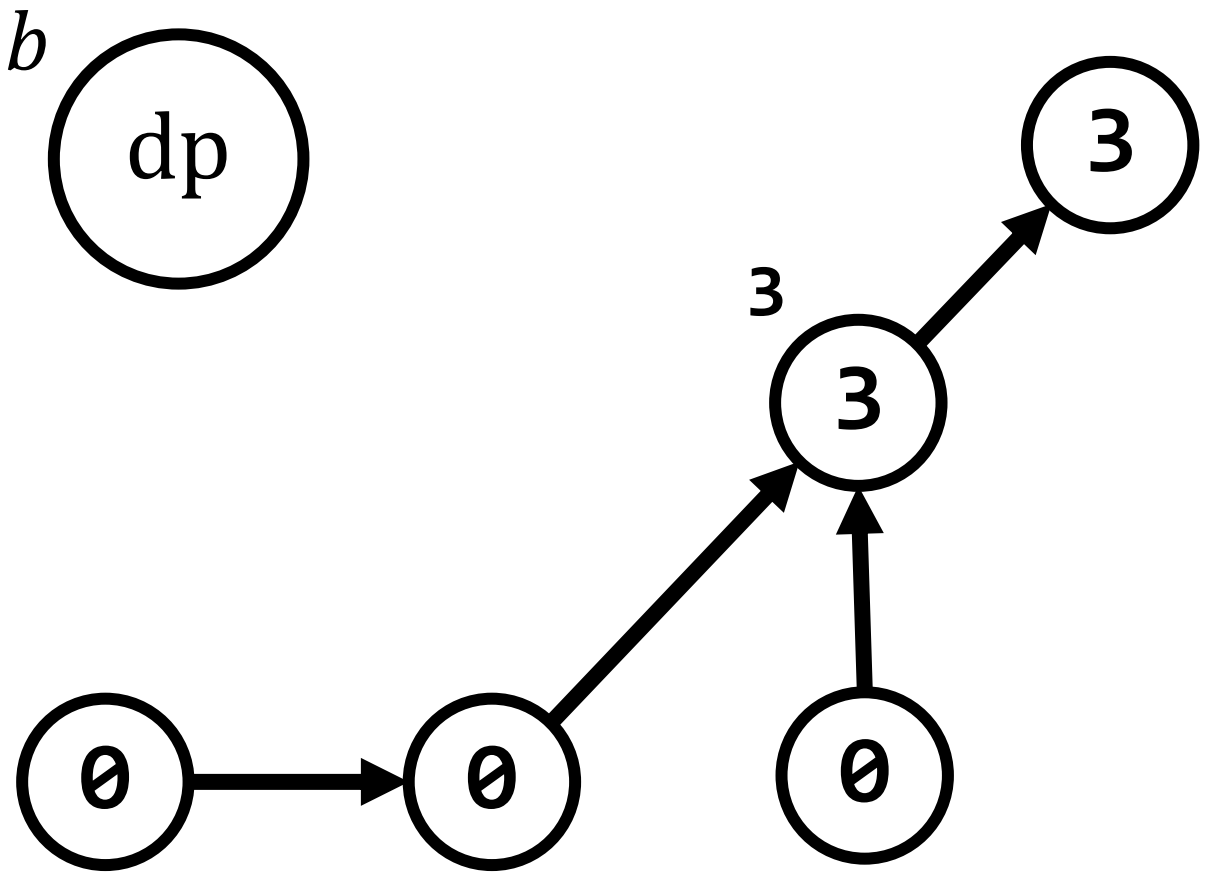
$$s[u][i] = b[u][i] - b[u][i+1] + \sum_{w \in N_u} (\mathrm{dp}[w][i] - \mathrm{dp}[w][i+1])$$

$$= b[u][i] - b[u][i+1] + s[u_v][i]$$

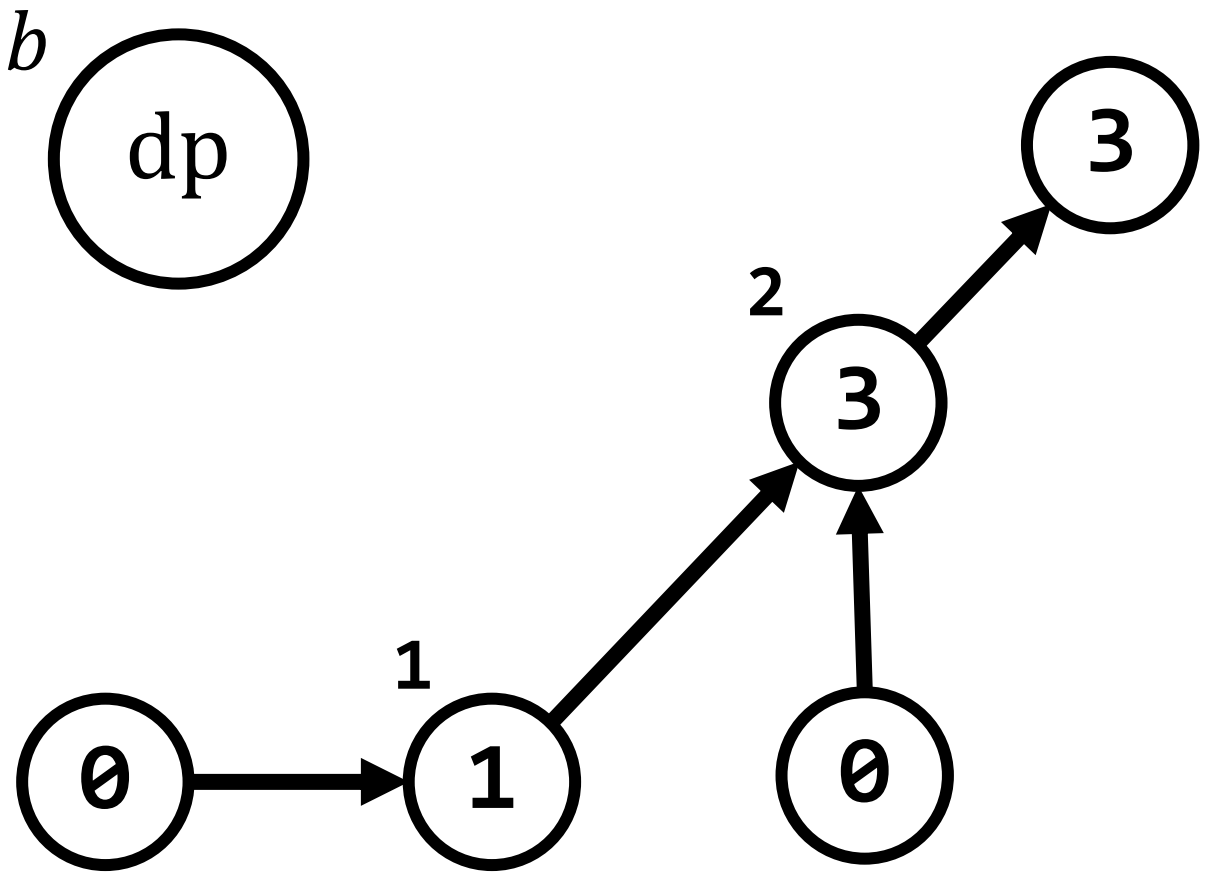
別解法



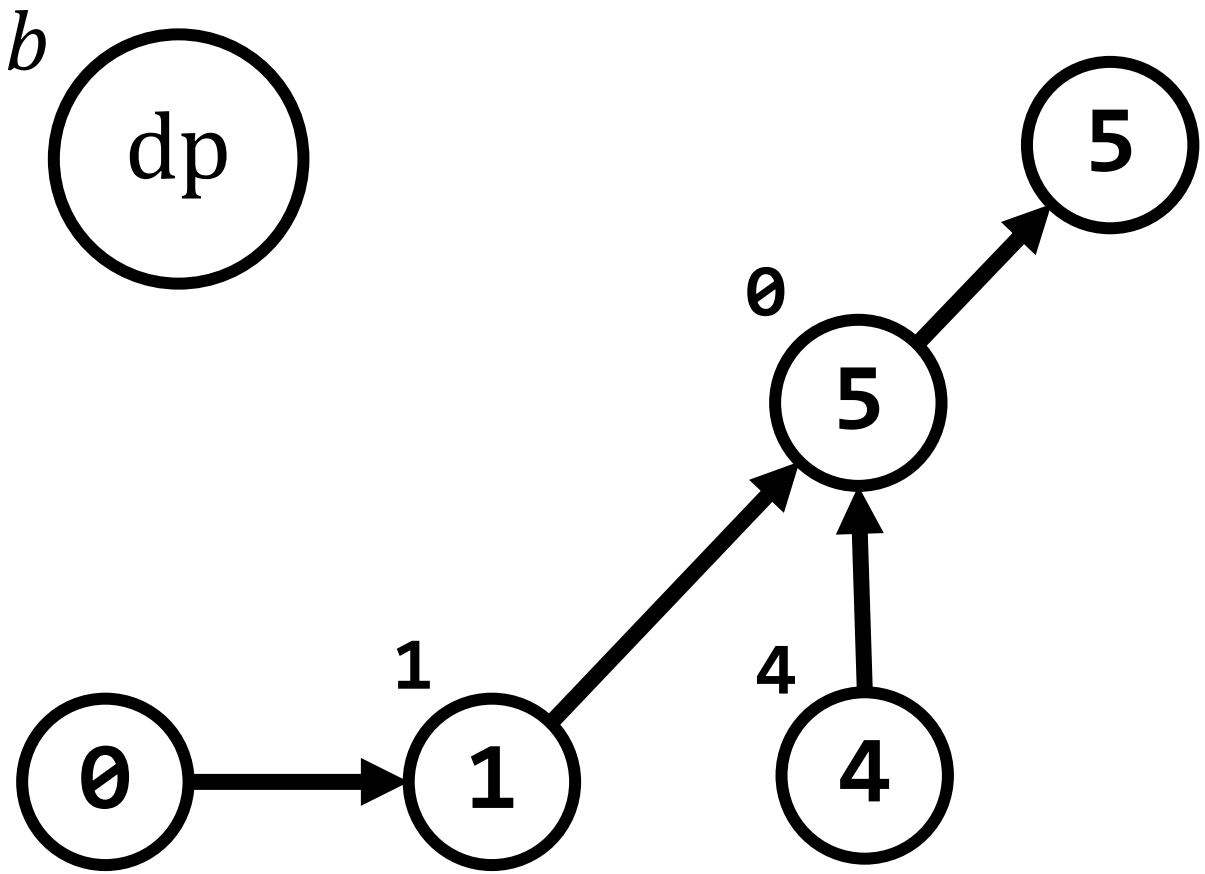
別解法



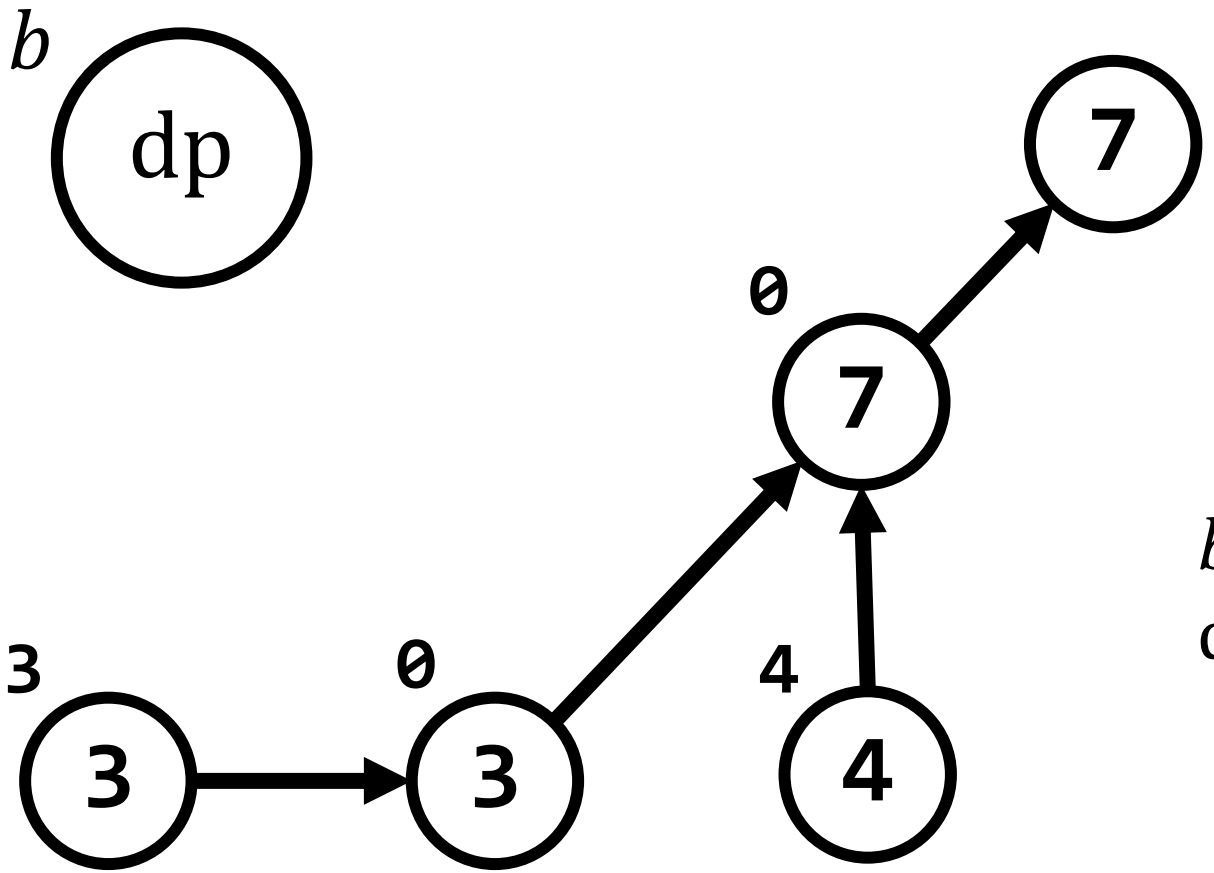
別解法



別解法



別解法



b の値を吸収するように
dp の値が増加する

別解法

- $\mathrm{dp}[root][1]$ だけ計算すれば十分

- v の祖先であり $b[u][i]$ の値が正であるような u の中で最も深い頂点を高速に計算できるようにしたい

別解法

- T を頂点集合として、以下のクエリを高速に処理したい

  Add : T に v を追加

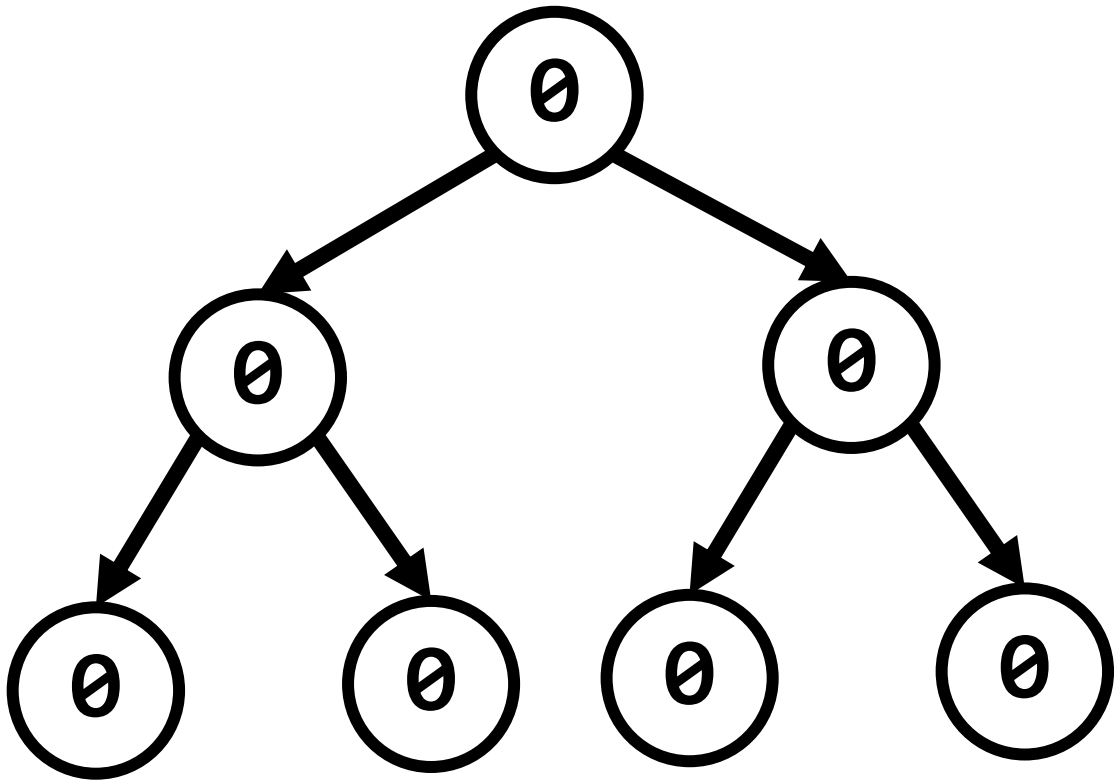
  Erase : T から v を削除

  Access

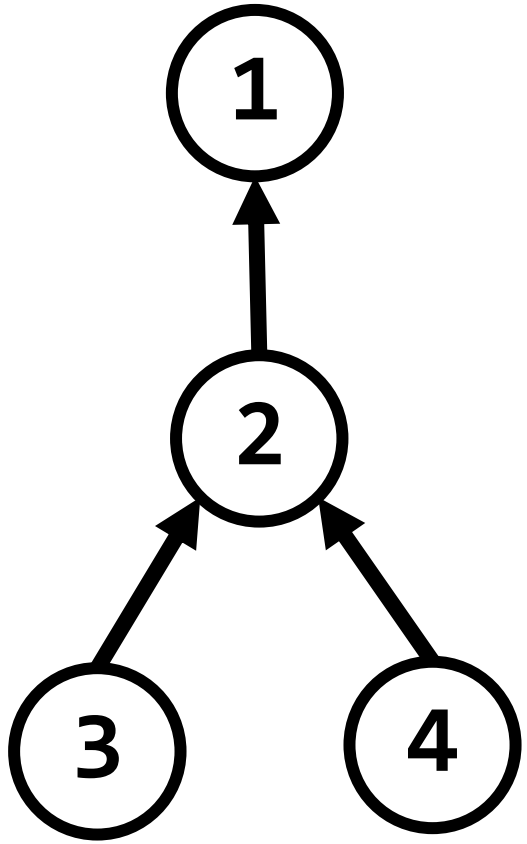
  v の祖先(v を含む)であり T に含まれている頂点で最も深い頂点を計算

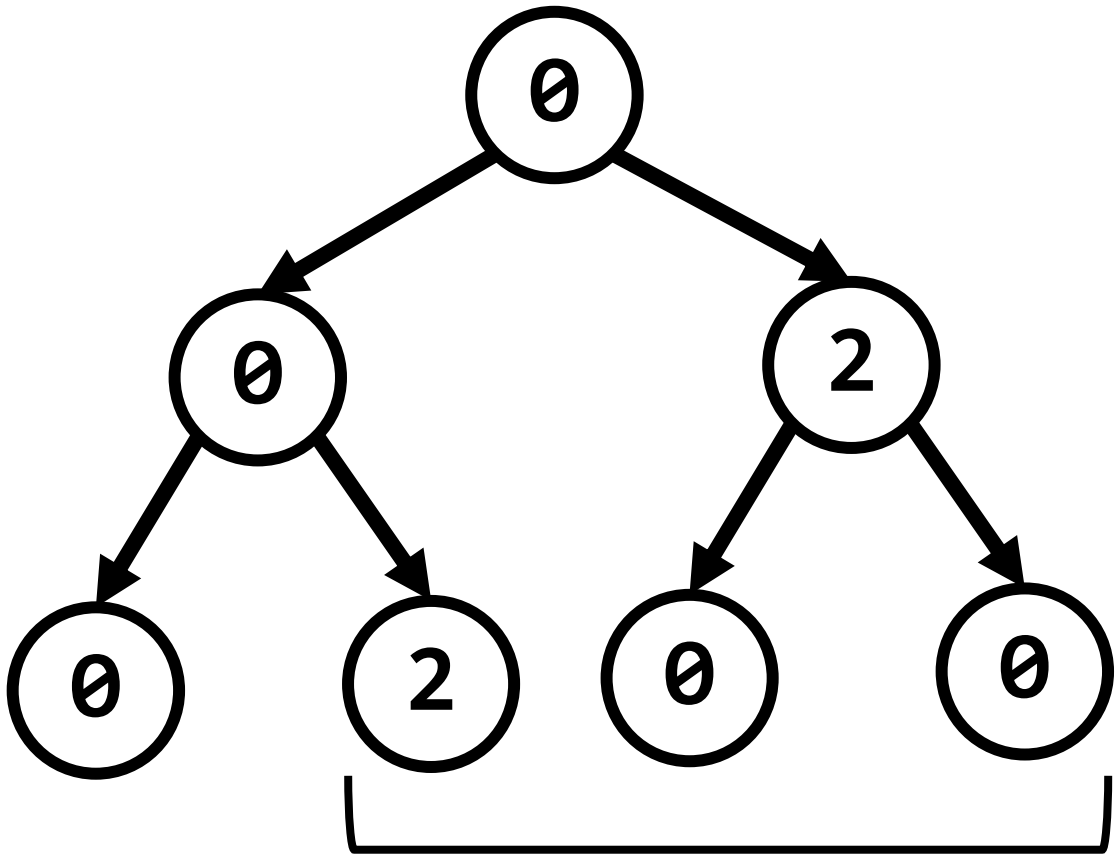
別解法

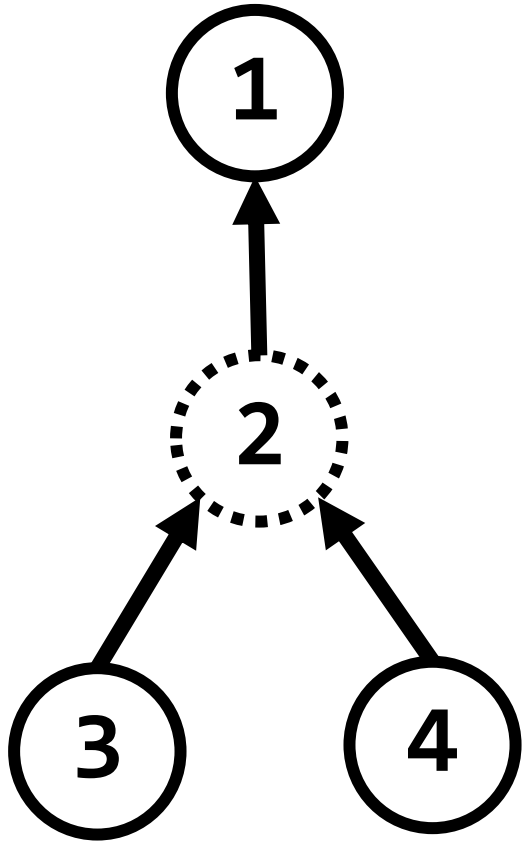
- Add, Erase は部分木内に影響を及ぼす
- 頂点をDFS順序に並べることで部分木を区間に変換することができる
- Add は区間に chmax するクエリとみなせる
- Segment tree で効率的に計算できる

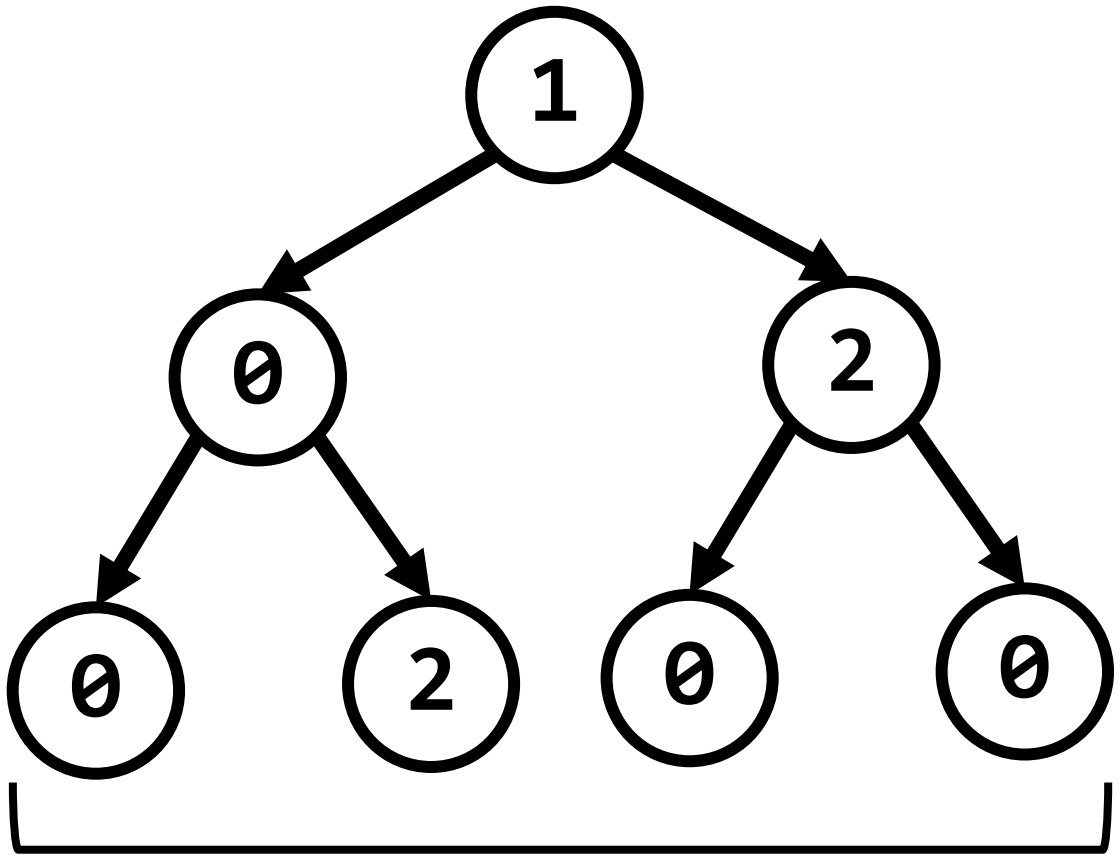


Segment tree

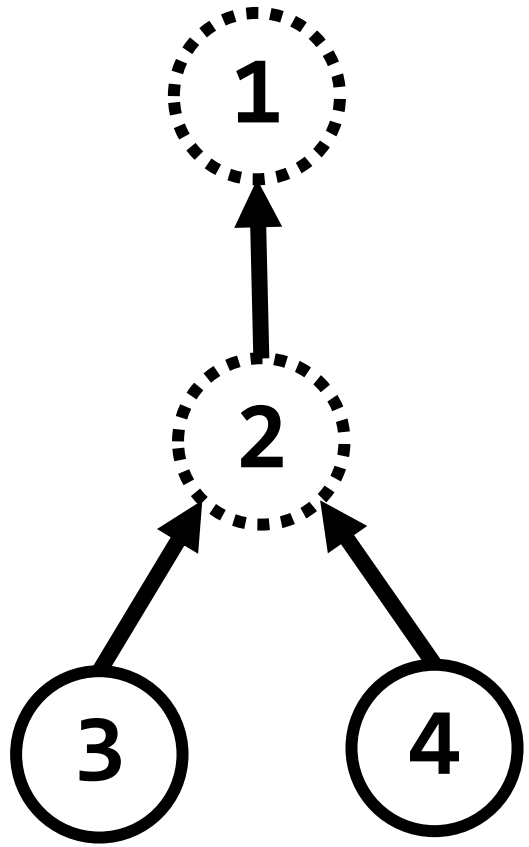


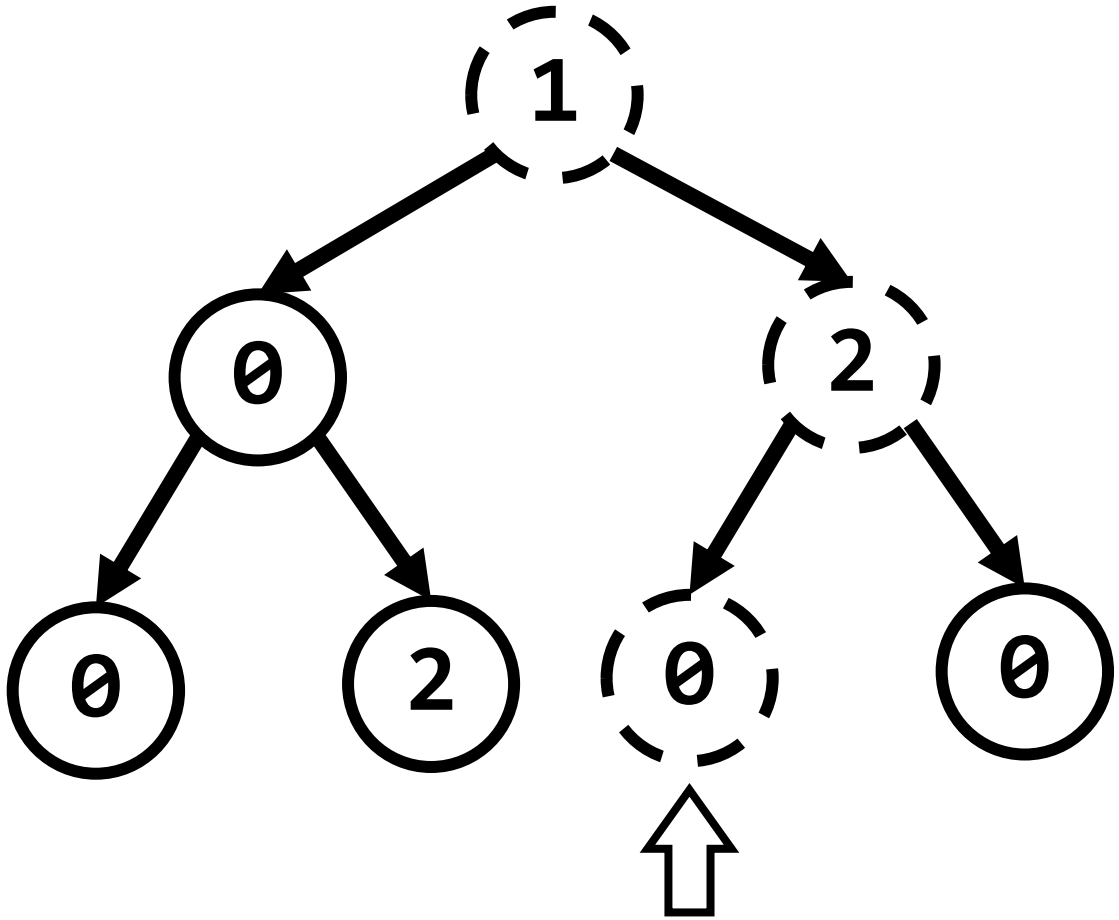

0
0
2
0
2
0
0
Add 2


1
2
3
4

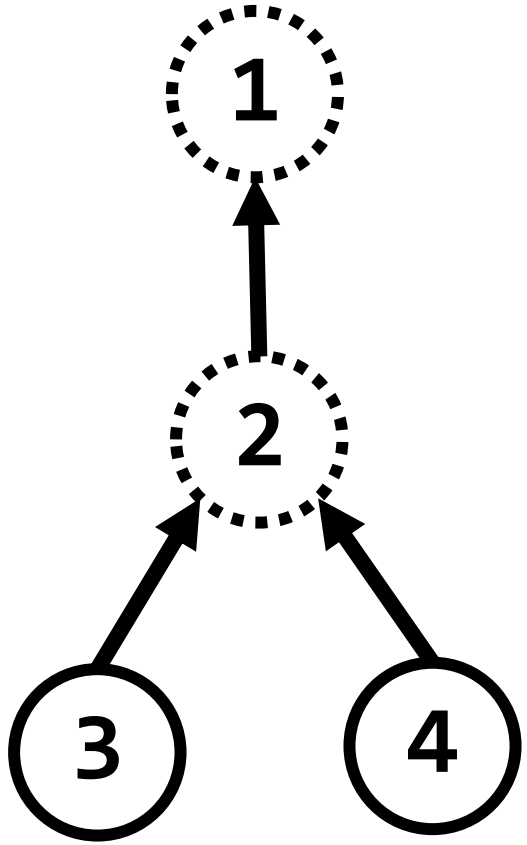


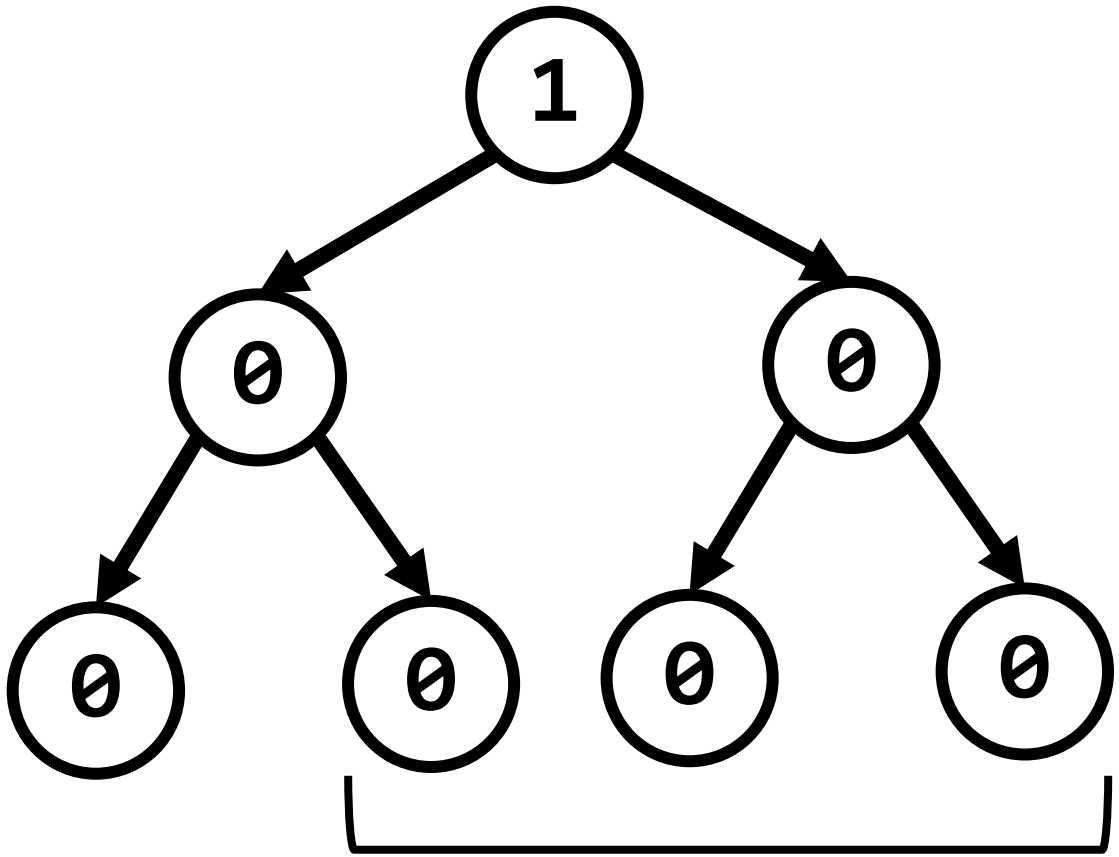
Add 1



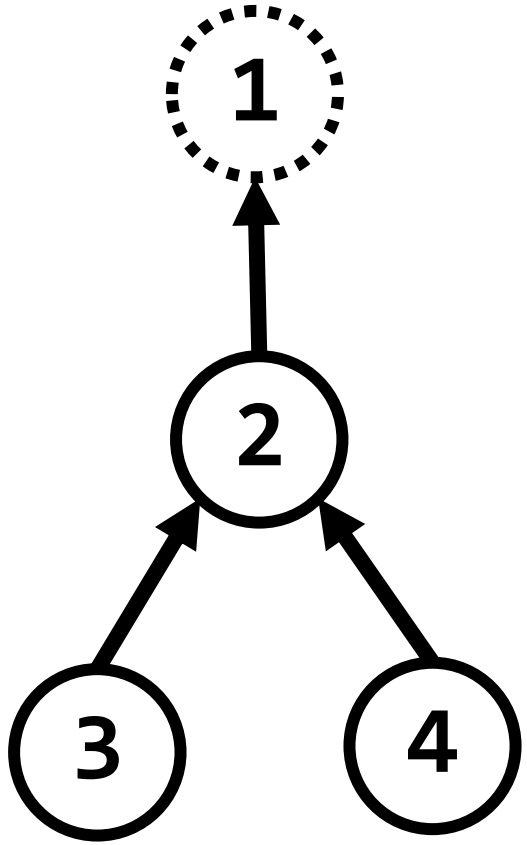


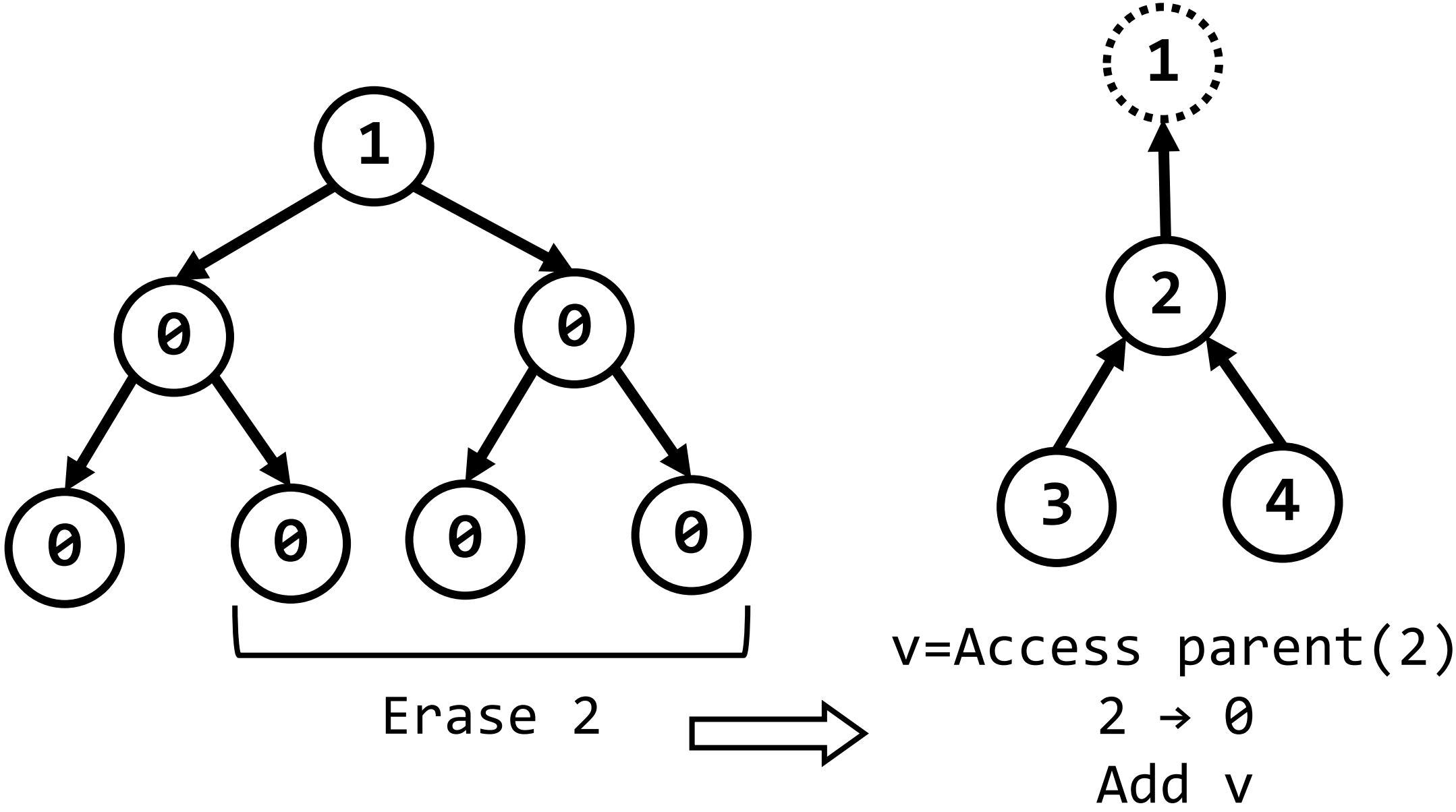
Access 3





Erase 2




1
0
0
0
0
0
0
Erase 2
1
2
3
4
v=Access parent(2)
2 → 0
Add v

別解法

- 各クエリ $O(\log N)$ 計算可能
- 全体の計算量は $O(N\log N)$

得点分布