蟻本輪講　データ構造(P69~86)

2014 / 6 / 11

アルゴリズム研究室　B4　竹内文登

データ構造

- データ構造とは、データの持ち方のこと。
- データの持ち方によって、効率的な操作方法が変わる。

- 例
  - 「配列」
  - 「スタック」
  - 「キュー」
  - 「ヒープ」
  - 「二分探索木」
  - 「Union-Find木」

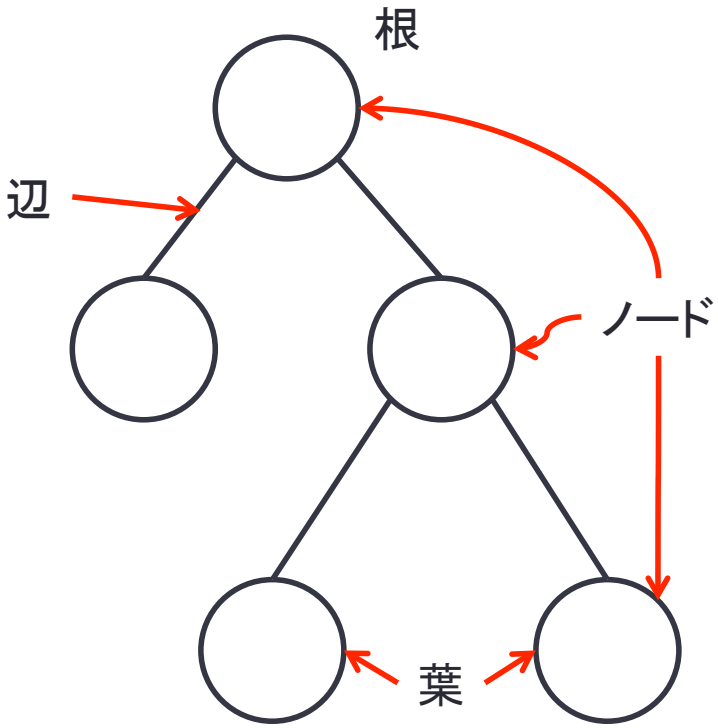
目次

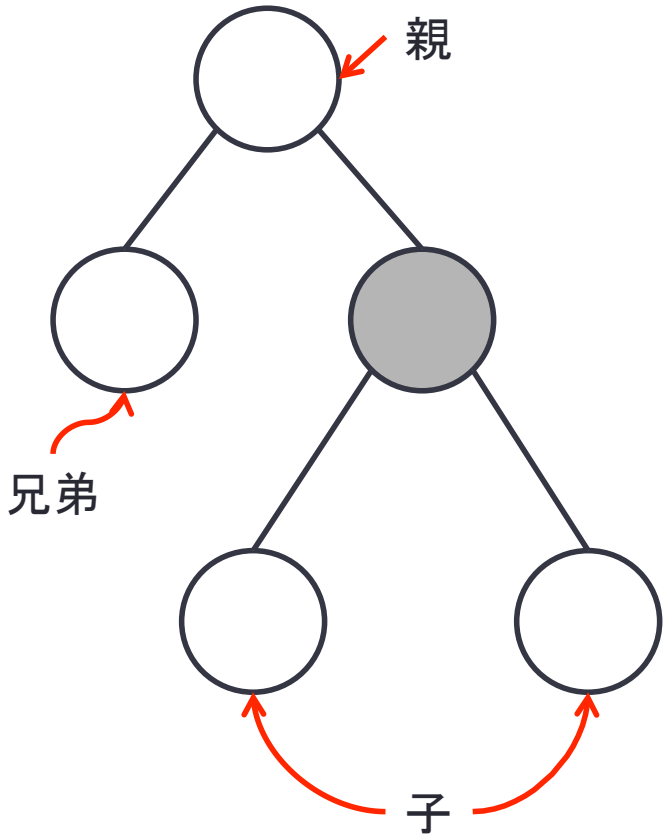
- 木・二分木
- プライオリティキューとヒープ
  - 仕組みと計算量
  - 実装例
  - 問題（2つ）
- 二分探索木
  - 仕組みと計算量
  - 実装例
  - （平衡二分木）
- Union-Find木
  - 仕組みと計算量
  - 実装例
  - 問題

目次

- 木・二分木
- プライオリティキューとヒープ
  - 仕組みと計算量
  - 実装例
  - 問題（2つ）
- 二分探索木
  - 仕組みと計算量
  - 実装例
  - （平衡二分木）
- Union-Find木
  - 仕組みと計算量
  - 実装例
  - 問題

木・二分木





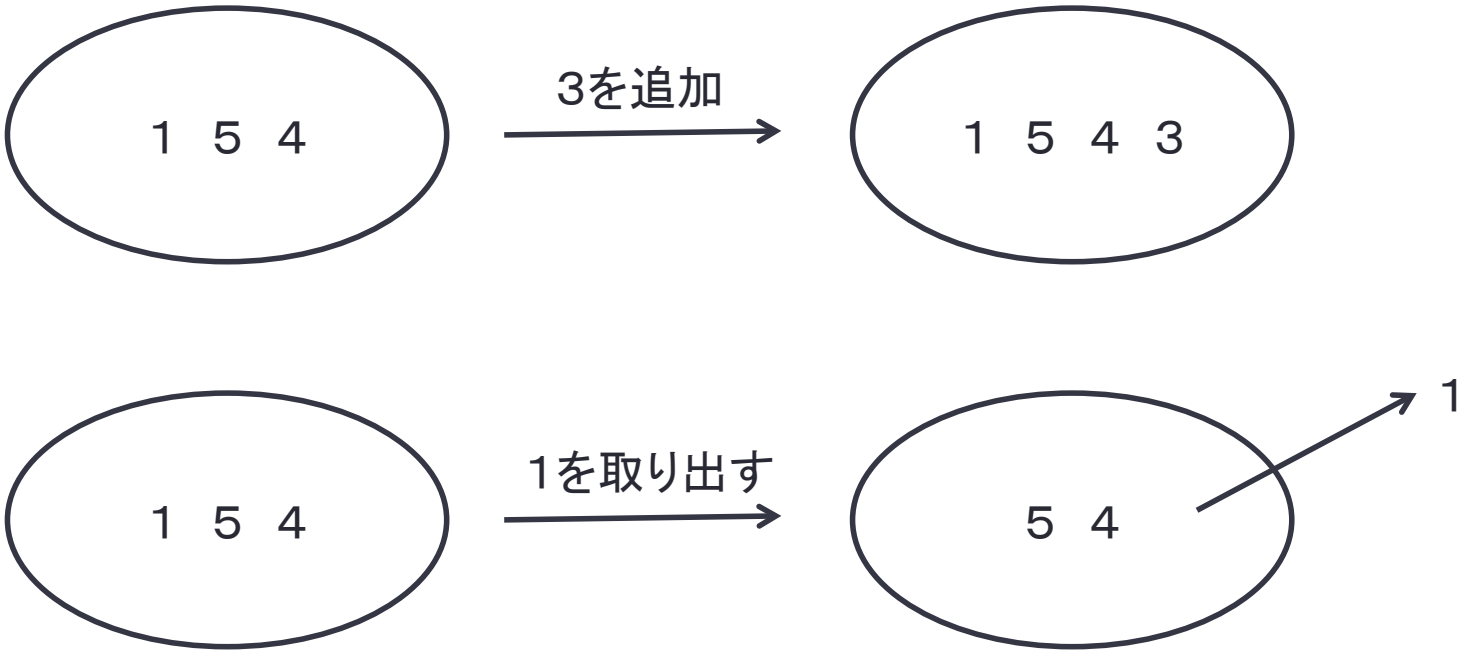
「二分木」とは、すべてのノードについて子が2個以下である木。

目次

- 木・二分木
- プライオリティキューとヒープ
  - 仕組みと計算量
  - 実装例
  - 問題（2つ）
- 二分探索木
  - 仕組みと計算量
  - 実装例
  - （平衡二分木）
- Union-Find木
  - 仕組みと計算量
  - 実装例
  - 問題

プライオリティキュー（順位キュー）

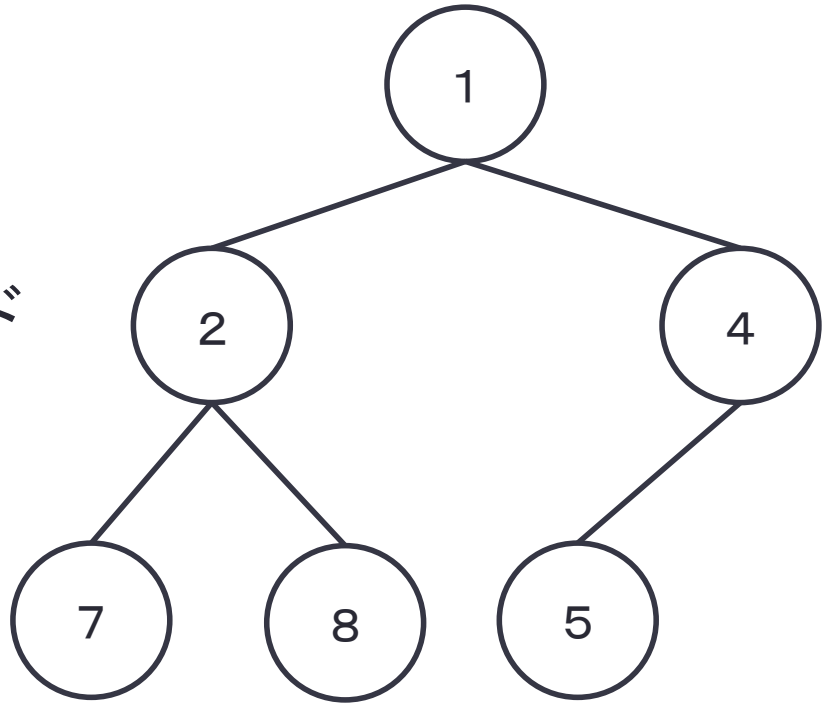
- 数を追加する。
- 最小の数値を取り出す（値を取得し、削除する）。



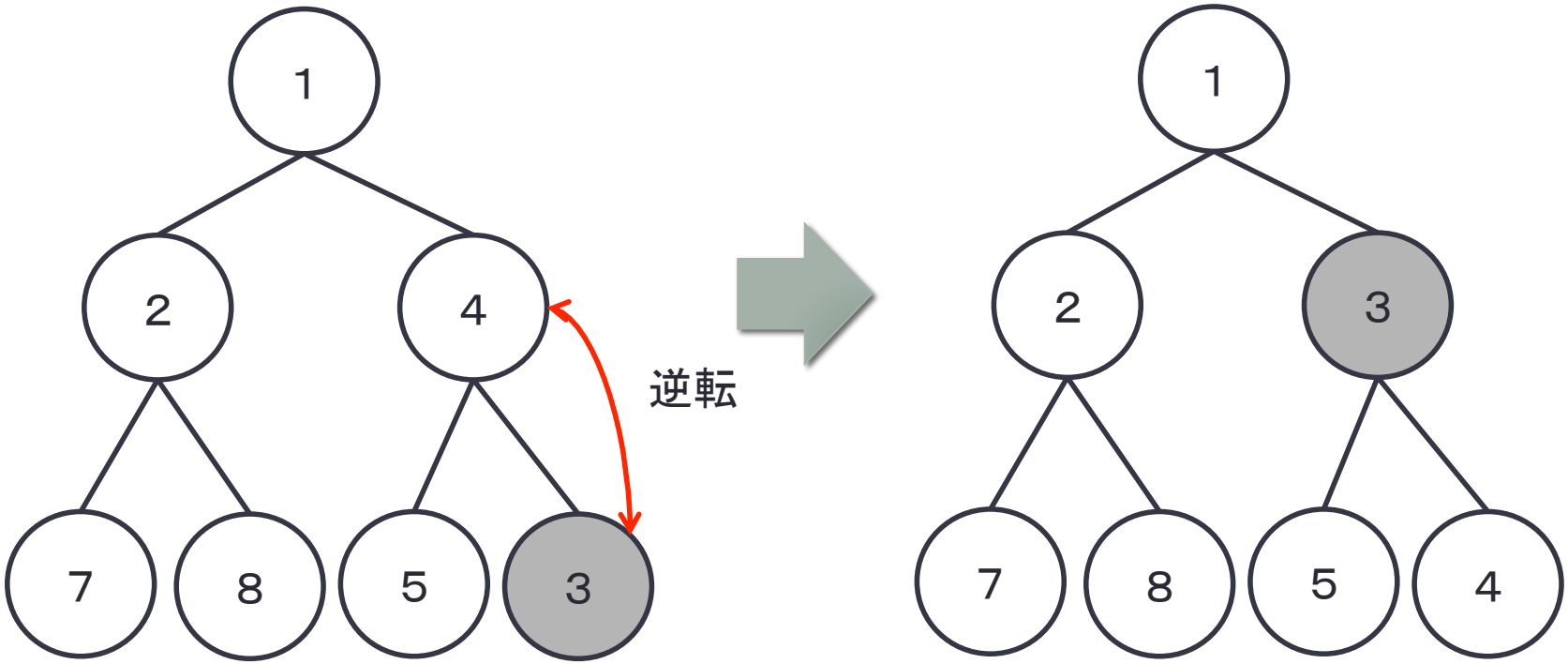
ヒープ

- 二分木を用いて実装したもの。

- 子の数字は親の数字より大きい
- 上から下へ、左から右へ順にノードが詰まっている



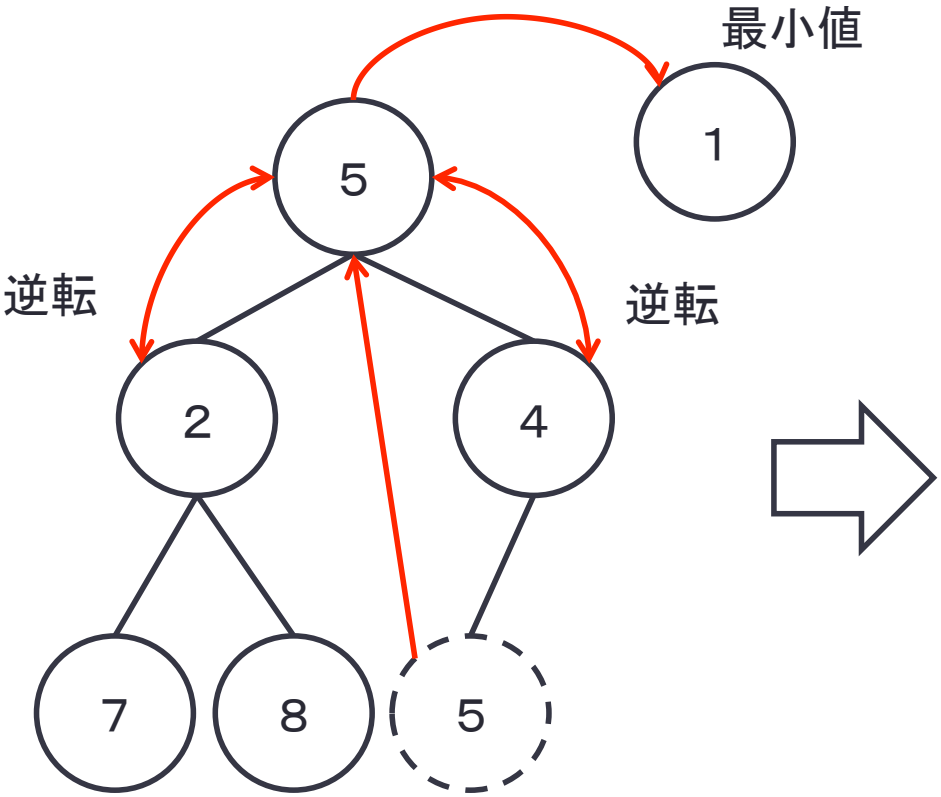
ヒープ　ー数字の追加ー

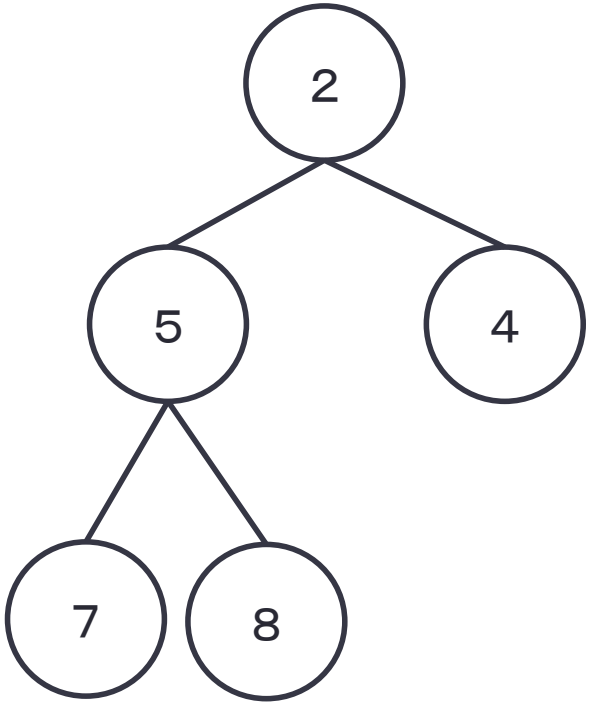


一番下の段の出来るだけ左に
ノードを追加。

逆転が無くなるまで上に
上げて行く

ヒープ　ー最小値の取り出しー





最小値（根）を取り出し、
最後尾のノードを根にコピーする。

逆転が無くなるまで下に下げる。
2つの子が両方逆転している時は、
小さい方と交換。

ヒープの操作の計算量

- 数字の追加

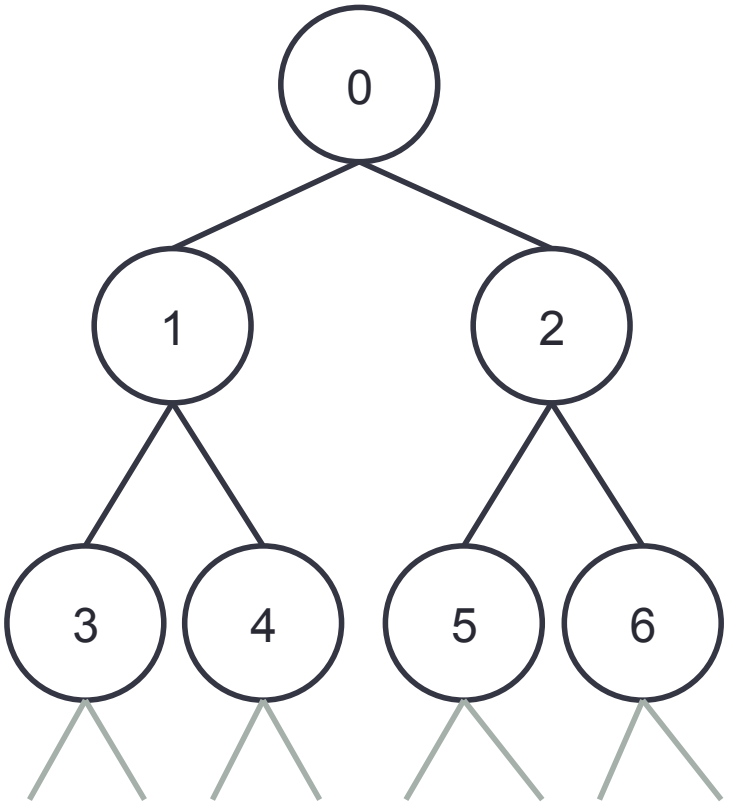
最悪の場合、根まで交換が必要。
➡木の深さに比例した時間　＝　O(log n)

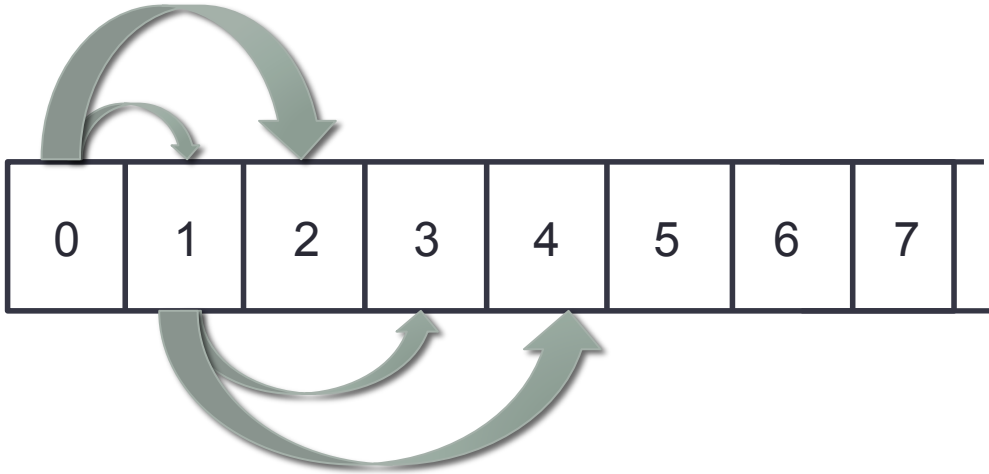
- 最小値の取り出し

最悪の場合、一番下の段まで交換が必要。
➡木の深さに比例した時間　＝　O(log n)

ヒープの実装例

- 二分木をポインタで表現するのではなく、配列で実装。
  - 左の子は自分の番号×2+1
  - 右の子は自分の番号×2+2





ヒープの実装例

- 蟻本p71-72 参照
  - void push(int x) ： 数の追加
  - int pop() ： 最小値の取り出し

- 標準ライブラリ（C＋＋）
  - 最大値から出てくるので注意！！

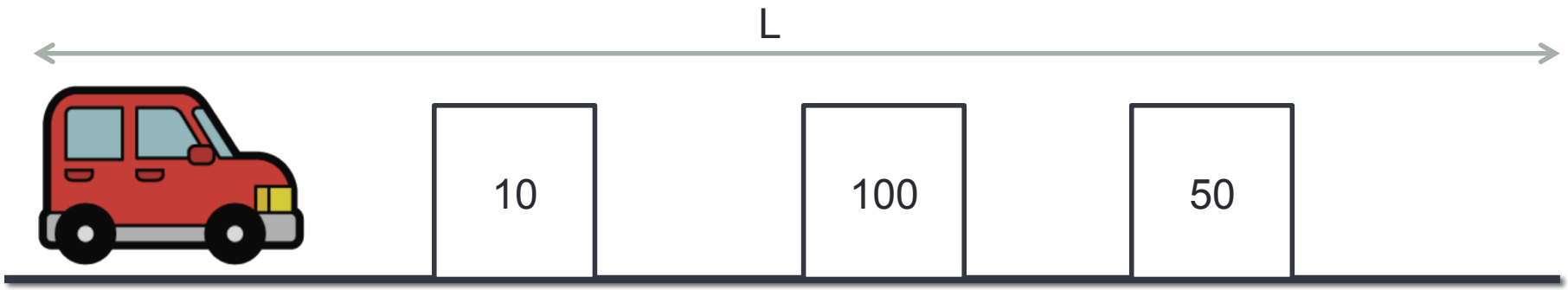
- 標準ライブラリ （C＋＋の場合）

```
#include <queue>
#include <iostream>
using namespace std;

int main(){
	priority_queue<int> pque;

	pque.push(3);			←3の追加
	pque.push(5); ;			←5の追加

	cout << pque.top(); ;		←の追加

	pque.pop();

	return 0;
}
```

問題　Expedition(POJ 2431)

- 距離Lの道を移動する。
- はじめガソリンはP積まれている。
- 距離1走るとガソリンを1消費。
- 途中N個のガソリンスタンドがある。
- 各スタンド ｉは、距離A_iの位置にあり、B_iのガソリンを補給可能。
- トラックのガソリンタンクに制限はない。
- このとき、車は移動を完了できるか？出来る場合は最小の補給回数を出力し、出来ない場合は-1を出力せよ。

- 制約
  $1 \leqq N \leqq 10000,\ 1 \leqq L \leqq 1000000,\ 1 \leqq P \leqq 1000000,\ 1 \leqq A_i < L,\ 1 \leqq B_i \leqq 100$



解法

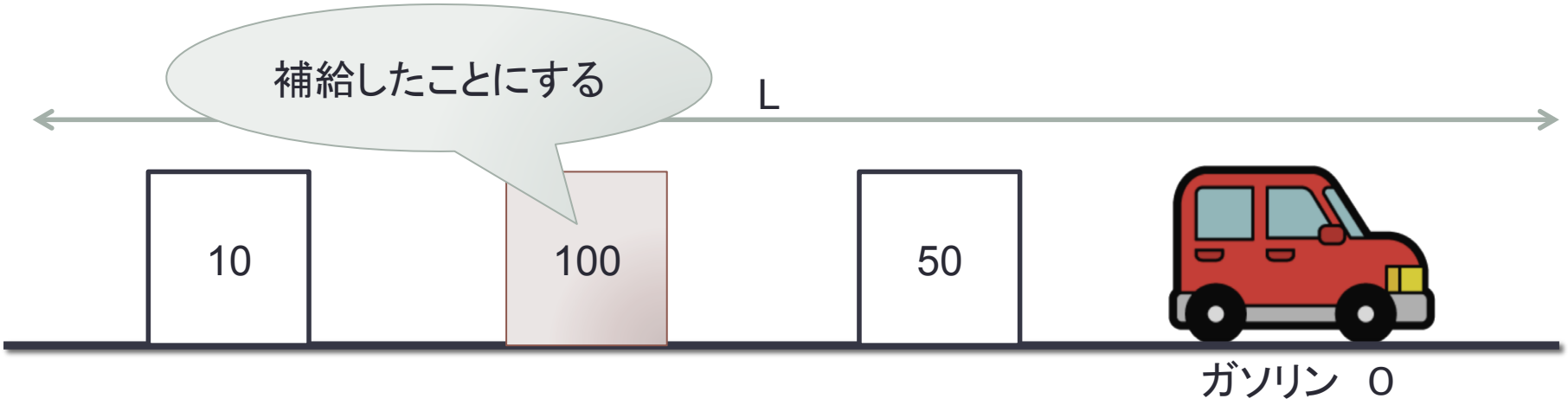
- 愚直な解法

  N個のガソリンスタンドに対して補給するかしないかを全探索

  ➡O(2^N)の計算量。

- 効率的な解法

  ガソリンが無くなったとき、今まで通ったガソリンスタンドのうち、(補給していない)補給できる量の最も大きいスタンドで補給すればよい。



解法

- プライオリティキューを用いて実装
  - ガソリンスタンドを通過した時に、プライオリティキューにB_iを追加
  - 燃料タンクが空になったとき、
    - プライオリティが空であれば、到達できない。
    - プライオリティキューから最大の値を取り出して、そこで補給したことにする。

＊コードはp74参照。

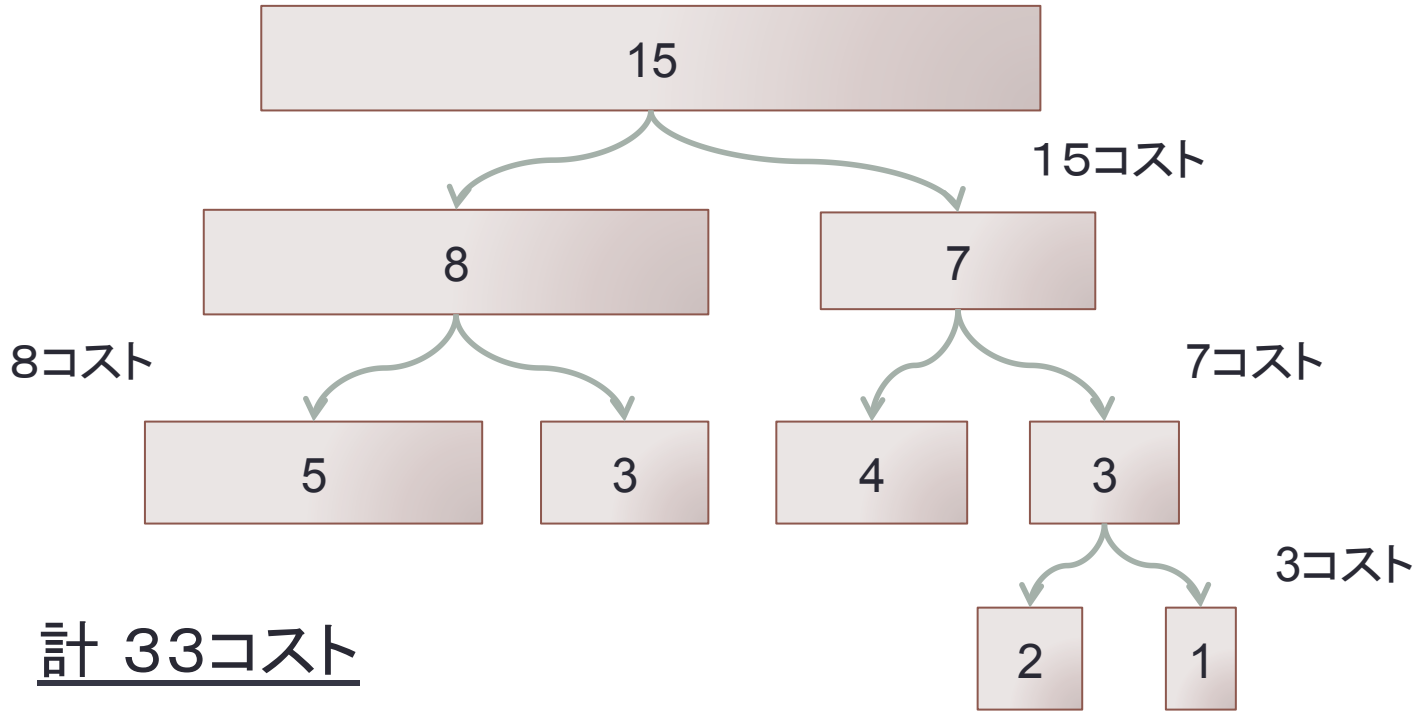
問題　Fence Repair

- 長さL_1+ … +L_NをN個に切り出す。
- 切り出す板の長さは、L_1, … , L_N
- 板を切断する際、板の長さ分のコストがかかる。

- 最小でどれだけのコストで切り出すことができるか？

- 制約
  - $1 \leqq N \leqq 20000$, $1 \leqq L_i \leqq 50000$

問題　Fence Repair



計 33コスト

解法

- 愚直な解法
  - O(N^2)の計算量。

- 効率的な解法
  - 板の集合から最も短い2つの板を取り出し、長さが和になる新しい板を板の集合に追加すればよく、プライオリティキューで実装可能。
  - O(log N)の操作を、O(N)回実行　➡　O(N log N)

  - ＊コードはp75参照。

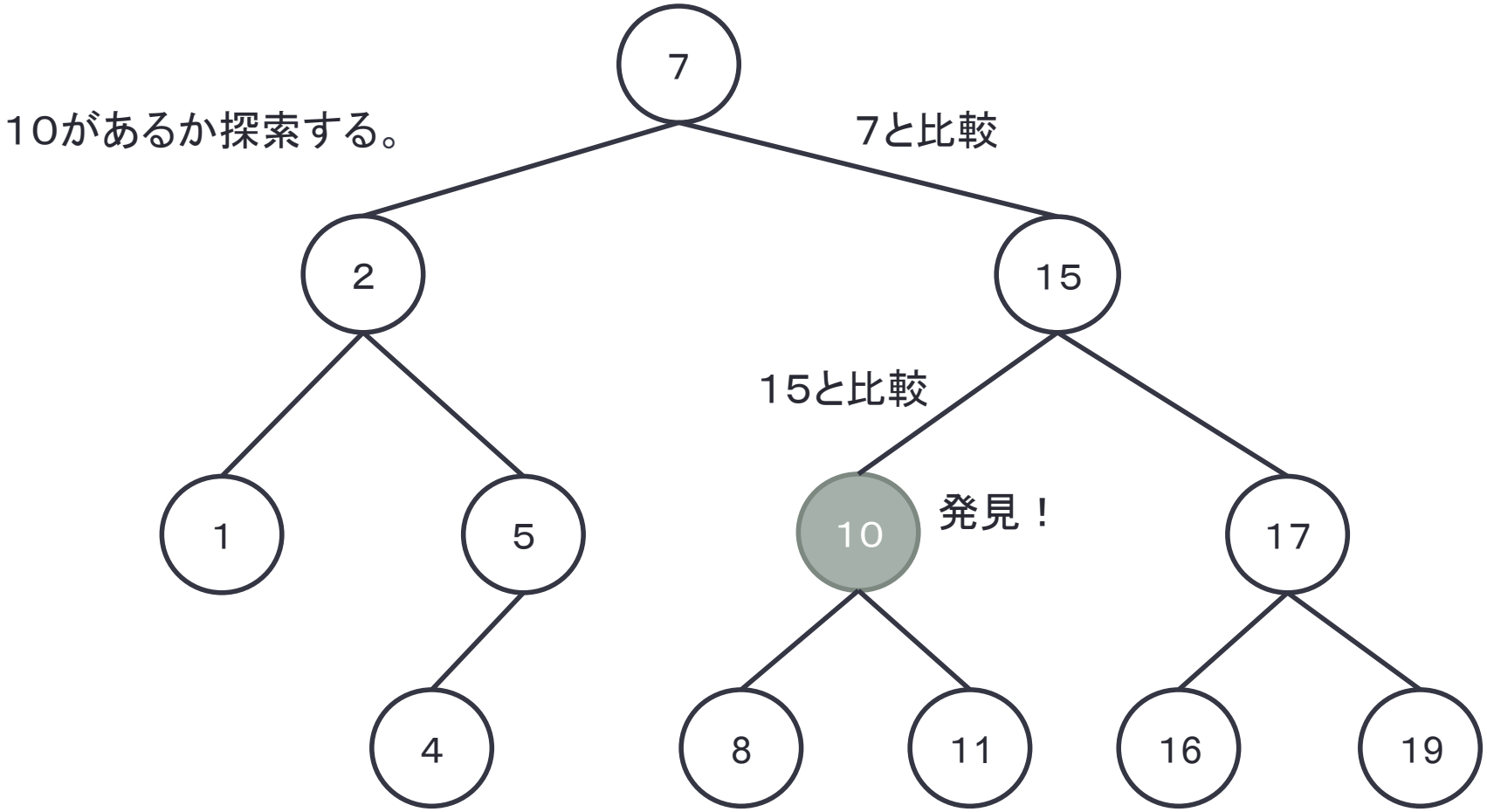
目次

- 木・二分木
- プライオリティキューとヒープ
  - 仕組みと計算量
  - 実装例
  - 問題（2つ）
- 二分探索木
  - 仕組みと計算量
  - 実装例
  - （平衡二分木）
- Union-Find木
  - 仕組みと計算量
  - 実装例
  - 問題

二分探索木

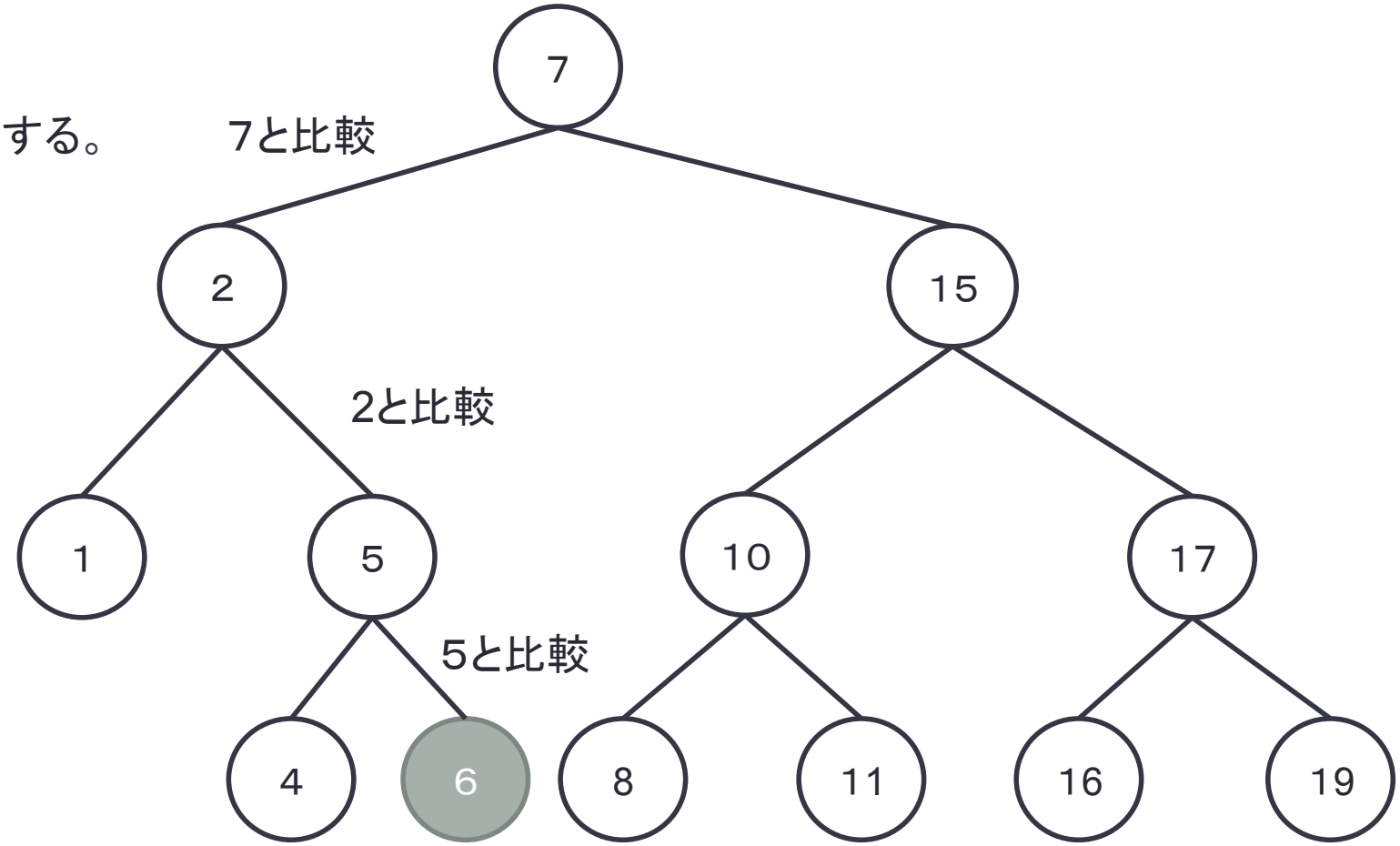
- 二分探索木は以下の操作を効率的に行うデータ構造である。
  - ある数値が含まれているか調べる。(探索)
  - 数値を追加する。
  - ある数値を削除する。
  - ＊実装によって他にも様々な操作が行える。➡応用力が高い。

二分探索木　ー探索ー



二分探索木　―数の追加―

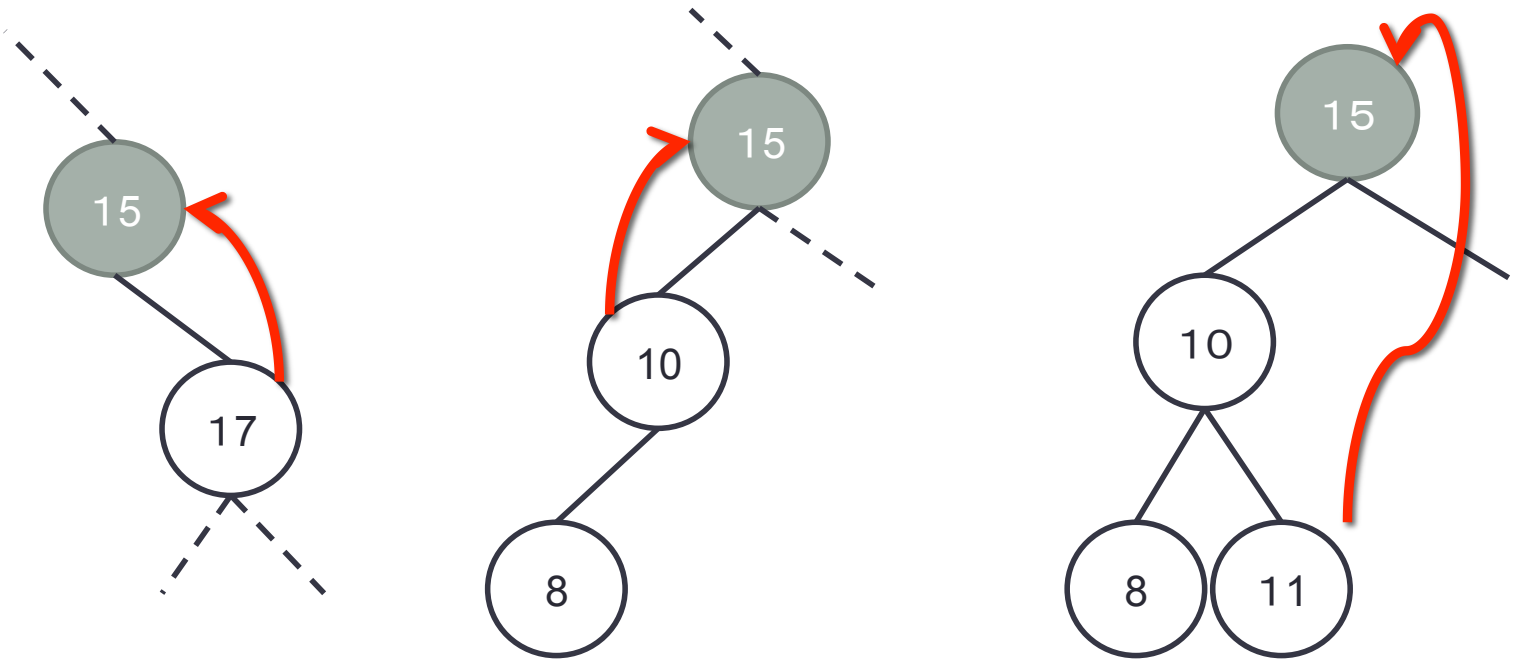
6を追加する。



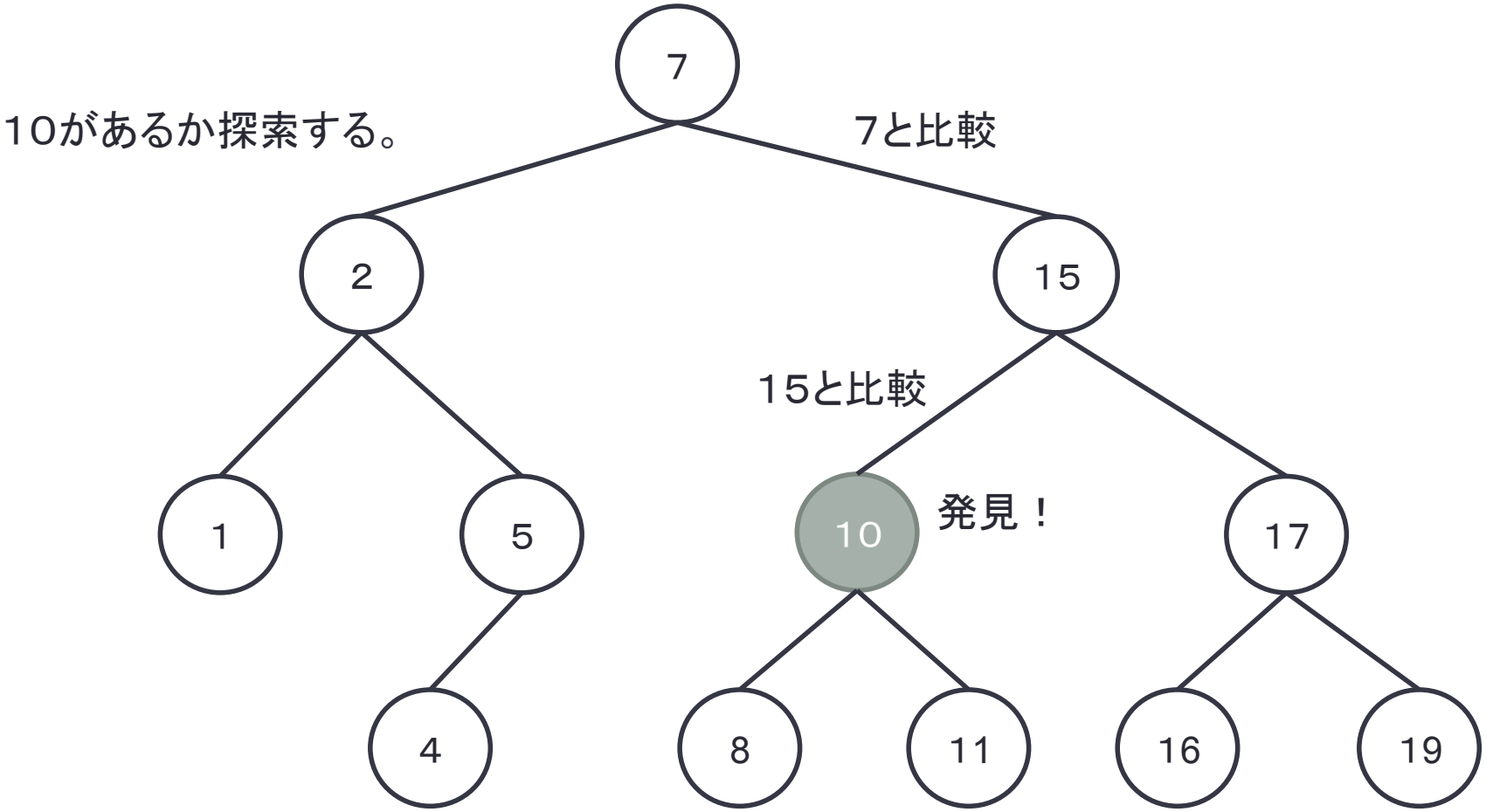
空き➡追加！

二分探索木　ー削除ー

- 以下のような場合分けが必要である。
  1. 削除したいノードが左の子を持っていない場合、右の子を持ってくる
  2. 削除したいノードの左の子が右の子をもっていなければ、左の子を持ってくる
  3. どちらでもなければ、左の子以下で最も大きいノードを持ってくる。



二分探索木　ー削除1ー



二分探索木の計算量

- どの操作も木の深さに比例した時間がかかる。
  - ➡平均的には、ノード数 n に対して、O(log n)時間。

二分探索木の実装例

- ＊蟻本p77-78参照。

- 標準ライブラリ（C＋＋）
  - →set
  - →map

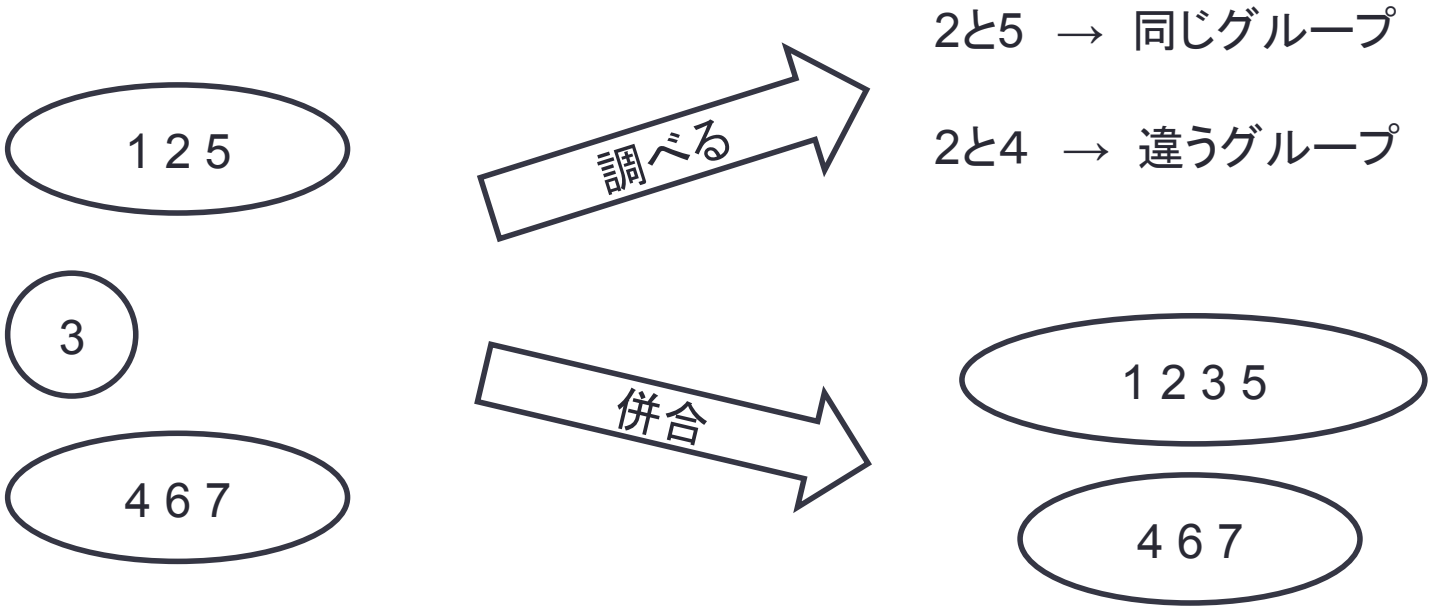
- 平衡二分木
  - 標準ライブラリに備わっている二分木は平衡二分木になっている！

目次

- 木・二分木
- プライオリティキューとヒープ
  - 仕組みと計算量
  - 実装例
  - 問題（2つ）
- 二分探索木
  - 仕組みと計算量
  - 実装例
  - （平衡二分木）
- Union-Find木
  - 仕組みと計算量
  - 実装例
  - 問題

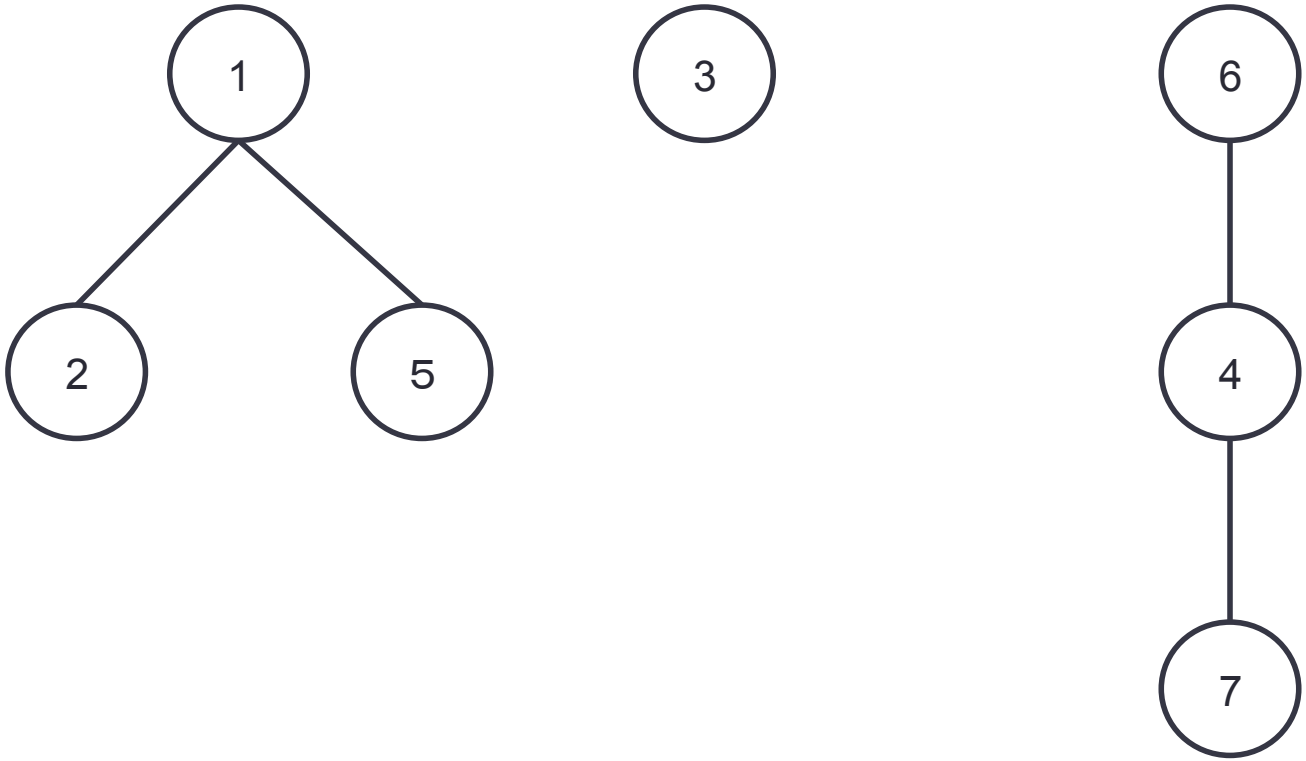
Union-Find木

- Union-Find木は、グループ分けを管理するデータ構造。
- 次の操作が行える。
  - 要素aと要素bが同じグループに属するか調べる
  - 要素aと要素bのグループを併合する
  - （分割はできない）



Union-Find木　ー仕組みー

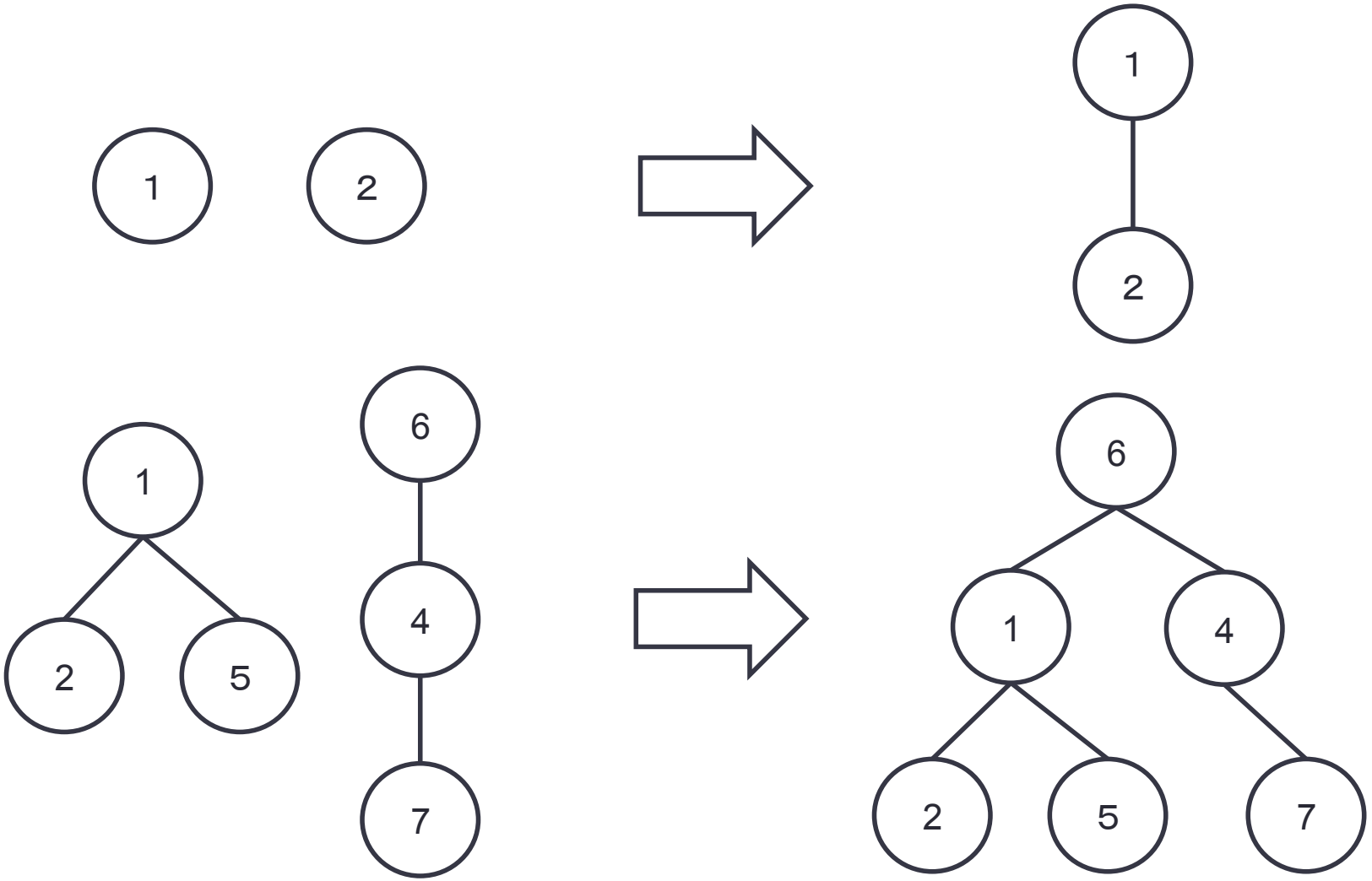
- 1つのグループに対して1つの木を構成する。
- 木の形などは本質的でなく、木であることが重要。



Union-Find木　―初期化―

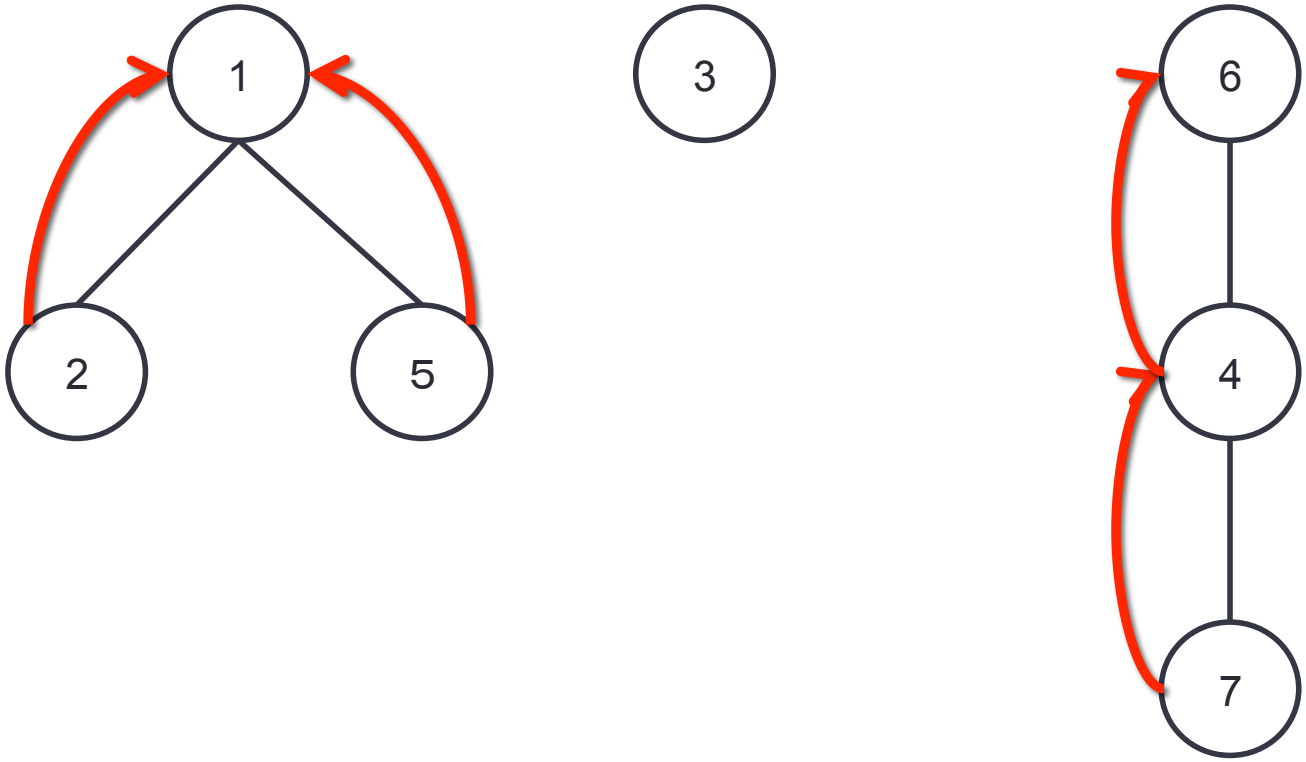
- n個の要素に対して、n個のノードを用意する
- 辺はない

Union-Find木 一併合一



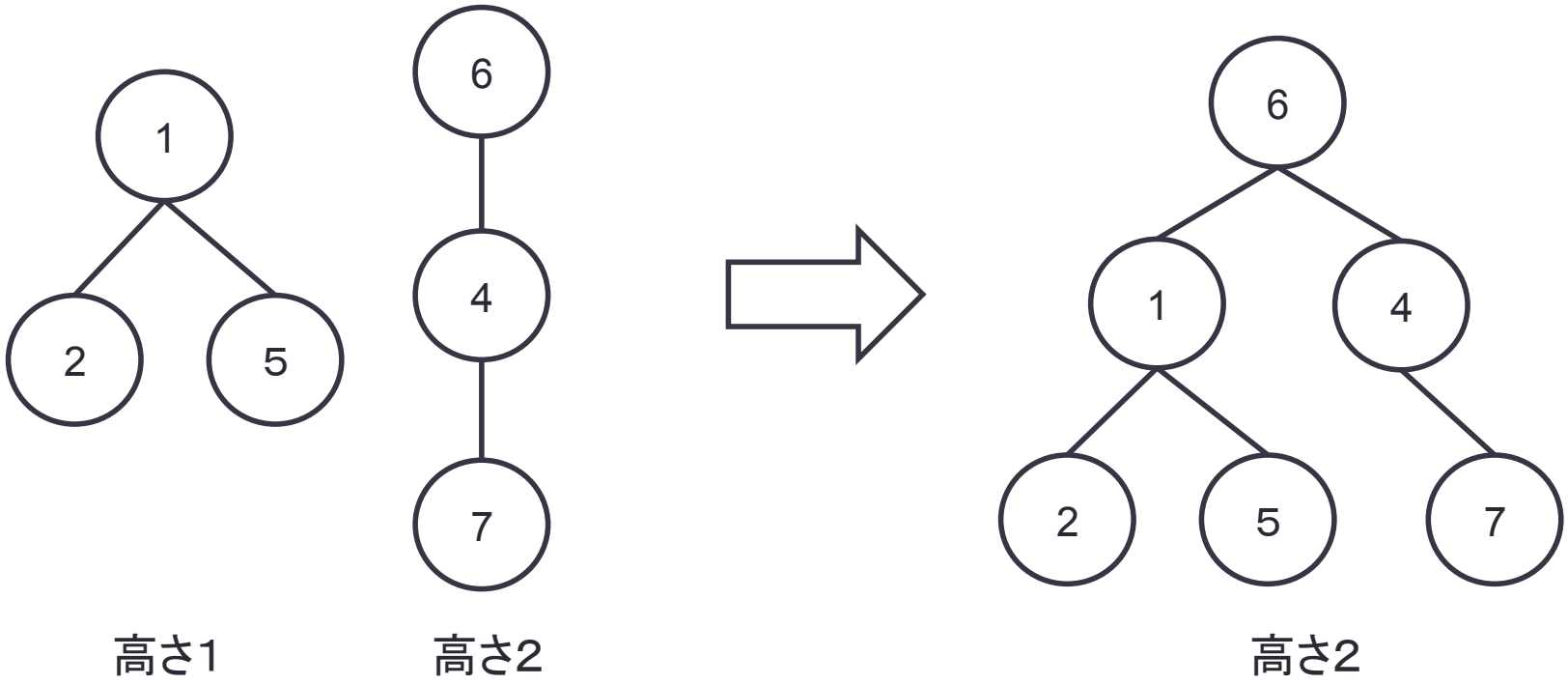
Union-Find木　ー判定ー

- 要素aと要素bが同じグループに属するか調べる
  - 木を上向きに辿り、木の根を調べ、同じ根にたどり着くか調べる



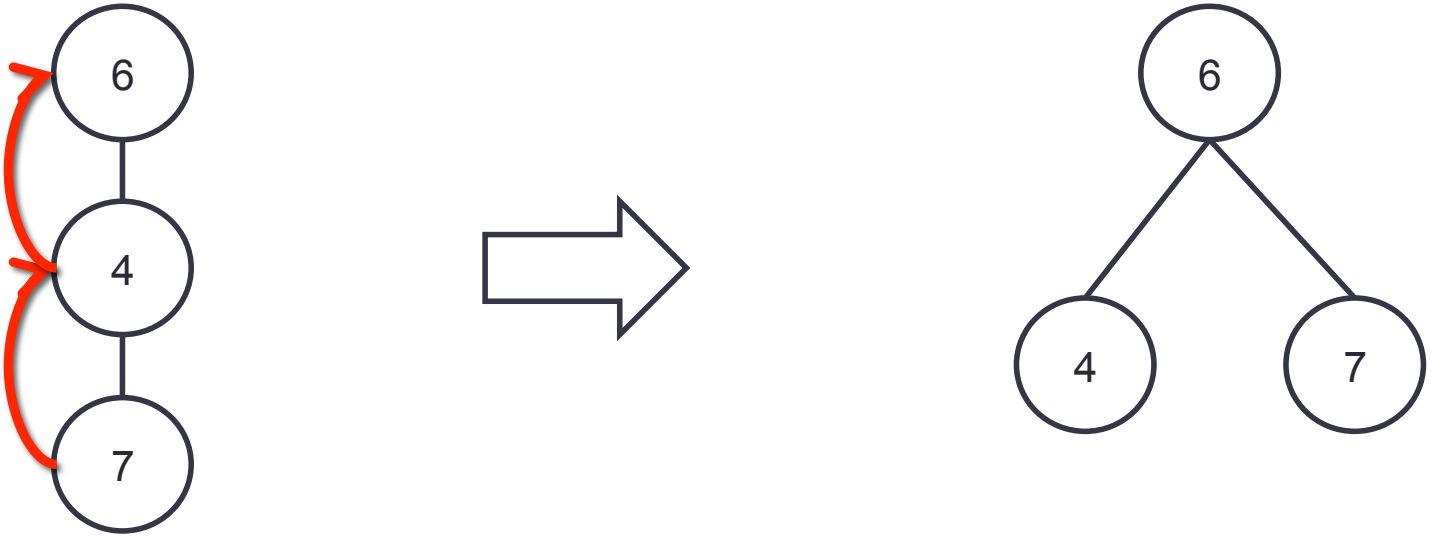
Union-Find木　ー実装の注意ー

- 偏りが発生しないようにする。
  - 木の高さ(rank)を記憶しておく。
  - 併合の際に2つの木のrankが異なれば、rankの小さいものから大きいものへ辺を張る。

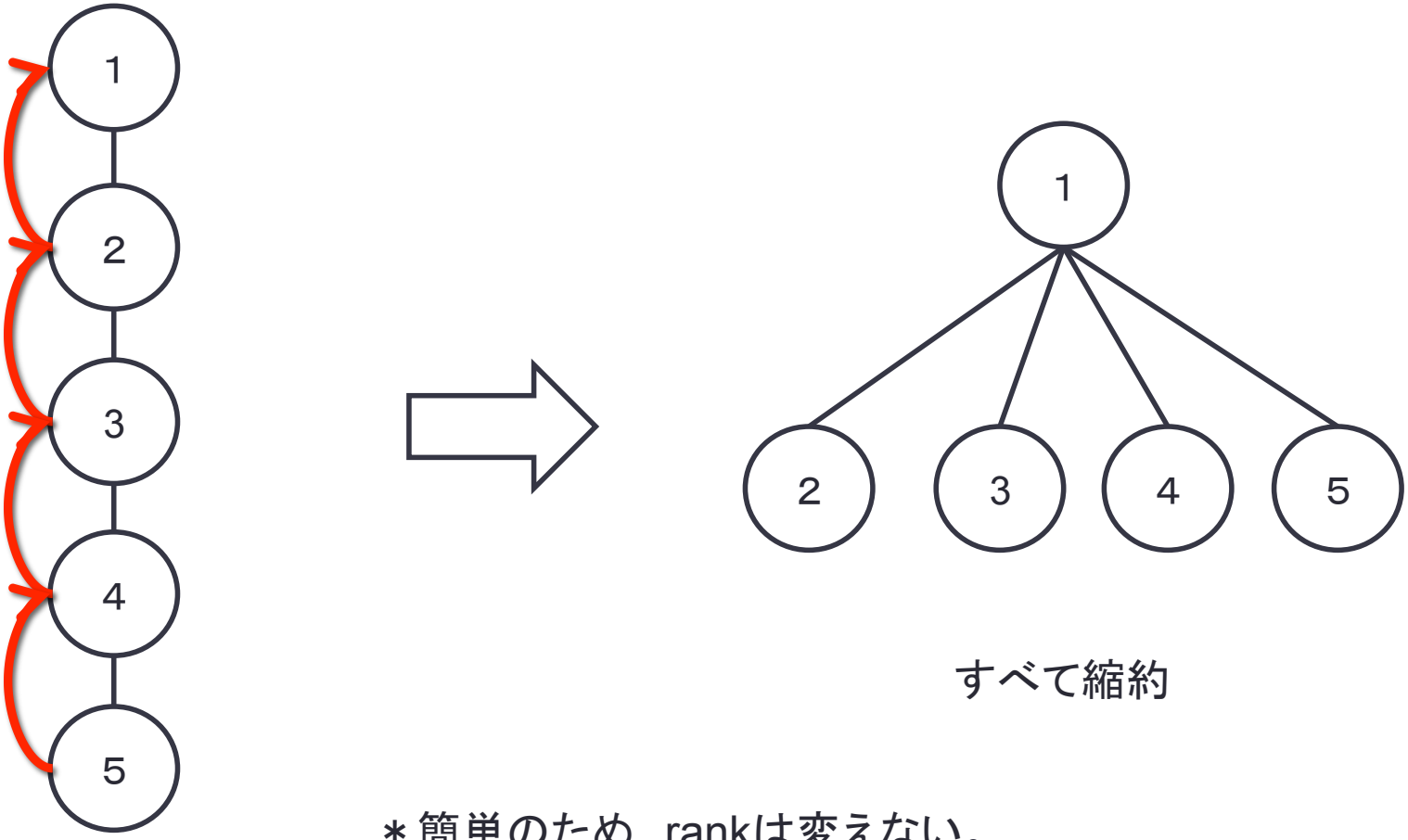


Union-Find木　一辺の縮約一

直接根に繋ぐ



Union-Find木　一辺の縮約一



すべて縮約

＊簡単のため、rankは変えない。

Union-Find木　ー計算量と実装ー

- 平均はO(α(n))らしい、、、
  - アッカーマン関数Aの、f(n) = A(n, n) の逆関数

- 実装
  - ＊p84参照

問題　食物連鎖 (POJ 1182)

- N匹の動物、1, … , Nがいる
- 動物はすべて3つの種類 A, B, C のいずれか
- A は B を食べ、B は C を食べ、C は Aを食べる
- 次の2種類の情報がk個与えられる
  - タイプ1：x と y は同じ種類です
  - タイプ2：x は y を食べます
- k個の情報は全て正しいと限らず、誤った情報（以前の情報に矛盾するものや、誤った番号など）も含まれている

- k個の情報のうち、誤った情報の個数を出力せよ

- 制約
  - 1<= N <= 50000
  - 0 <= k <= 100000

解法

- 各動物 i に対して、i-A, i-B, i-C を作り、3×N個の要素でUnion-Findを作る。
- i-X は「 i が種類Xである場合」を表す。
- 各情報に対して、矛盾が起きているかを調べ、
- 起きていなければ、以下の操作を行う。
  - タイプ1：x と y は同じ種類
  - → 「x-Aとy-A」、「x-Bとy-B」、「x-Cとy-C」の3つのペアを併合
  - タイプ2：x は y を食べる
  - → 「x-Aとy-B」、「x-Bとy-C」、「x-Cとy-A」の3つのペアを併合

- ＊コードはp86参照