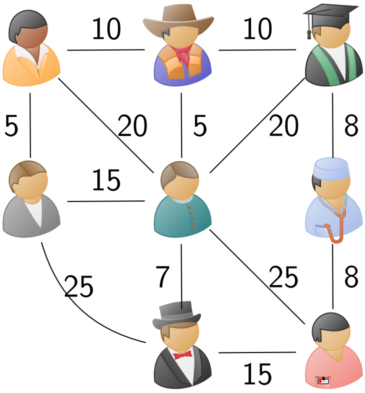
最小木

離散数学・オートマトン
2024年後期
佐賀大学理工学部　只木進一

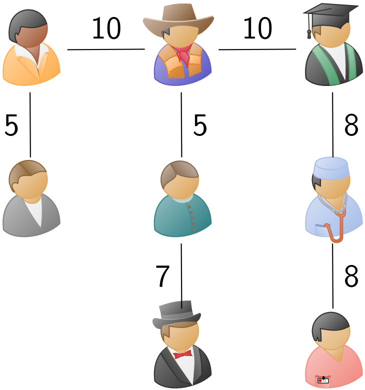
ネットワーク: Networks

- グラフの各辺に数値が対応したものをネットワークと呼ぶ
  - 道路網中の距離
  - パイプライン網中の容量
- 今日は、無向グラフの各辺に正の「重み (weight)」があるものを扱う

例 2.1: 最安の連絡経路(全員に連絡)
The cheapest communication route



例 2.1: 解



最小木の応用例: Applications of minimum trees

- 連絡網: communication network
- 油井のネットワーク: network of oil wells
  - 積出港へのパイプの長さを最小に
- 組織内のネットワーク配線: network wiring in a office

Jarník-Prim 法

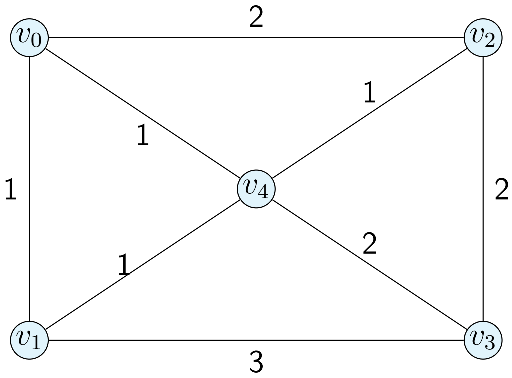
- 始点から開始して、連結した頂点の数を増やす: Incrementally connect vertices
- 構成途中でも木になっている: The intermediate result is always a tree
- 構成途中の木から、未連結の頂点への辺のうちの重み最小の辺を選んで、枝を伸ばす: Choose the edge with the smallest weight
    - 重みの増分が最小: The increment of the weight is the smallest

Jarník-Prim アルゴリズム

Algorithm 1 Jarník-Prim アルゴリズム

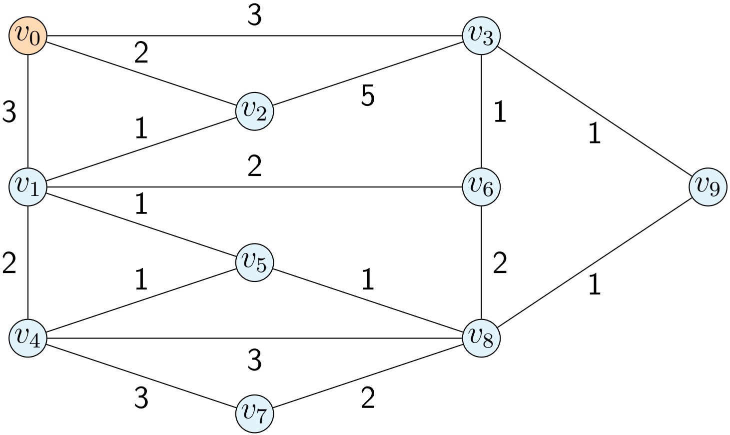
任意の頂点 $v \in V$ を選び、$U = \{v\}$、$T = \emptyset$ とする
while $U \neq V$ **do** ▷ 全ての頂点を結ぶまで繰返す
  U と $V \setminus U$ を結ぶ辺のうち、最小の重みのものを e とする
  e の $V \setminus U$ 側の端点を w とする
  U.append(w)
  T.append(e)
end while
T が最小木を構成する

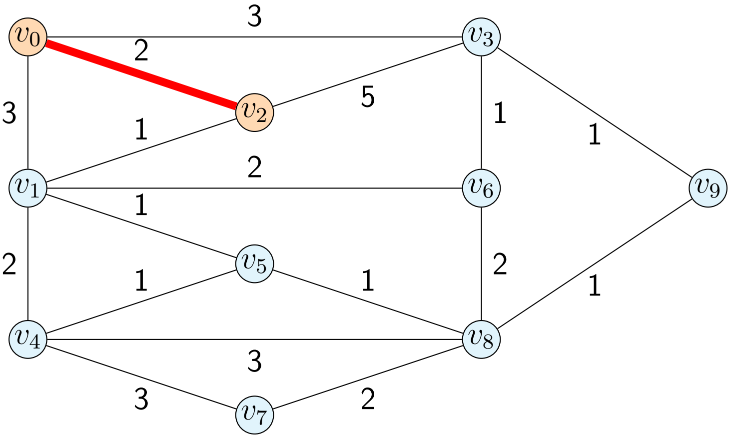
例 3.1:

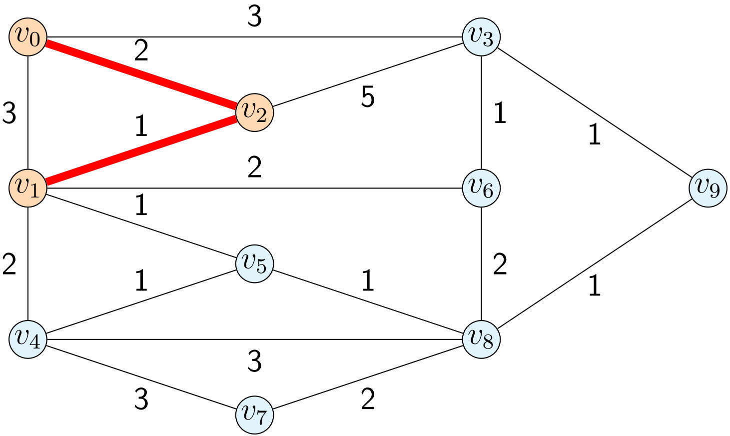


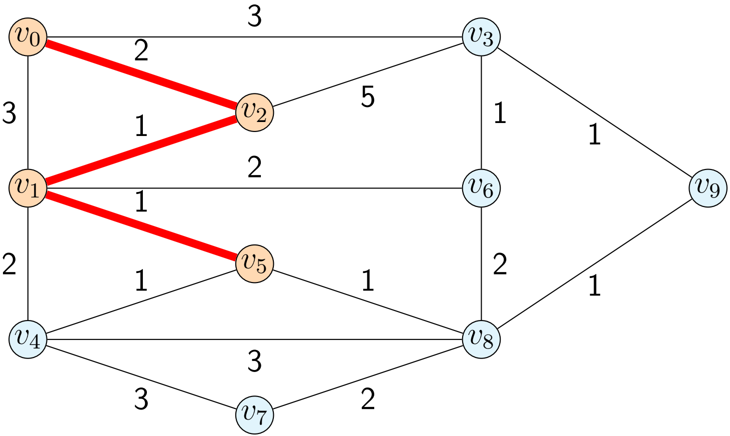
辺の数値が重みを表す

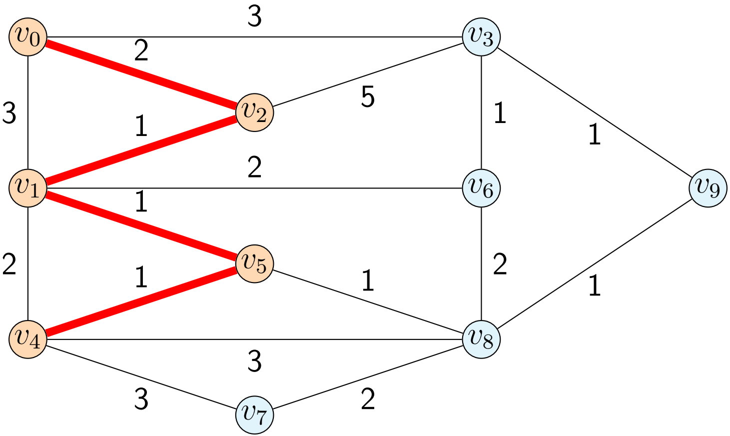
例 3.2:

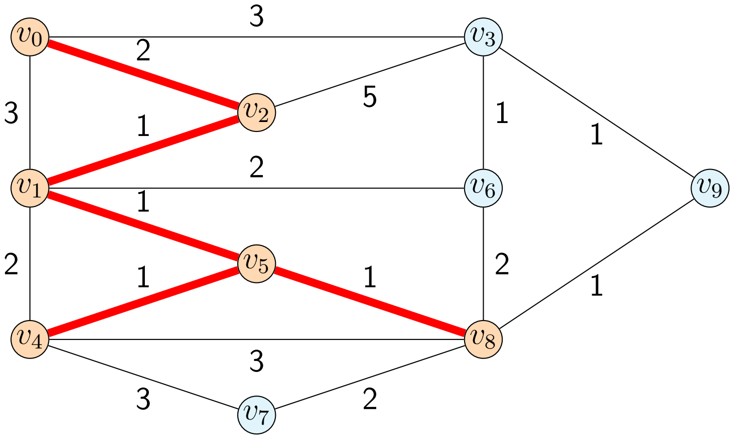


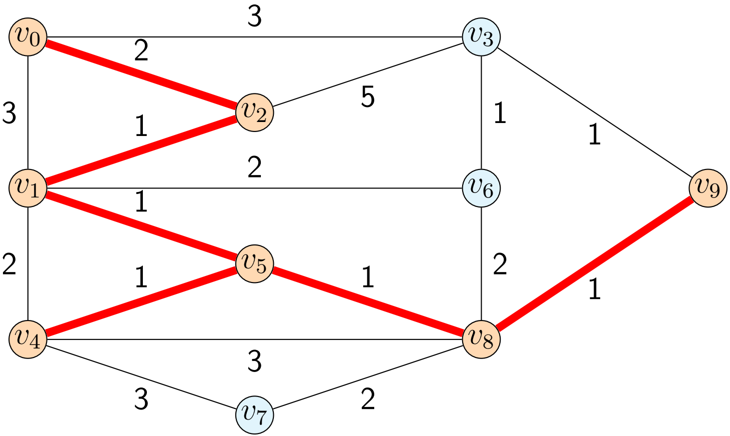

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

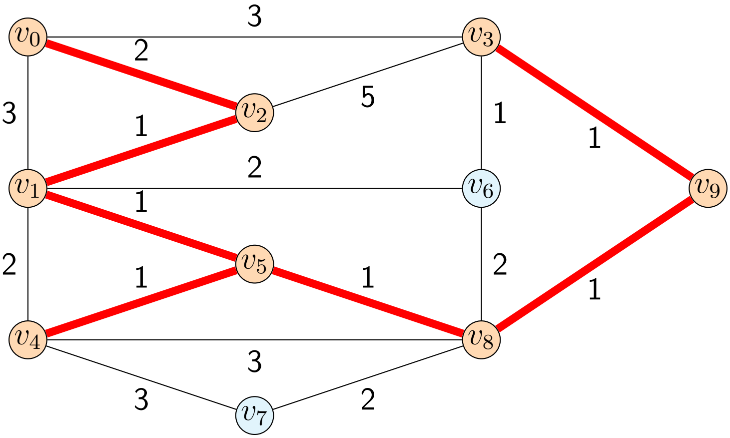

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
1
5
1
1
2
1
2
1
1
1
2
3
3
2
1

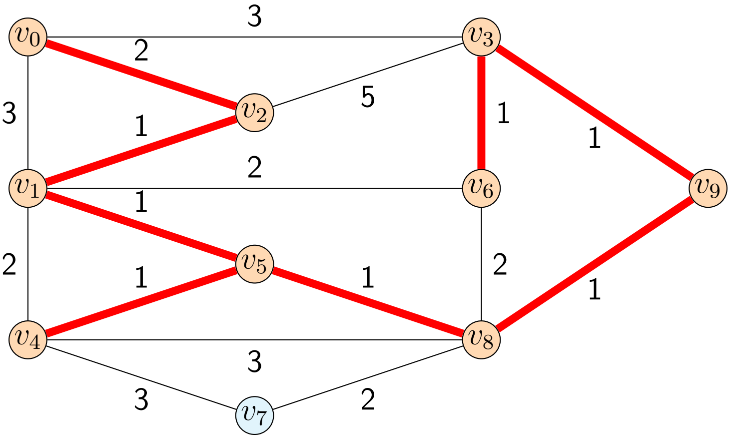

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

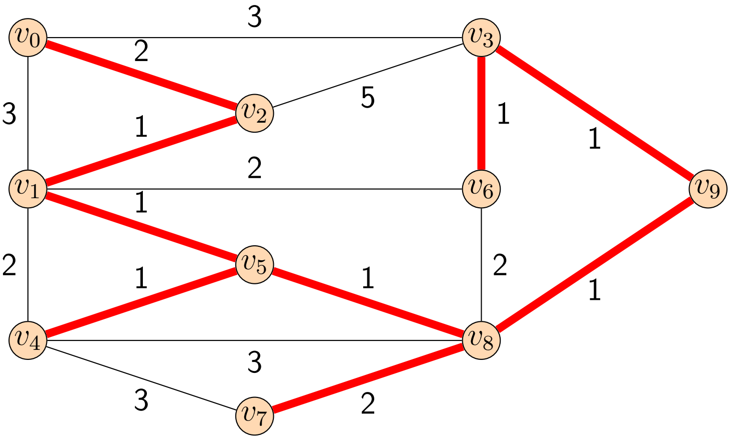

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2


v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
5
1
1
1
2
1
1
2
1
1
2
1
3
3
2


v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
1
5
1
1
2
1
2
1
1
1
2
1
3
3
2

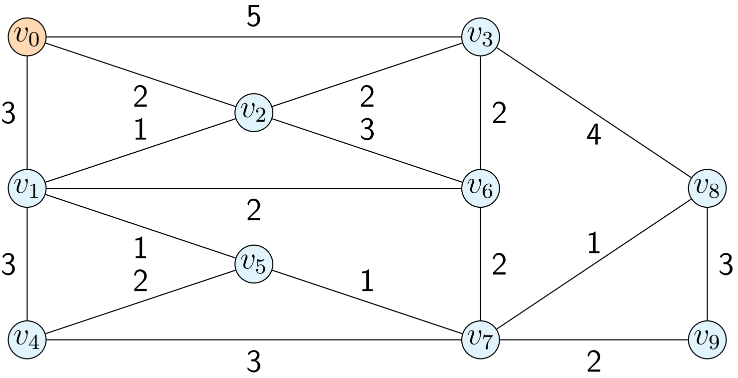

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

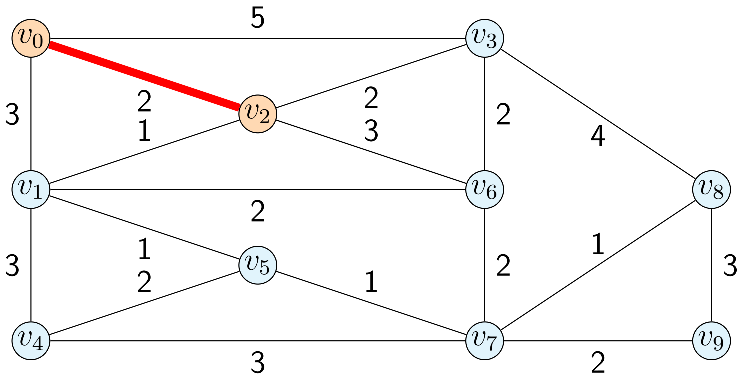

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

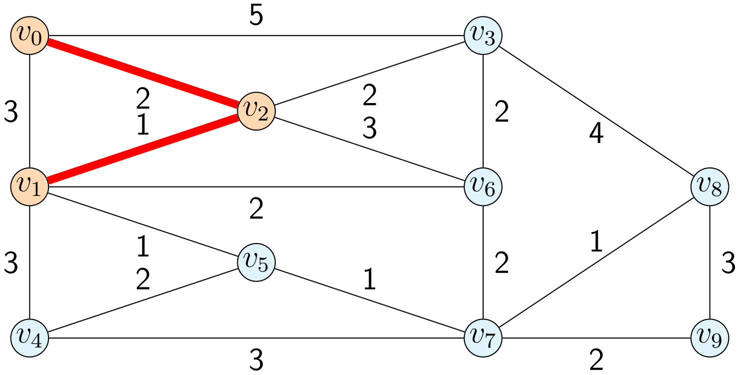


重み 1 の辺は全て使用。

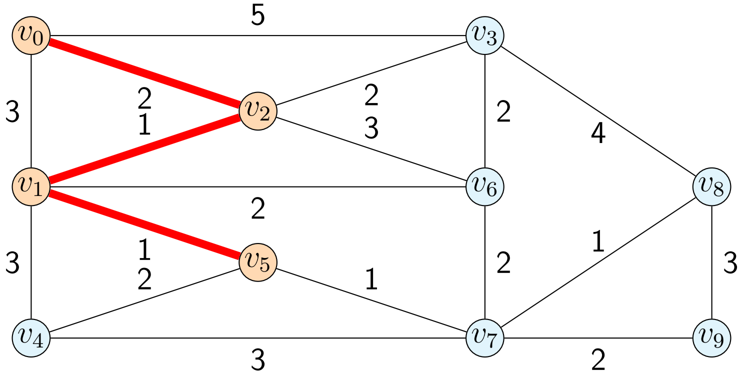
例 3.3:

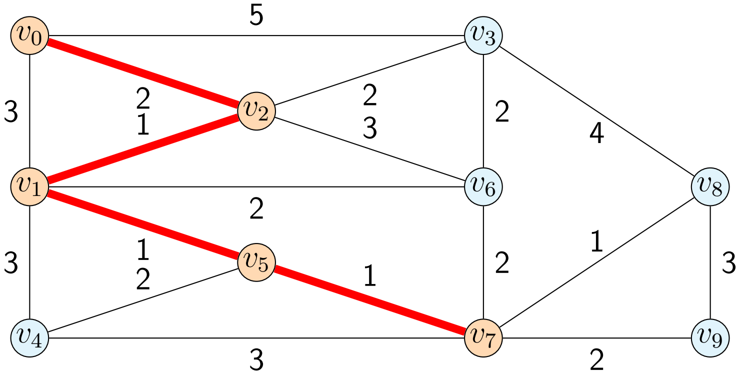


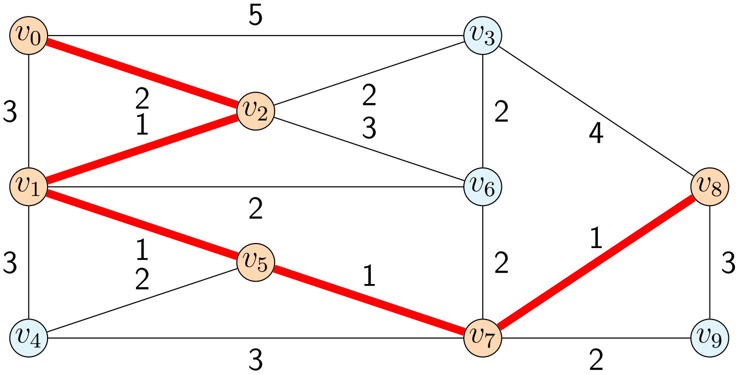

5
v_0
v_3
3
2
1
v_2
2
3
2
4
v_1
v_6
v_8
2
3
1
2
v_5
1
2
1
3
v_4
v_7
v_9
3
2



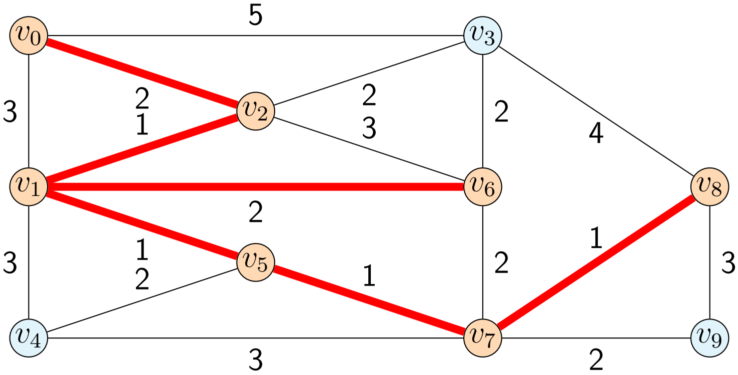
v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
5
3
2
1
2
3
2
4
2
3
1
2
1
1
2
3
3
2

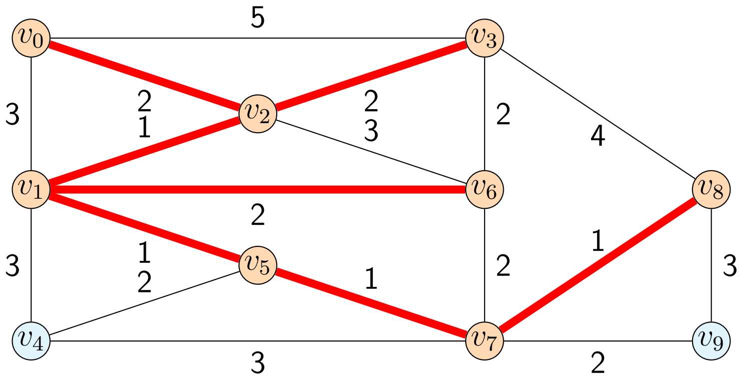

v_0
5
v_3
3
2
1
v_2
2
3
2
4
v_1
v_6
v_8
2
3
1
2
1
2
2
1
3
v_5
v_4
v_7
v_9
3
2


5
v_0
v_3
3
2
1
v_2
2
3
2
4
v_1
v_6
v_8
2
3
1
2
v_5
1
2
1
3
v_4
v_7
v_9
3
2

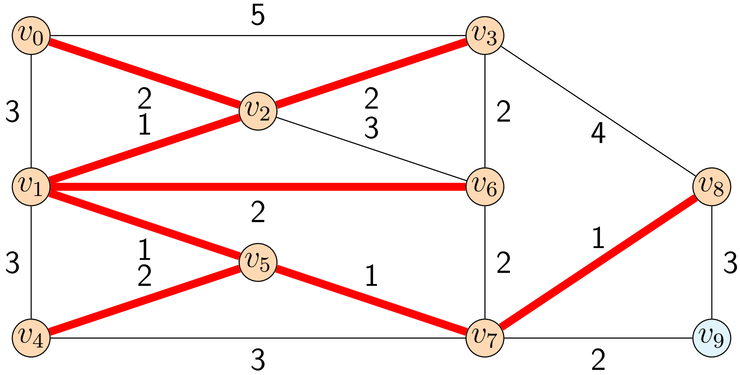


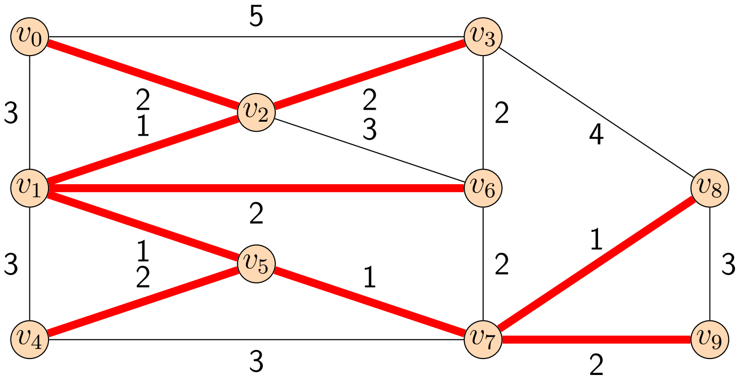
v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
5
3
2
1
2
3
2
4
2
3
1
2
1
2
1
3
3
2


5
2
1
2
3
2
4
3
2
3
1
1
2
2
1
3
3
3
2
v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9



5
v_0
v_3
3
2
1
v_2
2
3
2
4
v_1
v_6
v_8
2
1
3
1
2
v_5
1
2
3
v_4
v_7
v_9
3
2


5
v_0
v_3
3
2
1
v_2
2
3
2
4
v_1
v_6
v_8
2
3
1
2
v_5
1
2
1
3
v_4
v_7
v_9
3
2


5
v_0
v_3
3
2
1
v_2
2
3
2
4
v_1
v_6
v_8
2
3
1
2
v_5
1
2
1
3
v_4
v_7
v_9
3
2

例 3.3: 途中プロセス: Processes

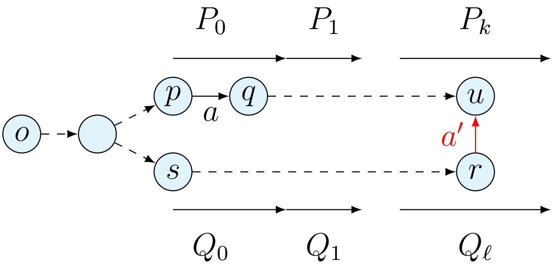
from	to	U
		$\{v_0\}$
v_0	v_2	$\{v_0, v_2\}$
v_2	v_1	$\{v_0, v_1, v_2\}$
v_1	v_5	$\{v_0, v_1, v_2, v_5\}$
v_5	v_7	$\{v_0, v_1, v_2, v_5, v_7\}$
v_7	v_8	$\{v_0, v_1, v_2, v_5, v_7, v_8\}$
v_1	v_6	$\{v_0, v_1, v_2, v_5, v_6, v_7, v_8\}$
v_2	v_3	$\{v_0, v_1, v_2, v_3, v_5, v_6, v_7, v_8\}$
v_5	v_4	$\{v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$
v_7	v_9	$\{v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$

Jarník-Prim 法が正しいこと: Correctness of the method

- Jarník-Prim アルゴリズム実行中の木 T は、U が誘導する G の部分グラフ $G(U)$ における最小木になっていることを示す。
- 証明の方針: ある辺 $\exists a \in T$ を、別のある辺 $\exists a' \notin T$ に置き換えることで、より小さい木ができる

$$w(a') < w(a) \tag{4.1}$$

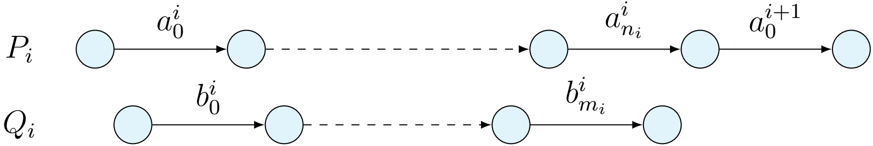
ことを仮定して、矛盾を導く。



- o を根とする木 T において、辺 $a \in T$ の代わりに辺 $a' \notin T$ としたときに、重みが小さくなると仮定する。

$$w(a') < w(a) \tag{4.2}$$

- 上の枝で、辺 a を先頭に連続して伸びた道を P_0 とし、その後に下の枝で連続して伸びた道を Q_0 とする。その後、P_1、Q_1 と交互に伸びるとする。他の道は無視する。
- 辺 a' の両端の頂点は道 P_k と Q_ℓ に属しているとする。



- P_i を構成する辺 $\left\{a_0^i, a_1^i, \cdots, a_{n_i}^i\right\}$
- Q_i を構成する辺 $\left\{b_0^i, b_1^i, \cdots, b_{m_i}^i\right\}$
- P_i の後で Q_i 伸びることから
  - P_i が伸びている最中は Q_i は伸び始めない
  - Q_i が伸びている最中は、P_i の次 P_{i+1} は伸び始めない

$$\forall i, 0 \leq \forall j \leq n_i, \qquad w(a_j^i) \leq w(b_0^i), \tag{4.3}$$

$$\forall i, 0 \leq \forall j \leq m_i, \qquad w(b_j^i) \leq w(a_0^{i+1}) \tag{4.4}$$

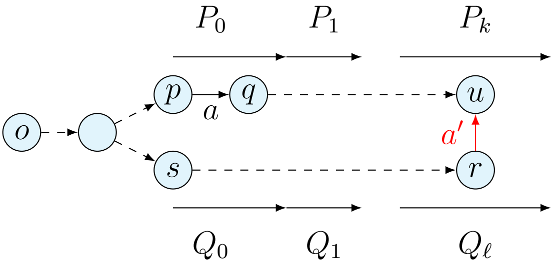
- 各道 P_i 及び Q_i の先頭の辺に注目

$$\forall i, w(a_0^i) \leq w(b_0^i) \leq w(a_0^{i+1}) \tag{4.5}$$

- 各道の先頭の辺の重みは以下を満たす

$$\forall i, w(a) \leq w(a_0^i), w(a) \leq w(b_0^i) \tag{4.6}$$

$k \leq \ell$

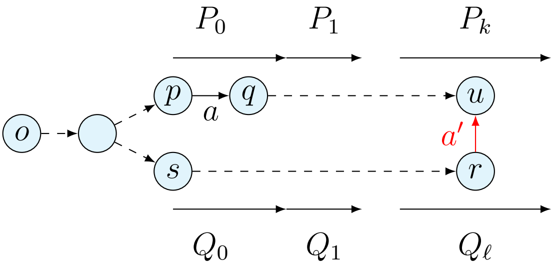


- 上の道が、頂点 u まで伸びたとき、下の道は頂点 r まで伸びていない
- 上の道 P_k が伸びるとき、つまり Q_k が始まる前に、辺 a' は採用されないことから

$$w(a) \leq w(b_0^k) \leq w(a') \tag{4.7}$$

となり、矛盾

$k > \ell$



- 上の道が、頂点 u まで伸びたとき、下の道は頂点 r を過ぎて伸びている
- 下の道 Q_ℓ が伸びるとき、つまり $P_{\ell+1}$ が始まる前に、辺 a' は採用されないことから、

$$w(a) \leq w(a_0^{\ell+1}) \leq w(a') \tag{4.8}$$

となり、矛盾

Binary Heap

- 要素の中から最小要素を取り出す
- 最小要素以外は完全に整列しているわけではない
- 実装が容易
- 処理が高速

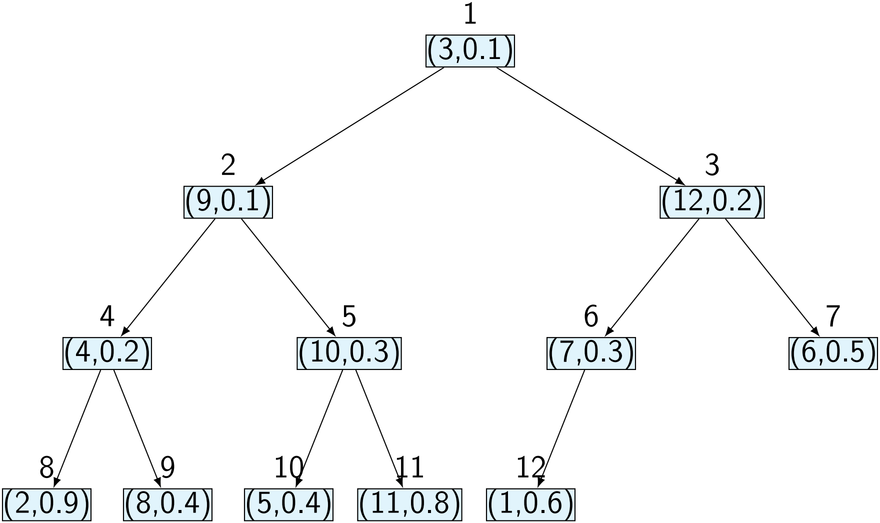
https://github.com/discrete-math-saga/BinaryHeap

例

- ラベルと値からなるデータ
  (label, value)
- 完全二分木
  - 最下層以外の第 k 層には、2^{l-1} 個の頂点
  - 最下層は左から詰めて配置
- ある頂点 p とその子の要素 c

$$p_{\text{value}} \le c_{\text{value}}$$

Binary Heap イメージ



頂点位置の番号つけに注目

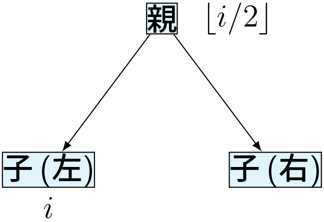
データ保持

- リスト L を用意
- L_0 は使用しない
- 二分木上の位置 i の要素を L_i に格納
- 要素数 n

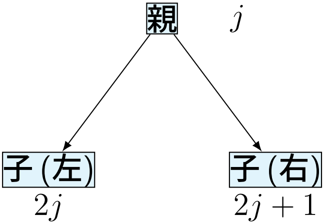
$$n = |L| - 1$$

親子のインデクス

親の番号



子の番号



最後尾に要素を追加し、適当な位置まで移動させる。

Algorithm 2 要素の追加

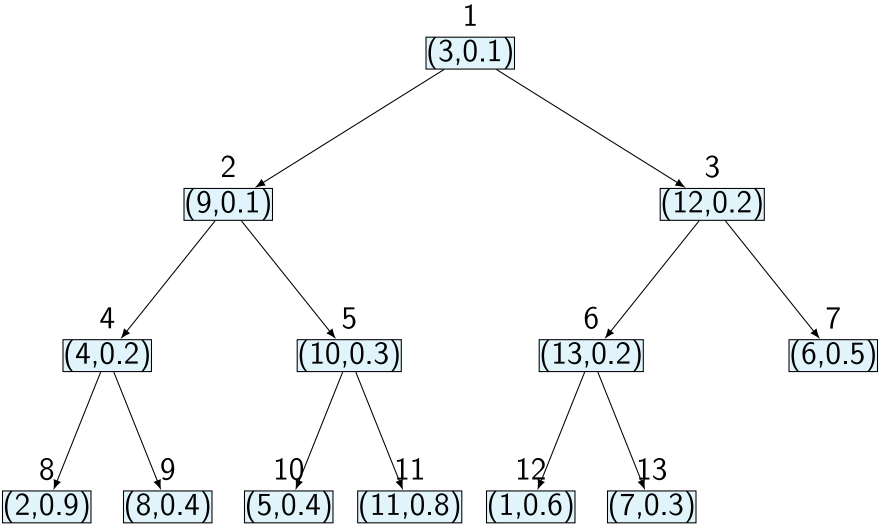
procedure $\textsc{Add}(o)$
  L.append(o)
  $n++$
  shiftUp(n)
end procedure

- 位置 k にある要素を適当な位置まで移動させる。
- そのために、親との大小を確認する。

Algorithm 3 シフトアップ

procedure $\textsc{ShiftUp}(k)$
    if $k > 1 \wedge \text{isLess}(k, \lfloor k/2 \rfloor)$ **then**
        $\text{swap}(k, \lfloor k/2 \rfloor)$
        $k = \lfloor k/2 \rfloor$
        $\text{shiftUp}(k)$
    end if
end procedure

(13,0.2) を追加した場合



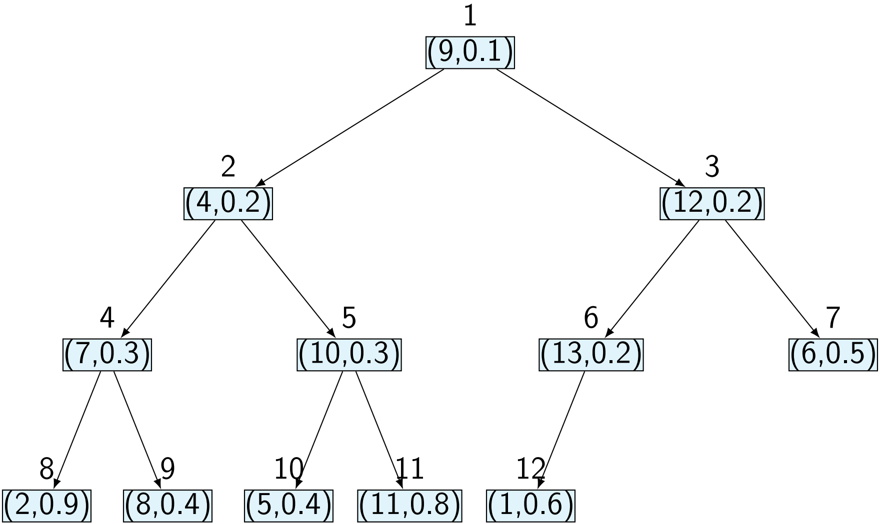
- 先頭要素を取り出す
- 末尾の要素を先頭に置き、適切な位置まで下げる

Algorithm 4 最小要素の取り出し

procedure POLL
  $t = L.\text{get}(1)$
  $x = L.\text{removeLast}()$
  $L.\text{set}(1, x)$
  $\text{shiftDown}(1)$
  return t
end procedure

Algorithm 5 シフトダウン

```
procedure SHIFTDOWN(k)
    if 2k ≤ n then
        j = 2k
        if j < n ∧ isLess(j + 1, j) then
            j++
        end if
        if isLess(k, j) then
            return
        end if
        swap(l.j)
        shiftDown(j)
    end if
end procedure
```


1
(9,0.1)
2
(4,0.2)
3
(12,0.2)
4
(7,0.3)
5
(10,0.3)
6
(13,0.2)
7
(6,0.5)
8
(2,0.9)
9
(8,0.4)
10
(5,0.4)
11
(11,0.8)
12
(1,0.6)

要素の値の変更

- 値を減少させた場合は、シフトアップ
- 値を増加させた場合は、シフトダウン