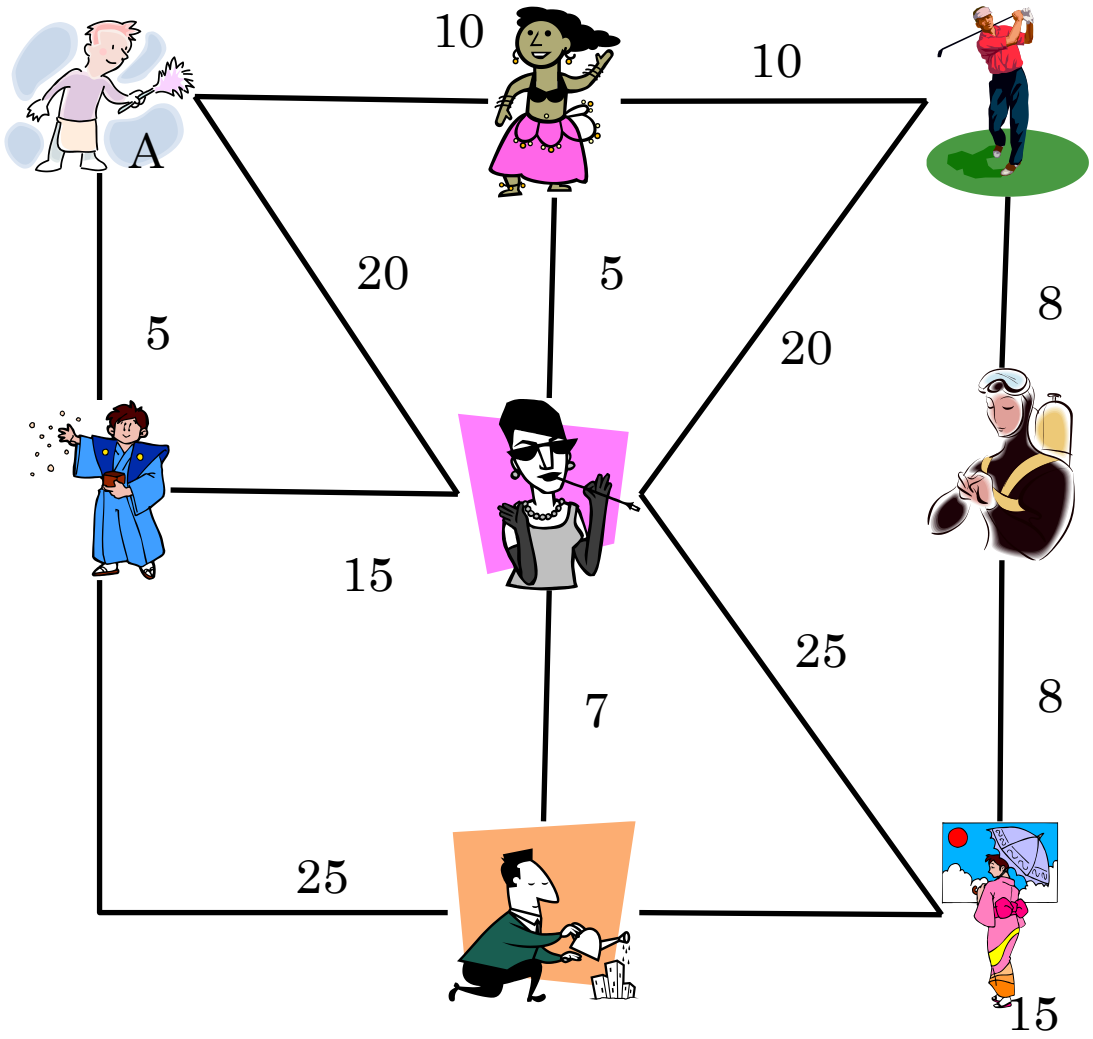
最小木問題
Minimum Tree Problem

ネットワーク(NETWORKS)

- 弧に長さ、重み、費用などの属性のあるグラフ
  - 都市とそれを結ぶ交通
    - 都市間の道路距離
    - 都市間の鉄道の運賃
    - 都市間の空路の最大輸送可能人数
  - コンピュータとネットワーク：帯域
  - 作業工程：所用時間、遅れ

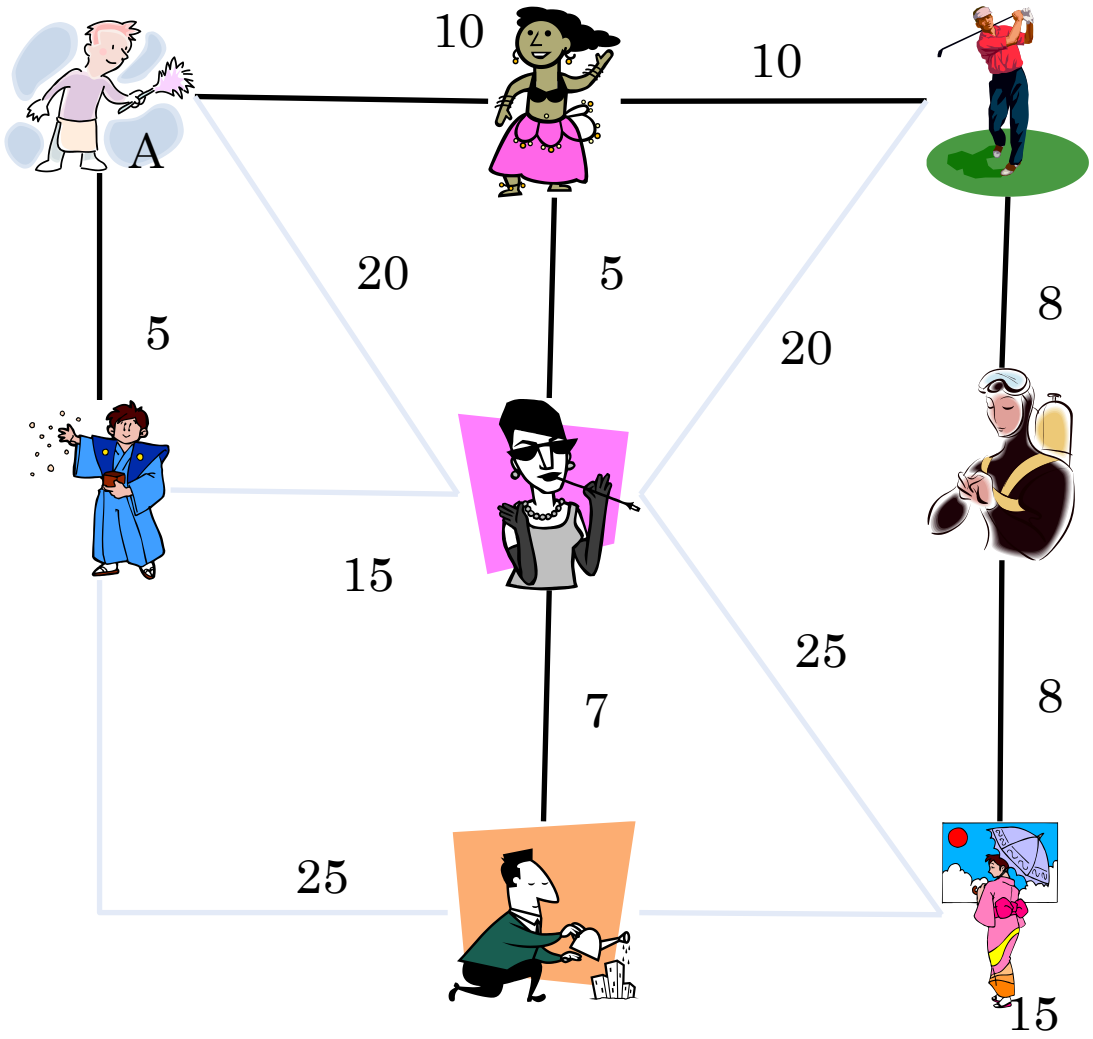
例:最安の連絡経路



Aから全員に最も安く連絡する経路総経費を考える

経路の経費が定義されている

例:最安の連絡経路:解



最小木問題(Minimum Tree Problem)

- 連結無向グラフ$G=(V,A)$
  - 弧に向きが無い
- 重み関数 $w: A \to R$
  - 各弧に実数が対応：重み、距離、etc.
- G の極大木 $T \subseteq A$
  - 重みが最小になる極大木Tを見付ける

$$\min_T w(T)$$

$$w(T) = \sum_{a \in T} w(a)$$

最小木問題の応用

- 油井（ゆせい）から精油所へパイプラインを引く
  - 最短(経費の最も安い)のパイプラインで一カ所に原油を集める
- 最小のコストでコンピュータを繋ぐ
- 通信コストを最小にして事業所を繋ぐ

二つのアプローチ

- Kruskal法 (貪欲法、Greedy法)
  - 重み最小の弧を順に選ぶ
  - 構成途中は木になっていない(部分木の集合)
  - 閉路ができないように制限しながら弧を選択する
- Jarník-Prim法
  - 始点から開始して、連結した頂点の数を増やす
  - 構成途中でも木になっている

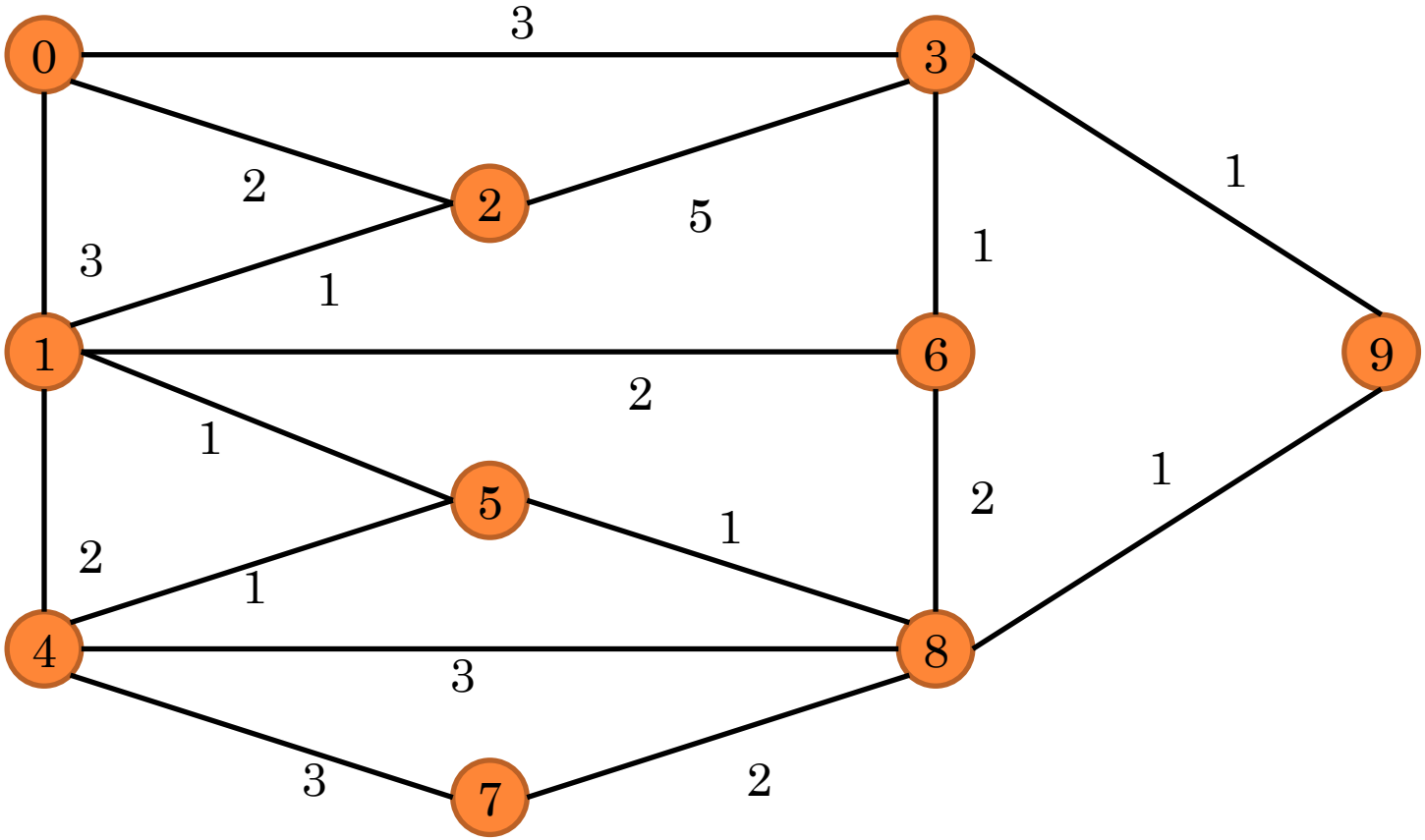
Kruskal法 (貪欲法、Greedy法)

- 重み最小の弧を順に選ぶ
- 構成途中は木になっていない(部分木の集合)
- 閉路ができないように制限しながら弧を選択する

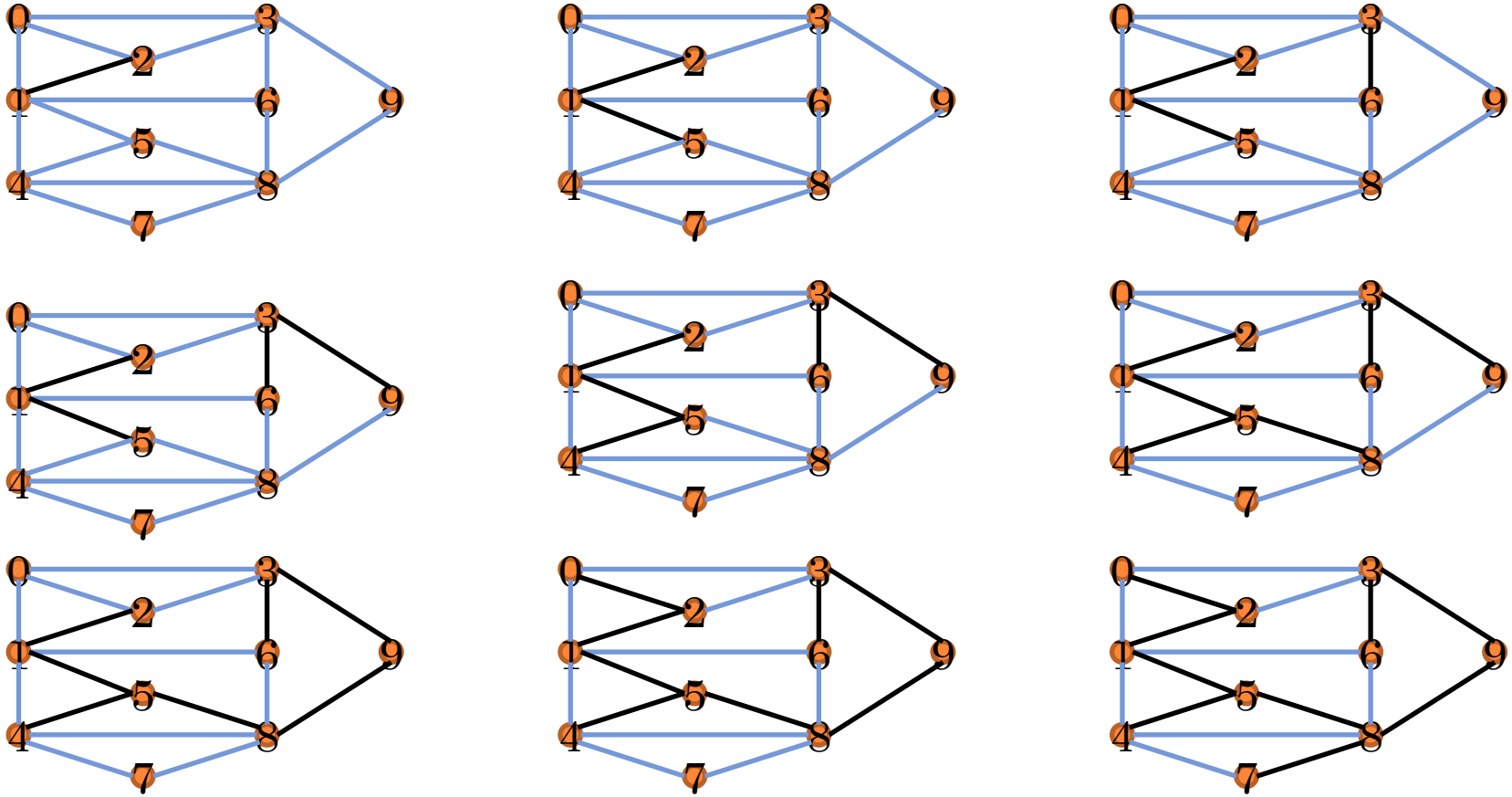
貪欲アルゴリズム (Greedy algorithm, Kruskal algorithm)

```
T = ∅
H : G の弧の重みに関するヒープ
while ( T は G の極大木ではない ) {
    a = H.poll() ヒープから最小要素を取得
    a_new = null
    while(a_new == null){
        if ( T ∪ {a} は閉路を持たない) {
            a_new = a
        }else{
            a = H.poll()
        }
    }
    T = T ∪ {a_new}
}
```

例1



例1:解の探索の様子



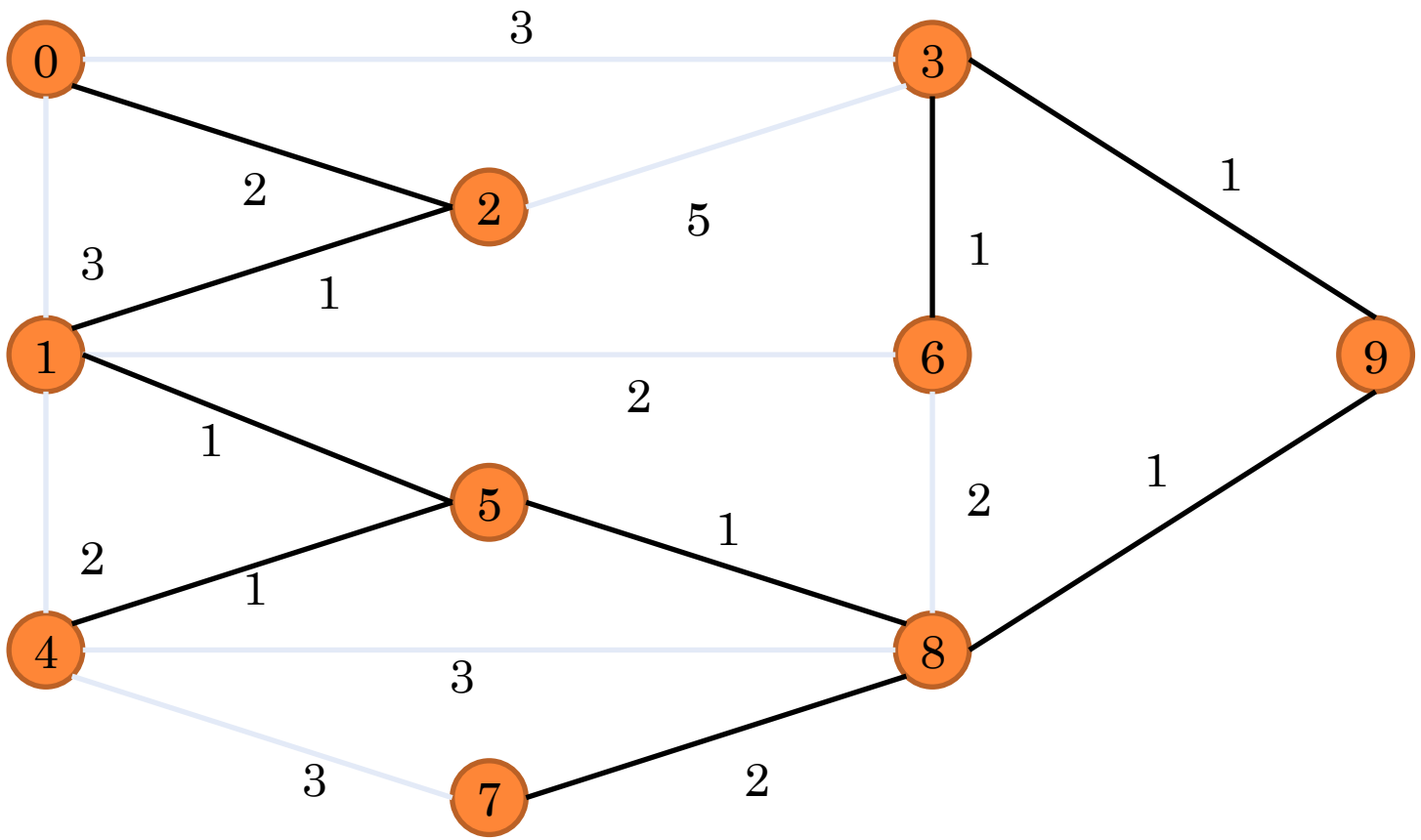
ここまでは、重み1の弧

重み2の弧 $e(v_0, v_2)$

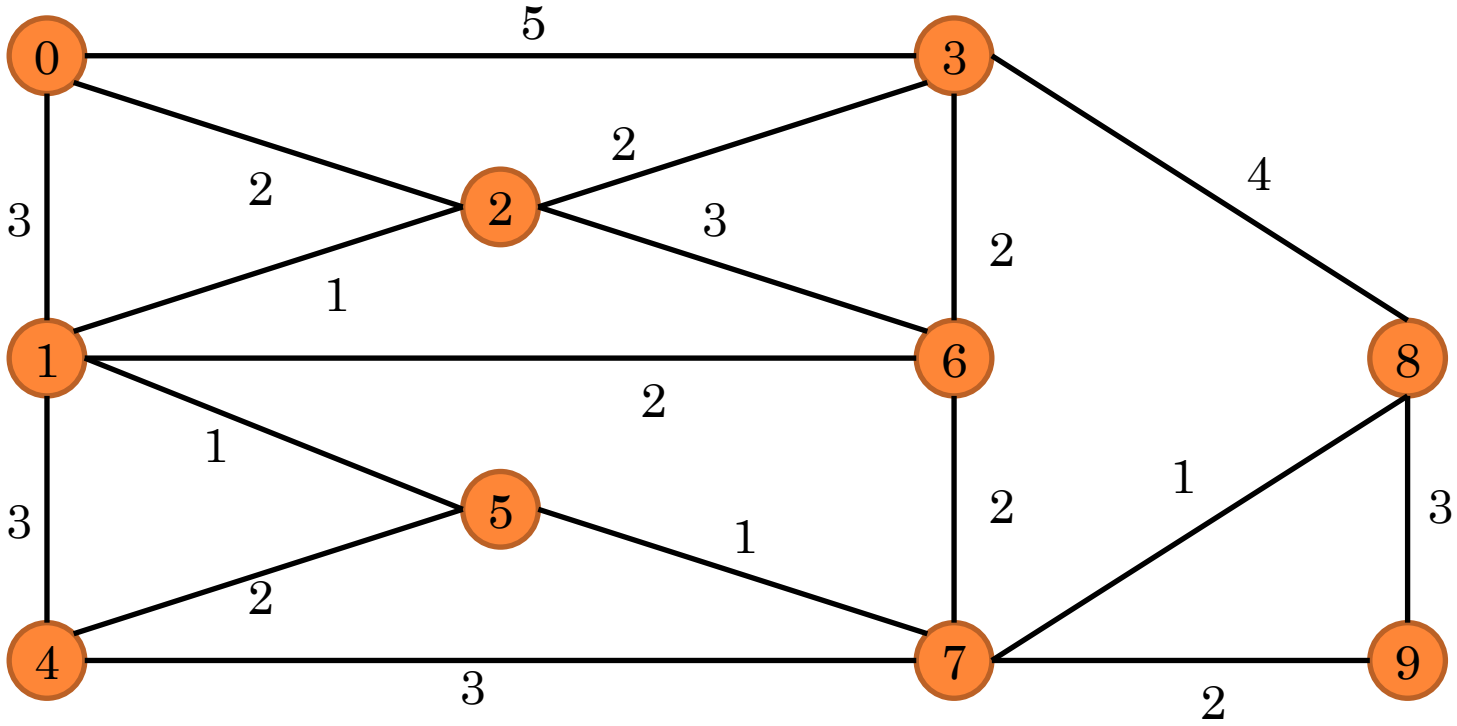
重み2の弧 $e(v_7, v_8)$

$e(v_1, v_6)$ を選択すると閉路ができる

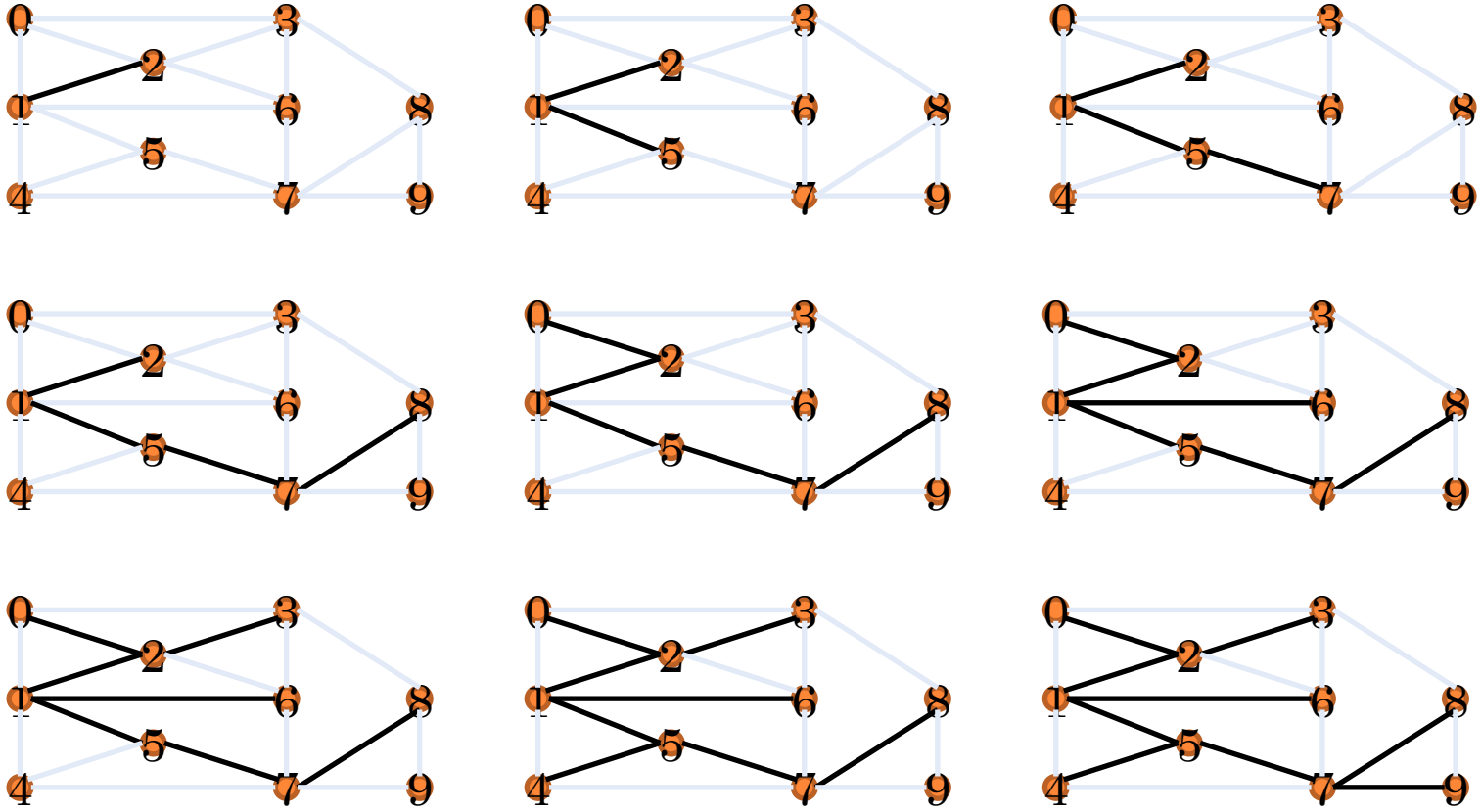
例1:結果

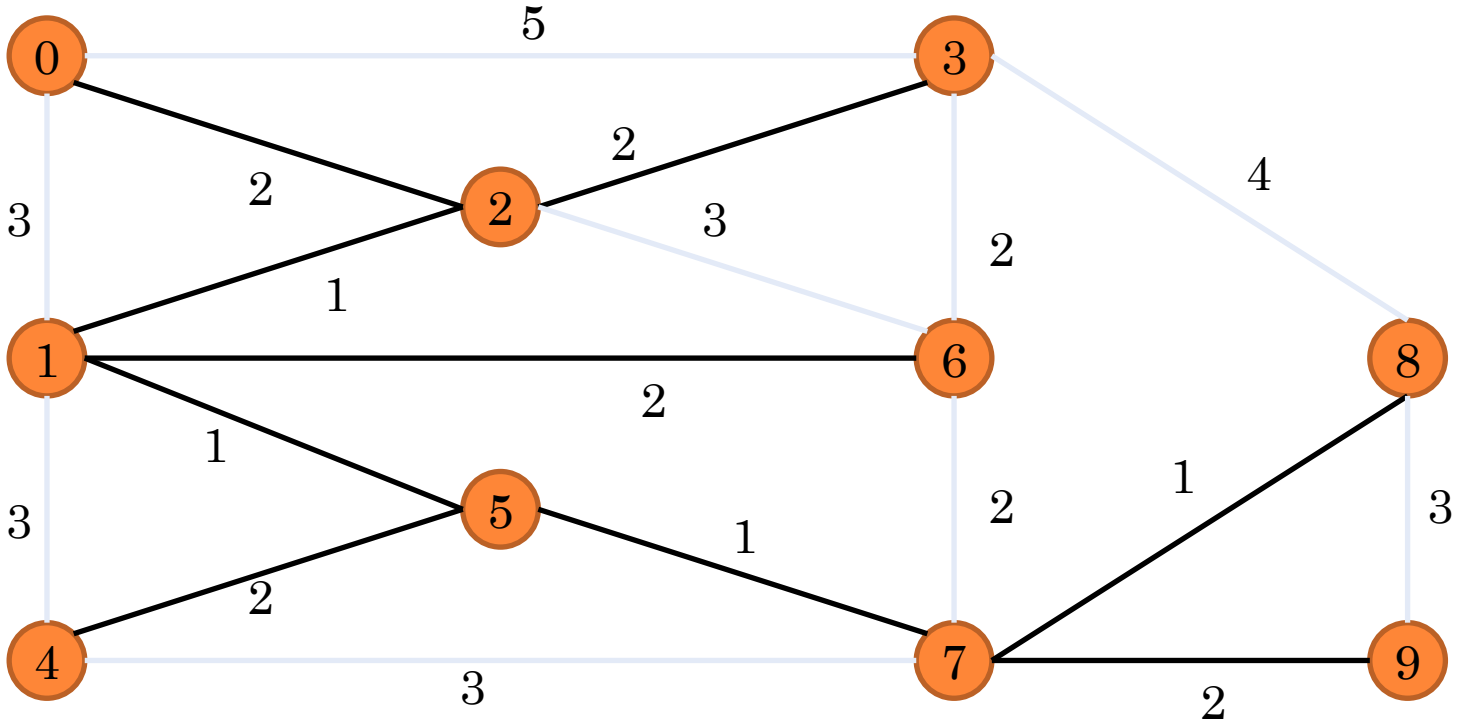


例2



例2:解の探索の様子


0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9
0 3 2 1 6 8 5 4 7 9


0
1
2
3
4
5
6
7
8
9
5
2
2
3
4
3
2
1
2
1
3
2
1
3
2
3
2

貪欲アルゴリズムが正しい理由

- 次の定理を証明すれば良い

「貪欲アルゴリズム実行中で得られるTは、弧数$|T|$を持ち、サーキットを含まない弧集合のうちで、その重みが最小である。」

  - 要するに、アルゴリズムの各段階で最小の重みのグラフであることを示す。

- 証明では、上記を性質(*)と呼ぶことにする。

数学的帰納法による証明

- $T=\varnothing$ の時、自明
- 操作を i 回行って、次の弧を選択する直前にある弧集合 T が性質(*)を満たしているとする。
- 次に選択された弧を a とする。
  - $a \in A \setminus T$
- $|T|+1$ 本の弧を持ちサーキットを有さない弧集合のうちで重みが最小のものを S とする。
  - $w(S)=w(T)+w(a)$ ならば性質(*)が成り立つ
  - $w(S)<w(T)+w(a)$ は矛盾する(起こりえないことを示す)

証明：準備

- T は $|T|$ 本の弧を持つサーキット(閉路)の無い弧集合の内で、重みが最小である。
  - 従って、Sから任意の弧bを取り除いた弧集合（$|T|$本の弧）の重みはTの重みより小さいことはない。

$$\forall b \in S,\ w\left(S \setminus \{b\}\right) = w(S) - w(b) \geq w(T)$$

- このことから、Sに含まれる任意の弧bとTにこれから追加する弧aの重みの大小関係がわかる。

$$w(S) < w(T) + w(a) \leq w(S) - w(b) + w(a)$$

$$\Downarrow$$

$$w(b) < w(a)$$

証明:矛盾の導出

- S とT はサーキットを含まず$|S|=|T|+1$
- ある　$a' \in S \setminus T$ に対して$T \cup \{a'\}$はサーキットを含まないとする。
  - a'はSの弧であることから、 $w(a') < w(a)$
    - なぜなら、Tは(*)を満たすから
  - これより

$$w\left(T \cup \{a'\}\right) = w(T) + w(a') < w(T) + w(a)$$

- これはa の選び方に反する。
- よって矛盾する。つまり、そのようなSは存在せず、手続きに従って構成した$T \cup \{a\}$は、最小木である。

注意

- ある弧を選択した際に、それが閉路を作らないことの確認が必要
  - 加えようとして孤aの両端の頂点(v,w)
  - T 内にvからwへの道があるかを調べる
- 深さ優先、幅優先の探索アルゴリズムが必要

最大補木を求める方法

- 最小木を求めるために、重み最大の補木 $A \setminus T$ を求める

```
T ← A
w_max = 0
While (T は G の極大木ではない ) {
        forall (a ∈ T ) {
                if (T \ {a} は G の極大木を含む) {
                        if ( w(a) > w_max ) {
                                a_selected = a
                                w_max = w(a)
                        }
                }
        }
        T ← T \ {a_selected}
}
```

JARNÍK-PRIM法

- 始点から開始して、連結した頂点の数を増やす
- 構成途中でも木になっている

- 途中の木から、未連結の頂点への弧のうちから、最小の弧を選んで、枝を伸ばす

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 ($U \neq V$){

U と $V \setminus U$ を結ぶ弧のうち最小の重みのものを a とする

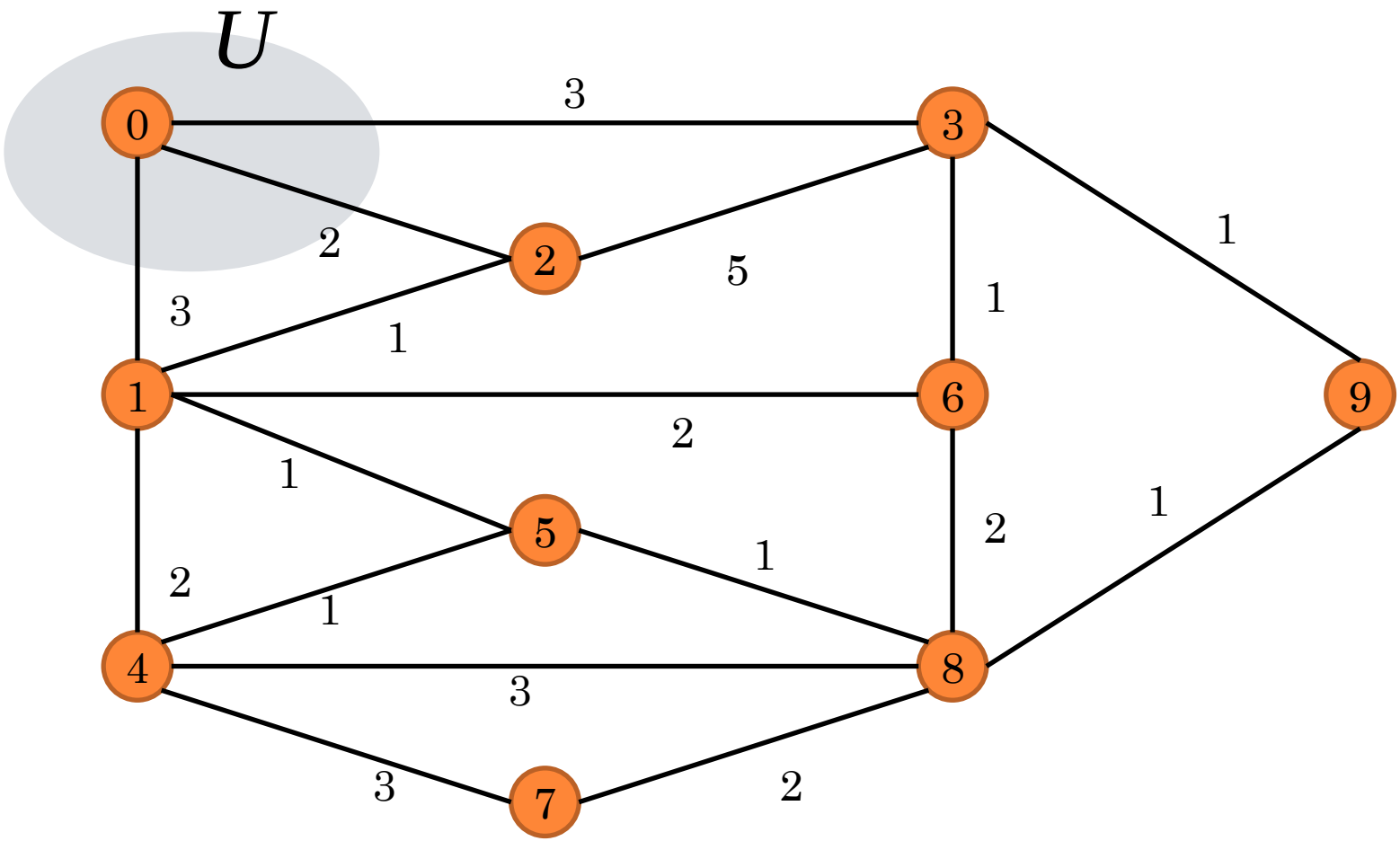
a の $V \setminus U$ 側の端点を w とする

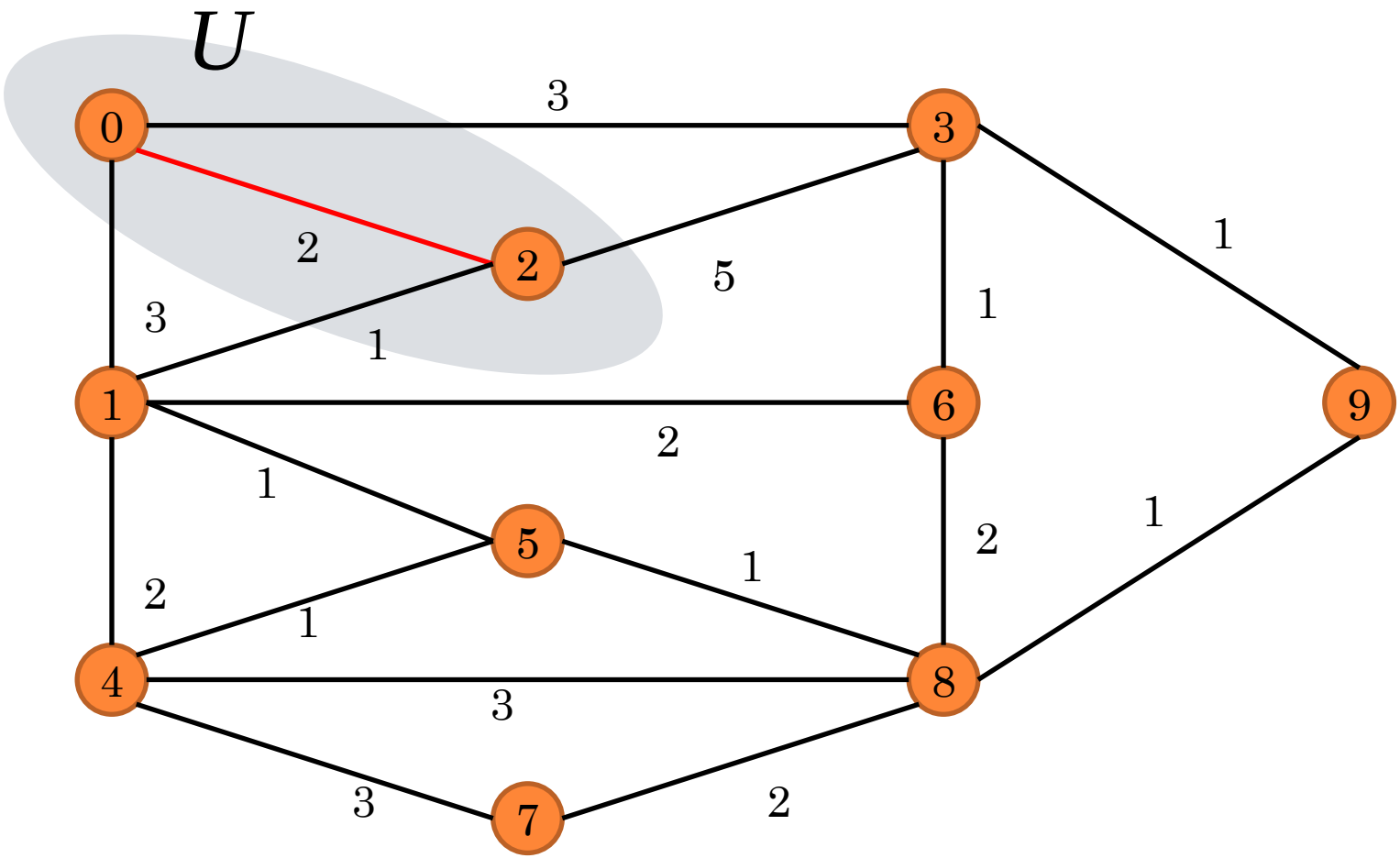
$U \leftarrow U \cup \{w\}$

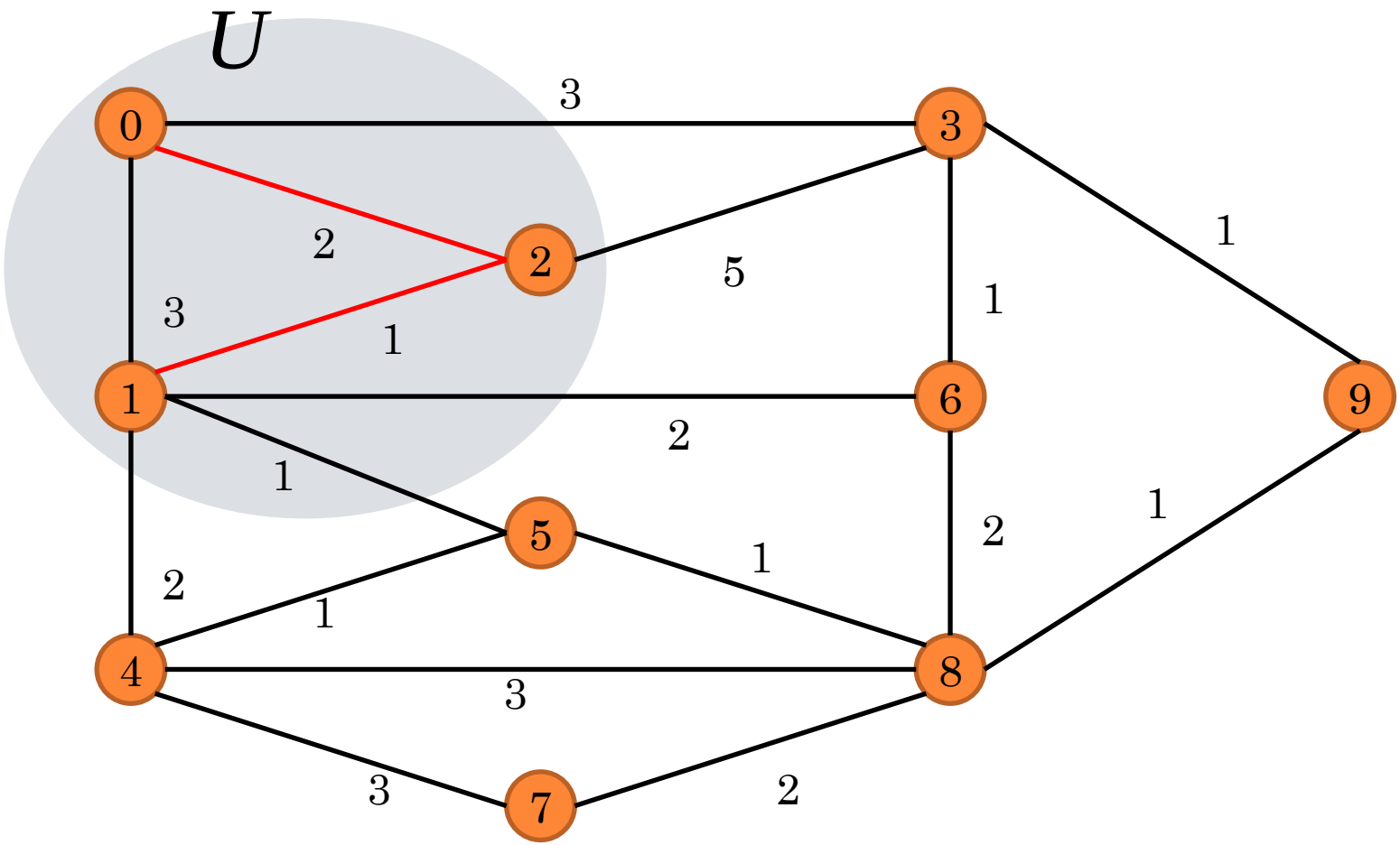
$T \leftarrow T \cup \{a\}$

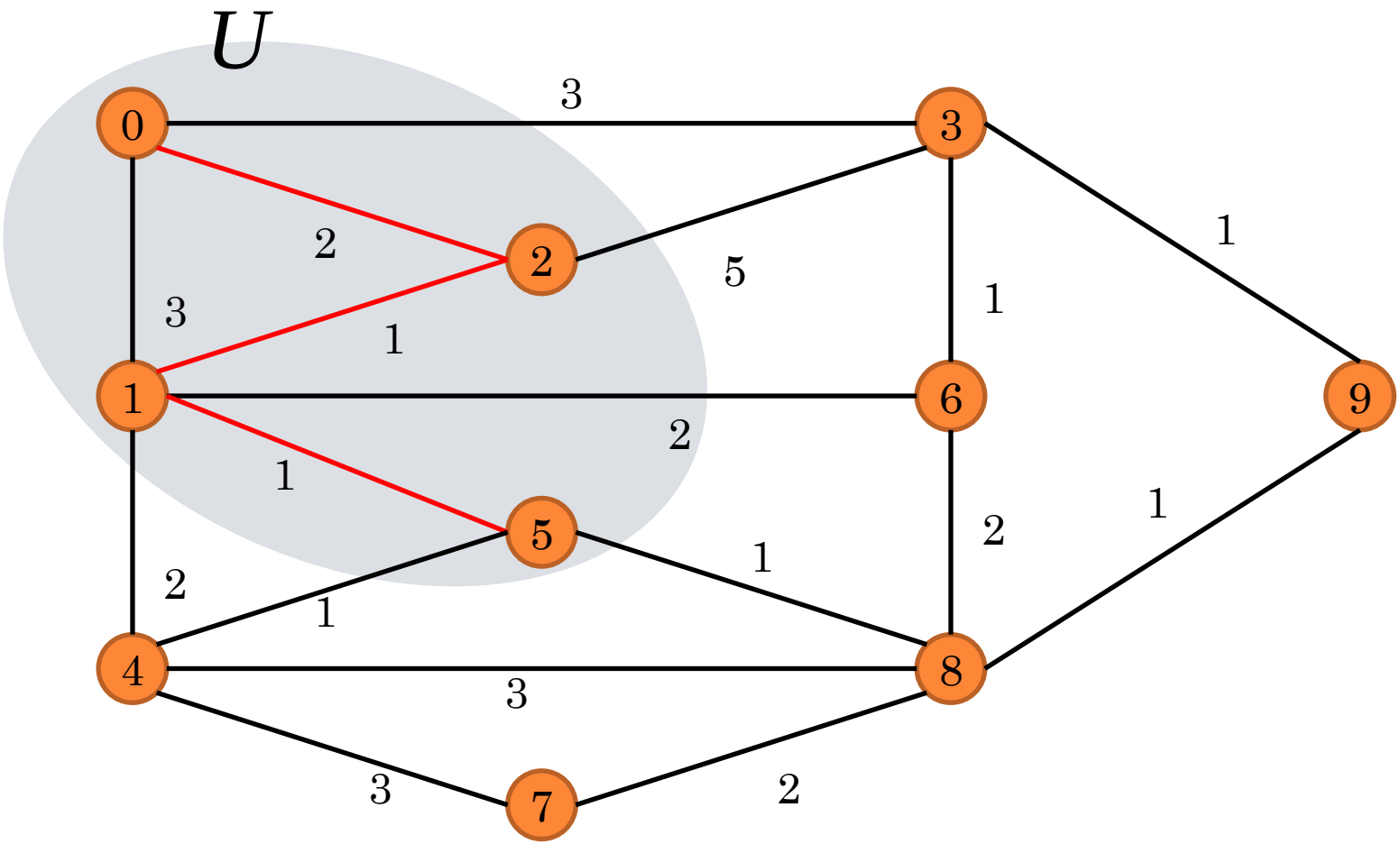
}

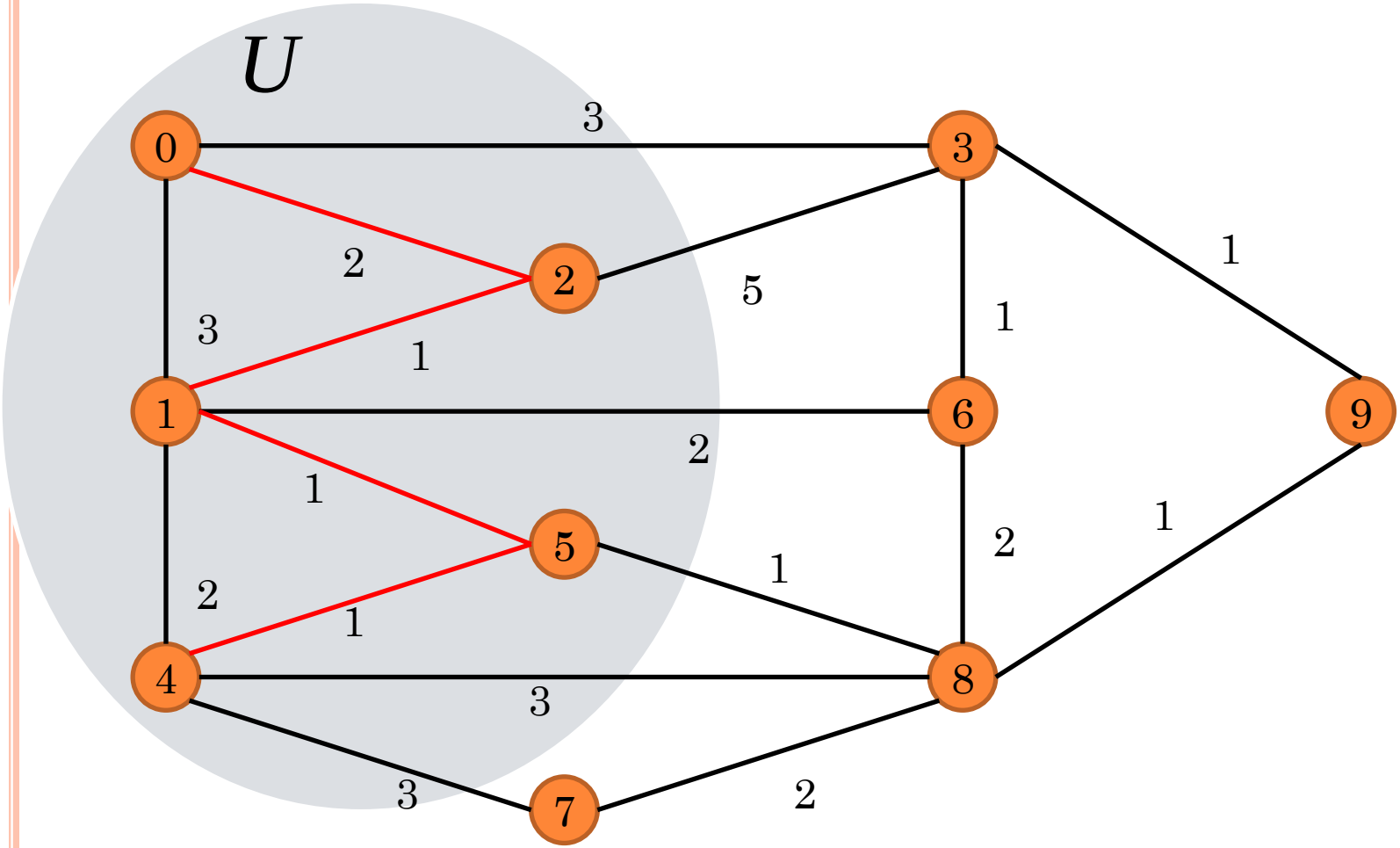
例1

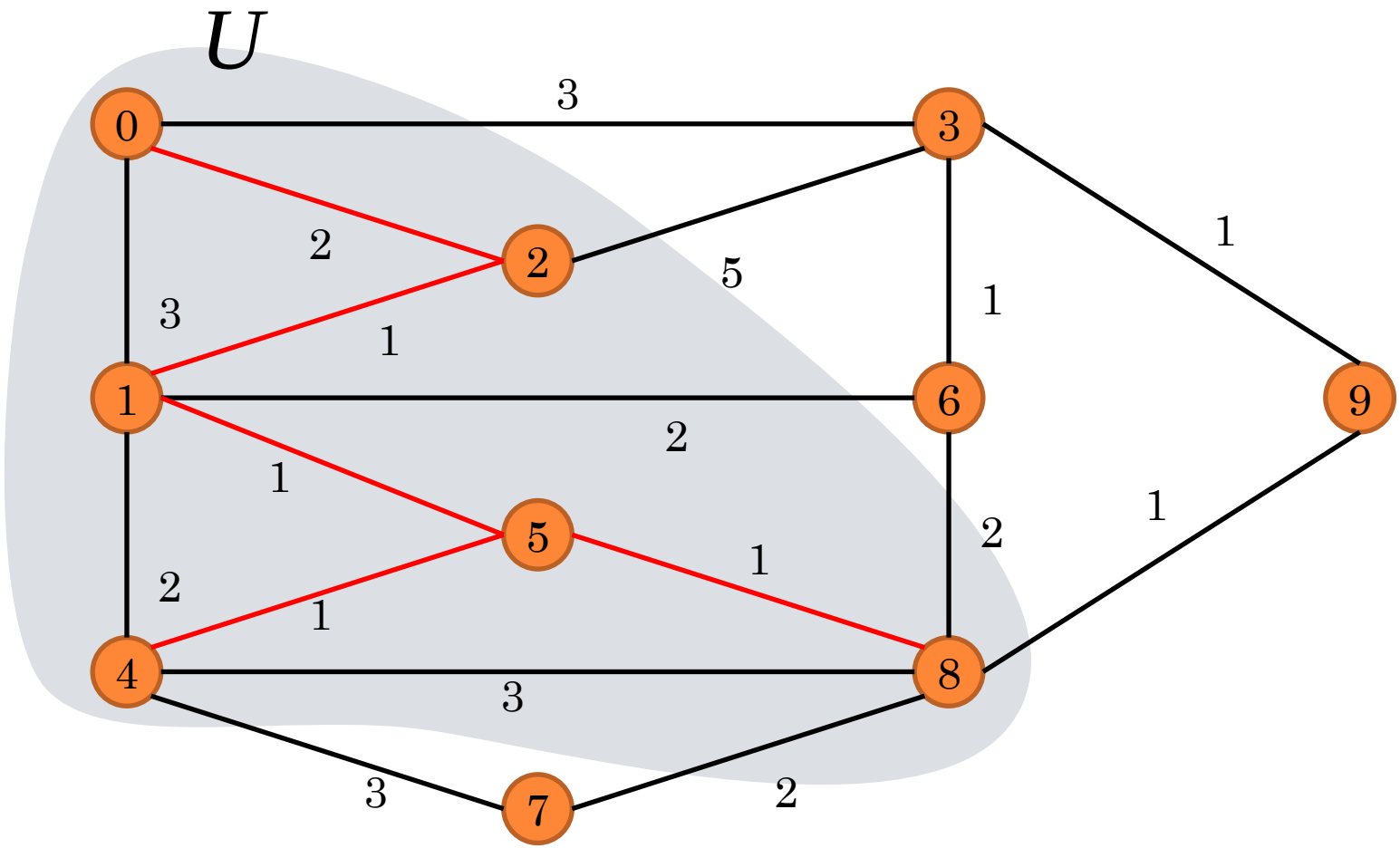


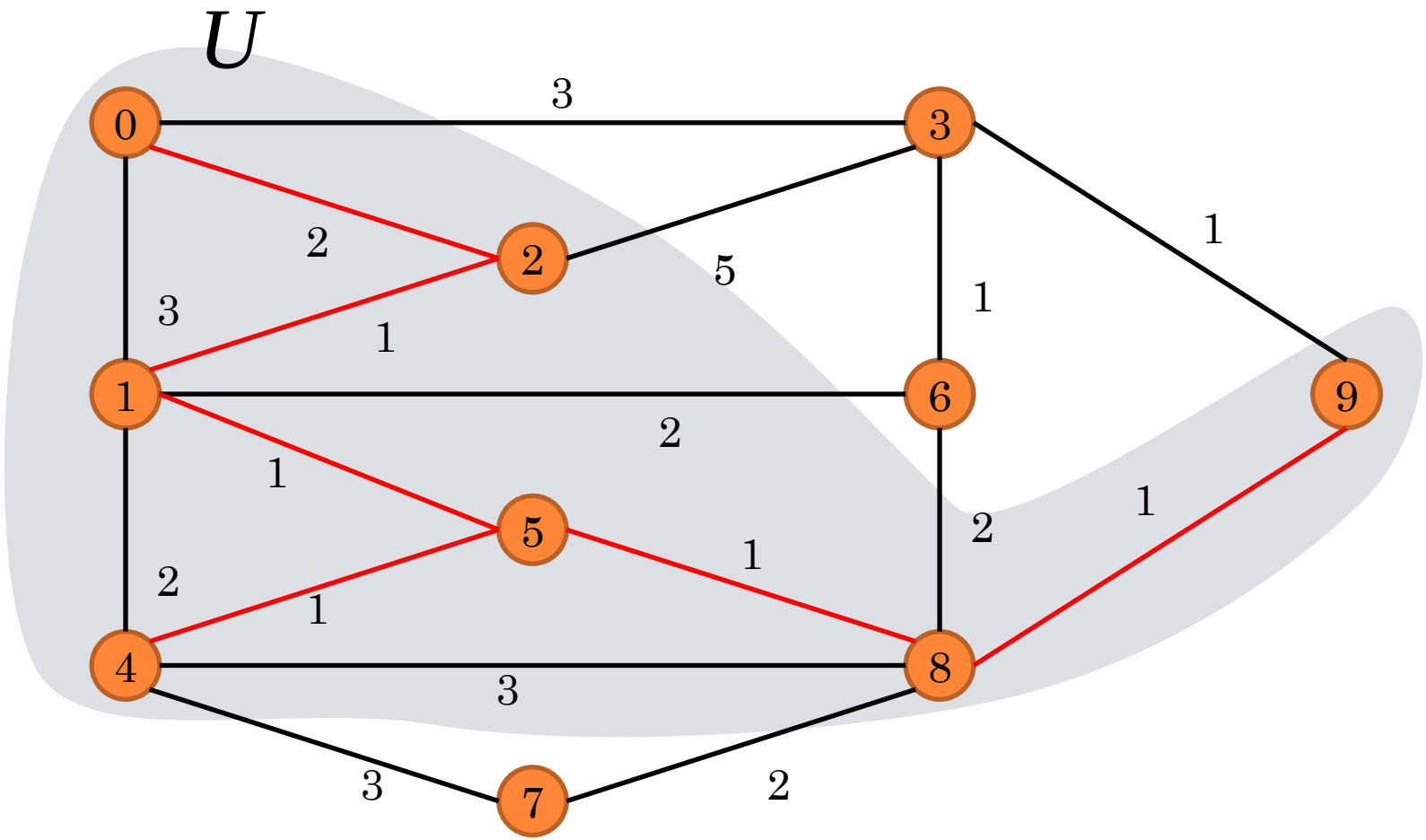

U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
2
1
3
3
2

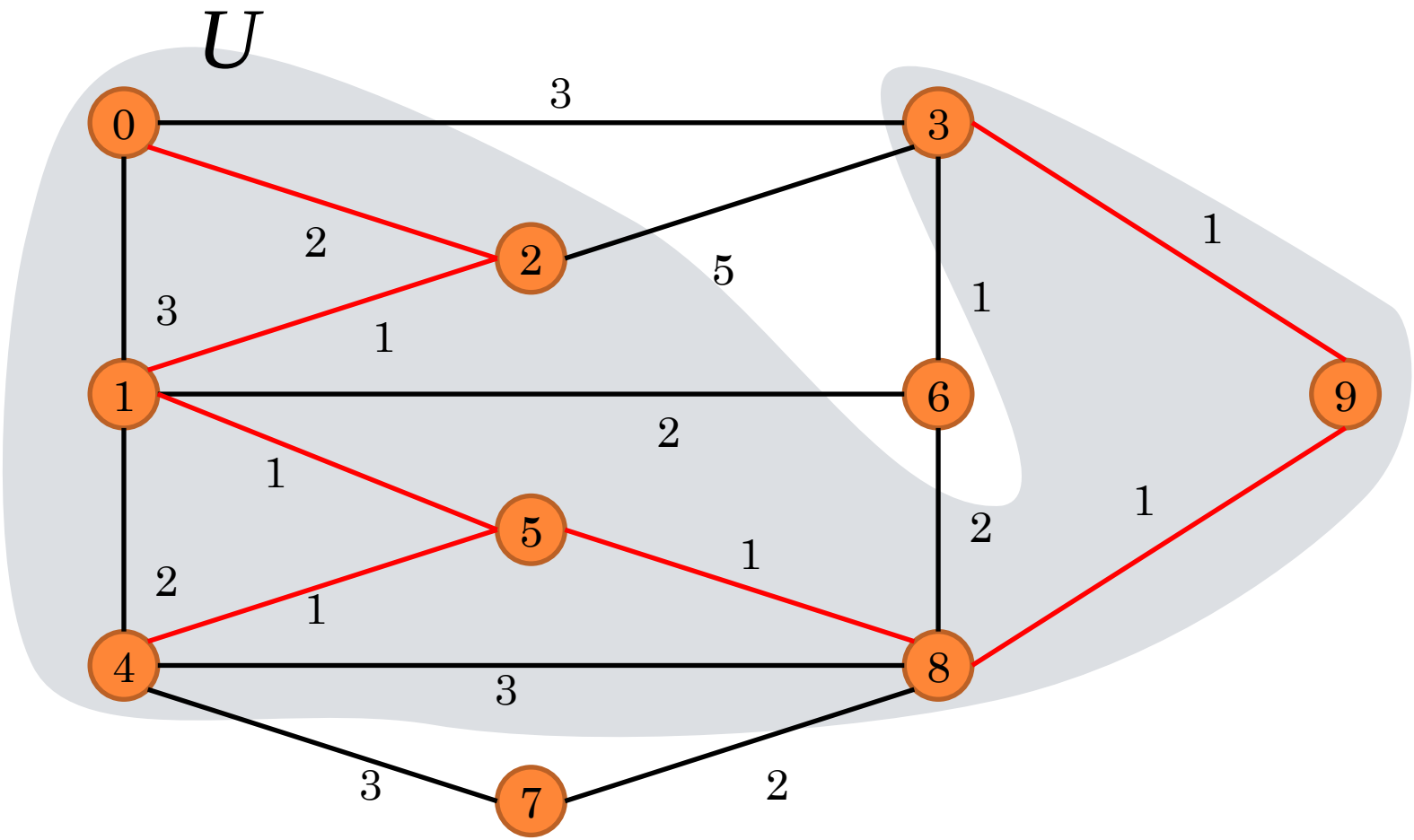

U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

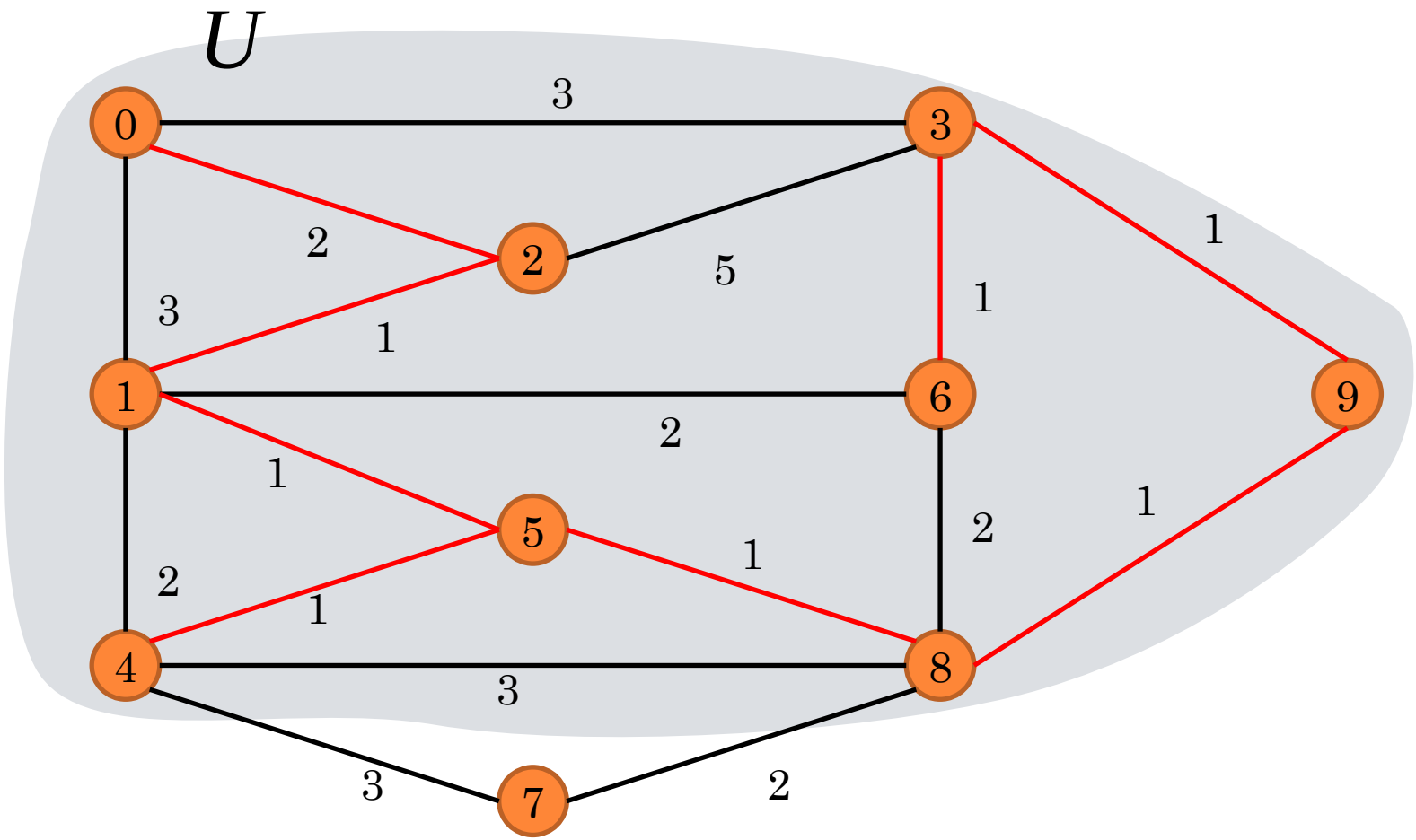

U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
2
1
1
3
3
2


U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
2
1
3
3
2
1

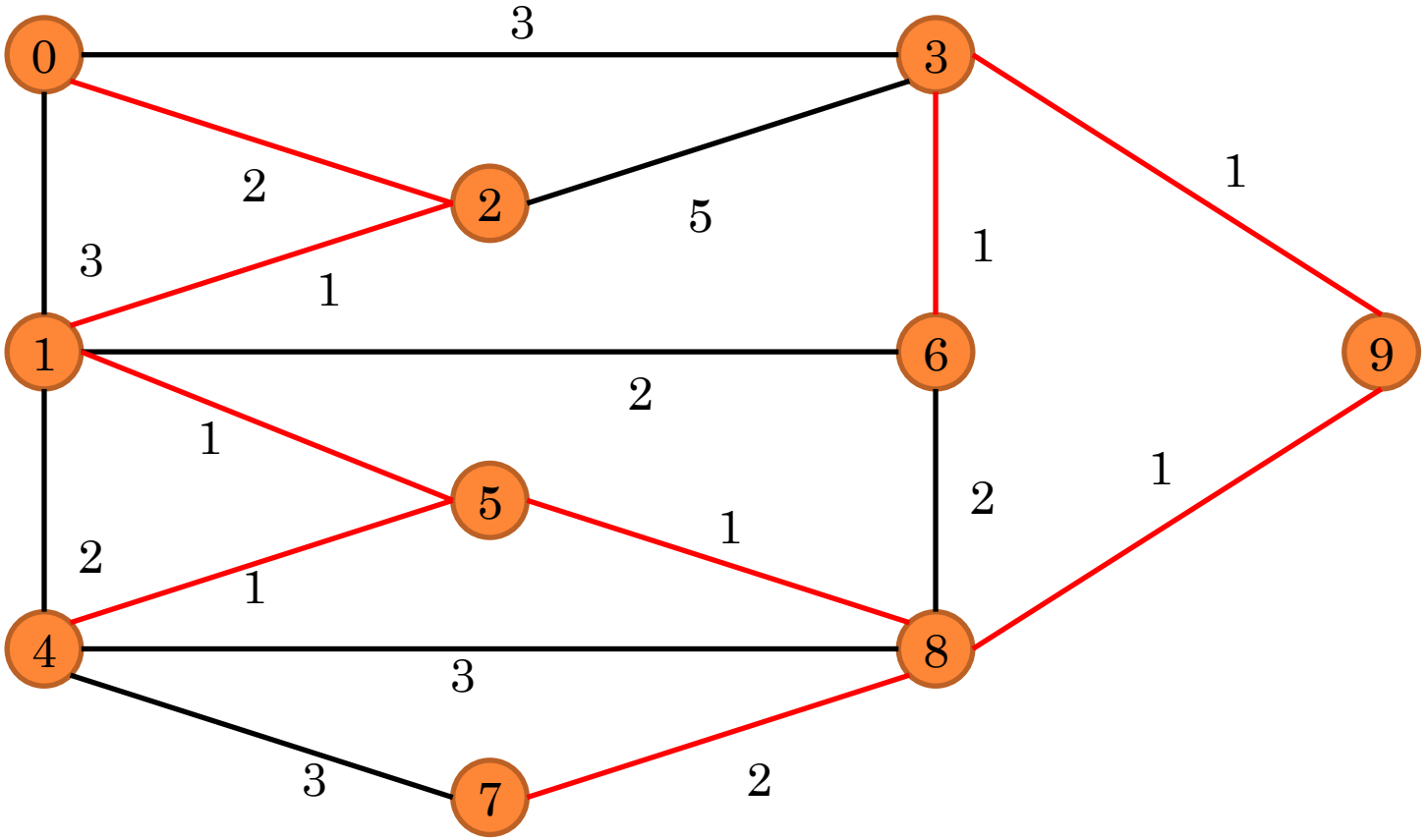

U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2


U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

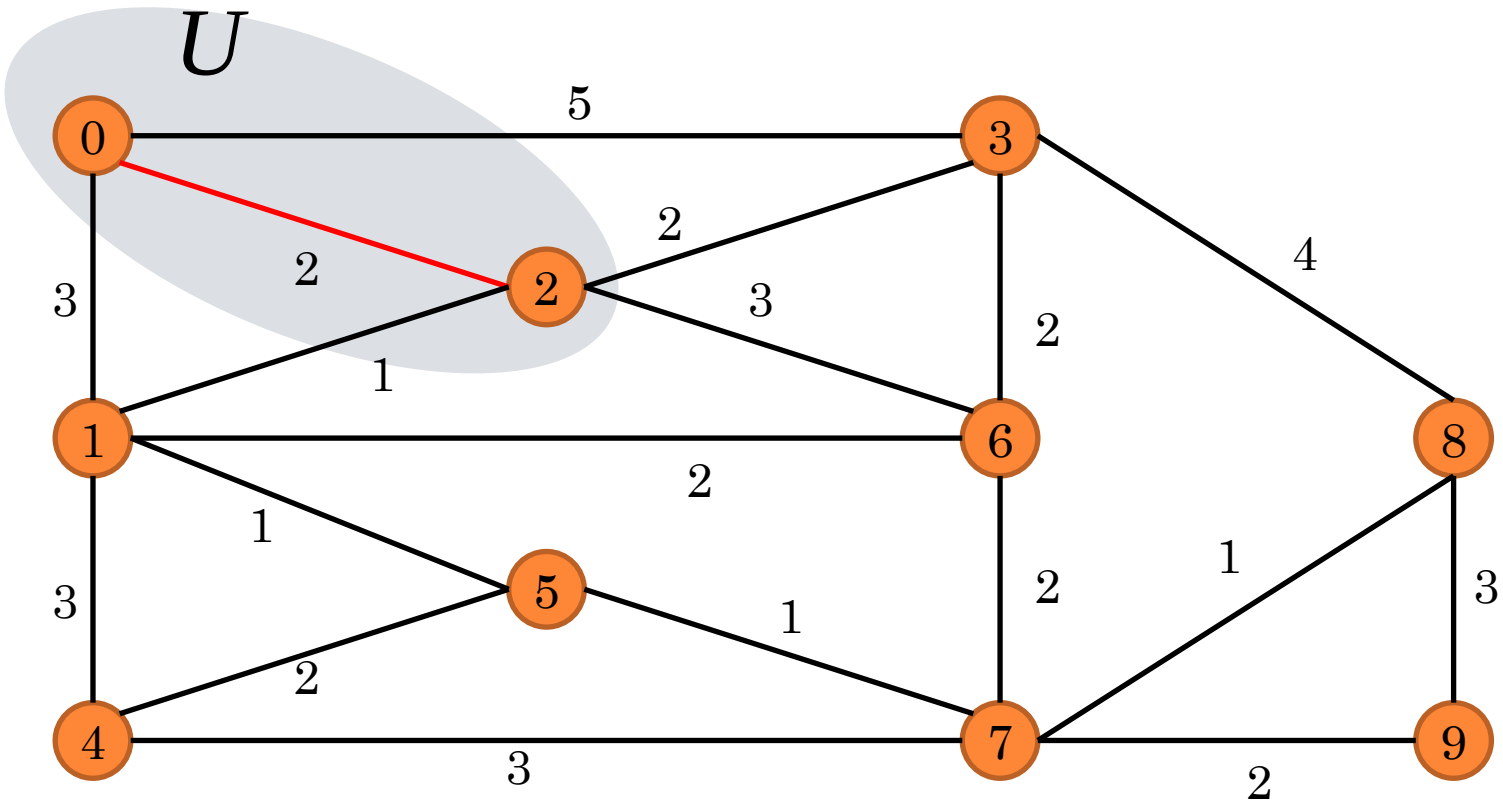

U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

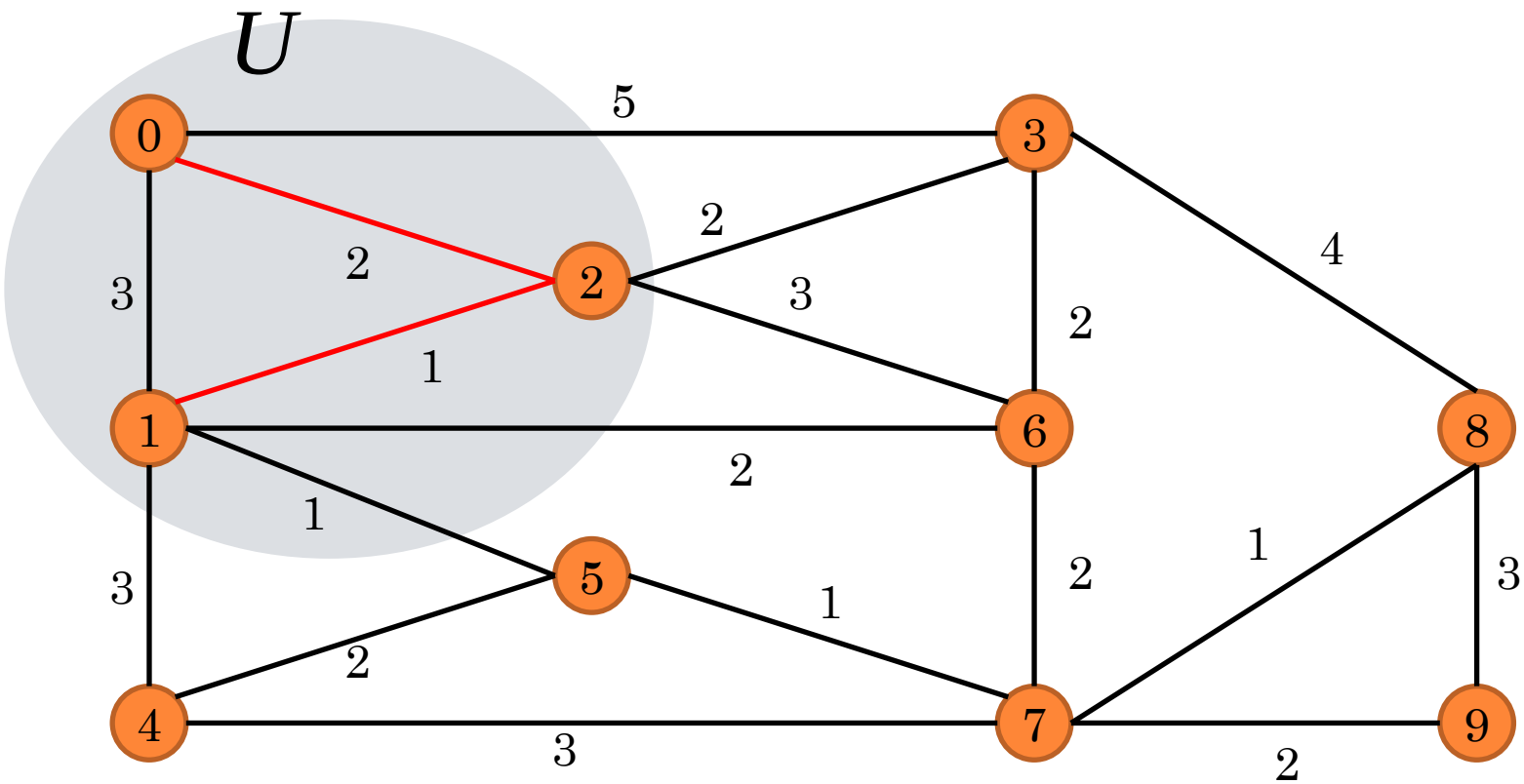

U
0
1
2
3
4
5
6
7
8
9
3
2
3
1
5
1
1
2
1
2
1
1
2
1
3
3
2

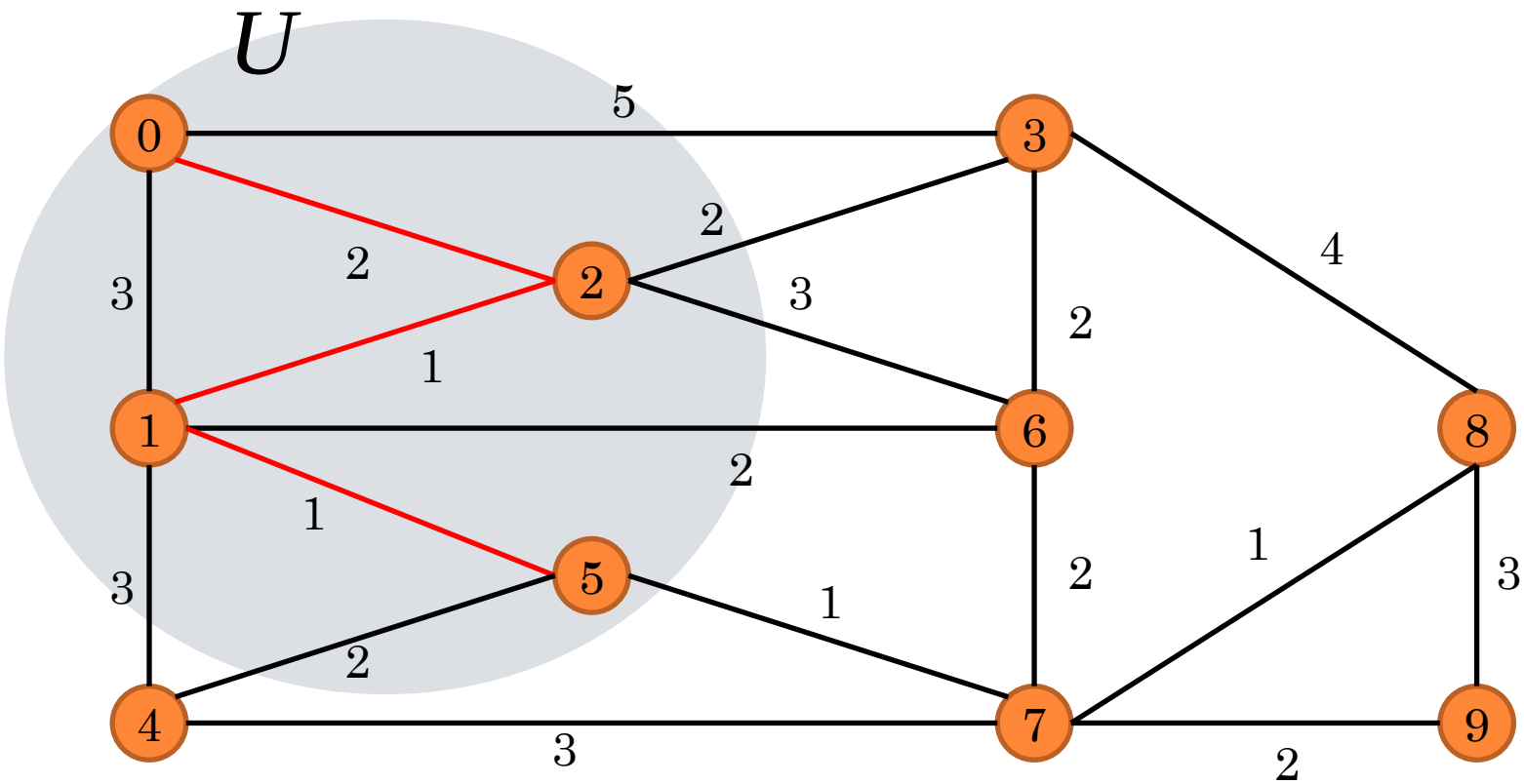
例1:最小木

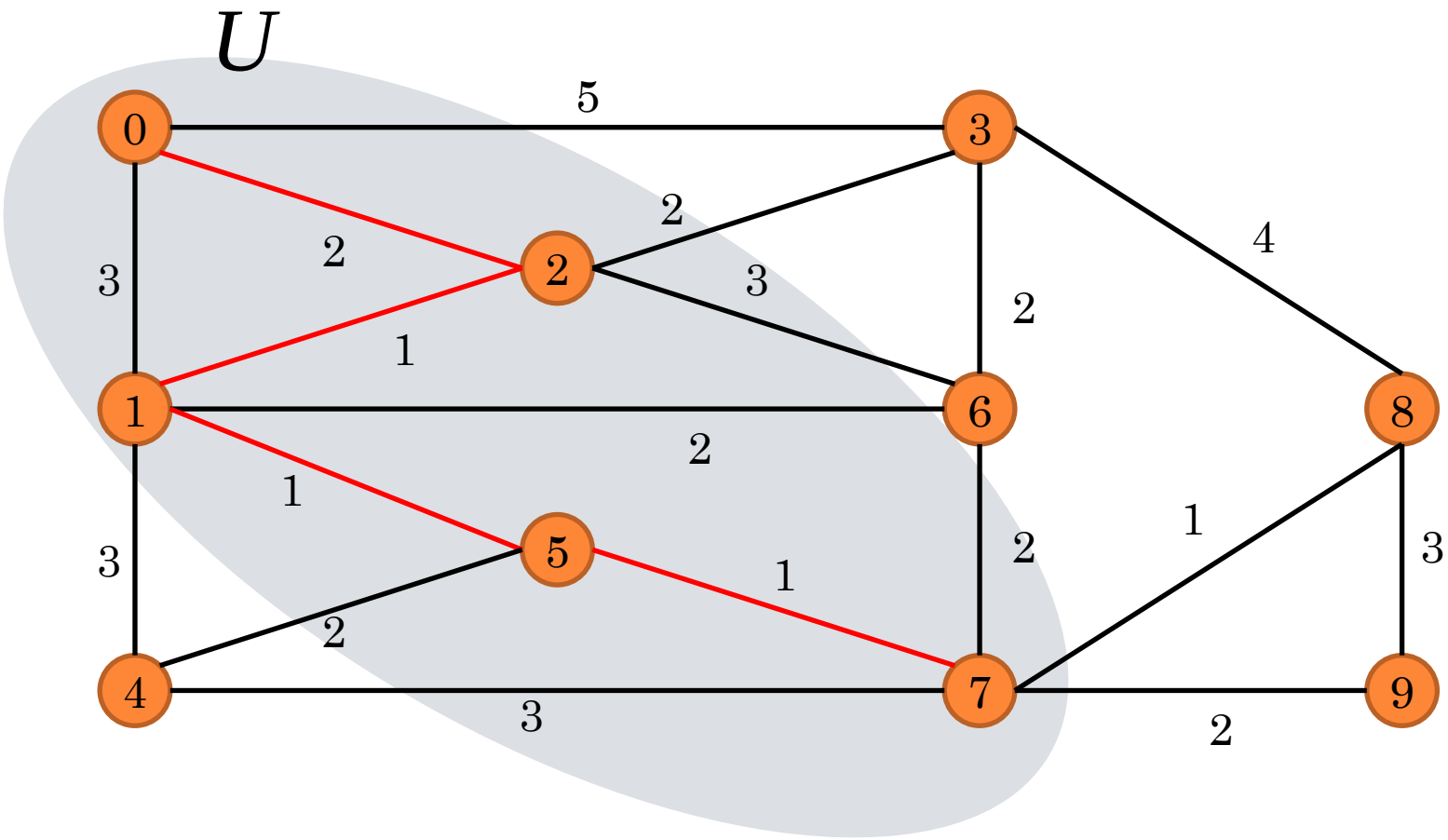


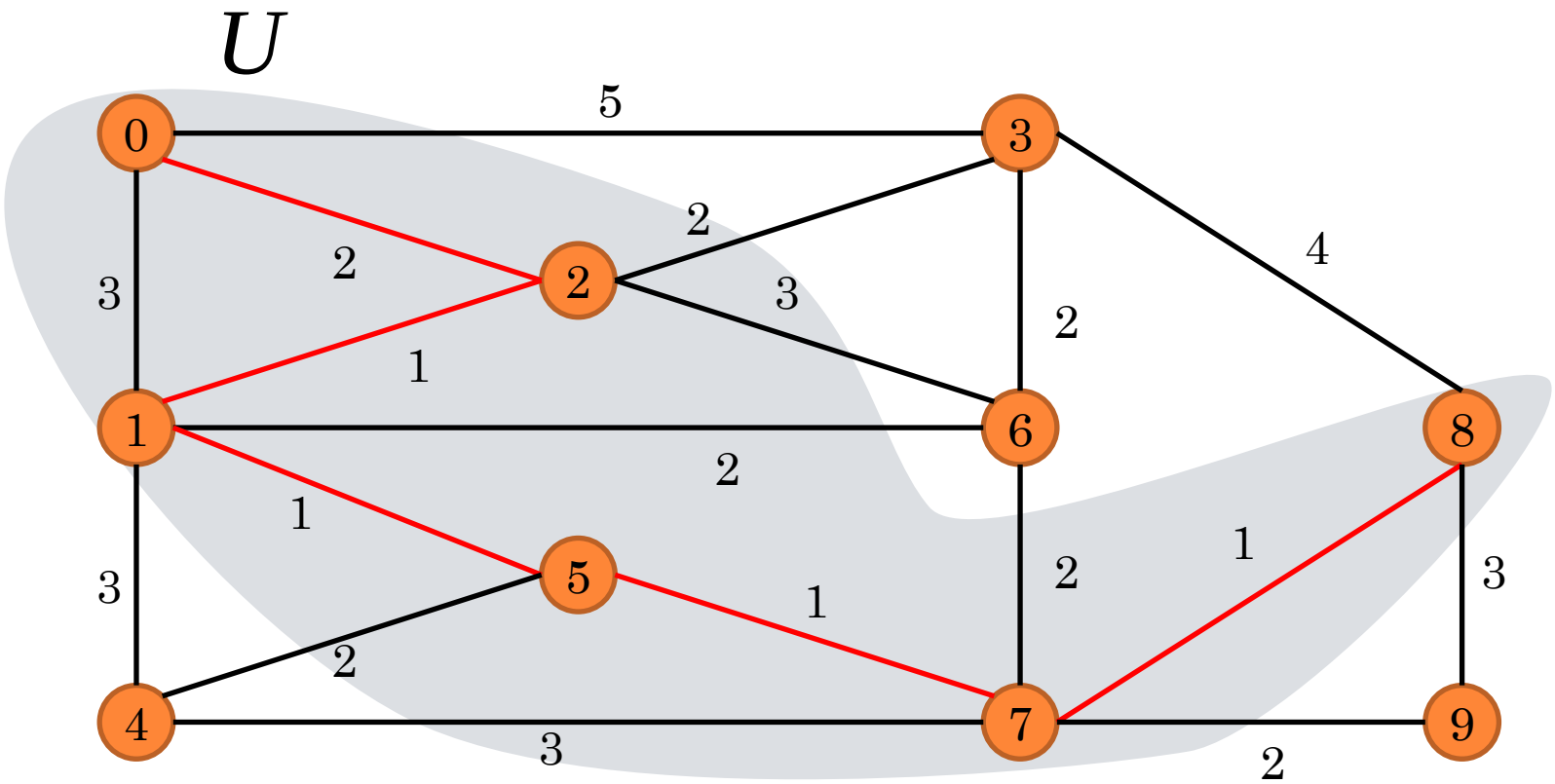
例2

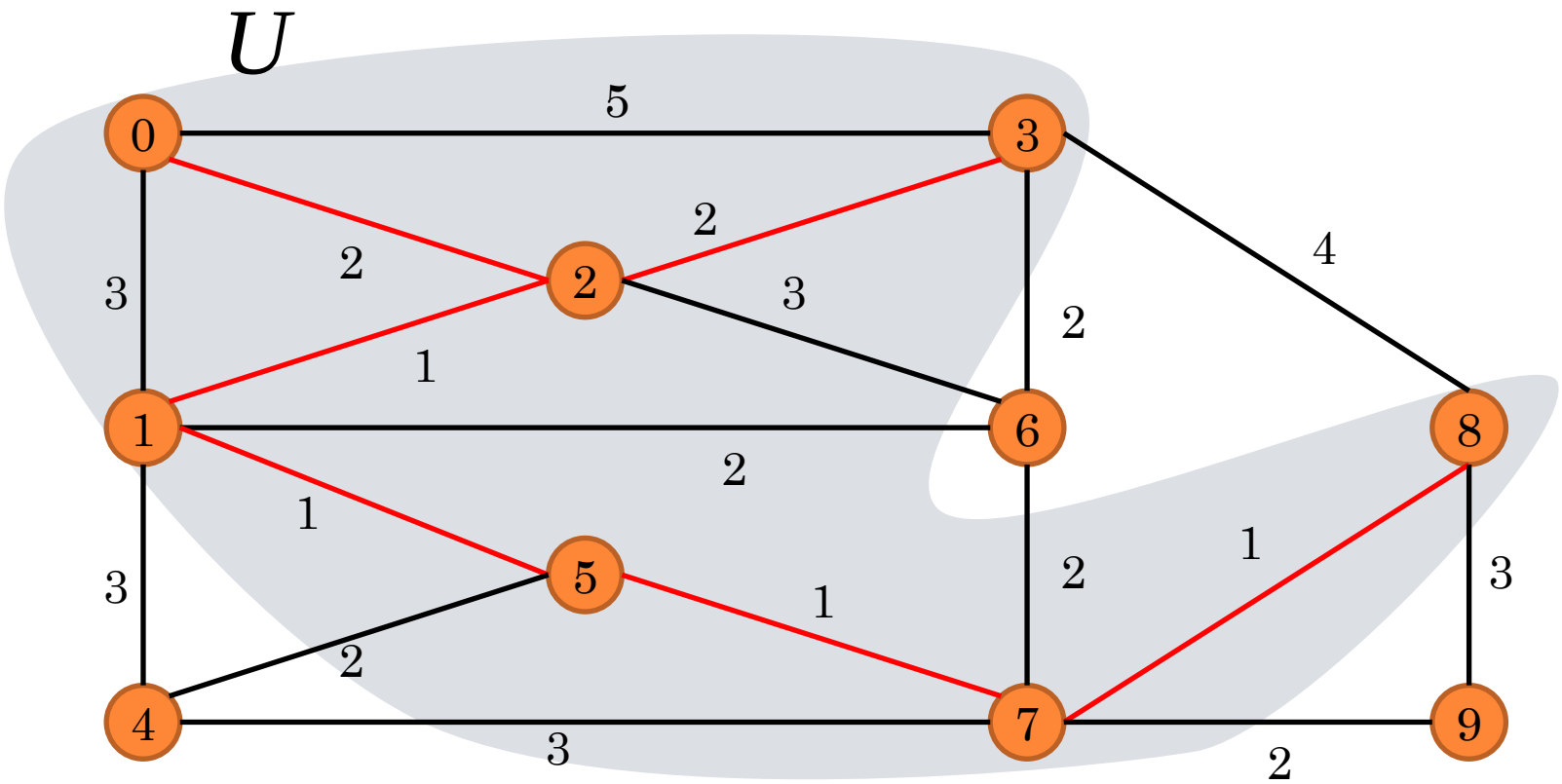


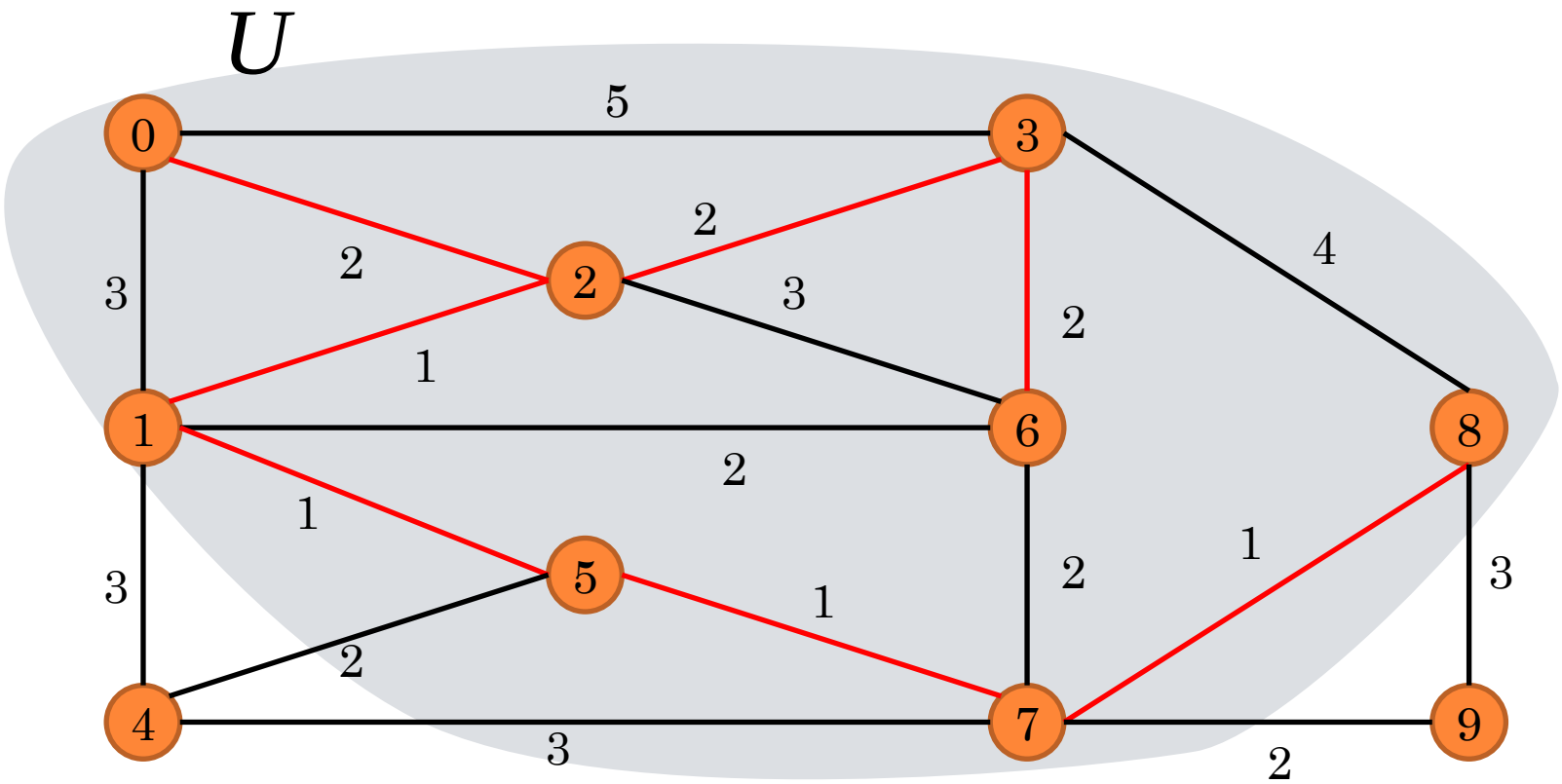

U
0
1
2
3
4
5
6
7
8
9
5
3
2
2
3
4
2
1
2
1
3
2
1
2
1
3
3
2

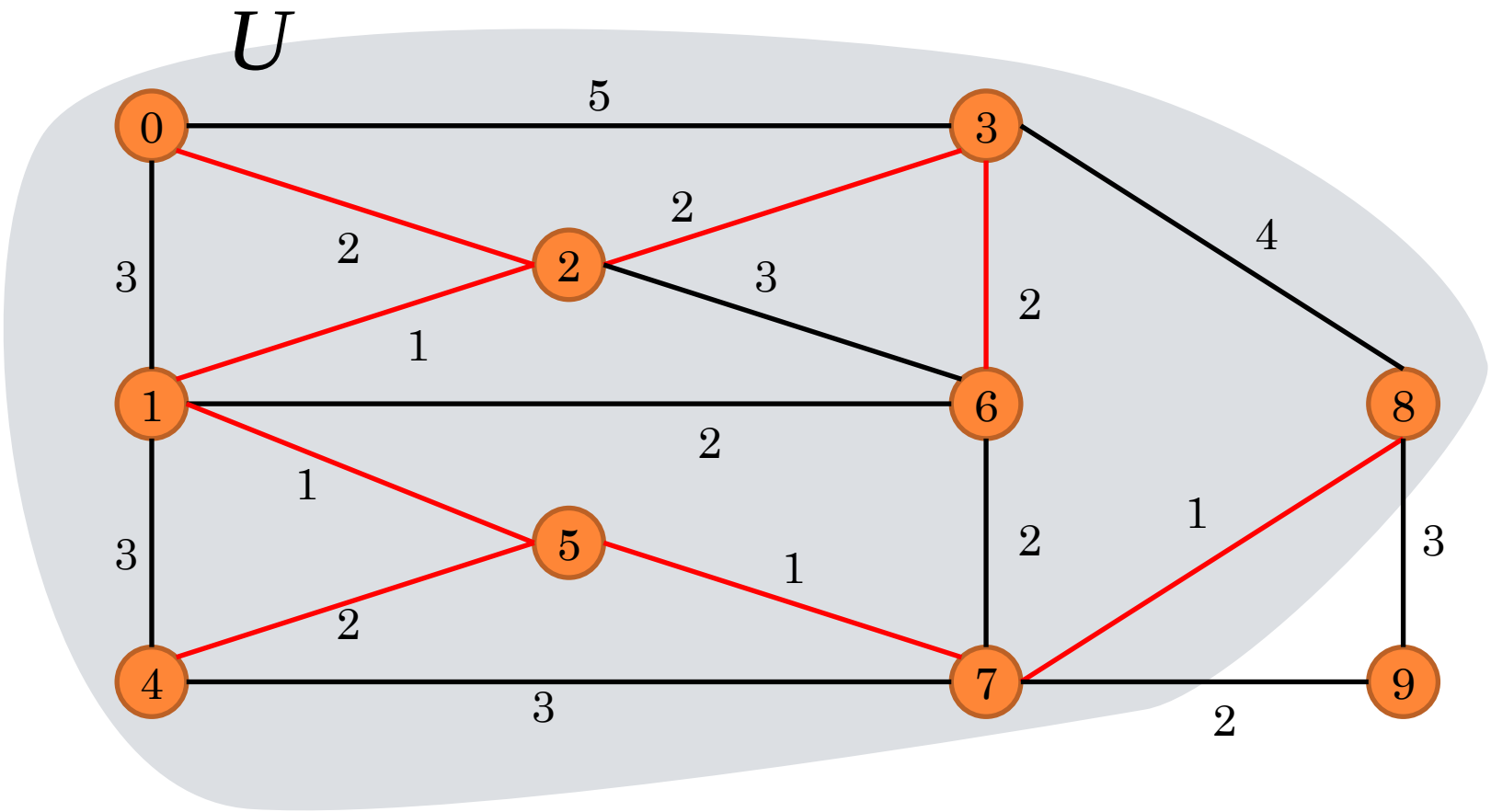

U
0
1
2
3
4
5
6
7
8
9
5
3
2
2
3
4
2
1
2
1
3
1
2
1
3
2
3
2

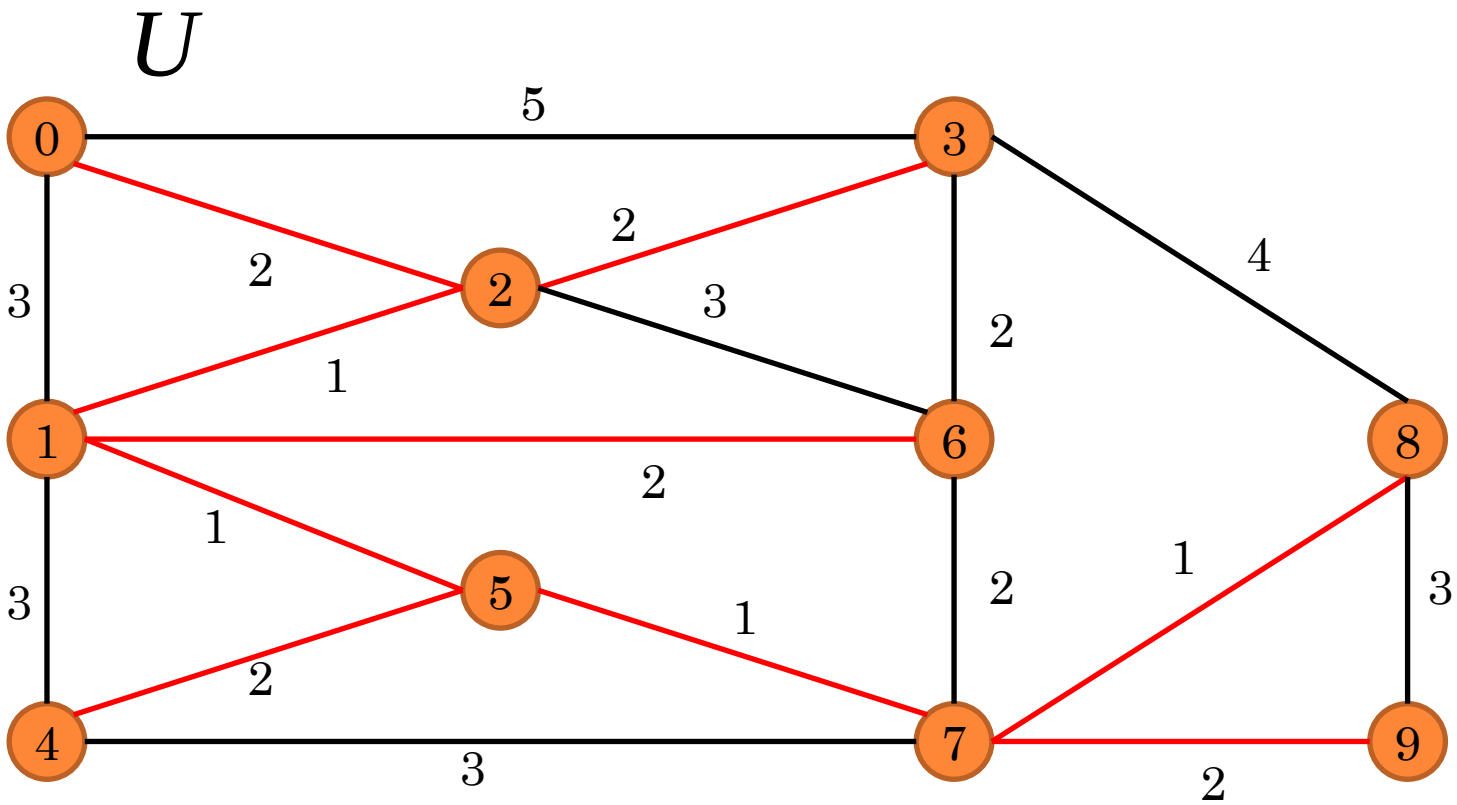

U
0
1
2
3
4
5
6
7
8
9
5
2
2
3
4
3
2
1
2
1
3
2
1
1
3
2
3
2


U
0
1
2
3
4
5
6
7
8
9
5
3
2
2
3
4
2
1
2
1
3
1
2
1
3
2
3
2


U
0
1
2
3
4
5
6
7
8
9
5
3
2
2
3
4
2
1
2
1
3
1
2
1
3
2
3
2

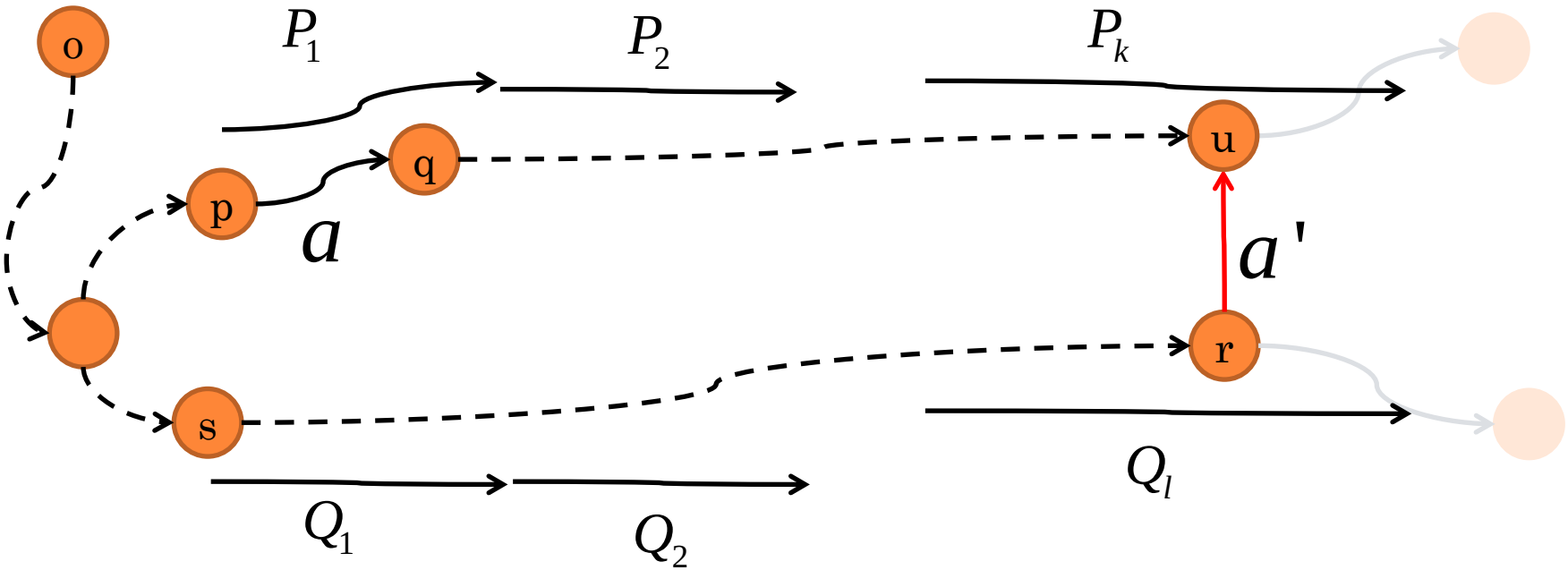

U
0
1
2
3
4
5
6
7
8
9
5
3
2
2
3
4
2
1
2
1
3
1
2
1
3
2
3
2


U
0
1
2
3
4
5
6
7
8
9
5
3
2
2
3
4
2
1
2
1
3
1
2
1
3
2
3
2


U
0
1
2
3
4
5
6
7
8
9
5
3
2
2
3
4
2
1
2
1
3
2
1
3
1
2
3
2

JARNÍK-PRIMアルゴリズムが正しいこと

- Jarník-Primアルゴリズム実行中の木Tは、Uによって誘導されるGの部分グラフ$G(U)$における最小木になっている。
- 証明
  - Tのある枝aをTに含まれない枝a'に置き換えることで、より小さい木ができる ($w(a')<w(a)$) ことを仮定して、矛盾を導く。



- Oを根とする木Tにおいて、弧aの代わりに弧a'としたほうが、重みが小さくなると仮定する。
- 上の枝で、弧aを先頭に連続して伸びた道をP_1とし、その後、下の枝で連続して伸びた道をQ_1とする。その後、P_2、Q_2と交互に伸びるとする。他の枝は無視する。
- 弧a'の両端の頂点は道P_k及びQ_lに属しているとする。

P_i を構成する弧を $\left\{a^i{}_0, a^i{}_1, \cdots, a^i{}_{n(i)}\right\}$ 、 Q_i を構成する弧を $\left\{b^i{}_0, b^i{}_1, \cdots, b^i{}_{n(i)}\right\}$ とする。P_i の後 Q_i が伸びることから

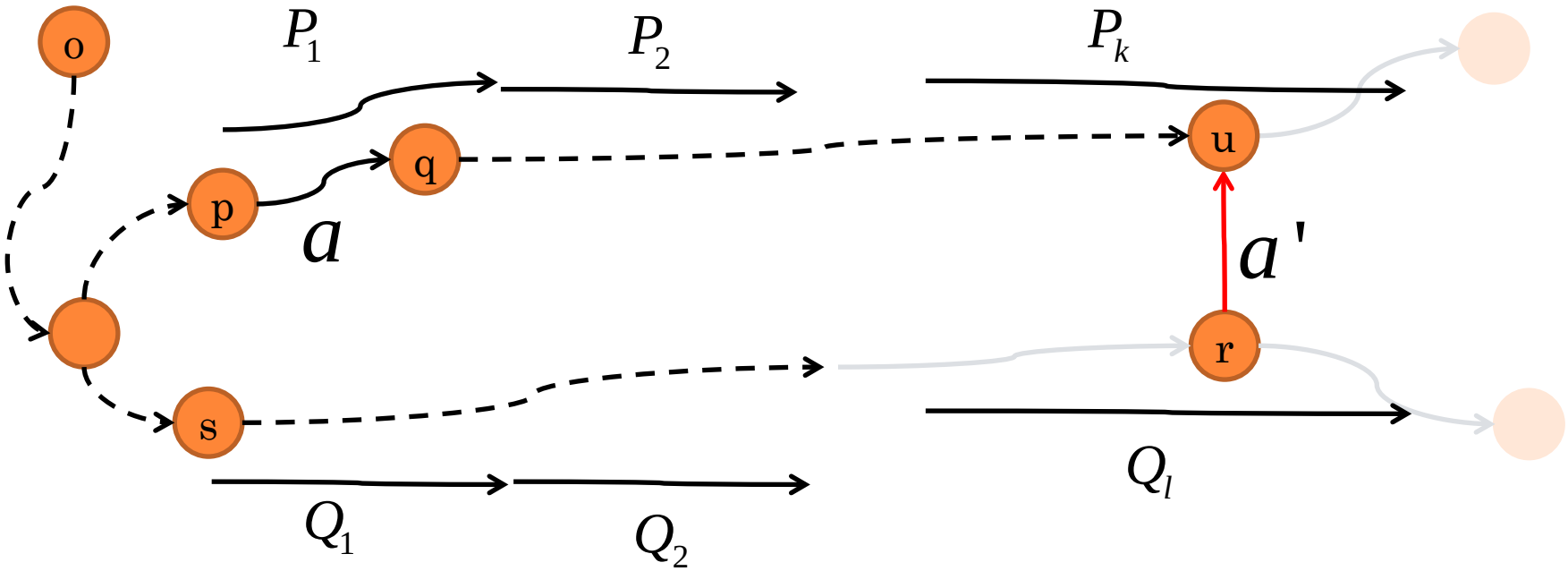
$$\forall i, \forall j, w\left(a_j^i\right) \leq w\left(b_0^i\right), w\left(b_j^i\right) \leq w\left(a_0^{i+1}\right)$$

先頭の弧に注目すると、以下が成り立つ

$$\forall i, w\left(a_0^i\right) \leq w\left(b_0^i\right) \leq w\left(a^{i+1}{}_0\right)$$

つまり、各道の先頭の弧の重みは以下を満たす。

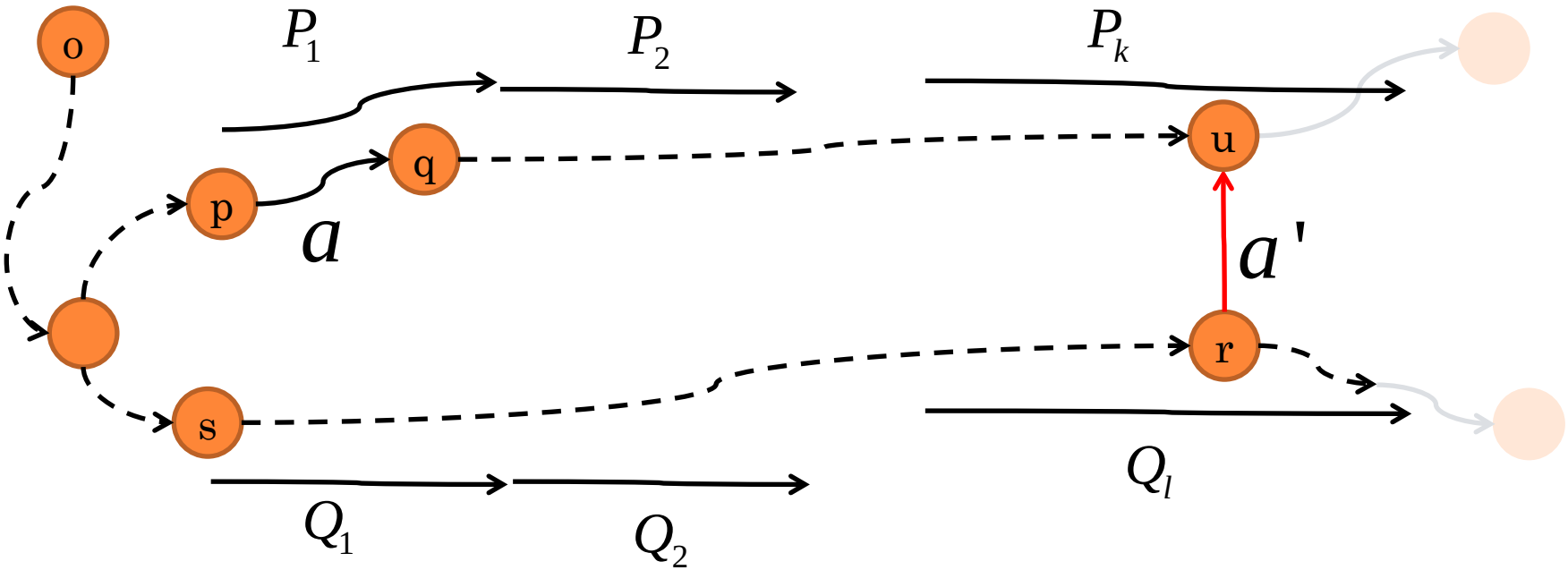
$$\forall i, w(a) \leq w\left(a_0^i\right), w(a) \leq w\left(b_0^i\right)$$



$k \le l$ の場合、つまり上の枝が頂点 u まで伸びたとき、下の枝は未だ頂点 r に伸びていない場合を考える。このとき、上の道 P_k が伸びるときに、枝 a' がアルゴリズムによって採用されなかったことから

$$w(a) \le w\left(b^k{}_0\right) \le w(a')$$

となり矛盾する。



逆の $k > l$ の場合、つまり上の枝が頂点 u まで伸びたとき、下の枝は頂点 r を過ぎて伸びていた場合を考える。下の道 Q_l が伸びるときに、枝 a' がアルゴリズムによって採用されなかったことから

$$w(a) \le w\left(a^k{}_0\right) \le w(a')$$

となり矛盾する。