グラフを扱おう：最短路問題

北海道大学 情報科学研究科

修士1年 栗田和宏

グラフの定義(数学)

- グラフとは集合の組であり，$G = (V, E)$と表される．ここで$E \subseteq (V, V)$である．
- Vを**頂点集合**と呼び，Eを**辺集合**と呼ぶ．

グラフの表現法

- グラフを表現する方法として2つ考えられる.

1. 隣接行列
2. 隣接リスト

グラフの表現法

- グラフを表現する方法として2つ考えられる.

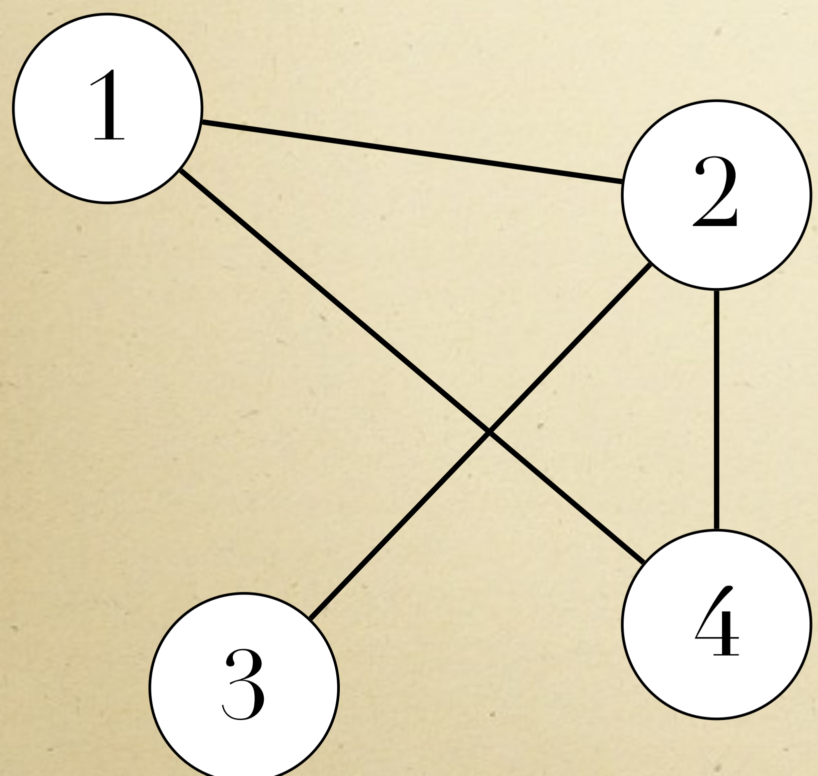
1. 隣接行列 ← メモリ：大　検索：速
2. 隣接リスト← メモリ：小　検索：遅

隣接行列

- グラフの各頂点間の隣接関係を行列で表す.

隣接行列

- グラフの各頂点間の隣接関係を行列で表す.



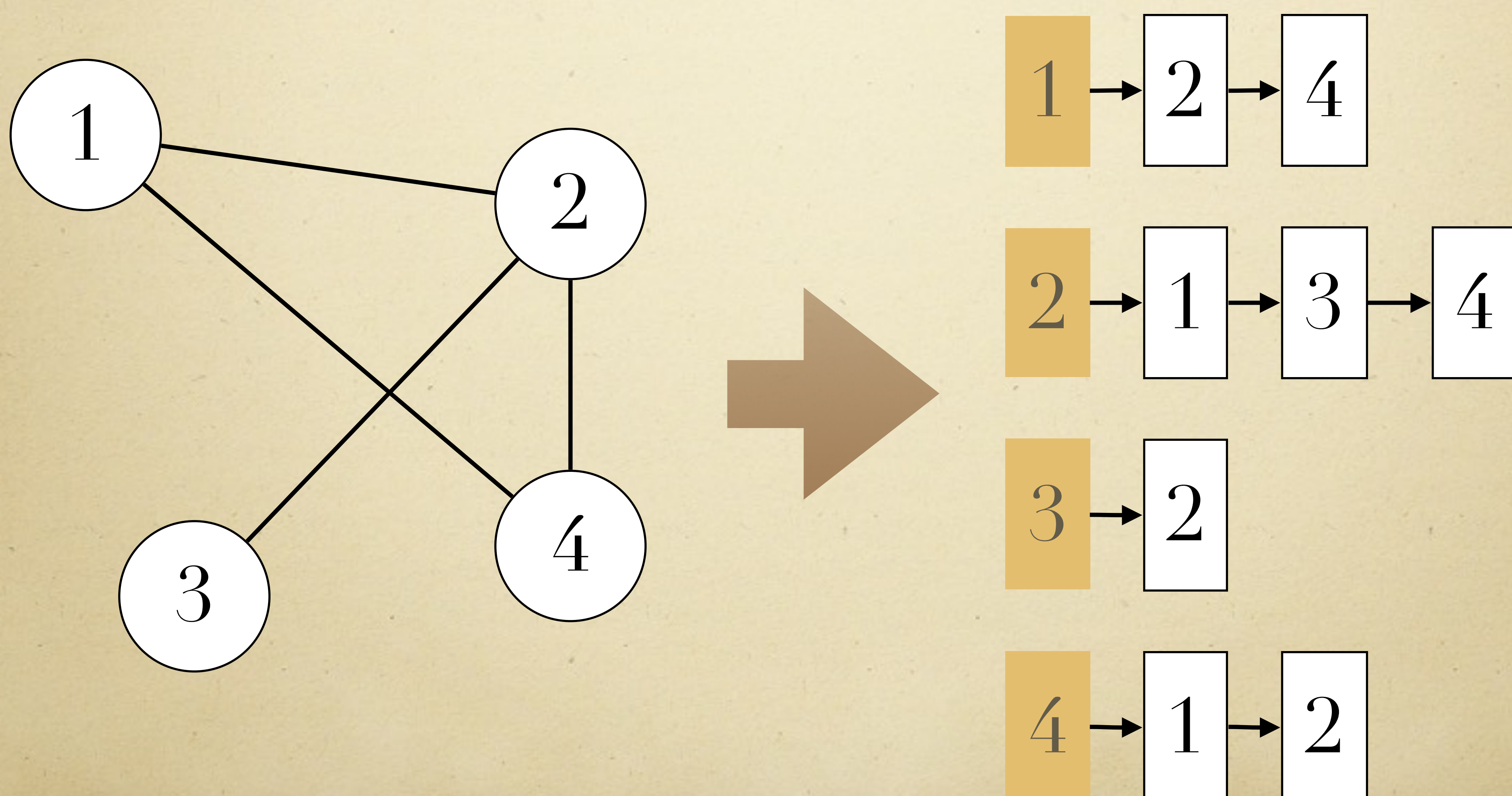
0	1	0	1
1	0	1	1
0	1	0	0
1	1	0	0

隣接行列

- 隣接行列には$O(V^2)$のメモリとある2頂点間に辺があるかを検索するのに$O(1)$時間がかかる

隣接リスト

- グラフの各頂点の隣接関係をリストで保持する.



隣接リスト

- 隣接リストには$O(V + E)$のメモリとある2頂点間に辺があるかを検索するのに$O(E)$時間がかかる.

単一始点最短路問題

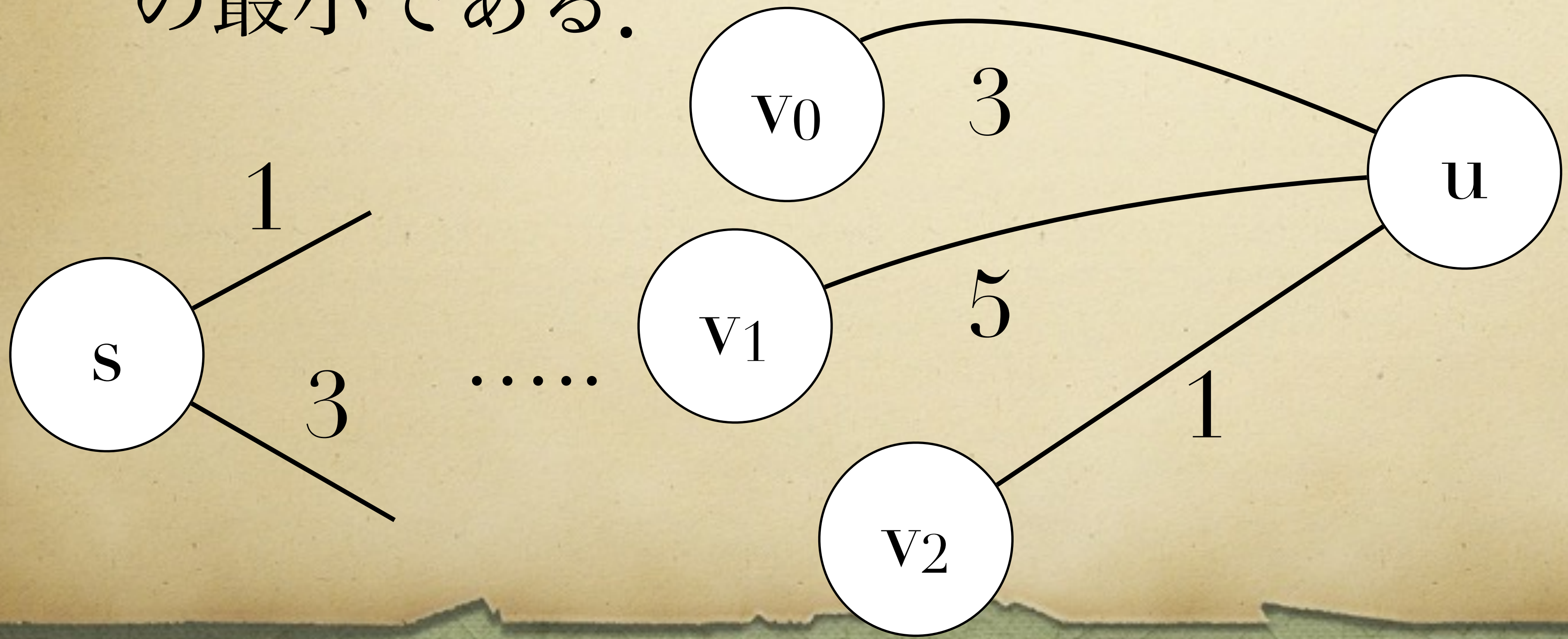
- **単一始点最短路問題**とは与えられた頂点sから他のすべての頂点への最短路を求める問題である.

単一始点最短路問題

- 単一始点最短路問題とは与えられた頂点sから他のすべての頂点への最短路を求める問題である.
- ベルマンフォード法
- ダイクストラ法

最短路の性質

- 最小コストの重要な性質として次のことが言える.
- 頂点uの最小コストは
  uと隣接する頂点の最短路 + uへ移動するコスト
  の最小である.

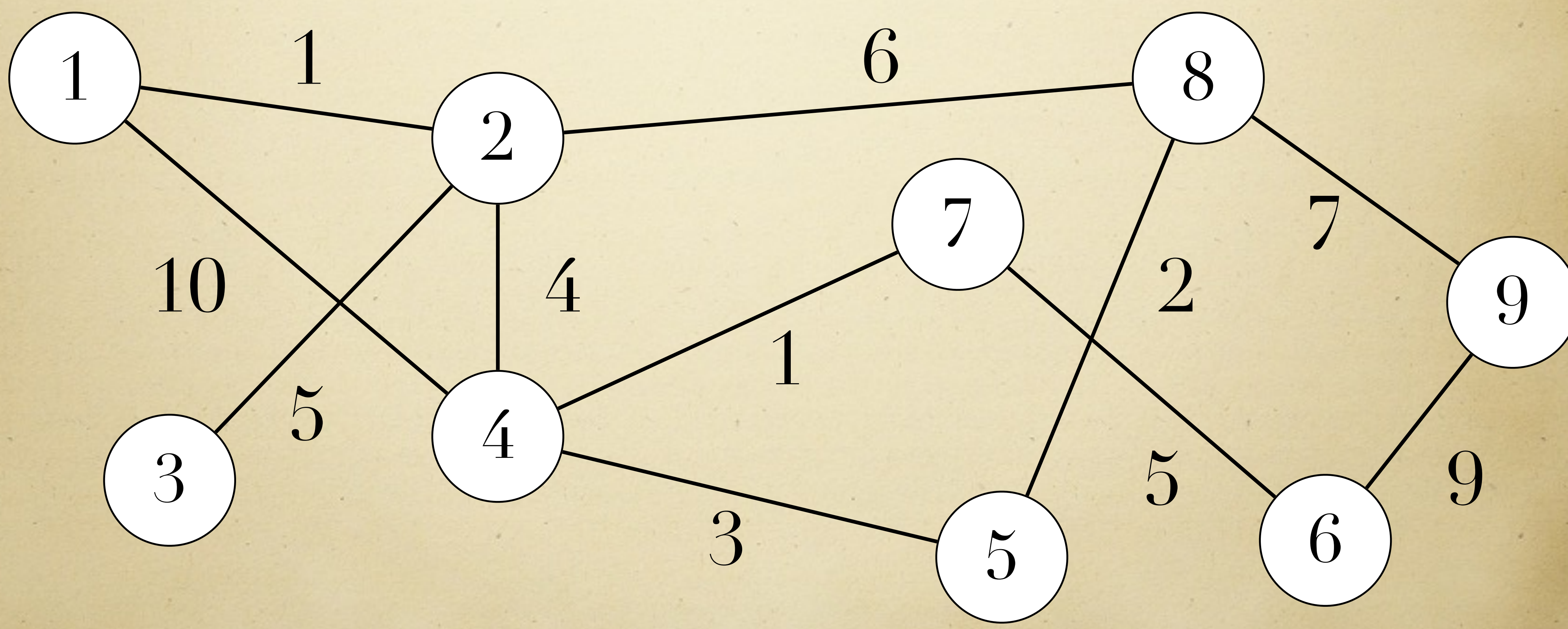


ベルマンフォード法

1. 始点以外のコストをINFに，始点のコストを0にする.
2. すべての辺を使い，コストを更新する
3. コストの更新ができなかったら終了，更新ができたら2に戻る.

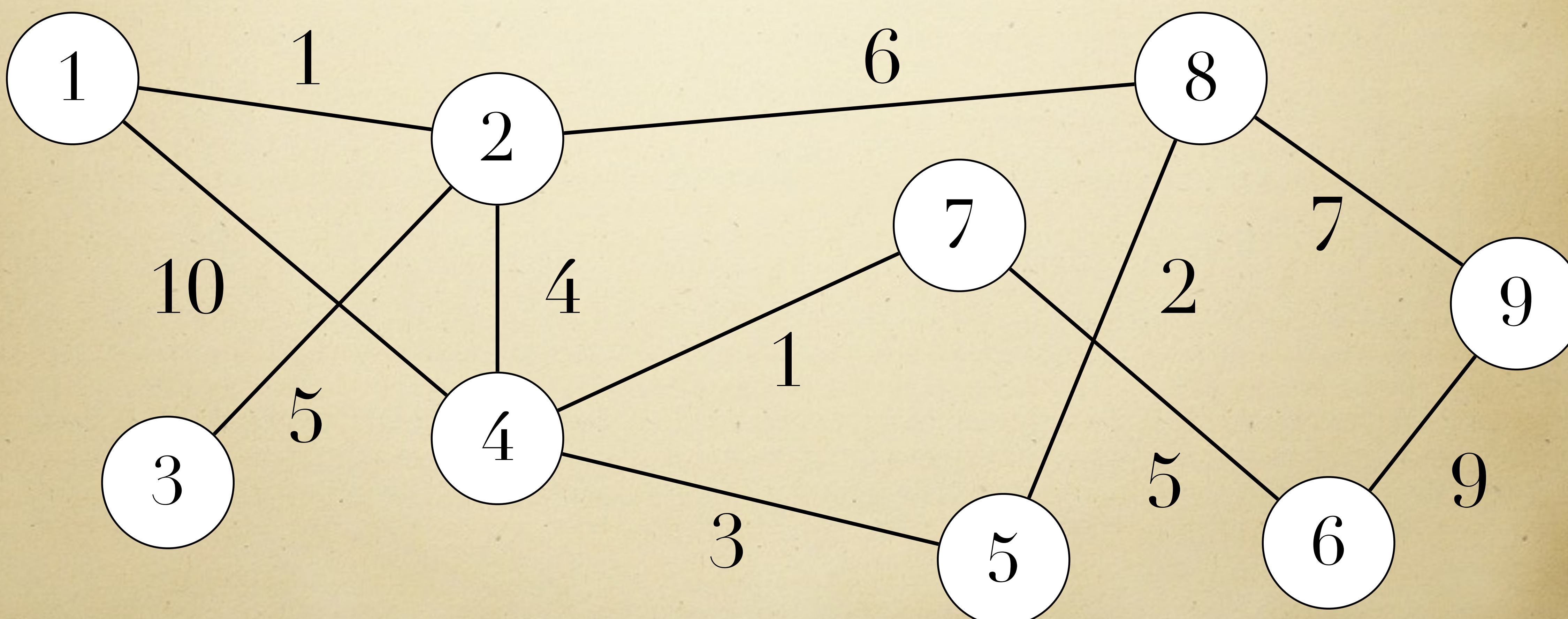
コストの更新

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



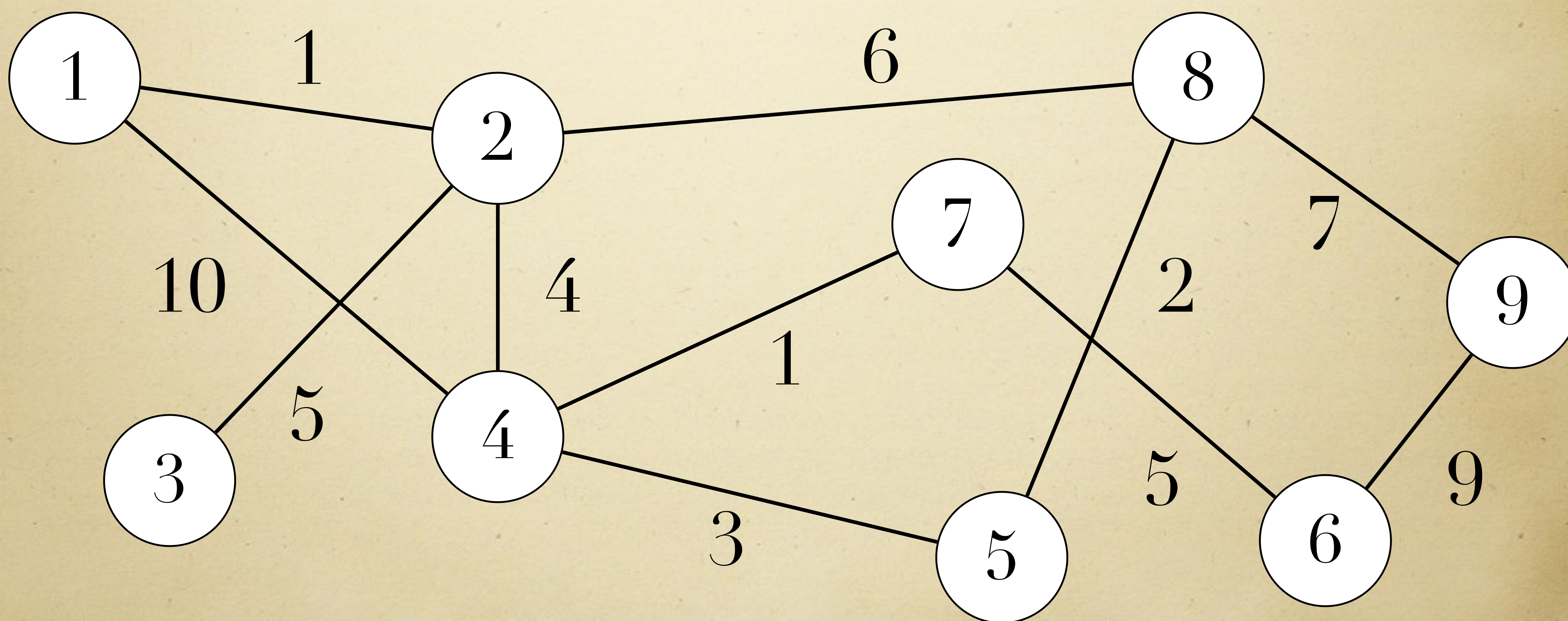
コストの更新

1	2	3	4	5	6	7	8	9
0	1	INF	10	INF	INF	INF	INF	INF



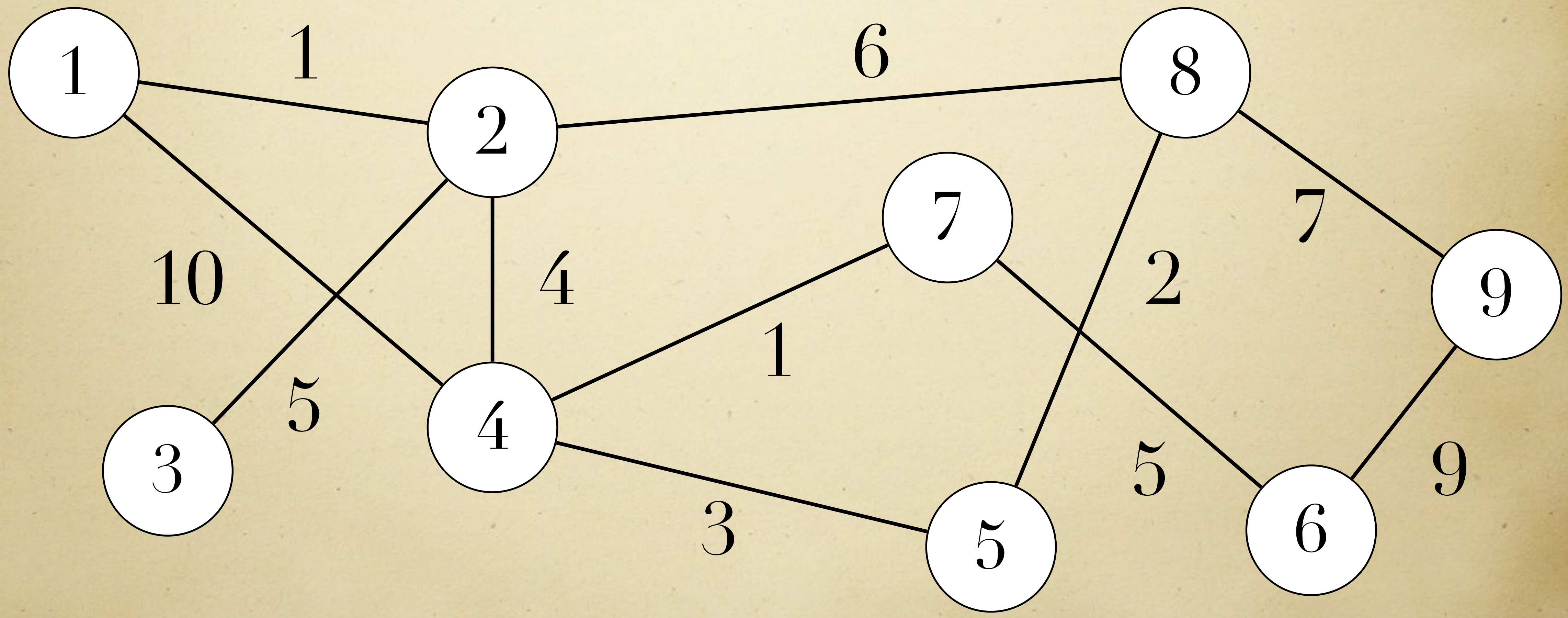
コストの更新

1	2	3	4	5	6	7	8	9
0	1	INF	10	INF	INF	INF	INF	INF



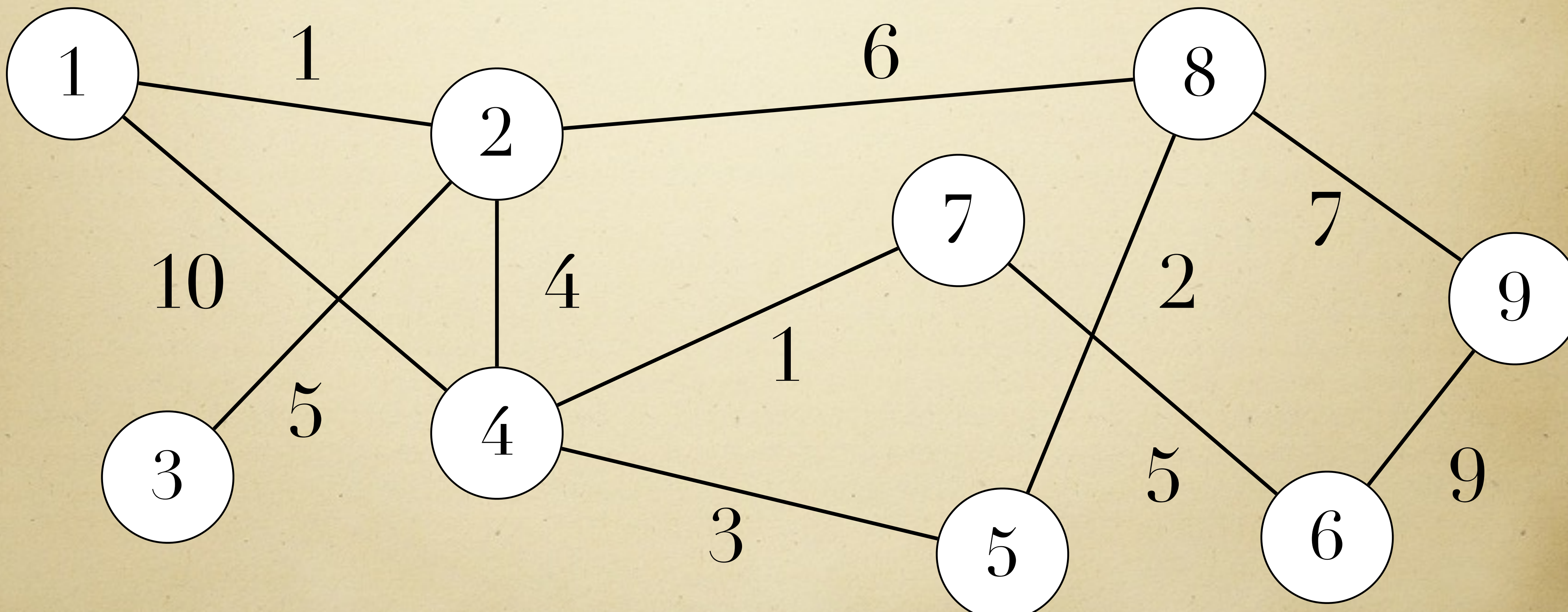
コストの更新

1	2	3	4	5	6	7	8	9
0	1	6	5	13	INF	11	7	INF



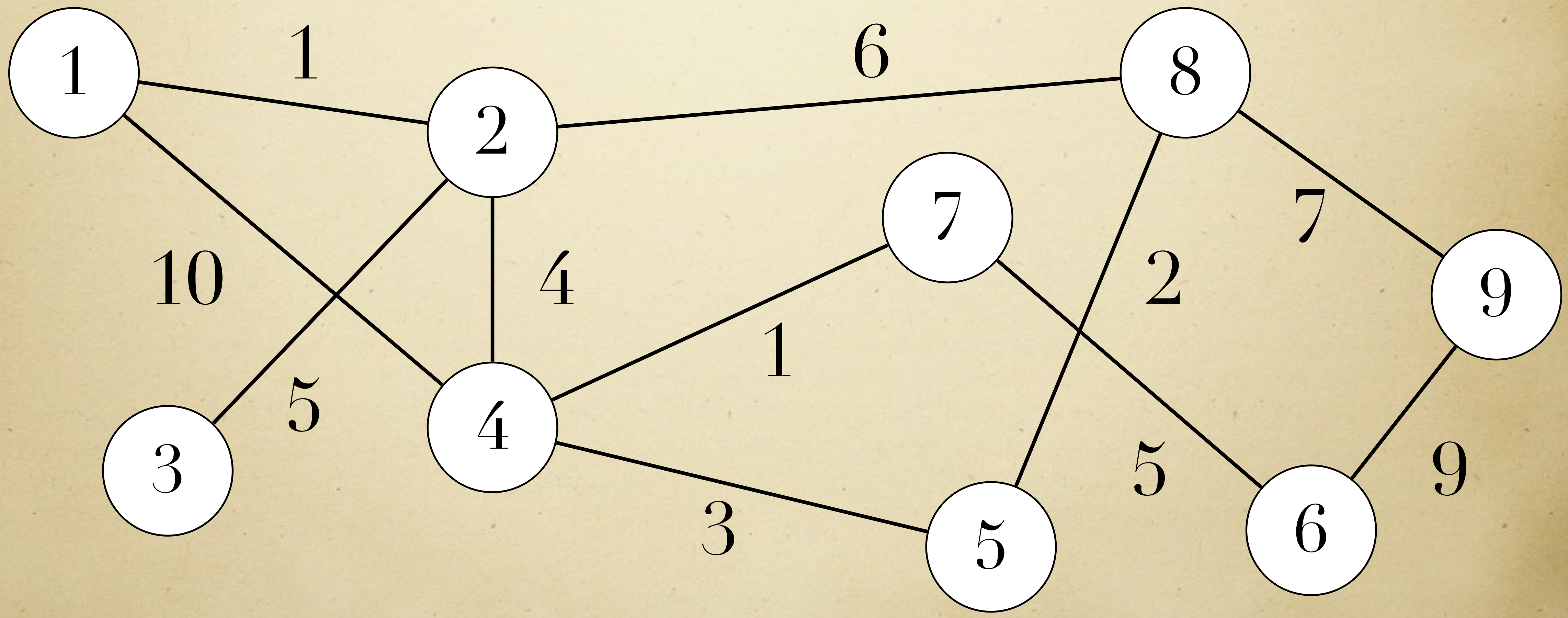
コストの更新

1	2	3	4	5	6	7	8	9
0	1	6	5	13	INF	11	7	INF



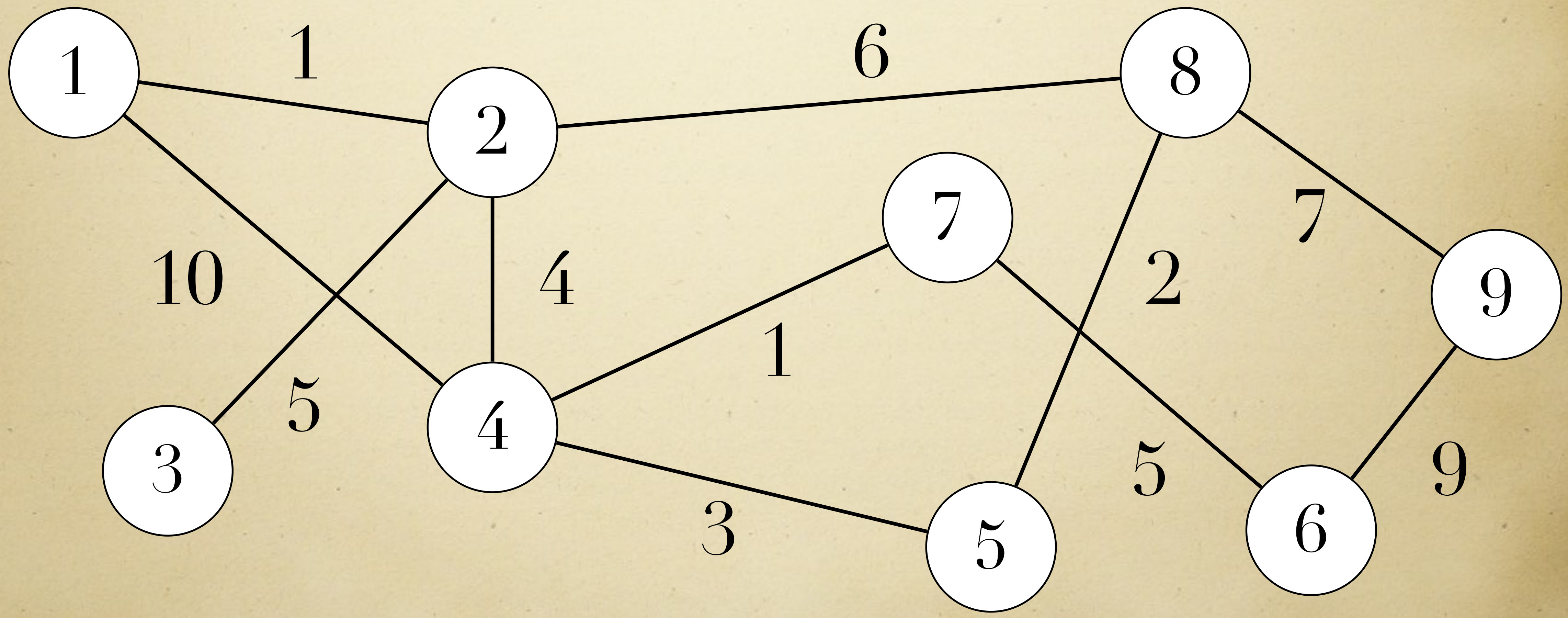
コストの更新

1	2	3	4	5	6	7	8	9
0	1	6	5	8	16	6	7	14



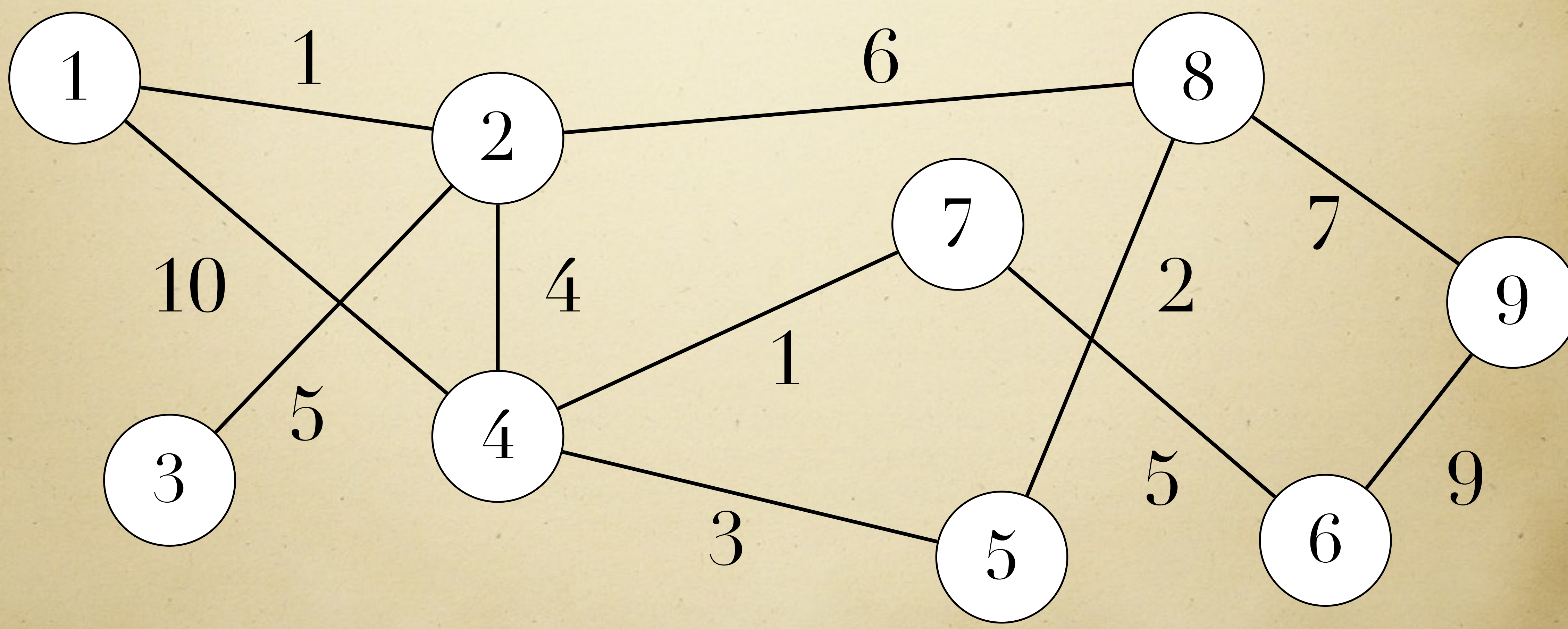
コストの更新

1	2	3	4	5	6	7	8	9
0	1	6	5	8	16	6	7	14



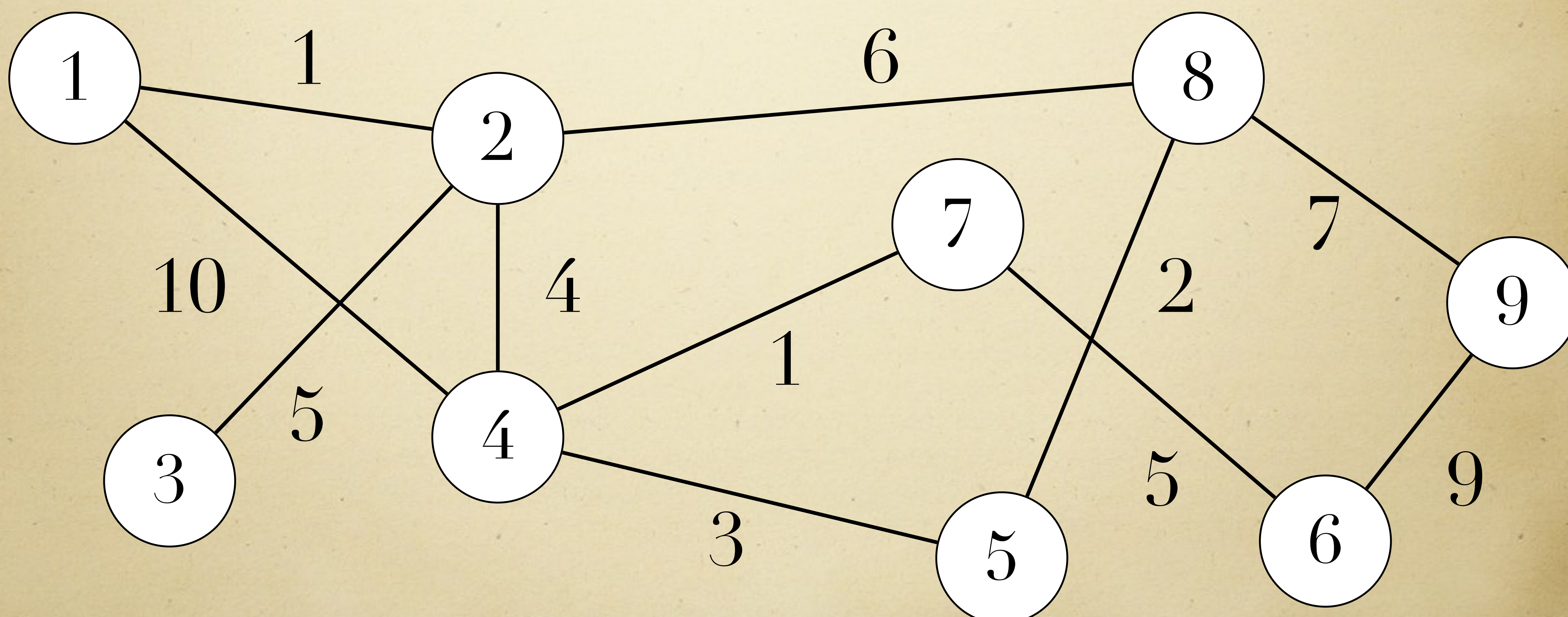
コストの更新

1	2	3	4	5	6	7	8	9
0	1	6	5	8	11	6	7	14



コストの更新

1	2	3	4	5	6	7	8	9
0	1	6	5	8	11	6	7	14



計算量

- 時間計算量：$O(VE)$ もしくは $O(V^3)$
- 空間計算量：$O(E)$ もしくは$O(V^2)$
- 時間計算量はグラフを隣接リストで持っておくと$O(VE)$になり，隣接行列で持っておくと$O(V^3)$となる．

コードの例：
ベルマンフォード法

- https://github.com/kazu0423/procon_example/blob/master/bellman_ford.cpp

計算量の解析

- while文が入っていて，ぱっと見での計算量がわかりづらい.
- このwhile文は最大V - 1回のループをする
→ なぜなら1回のループで少なくとも1つの頂点の最短コストが決定するからである.

計算量の解析

- while文中の計算量はすべての辺についての操作をしているのでO(E)となる.
- 全体としてはO(VE)となる.

ダイクストラ法

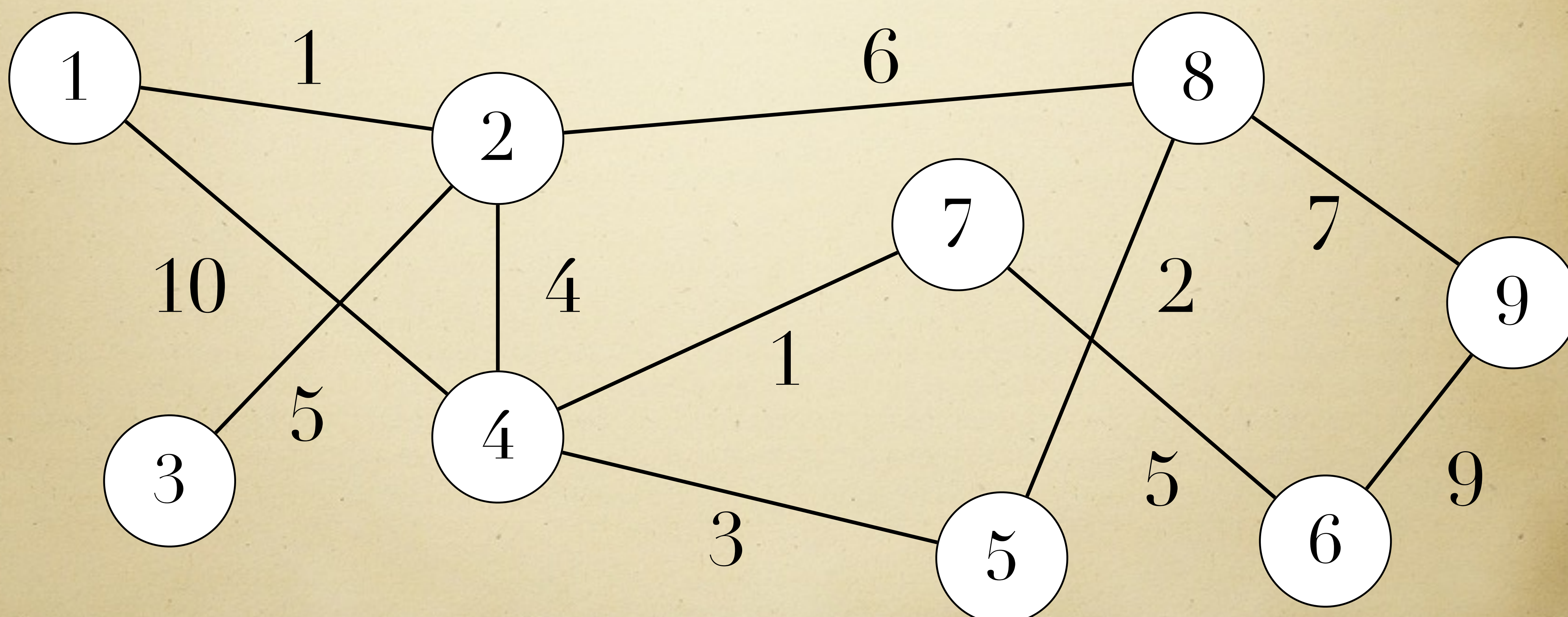
- ベルマンフォード法よりももっと高速に最短路を求めるアルゴリズム.
- 仮定としてすべての辺のコストを非負とする.

ダイクストラ法

- ベルマンフォード法よりももっと高速に最短路を求めるアルゴリズム.
- 仮定としてすべての辺のコストを非負とする.
- ベルマンフォード法では最小コストが決定していない頂点に対しても更新をしていた.
- このような更新は無駄である.

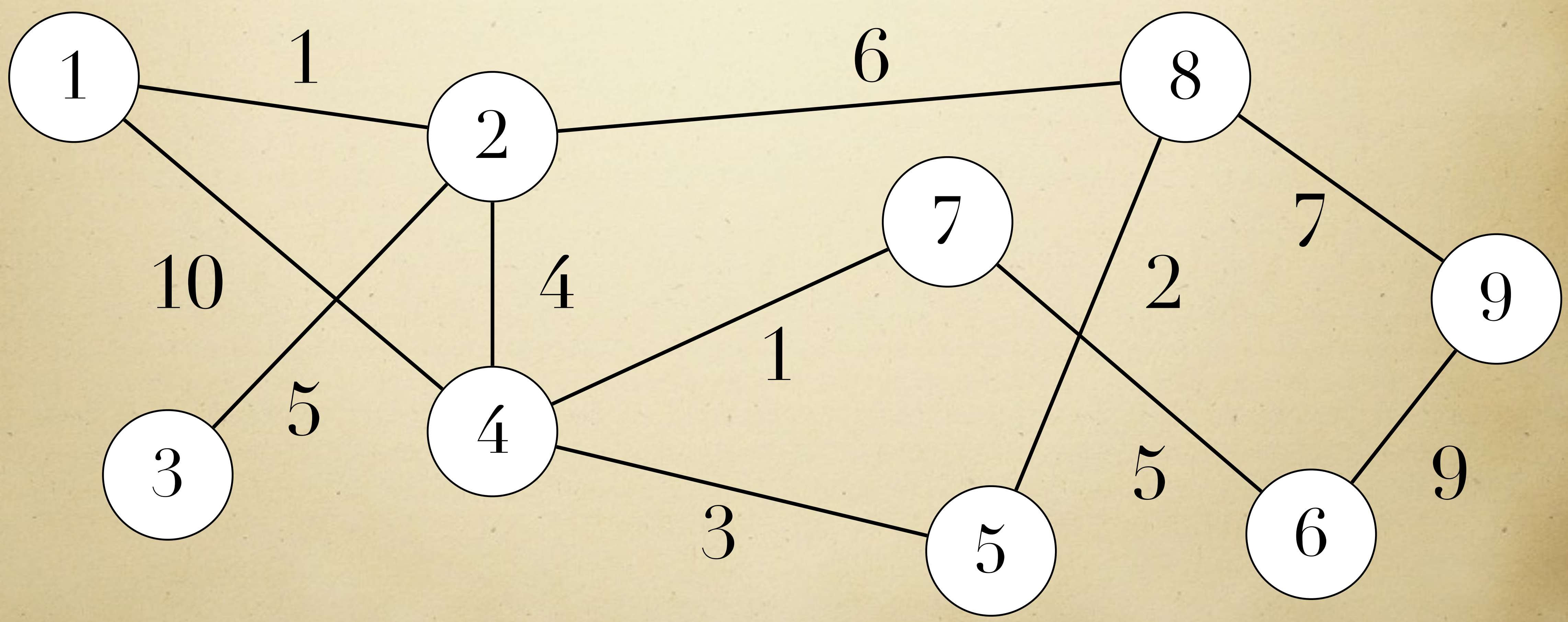
不要な更新

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



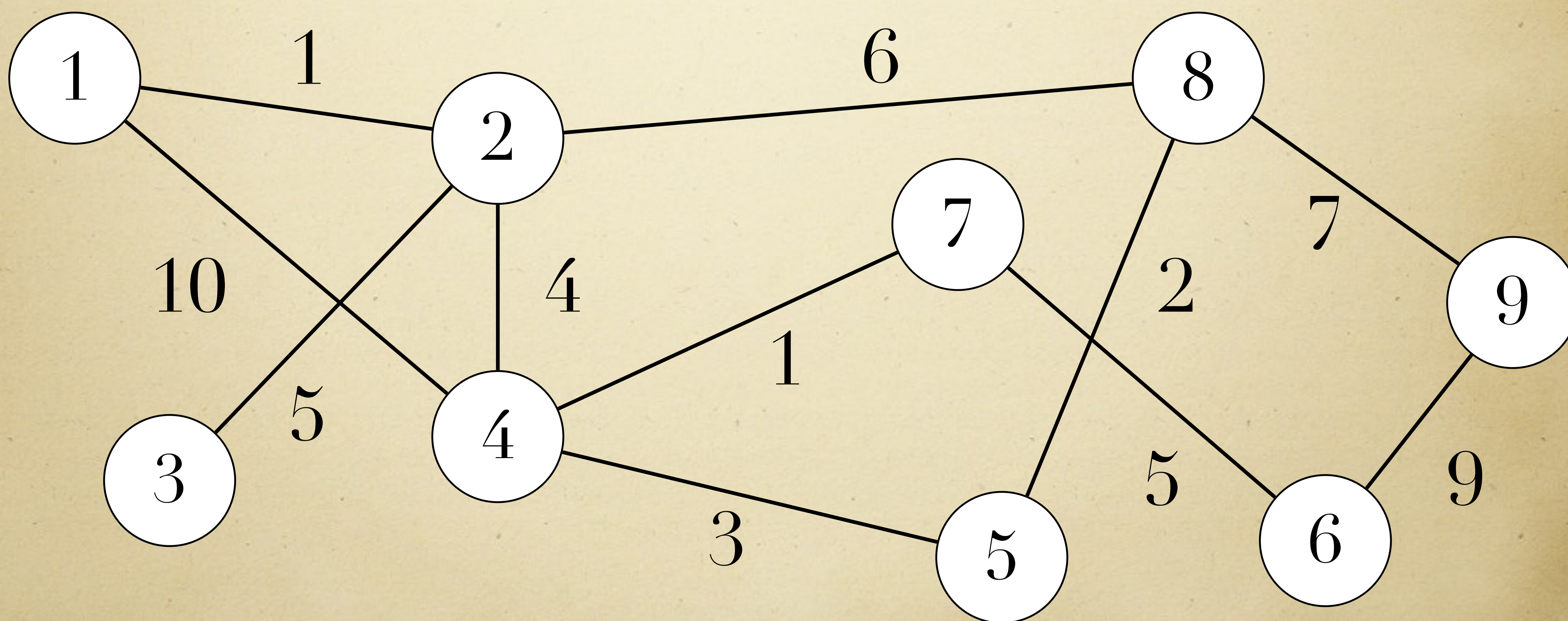
不要な更新

1	2	3	4	5	6	7	8	9
0	1	INF	10	INF	INF	INF	INF	INF



不要な更新

1	2	3	4	5	6	7	8	9
0	1	INF	10	INF	INF	INF	INF	INF

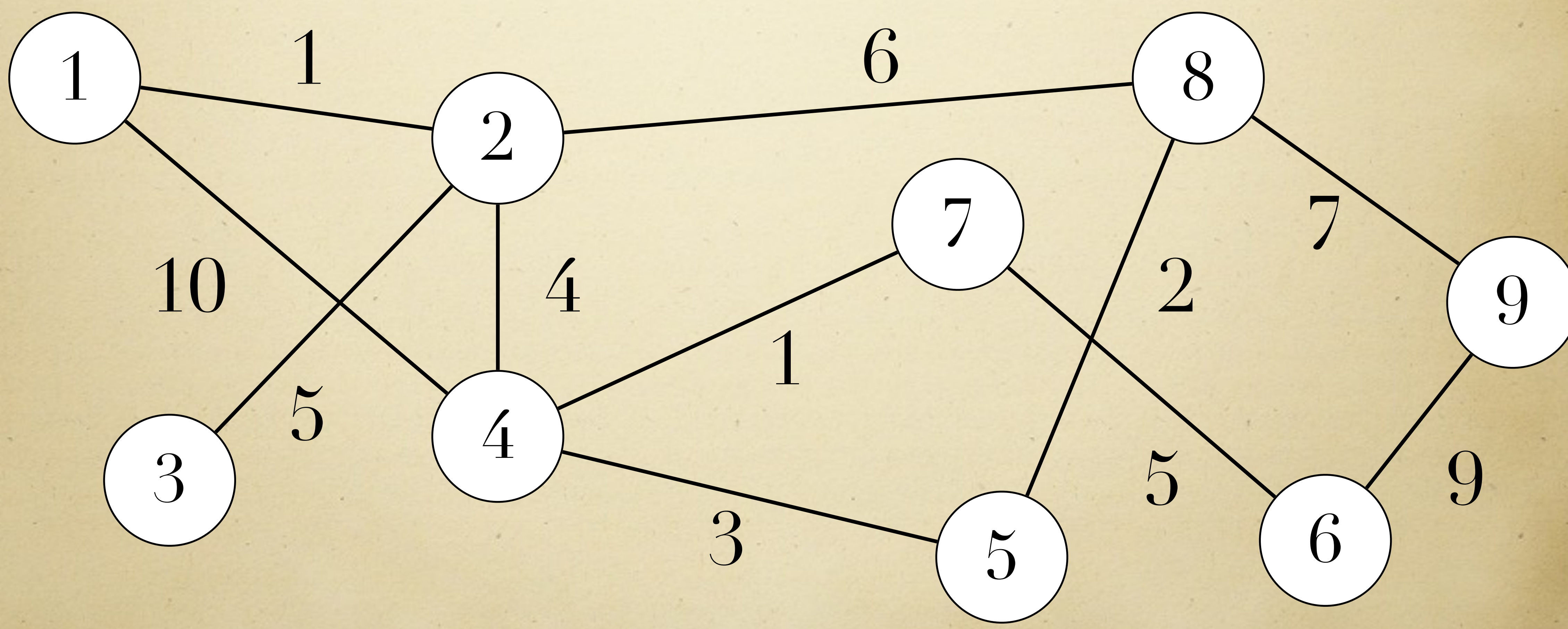


なぜ不要か？

- 背理法で証明する.
- 仮定として現在最小コストが決まっていない頂点の中でコスト最小の頂点がもっと小さいコストでその頂点に行けると別のパスがあると仮定する.
- そのようなパスは存在しないことが証明できる.

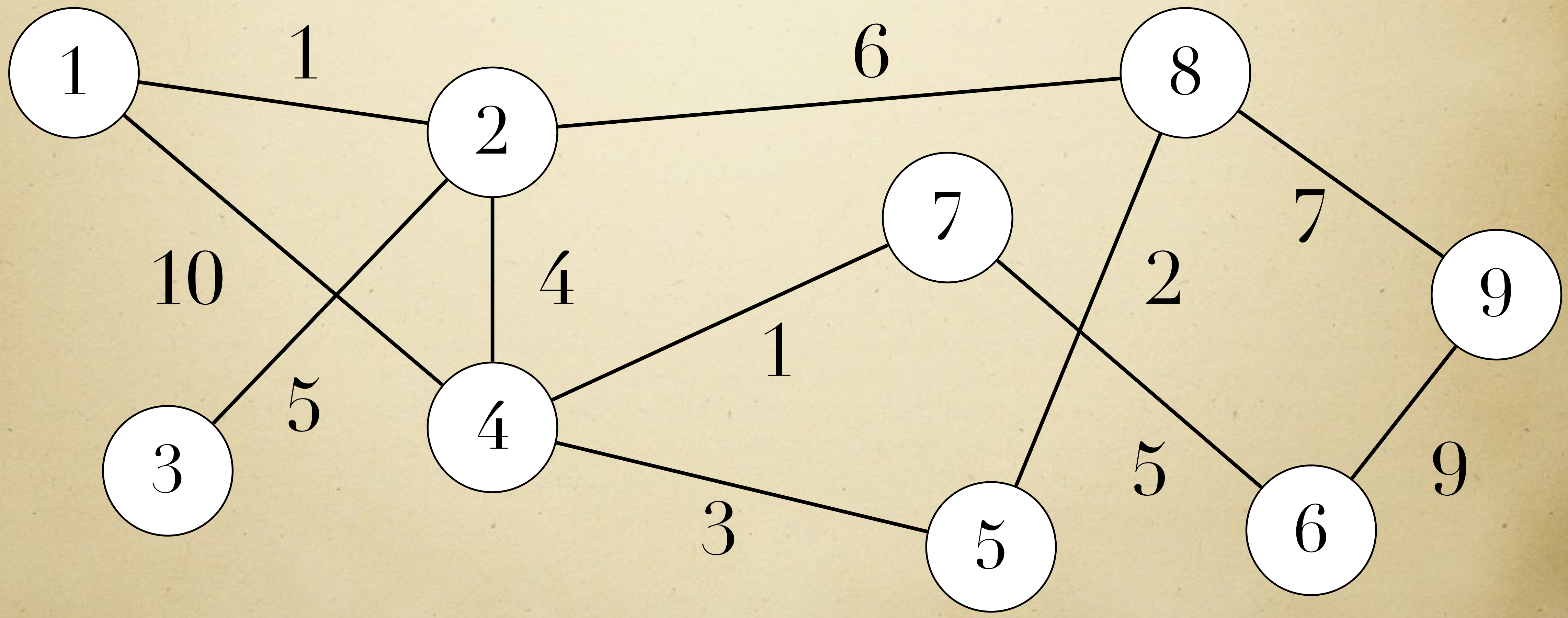
コストの更新

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



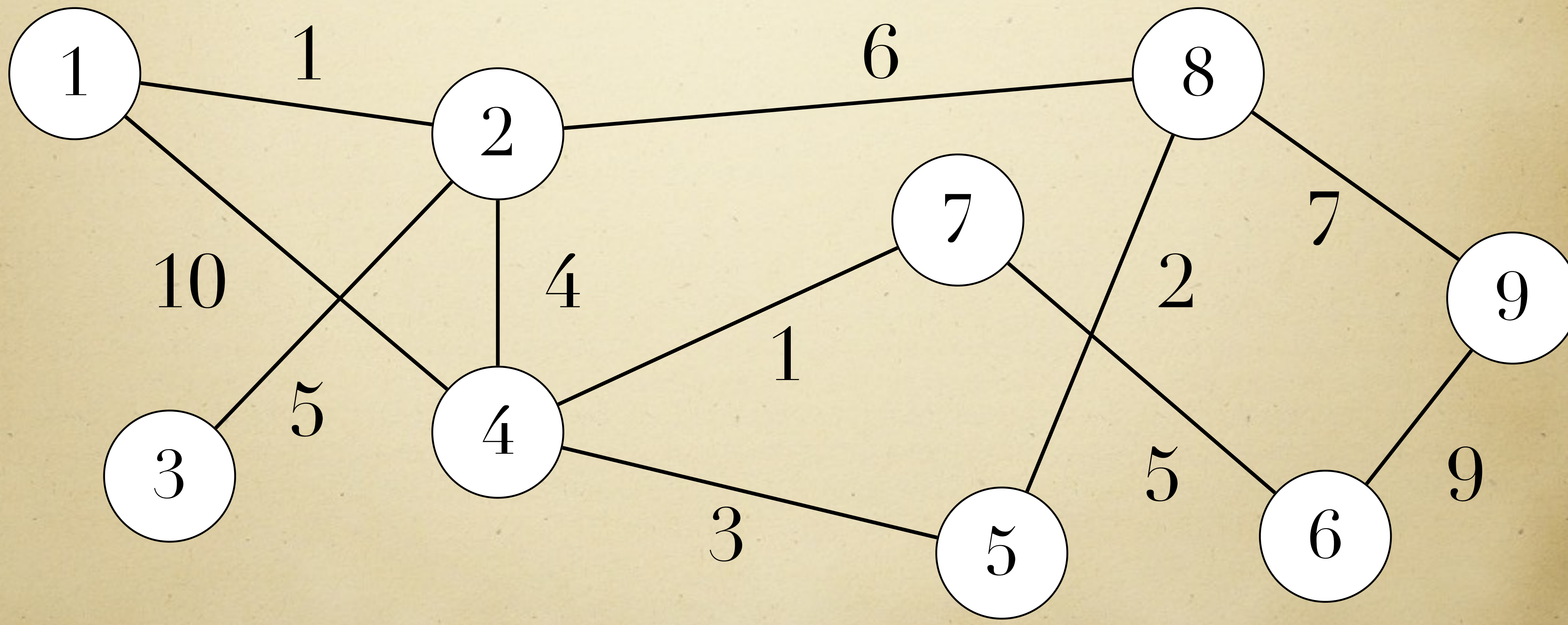
コストの更新

1	2	3	4	5	6	7	8	9
0	1	INF	10	INF	INF	INF	INF	INF



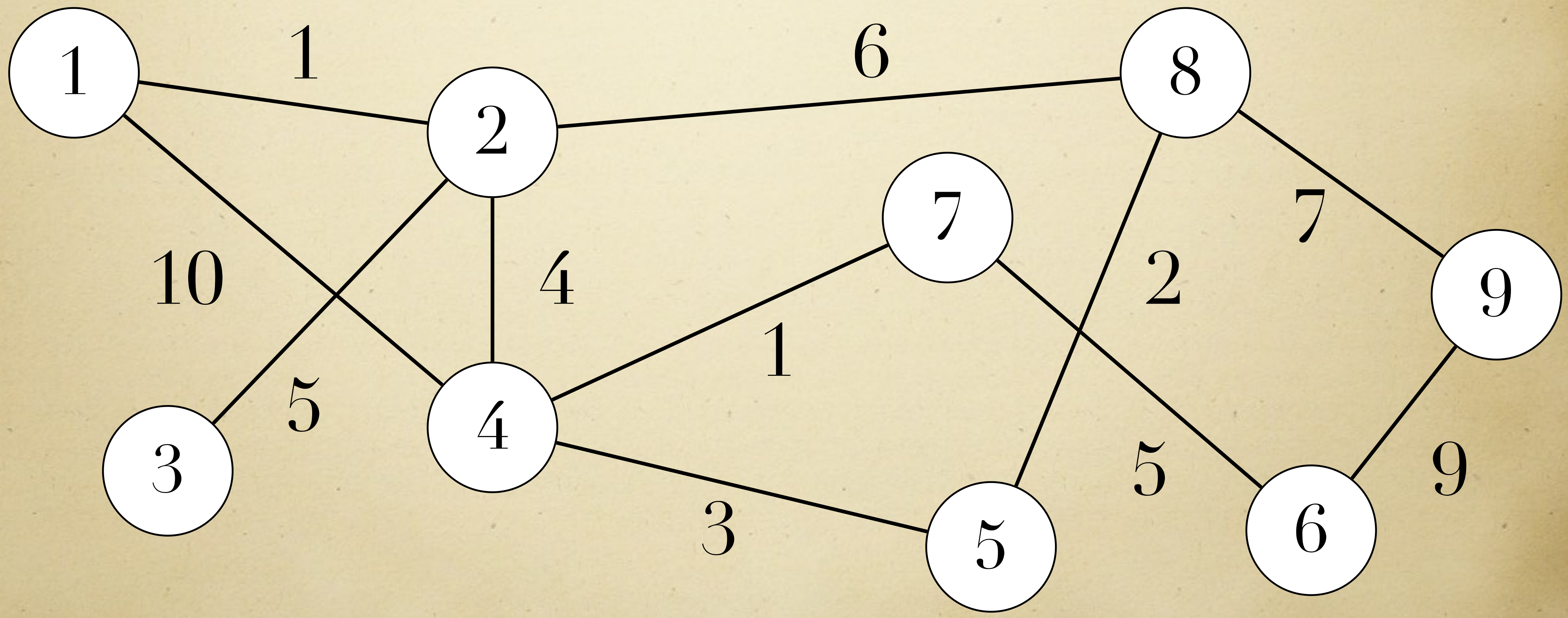
コストの更新

1	2	3	4	5	6	7	8	9
0	1	INF	10	INF	INF	INF	INF	INF



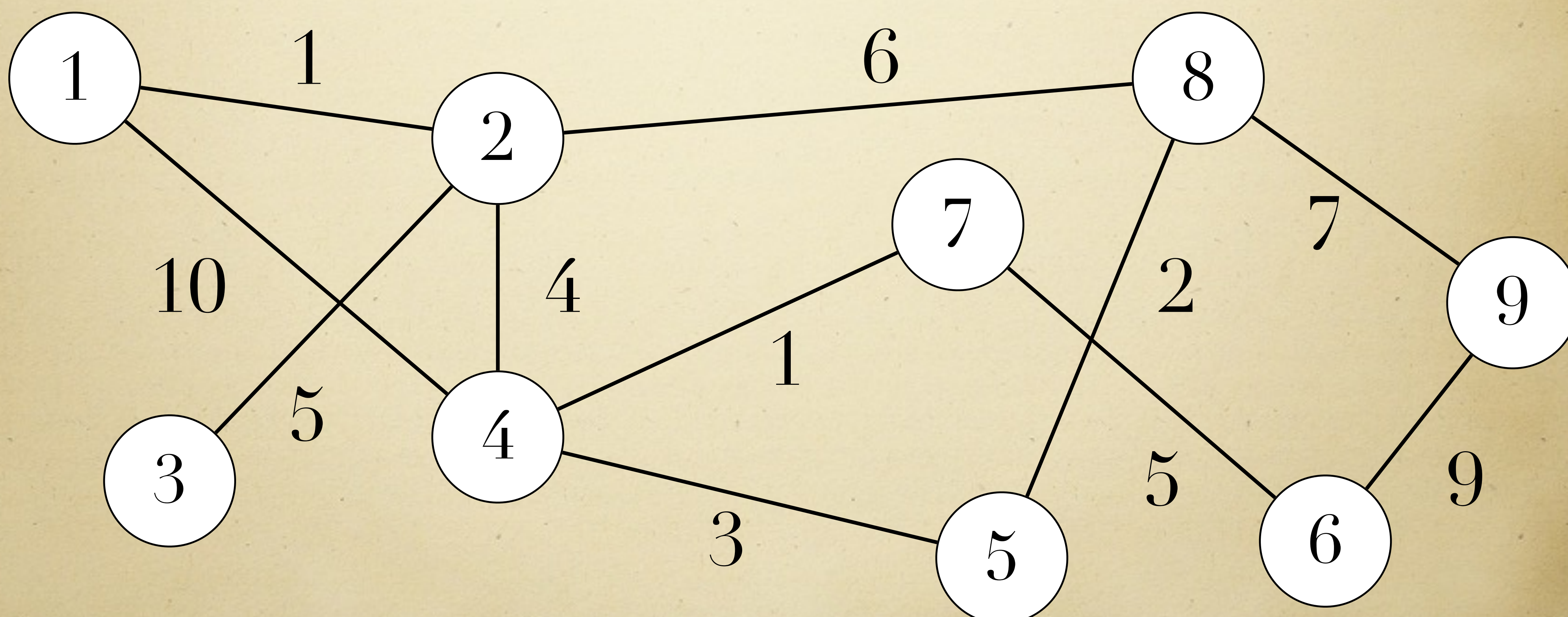
コストの更新

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



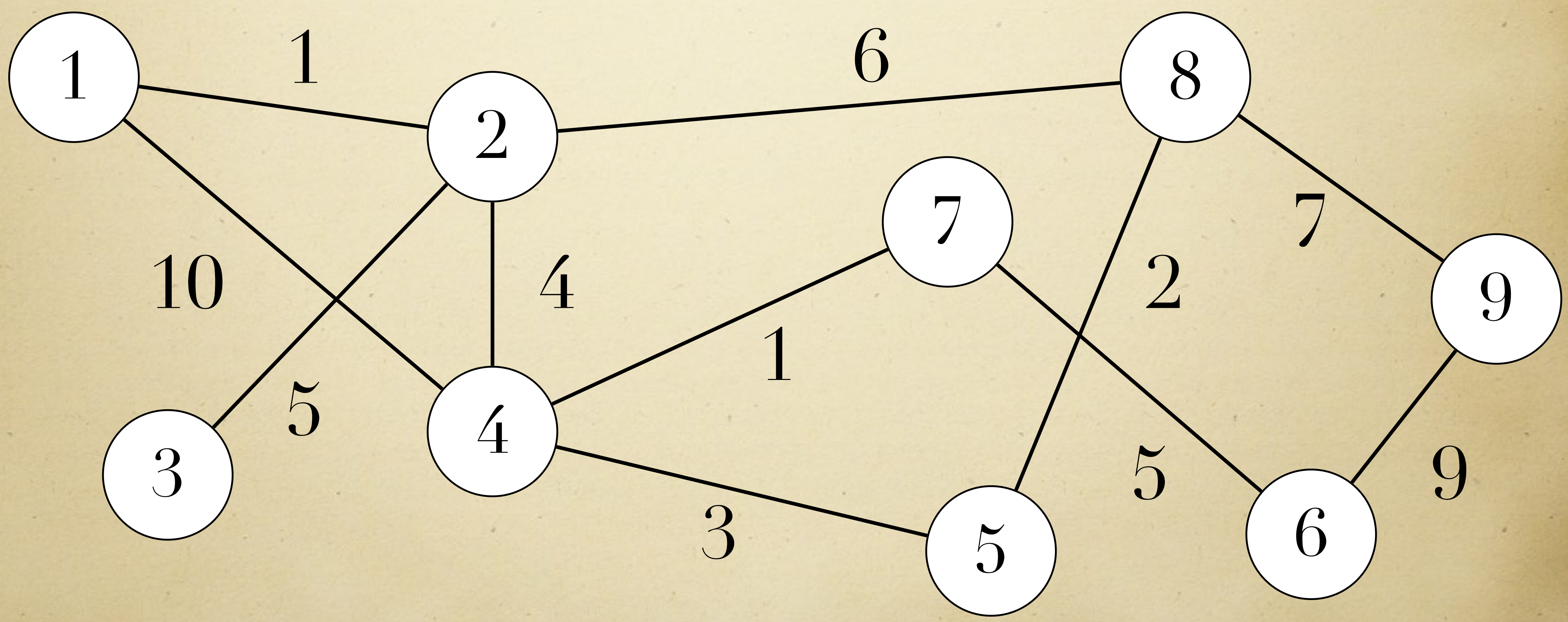
コストの更新

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



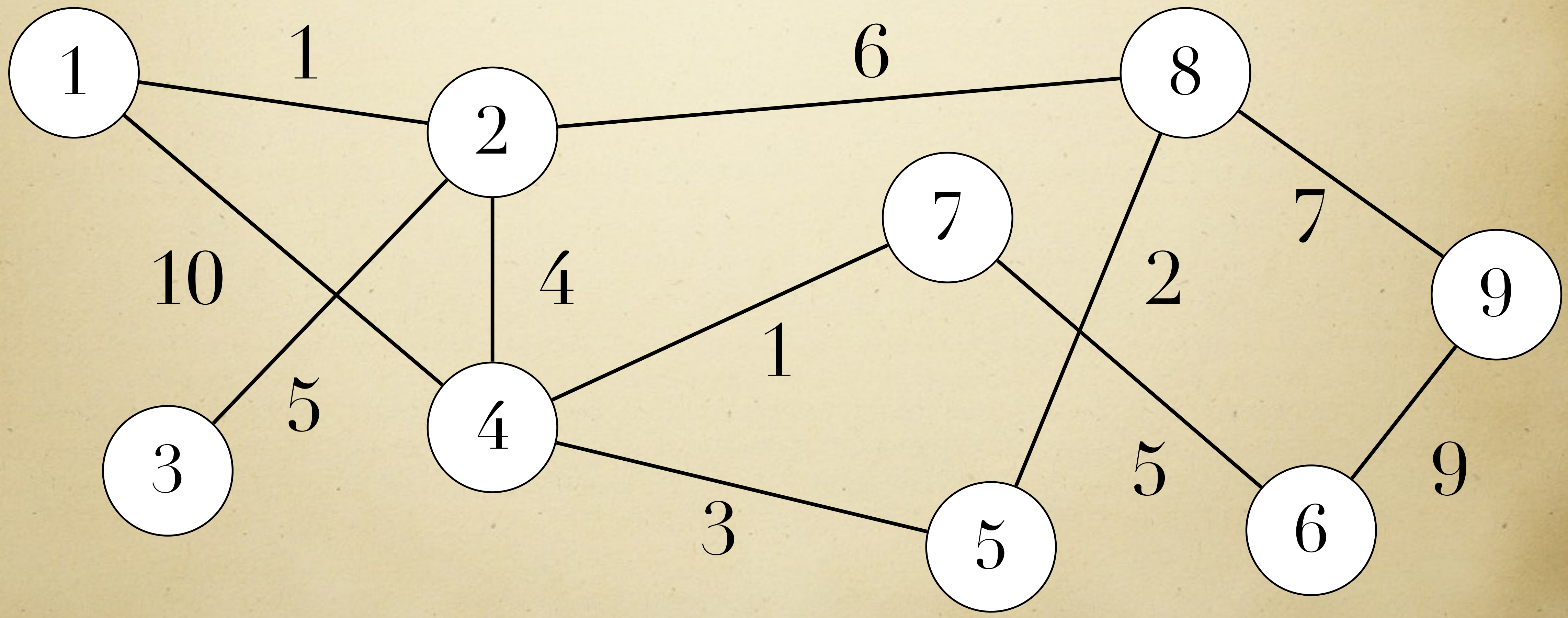
コストの更新

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



コストの更新

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



コードの例：
ダイクストラ法

- https://github.com/kazu0423/procon_example/blob/master/dijkstra.cpp

全点対最短路問題

- **全点対最短路問題**とは任意の2点間の最短路を求める問題である.

全点対最短路問題

- **全点対最短路問題**とは任意の2点間の最短路を求める問題である.
- **ワーシャルフロイド法**

ワーシャルフロイド法

- dp[i][j]を頂点iから頂点jまでの最小コストを記憶する変数とする.
- dpの初期状態は隣接行列になっており，隣接していない頂点同士のコストはINFとする.
- dp[i][i] = 0とする. （自分から自分はコスト0）

ワーシャルフロイド法

```
for(int k = 0; k< n; k++)
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            dp[i][j] = min(dp[i][j], dp[i][k] + dp[k][j]);
```

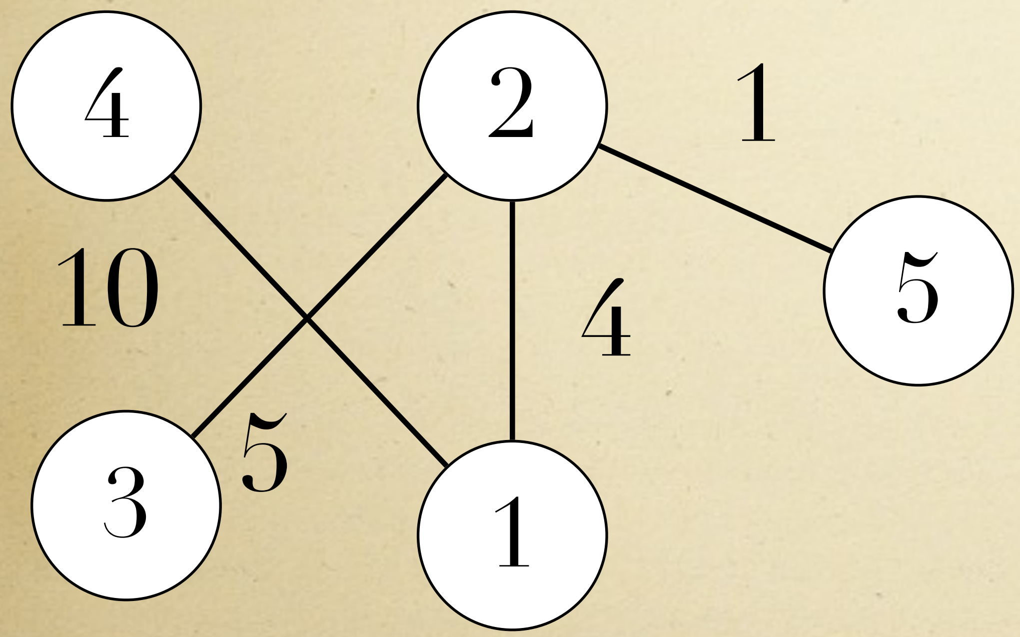
kのループ

- kのループを以外の部分では，大体のイメージとして任意の2点間の距離について何かの更新をしていることがわかる．

kのループ

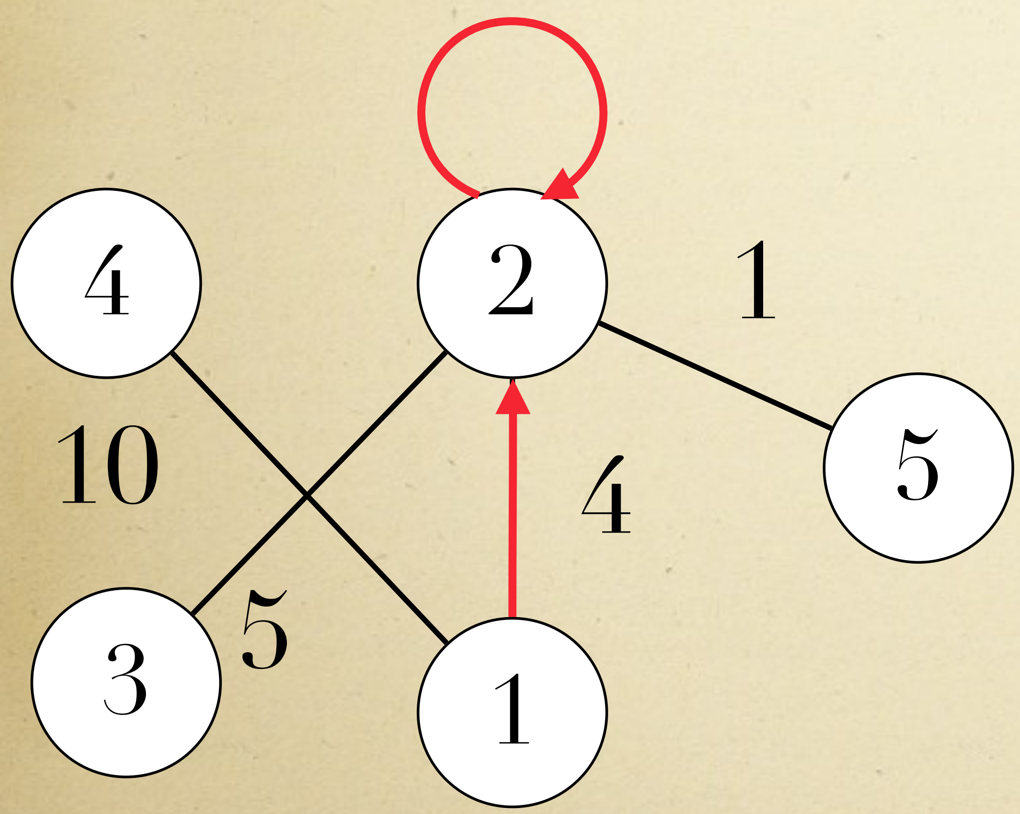
- kのループを以外の部分では，大体のイメージとして任意の2点間の距離について何かの更新をしていることがわかる．
- kは途中で立ち寄る頂点がどこなのかを決めている．

コストの更新



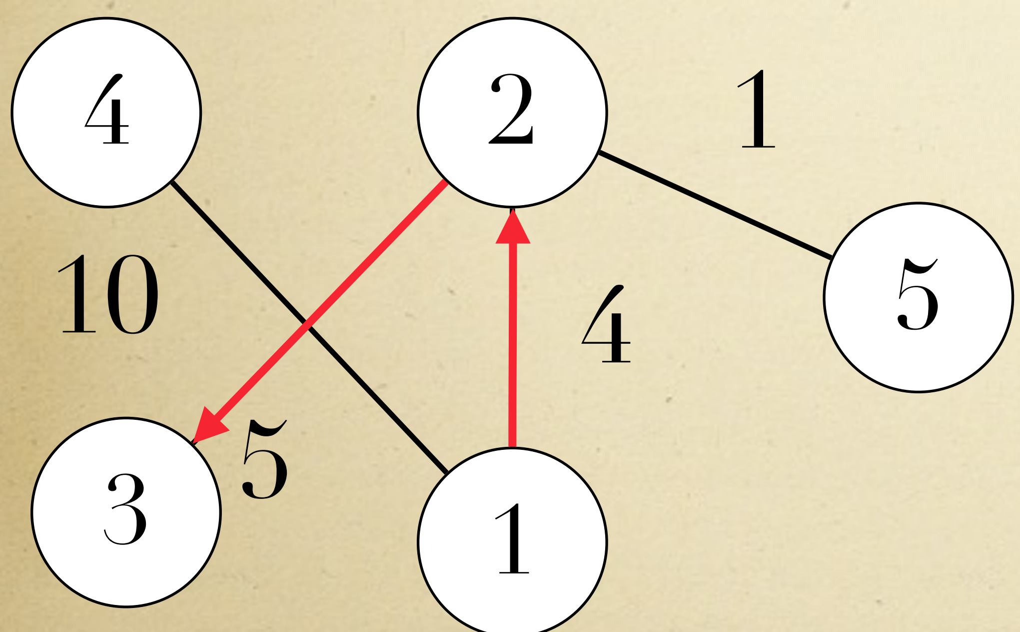
	1	2	3	4	5
1	0	4	INF	10	INF
2	4	0	5	INF	1
3	INF	5	0	INF	INF
4	10	INF	INF	0	INF
5	INF	1	INF	INF	0

コストの更新：k = 2



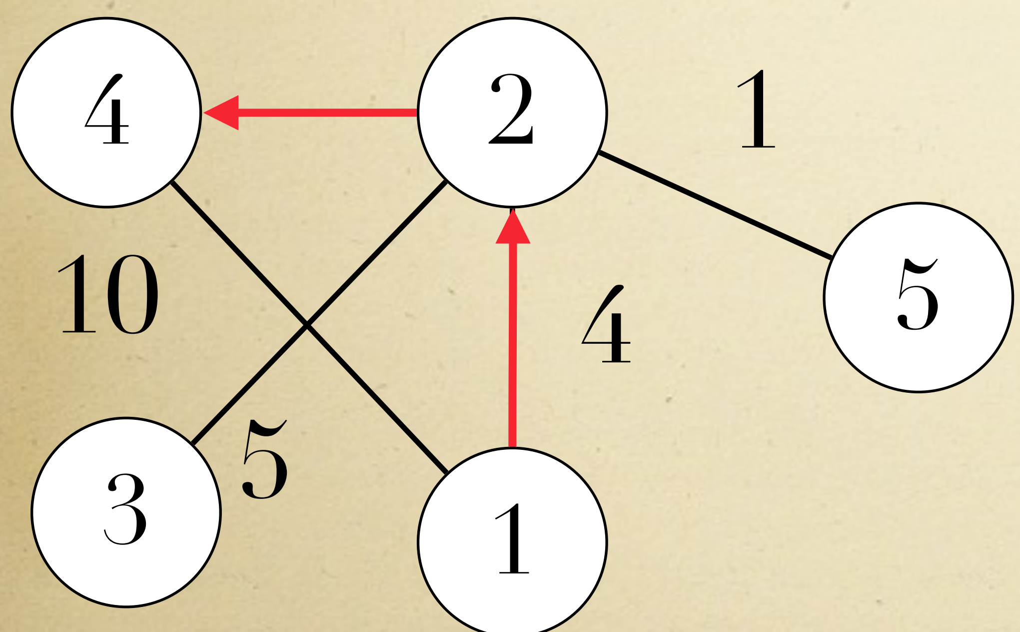
	1	2	3	4	5
1	0	4	INF	10	INF
2	4	0	5	INF	1
3	INF	5	0	INF	INF
4	10	INF	INF	0	INF
5	INF	1	INF	INF	0

コストの更新：k = 2



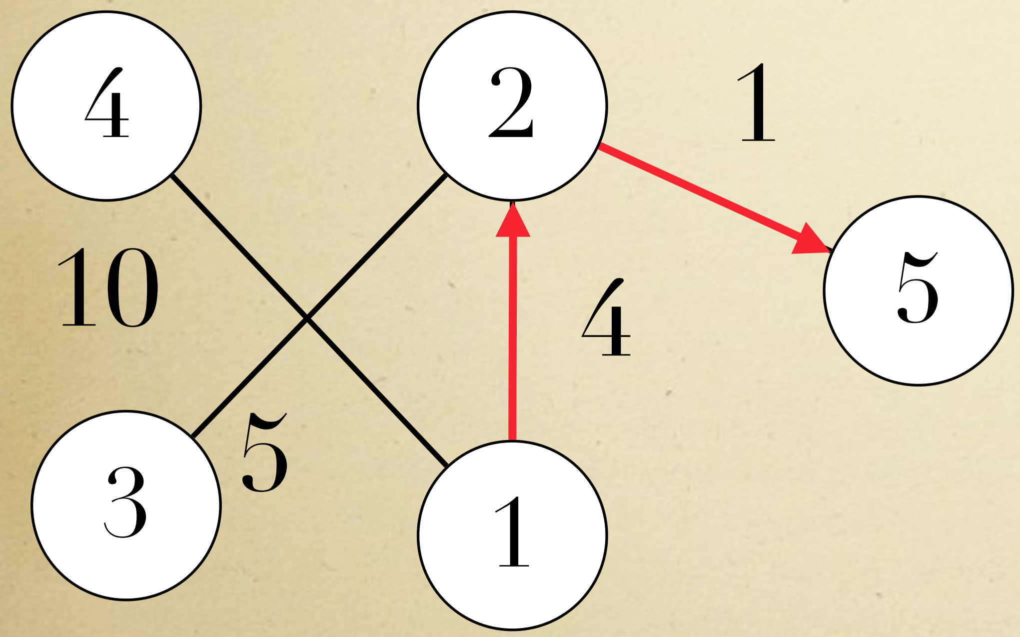
	1	2	3	4	5
1	0	4	9	10	INF
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	INF	1	INF	INF	0

コストの更新：k = 2



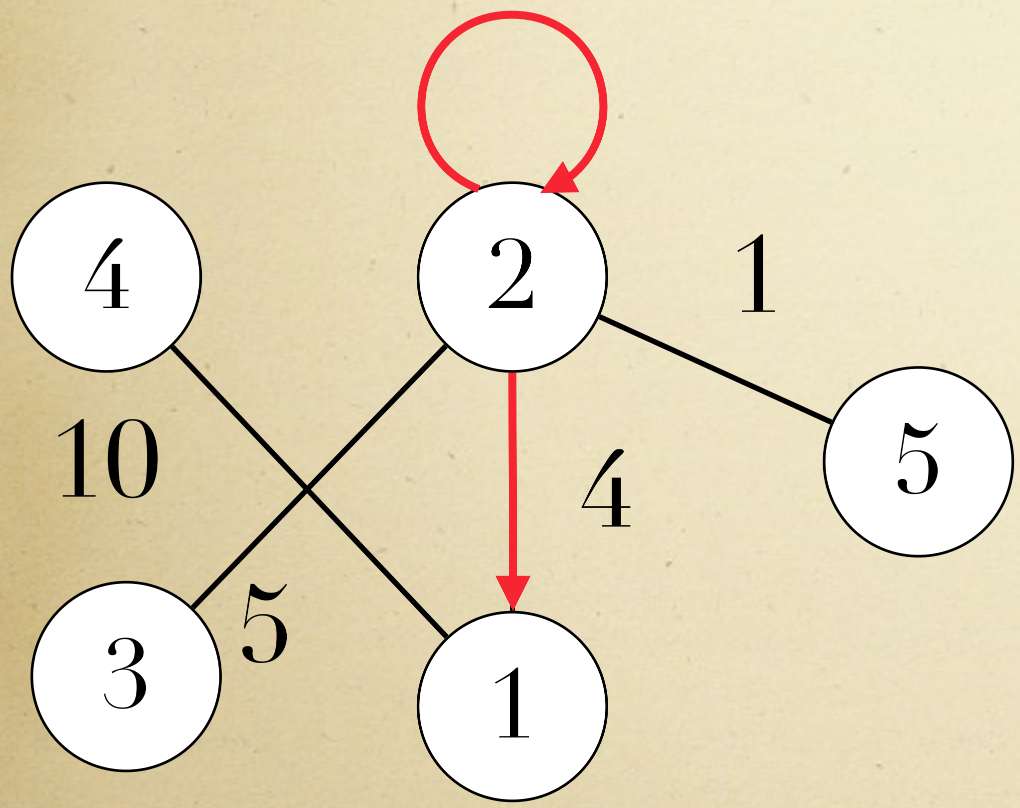
	1	2	3	4	5
1	0	4	9	10	INF
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	INF	1	INF	INF	0

コストの更新：k = 2



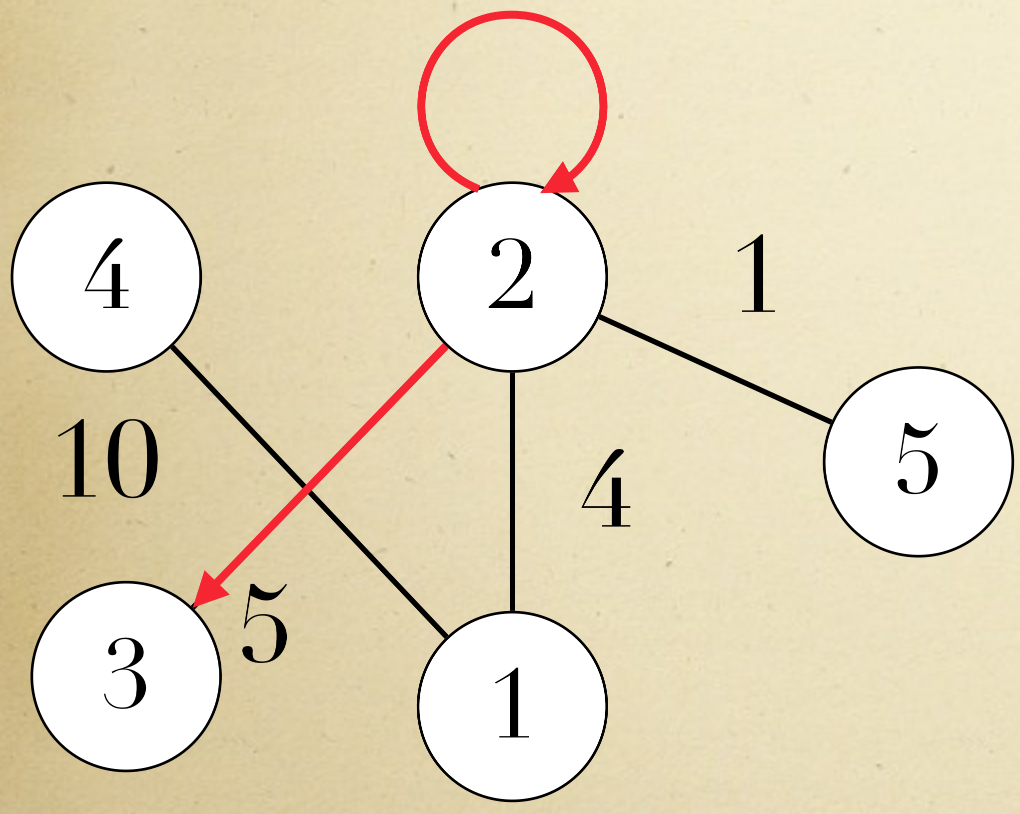
	1	2	3	4	5
1	0	4	9	10	5
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	5	1	INF	INF	0

コストの更新：k = 2



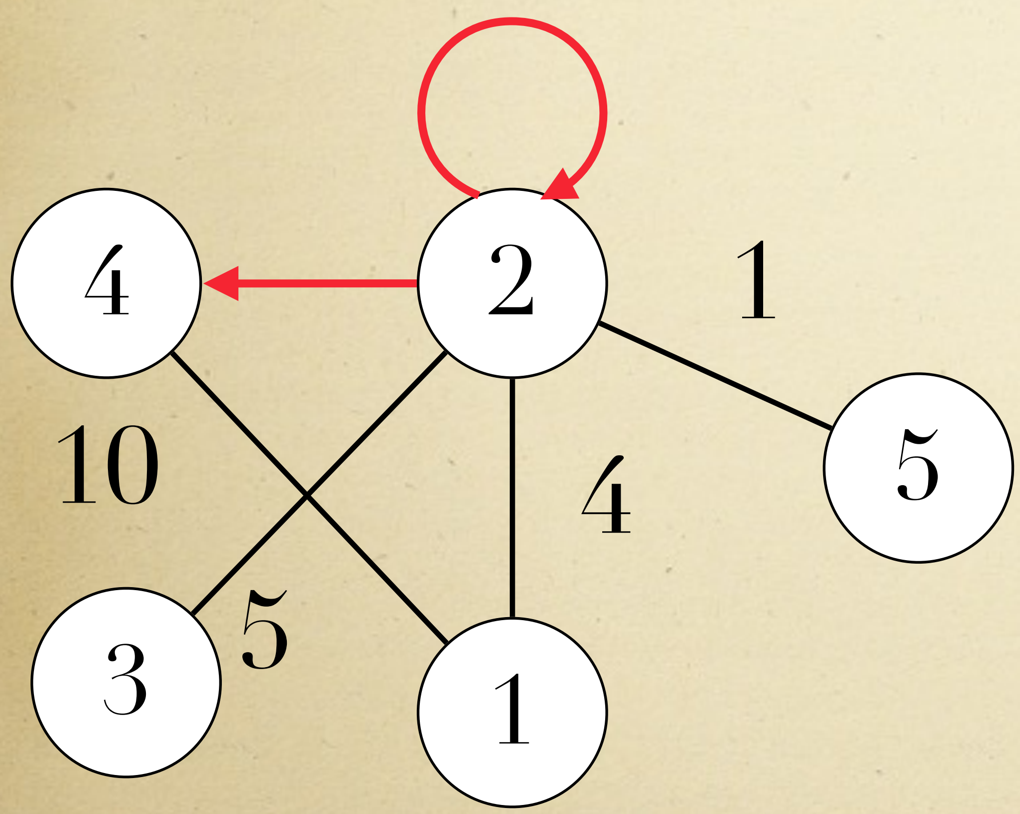
	1	2	3	4	5
1	0	4	9	10	5
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	5	1	INF	INF	0

コストの更新：k = 2



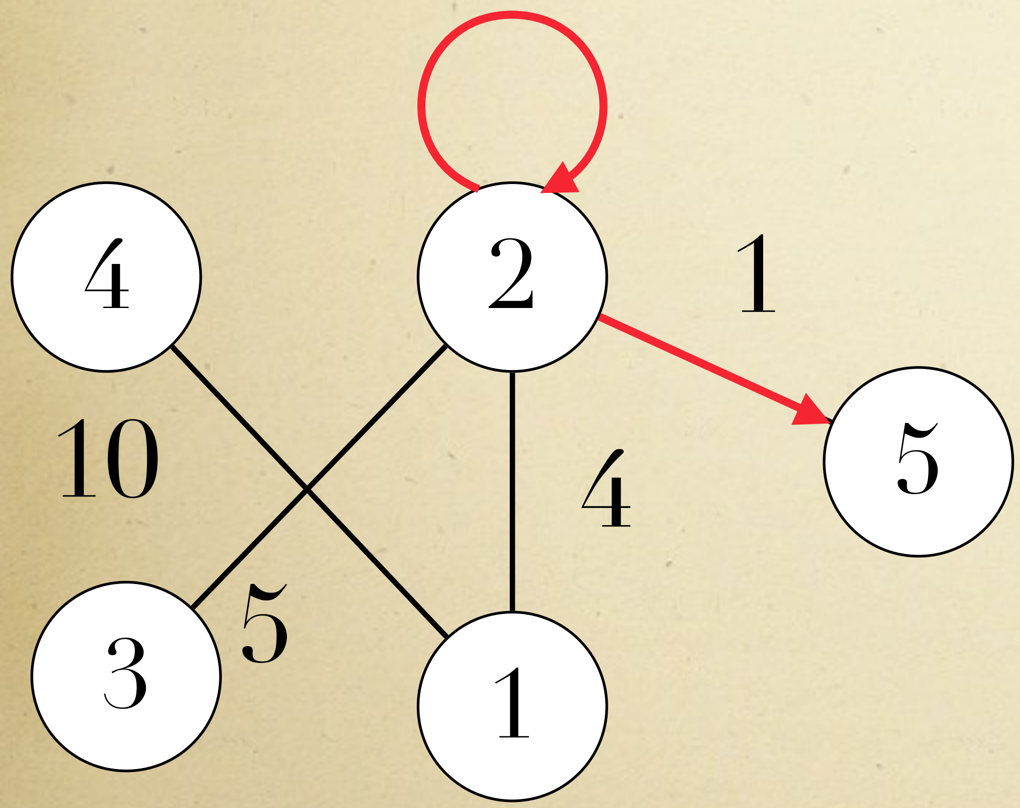
	1	2	3	4	5
1	0	4	9	10	5
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	5	1	INF	INF	0

コストの更新：k = 2



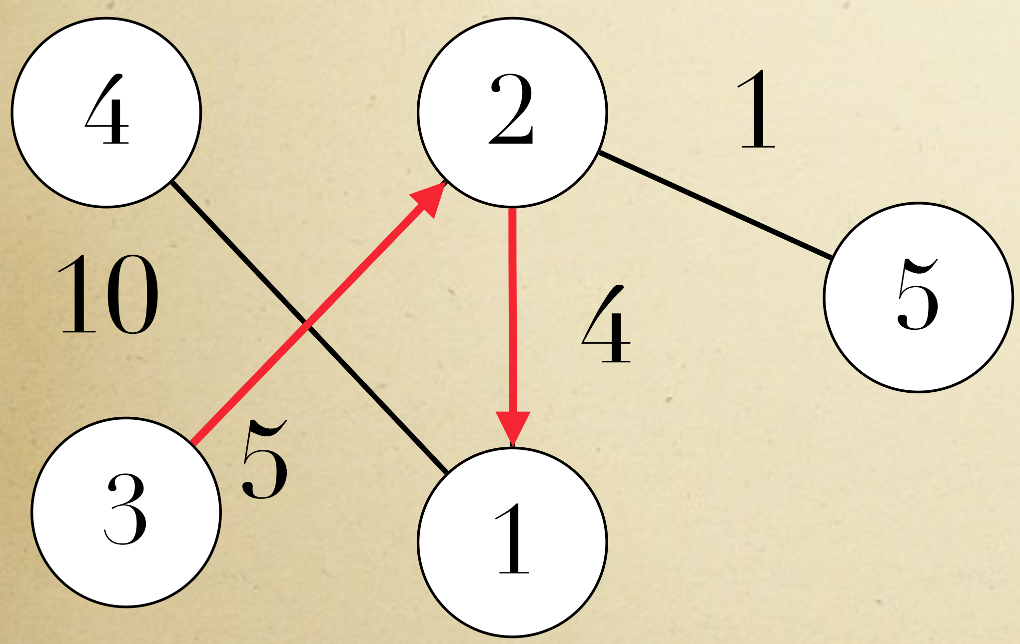
	1	2	3	4	5
1	0	4	9	10	5
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	5	1	INF	INF	0

コストの更新：k = 2



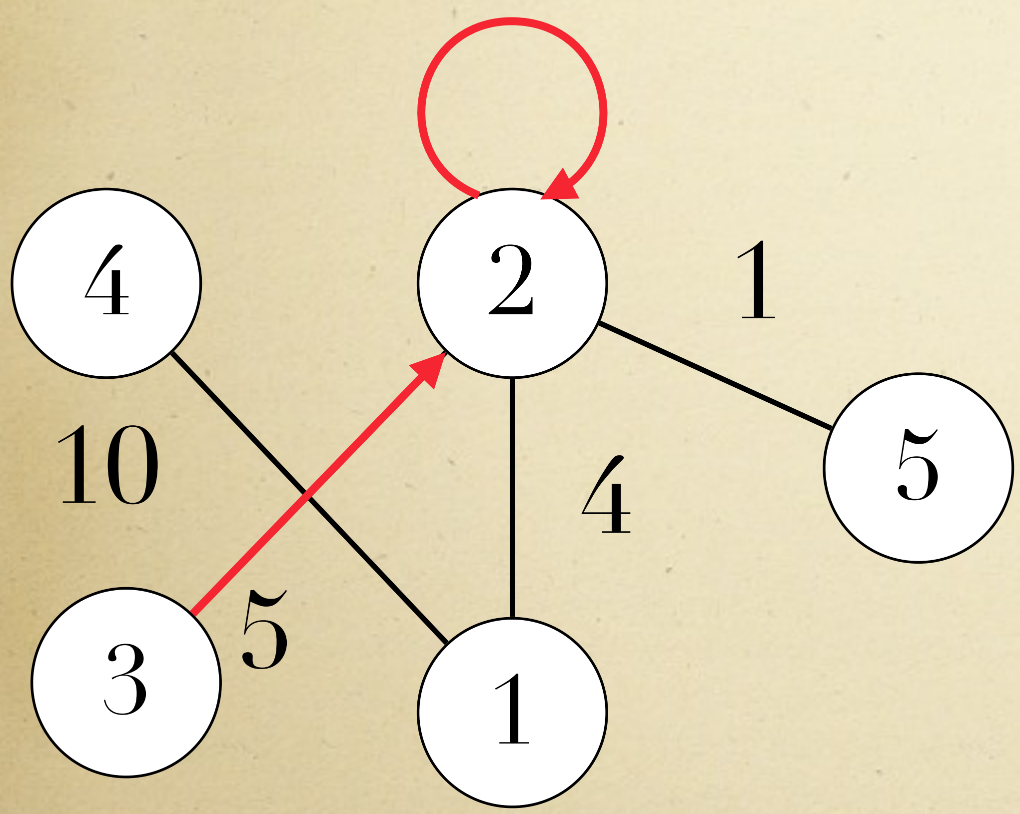
	1	2	3	4	5
1	0	4	9	10	5
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	5	1	INF	INF	0

コストの更新：k = 2



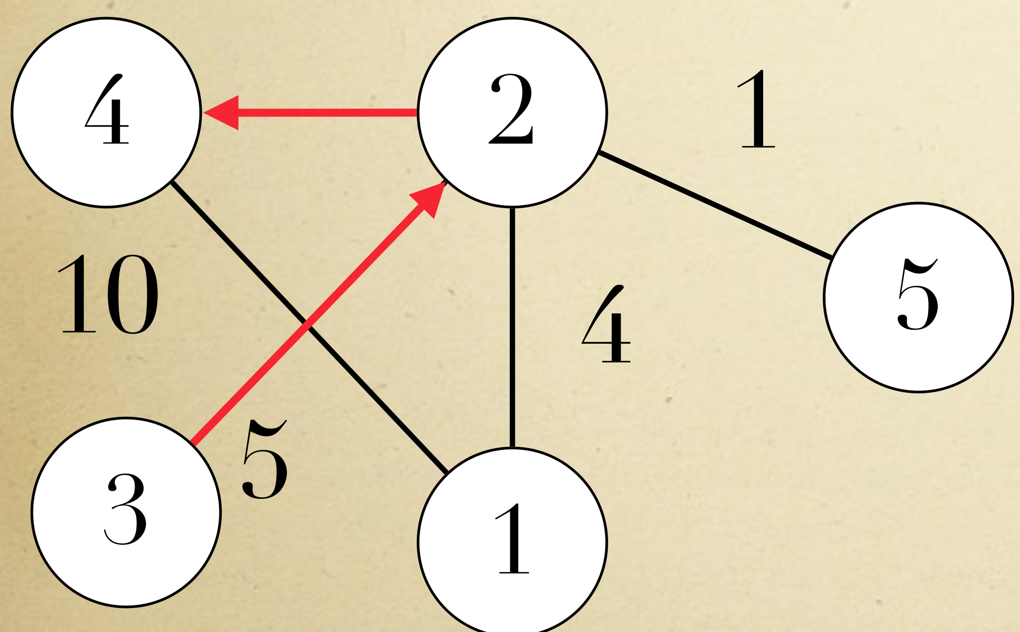
	1	2	3	4	5
1	0	4	9	10	5
2	4	0	5	INF	1
3	9	5	0	INF	INF
4	10	INF	INF	0	INF
5	5	1	INF	INF	0

コストの更新：k = 2



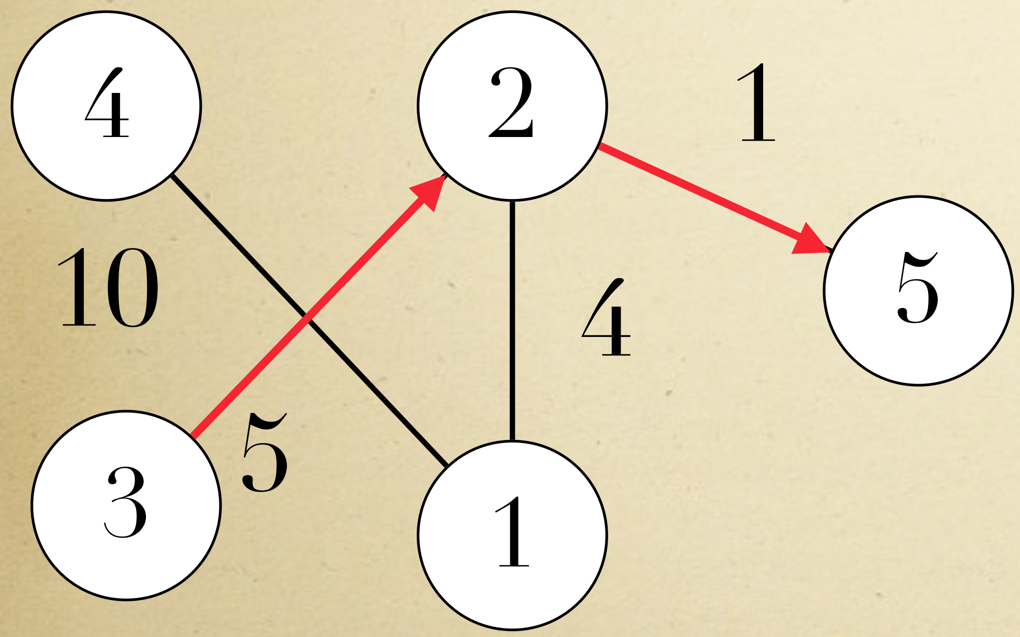
	1	2	3	4	5
1	0	4	INF	10	INF
2	4	0	5	INF	1
3	INF	5	0	INF	INF
4	10	INF	INF	0	INF
5	INF	1	INF	INF	0

コストの更新：k = 2



	1	2	3	4	5
1	0	4	INF	10	INF
2	4	0	5	INF	1
3	INF	5	0	INF	INF
4	10	INF	INF	0	INF
5	INF	1	INF	INF	0

コストの更新：k = 2



	1	2	3	4	5
1	0	4	INF	10	INF
2	4	0	5	INF	1
3	INF	5	0	INF	6
4	10	INF	INF	0	INF
5	INF	1	6	INF	0

コードの例：
ワーシャルフロイド法

- https://github.com/kazu0423/procon_example/blob/master/warshall-floyed.cpp

計算量

- ほぼ自明
- 時間計算量：$O(V^3)$
- 空間計算量：$O(V^2)$

まとめ

- グラフの表現
- 単一始点最短路問題
  - ベルマンフォード法
  - ダイクストラ法
- 全点対最短路問題
  - ワーシャルフロイド法