Dijkstra Method

2012 / 4 / 23

目的

グラフにおいて

あるノードから他のノードへの
最短経路を求める

例

- 東京から大阪へ最速で
  - ノード：駅
  - コスト：時間

前提

- 負のコストを持った経路がないこと

データ構造

- 確定ノード
  - 最短経路がわかったノード
- 候補ノード
  - 確定ノードから１手で行けるノード

アルゴリズム

1)候補ノードの中から、経路が最小のものを確定ノードとする

- そこへのより良いルートは存在しない

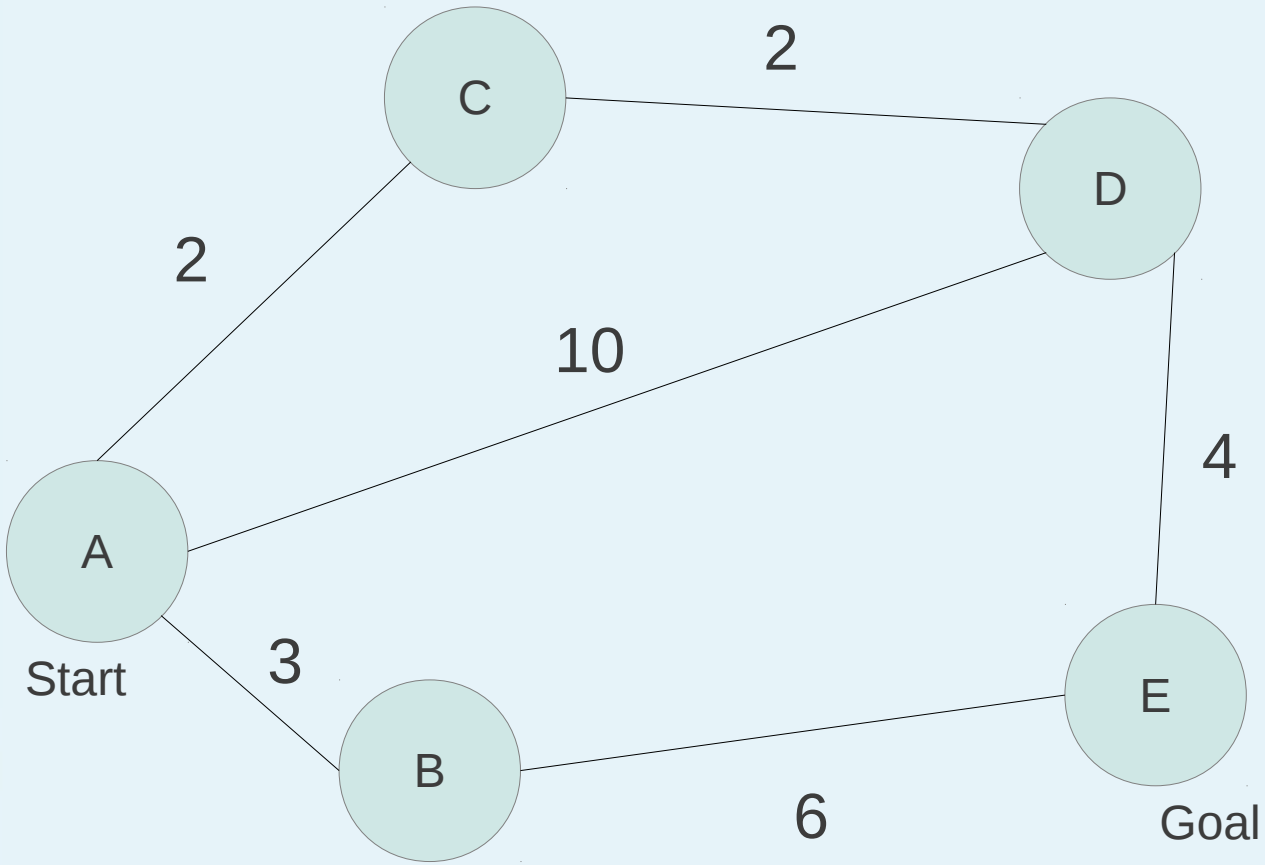
  (∵負の経路が存在しない)

2)新たな確定ノードから１手で行けるノードを、候補ノードとして登録する

- 既により良い経路が見つかっていたら

  スキップ

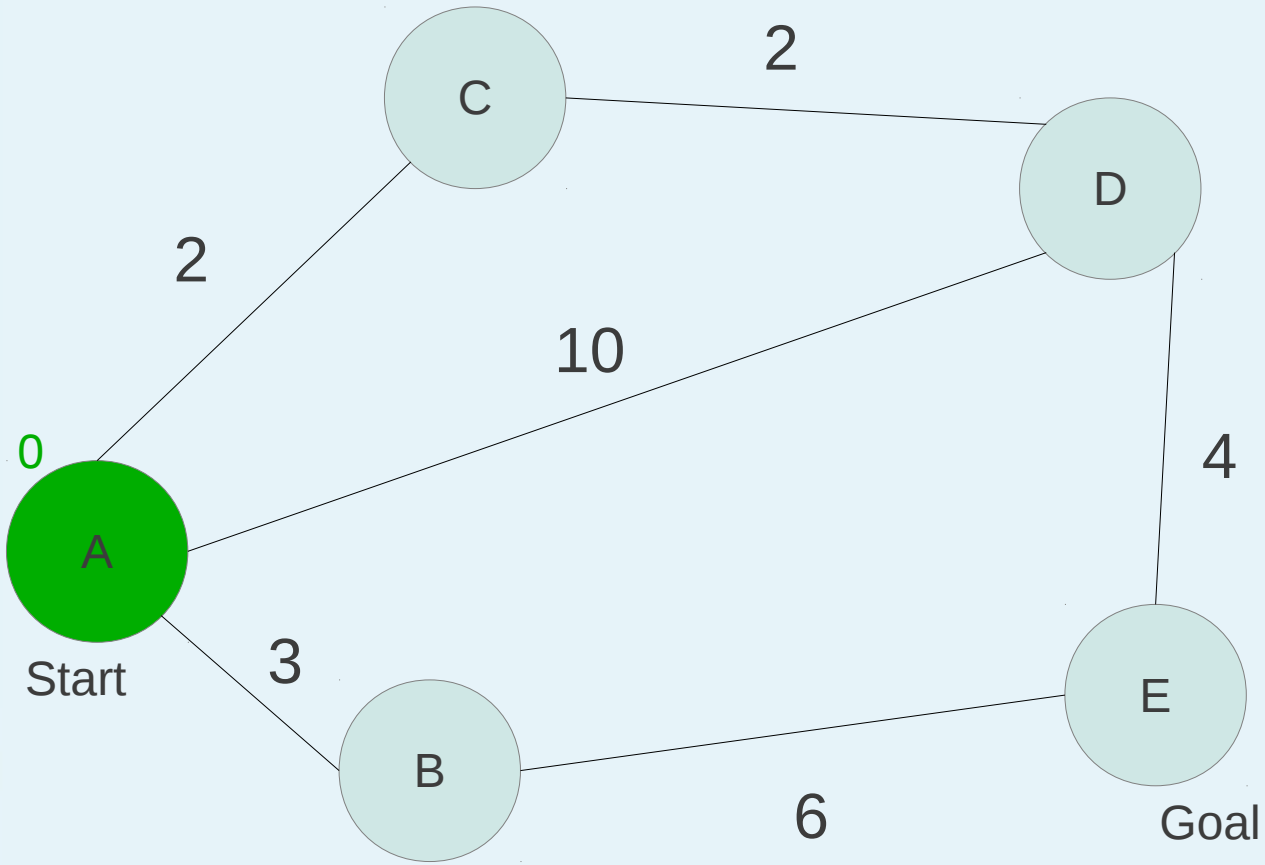
アルゴリズム



候補ノード

Dist	Node

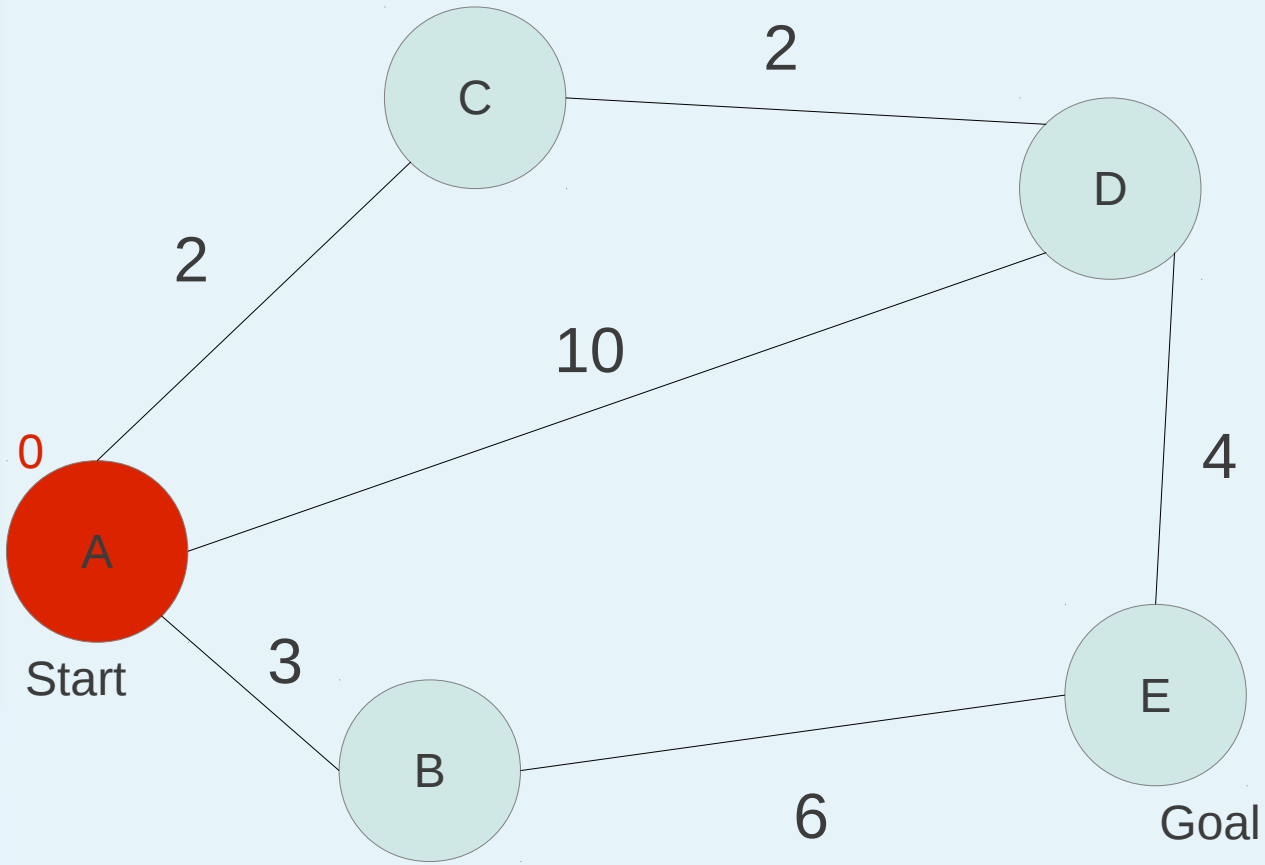
アルゴリズム



候補ノード

Dist	Node
0	A

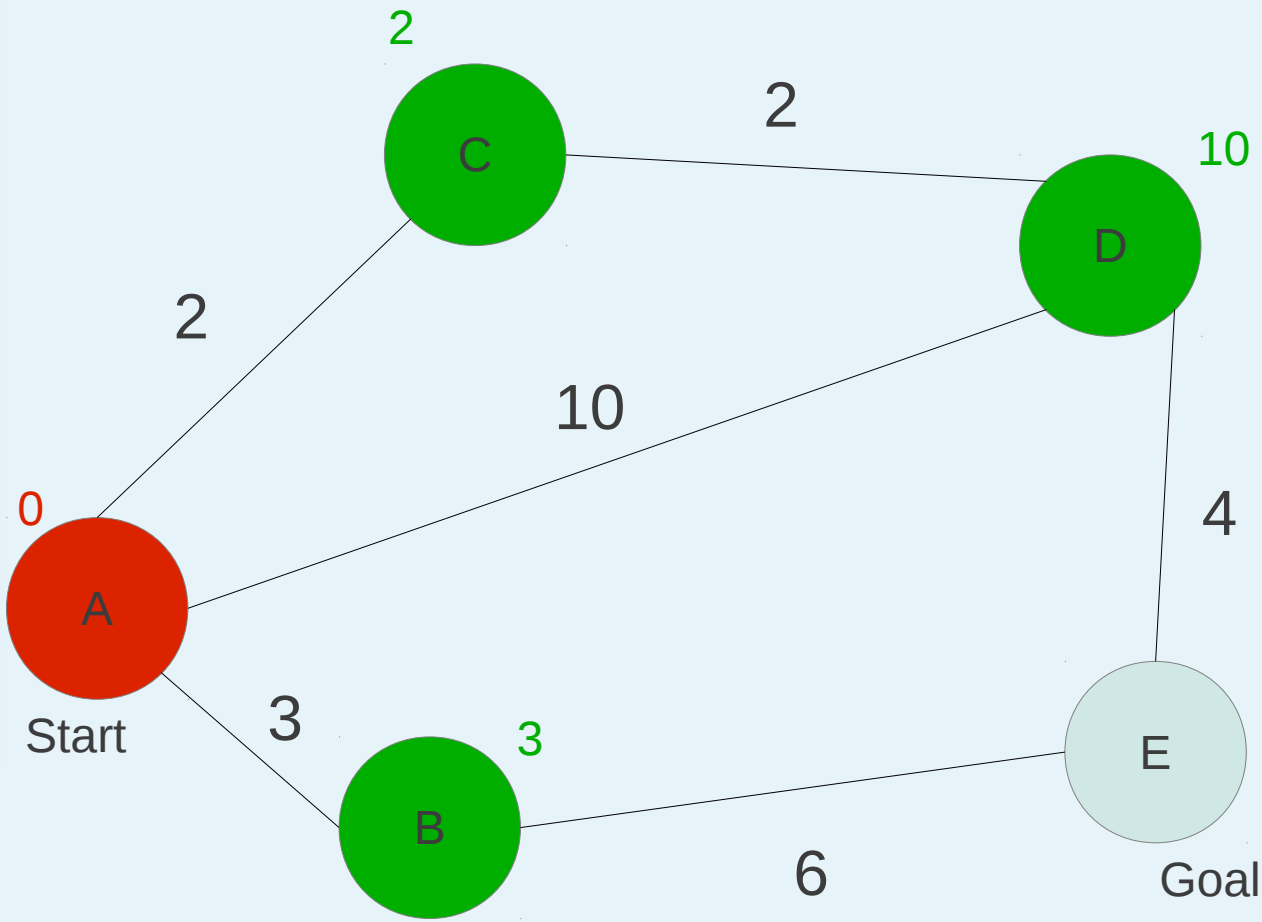
アルゴリズム



候補ノード

Dist	Node

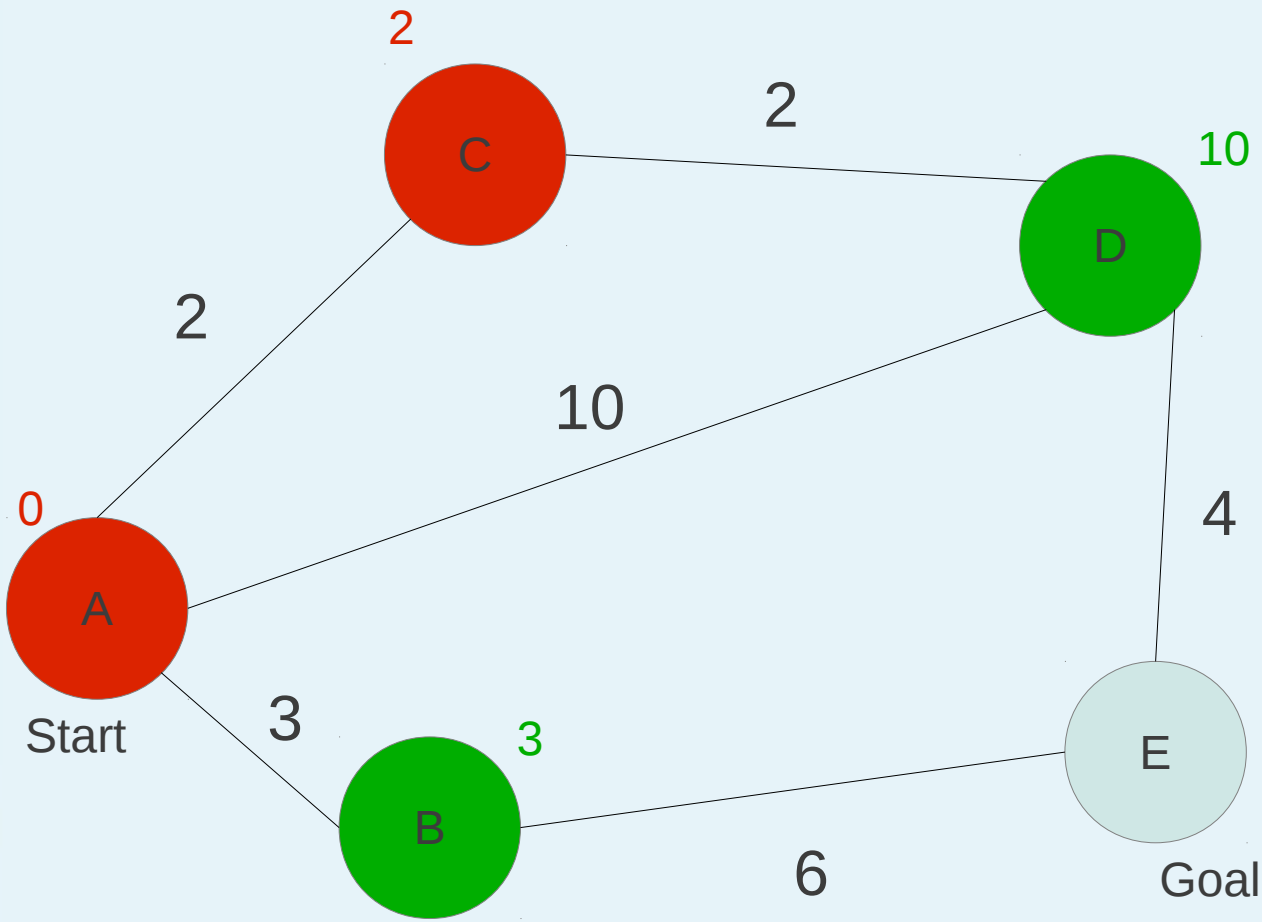
アルゴリズム



候補ノード

Dist	Node
2	C
3	B
10	D

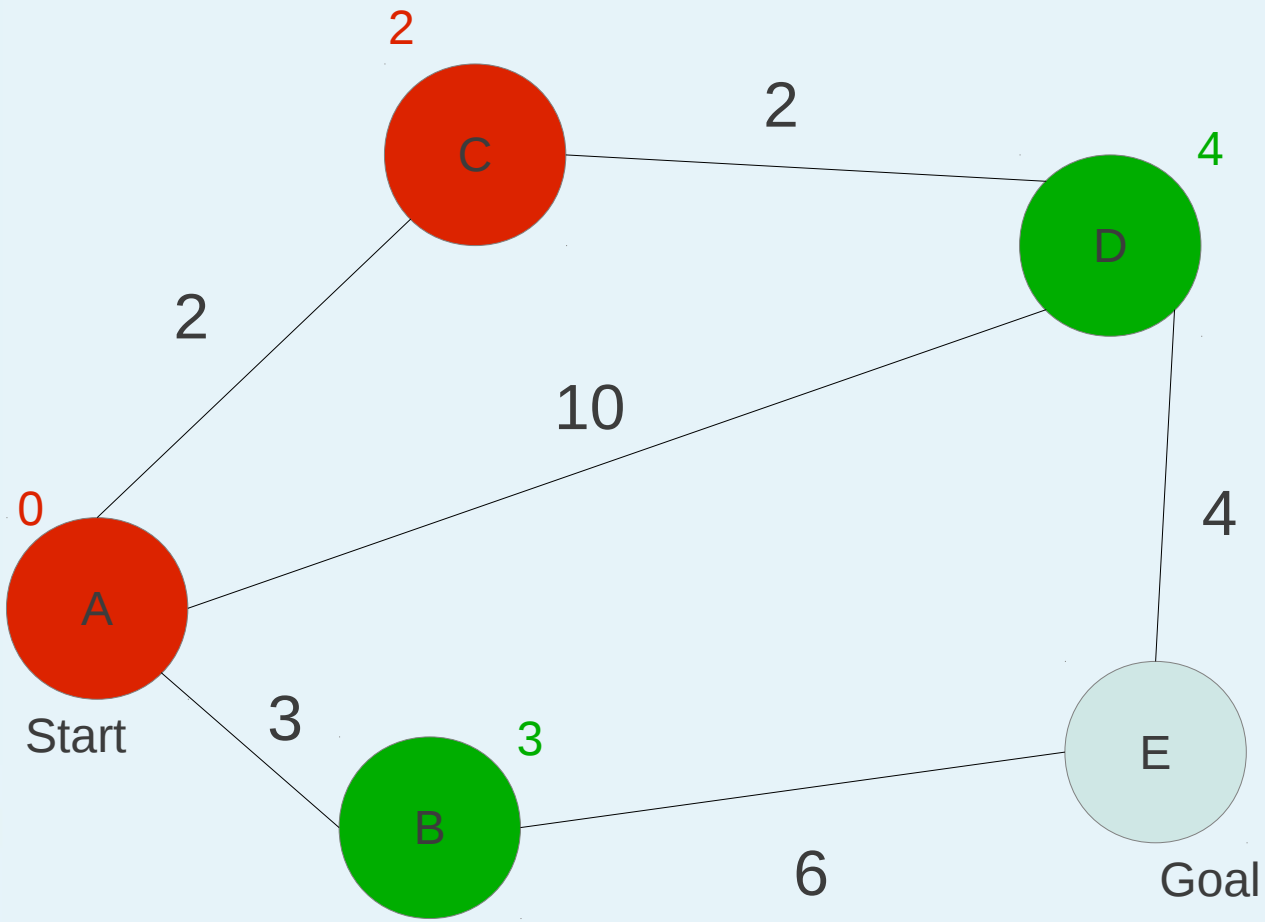
アルゴリズム



候補ノード

Dist	Node
3	B
10	D

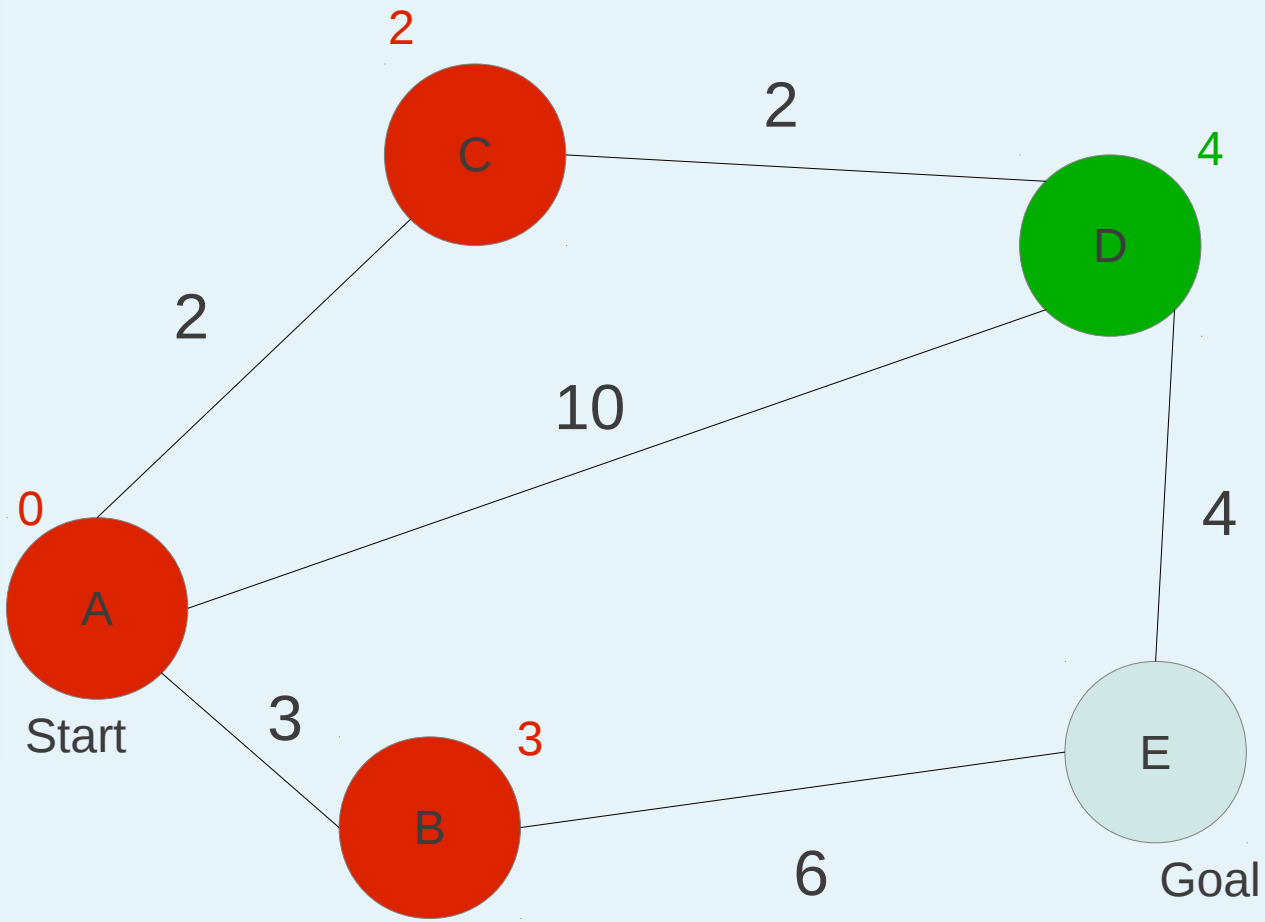
アルゴリズム



候補ノード

Dist	Node
3	B
4	D
10	D

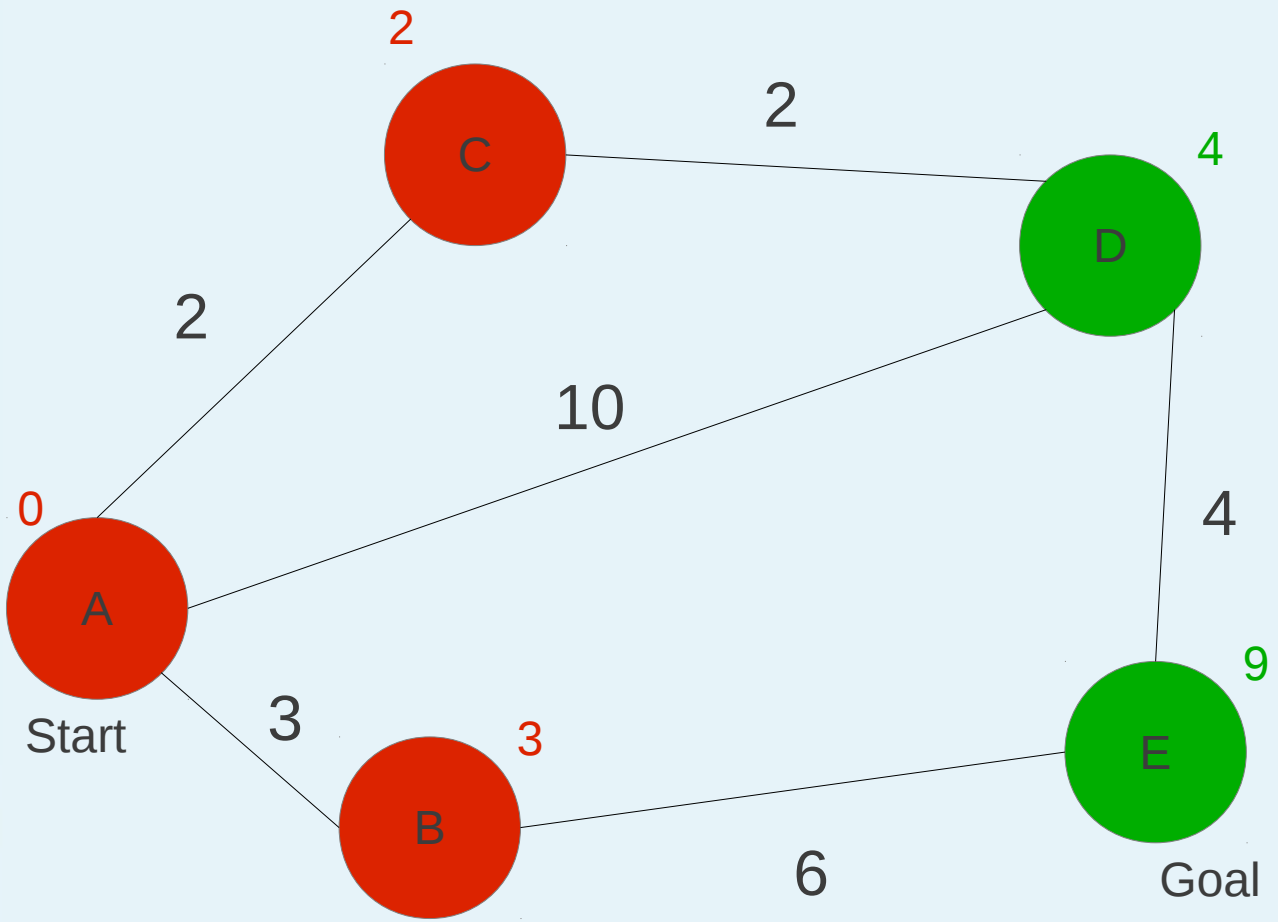
アルゴリズム



候補ノード

Dist	Node
4	D
10	D

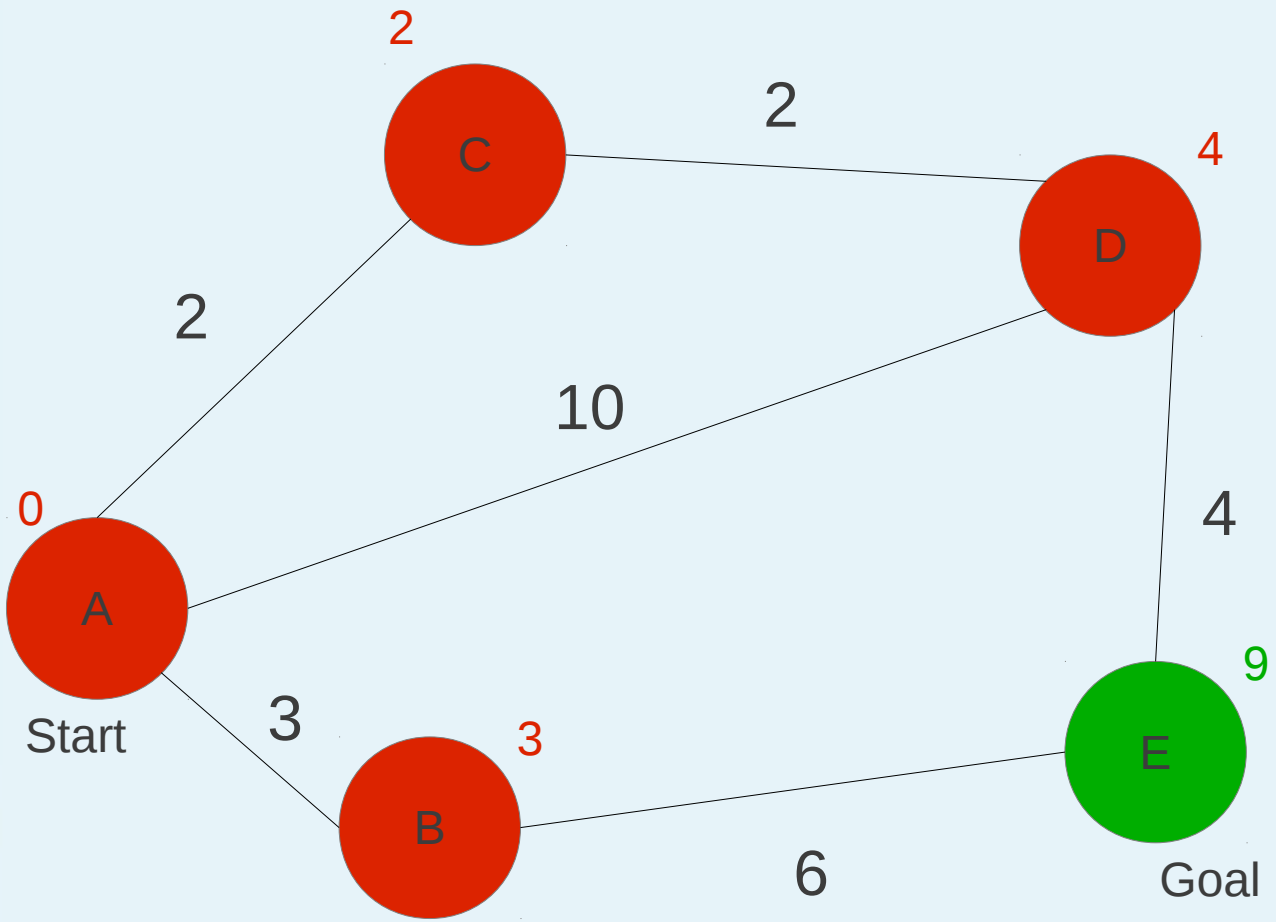
アルゴリズム



候補ノード

Dist	Node
4	D
9	E
10	D

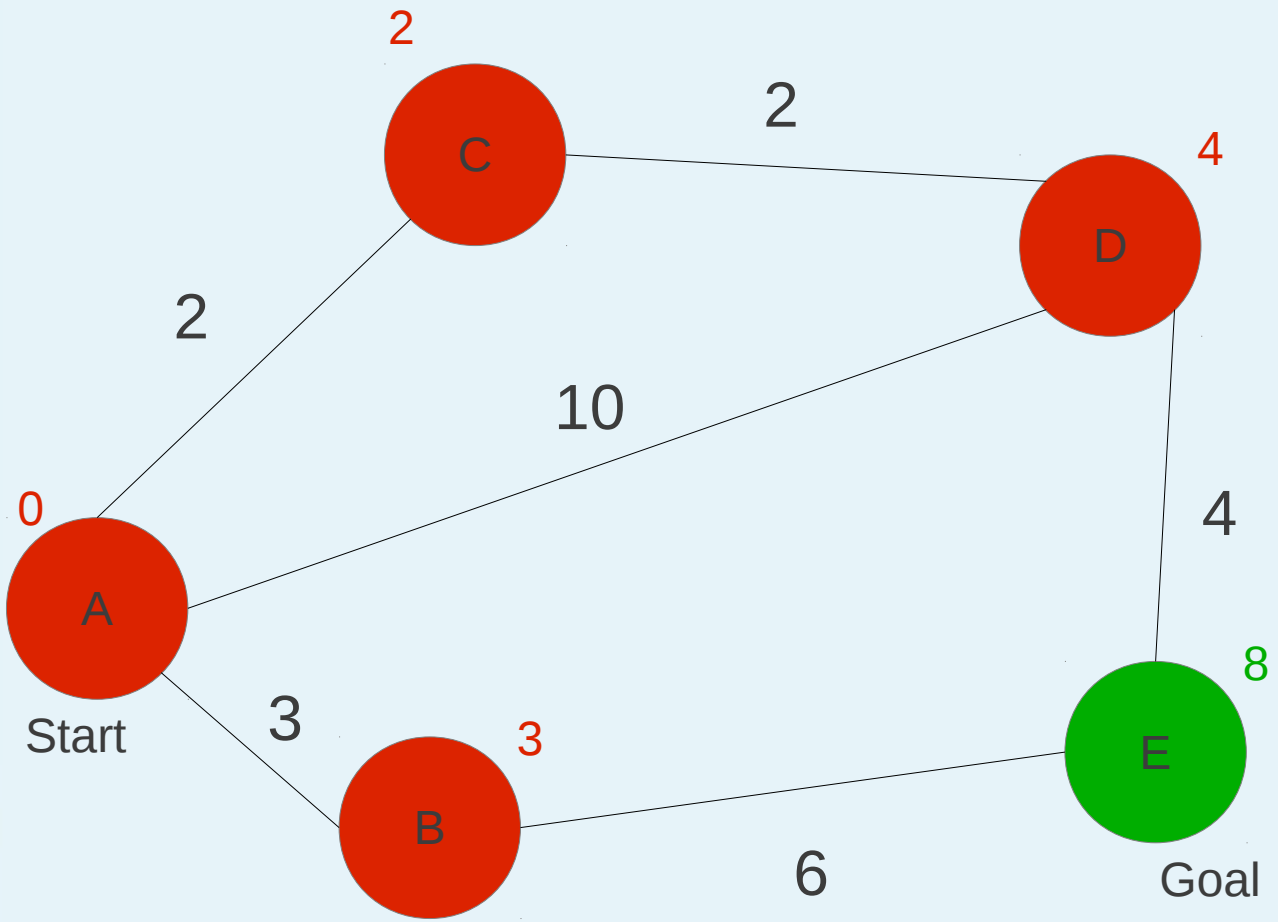
アルゴリズム



候補ノード

Dist	Node
9	E
10	D

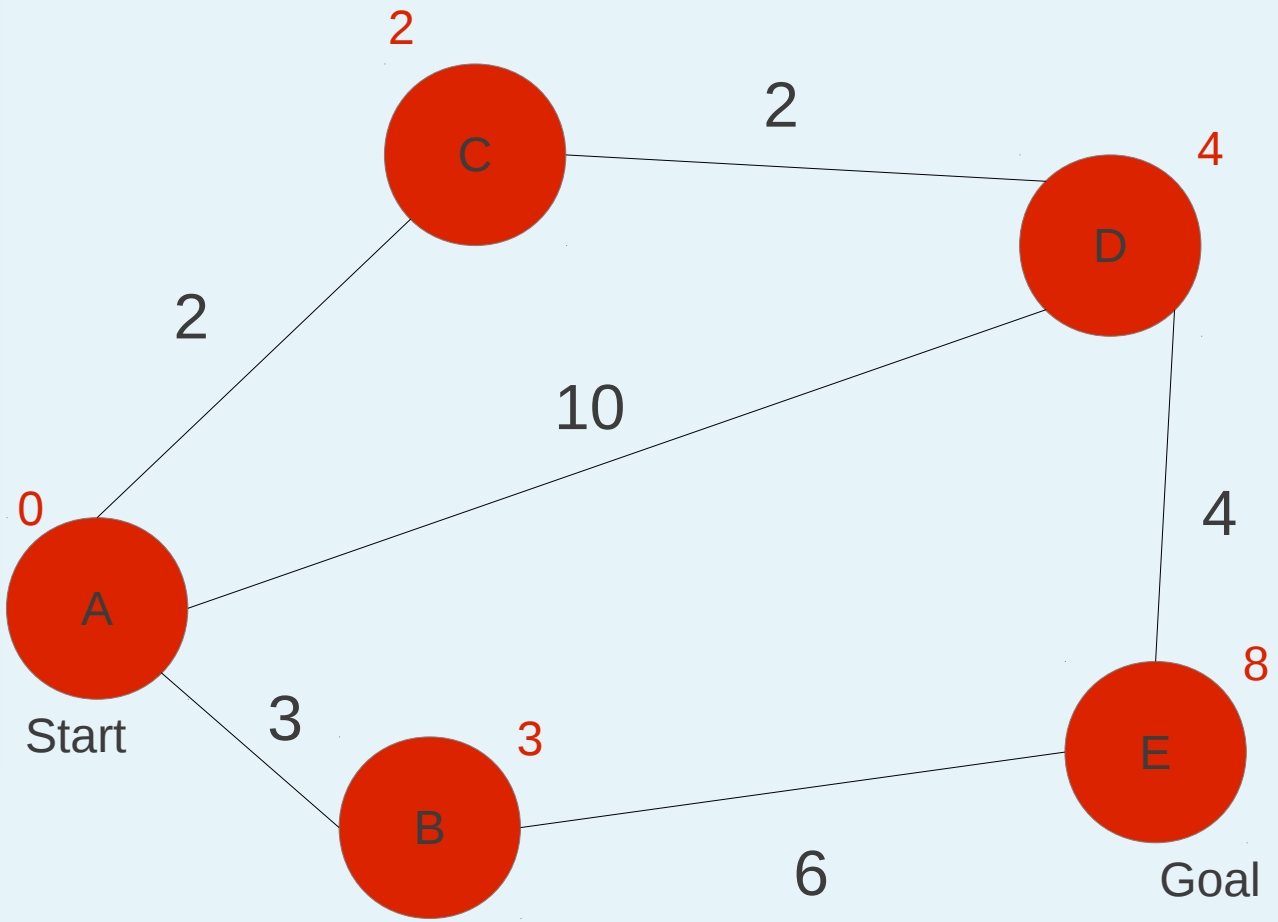
アルゴリズム



候補ノード

Dist	Node
8	E
9	E
10	D

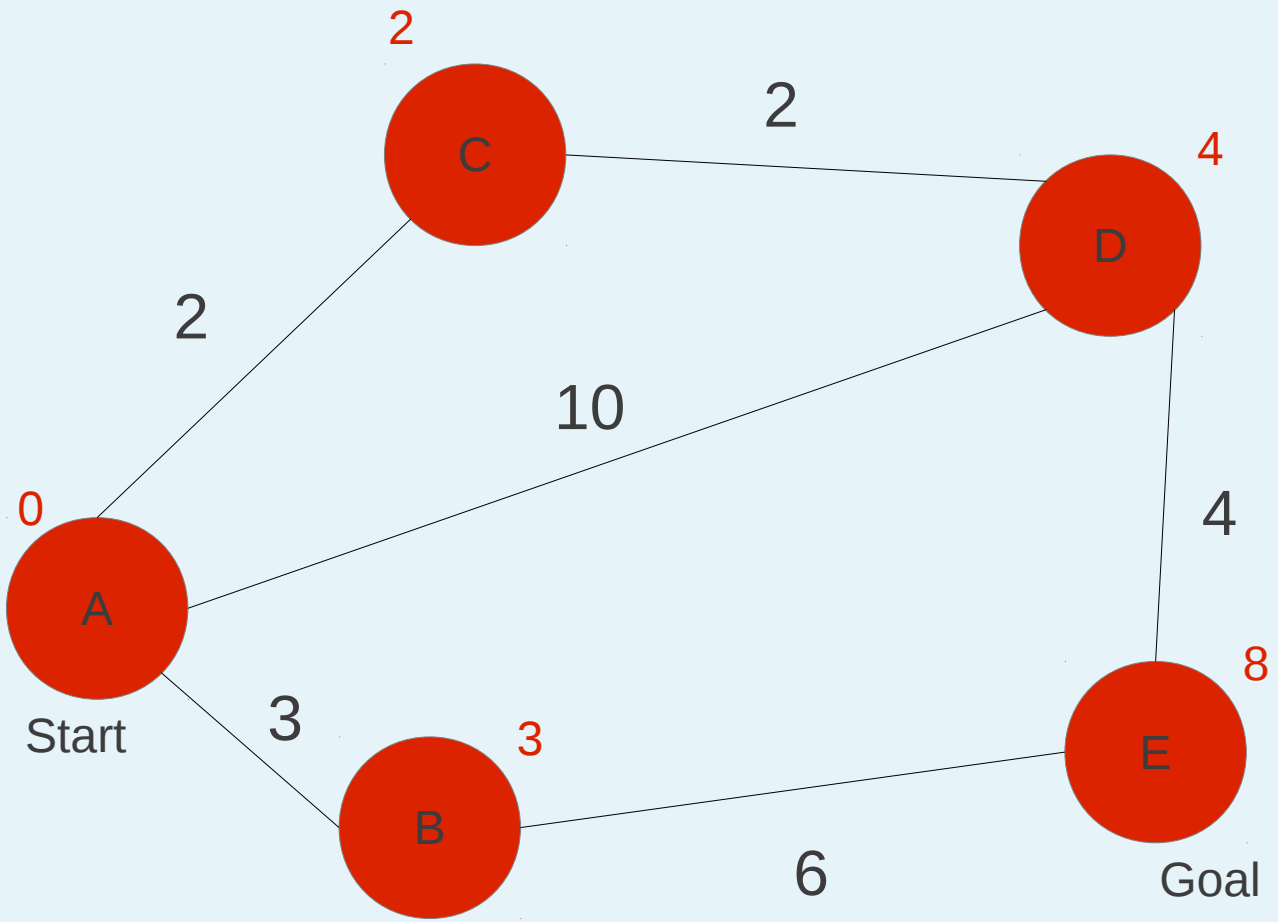
アルゴリズム



候補ノード

Dist	Node
9	E
10	D

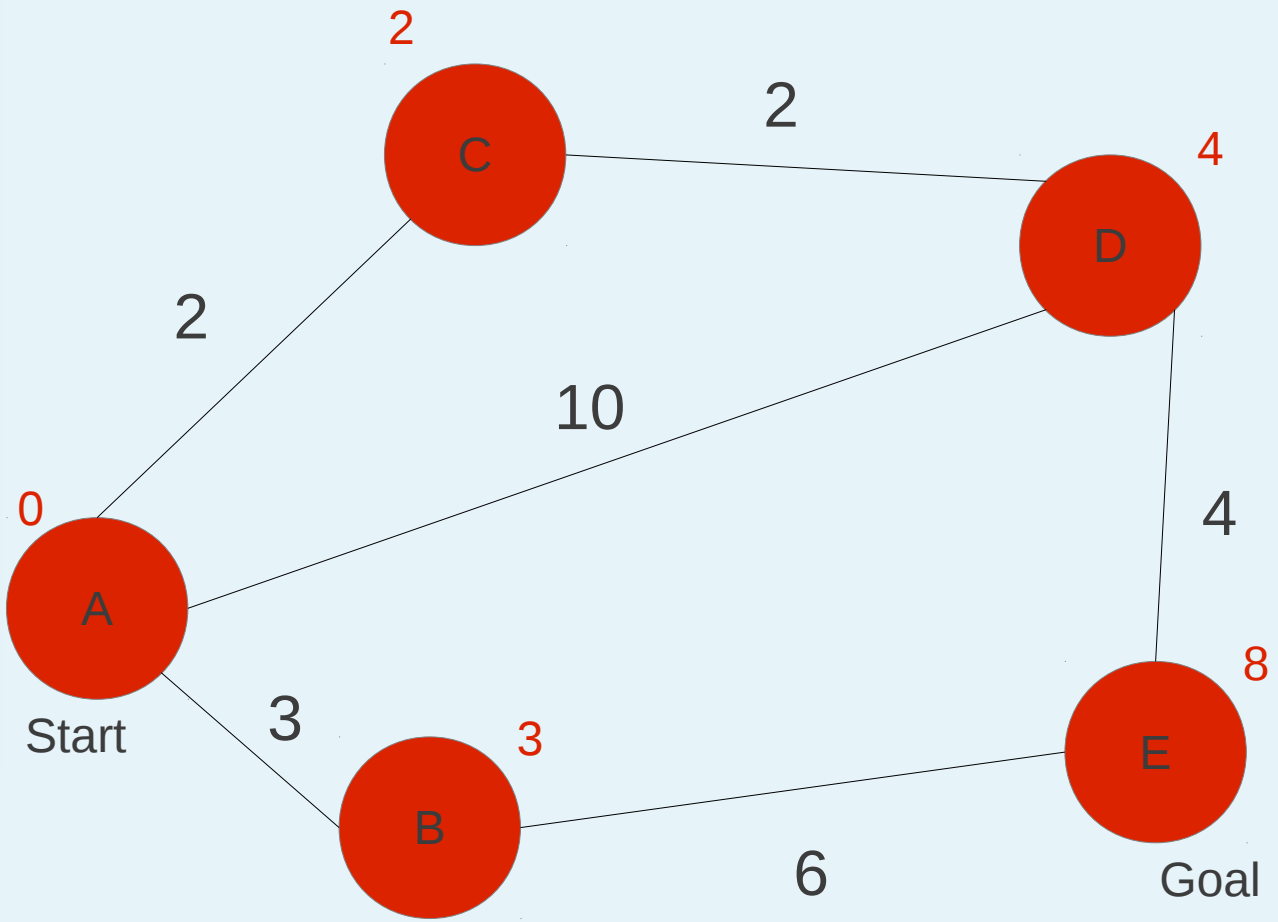
アルゴリズム



候補ノード

Dist	Node
10	D

アルゴリズム



候補ノード

Dist	Node

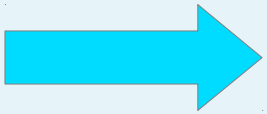
Princess in Danger

- ダイクストラ法の応用
  - タイムリミットをどう扱うか
  - 再冷凍をどう扱うか

アルゴリズム

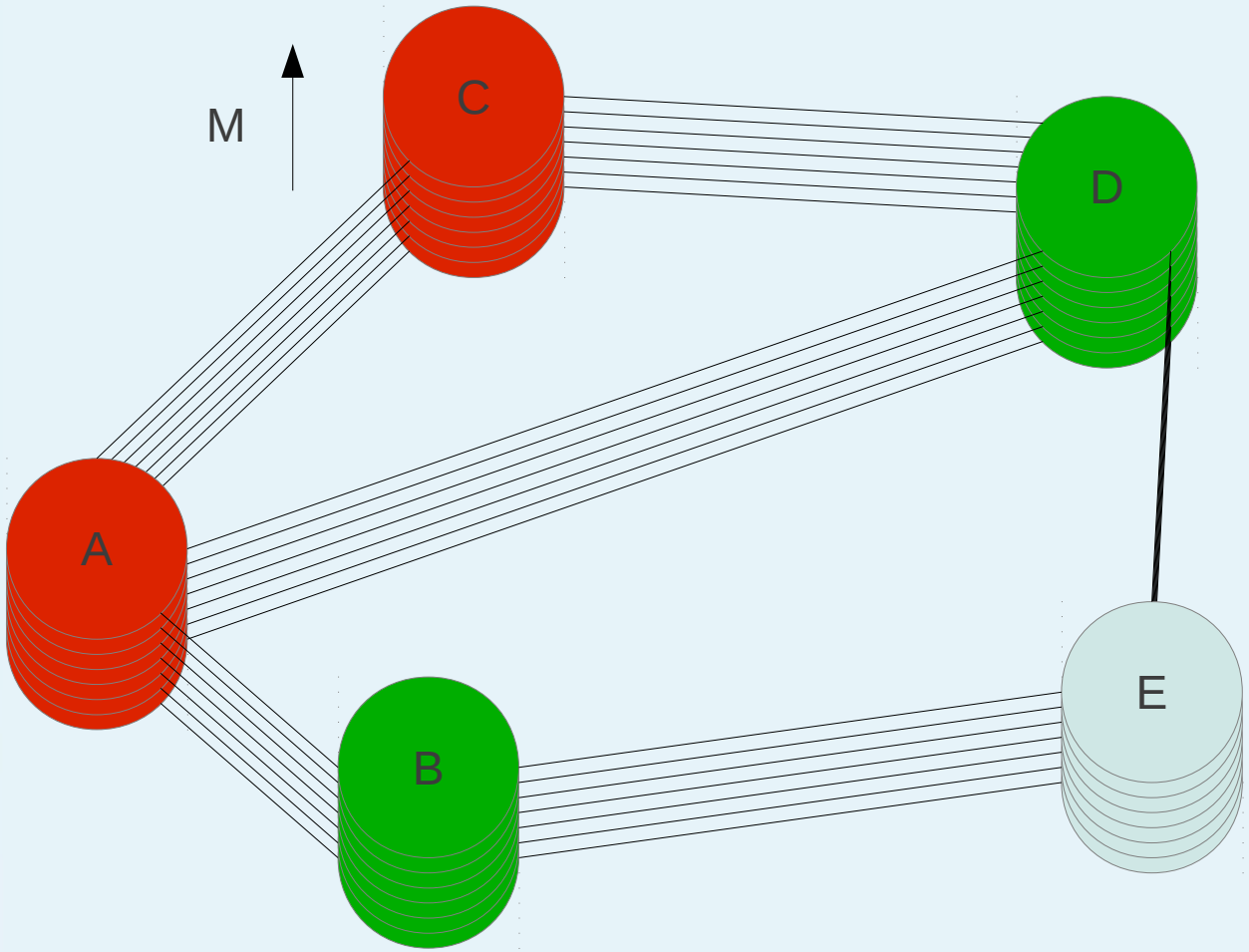
- 残りリミットに応じた多レイヤーでのダイクストラ法
  - 残り M でノード V に着く最短経路

```
struct vertex {
	vector<edge> edges;

	bool fixed;
	int distance;
};
```



```
struct vertex {
	vector<edge> edges;

	bool fixed[MAX_M];
	int distance[MAX_M];
};
```

アルゴリズム



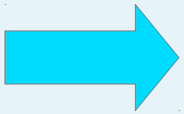
候補ノード

Dist	Node	M
3	B	5
4	D	4
10	D	10
12	B	5

アルゴリズム

- 冷凍は最大限行う
- 最後に、余分な分を冷凍しなかったことにする

```
struct vertex {
        vector<edge> edges;

        bool fixed[MAX_M];
        int distance[MAX_M];
};
```



```
struct vertex {
        vector<edge> edges;

        bool fixed[MAX_M];
        int distance[MAX_M];
        Int total_freezing_time[MAX_M];
};
```

アルゴリズム

- 最後に、ゴールを全てのＭについて調べる
- 最短経路を見つけて、終了！