Shortest Path - Dijkstra

Shortest Path

👉 최단 거리 문제

- **Graph Application**
- Input : Weighted Graph

- **거리** : 특정 노드 A, B사이의 경로(Path)의 가중치(Weight)의 합
- '거리가 최소이도록 하는 경로'

- 최단 거리 **특징**
  - Acylic
  - 최적화 구조(Optimal Structure)가 존재
    => Greedy Algorithm 접근 가능

Dijkstra Algorithm

👉 정의

- **Input**

  1️⃣ Weighted Graph 2️⃣ Starting Node

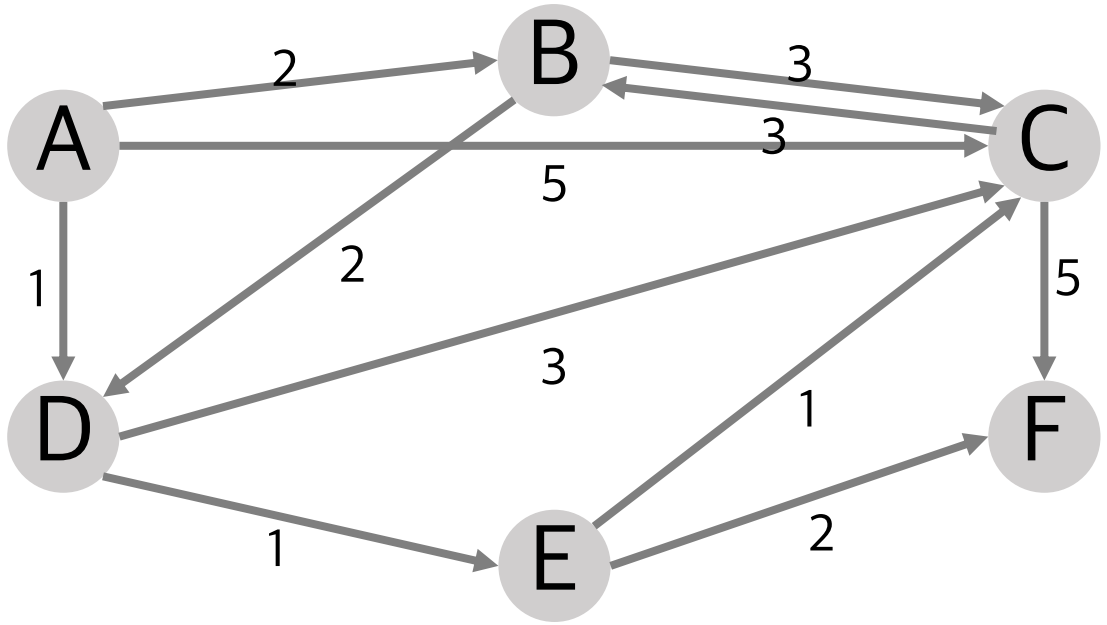
- **Output**

  Shortest Paths from **Starting** to **each node**

Dijkstra Algorithm

👉 예시

- **Input**
  Following Graph,
  Starting - **A**



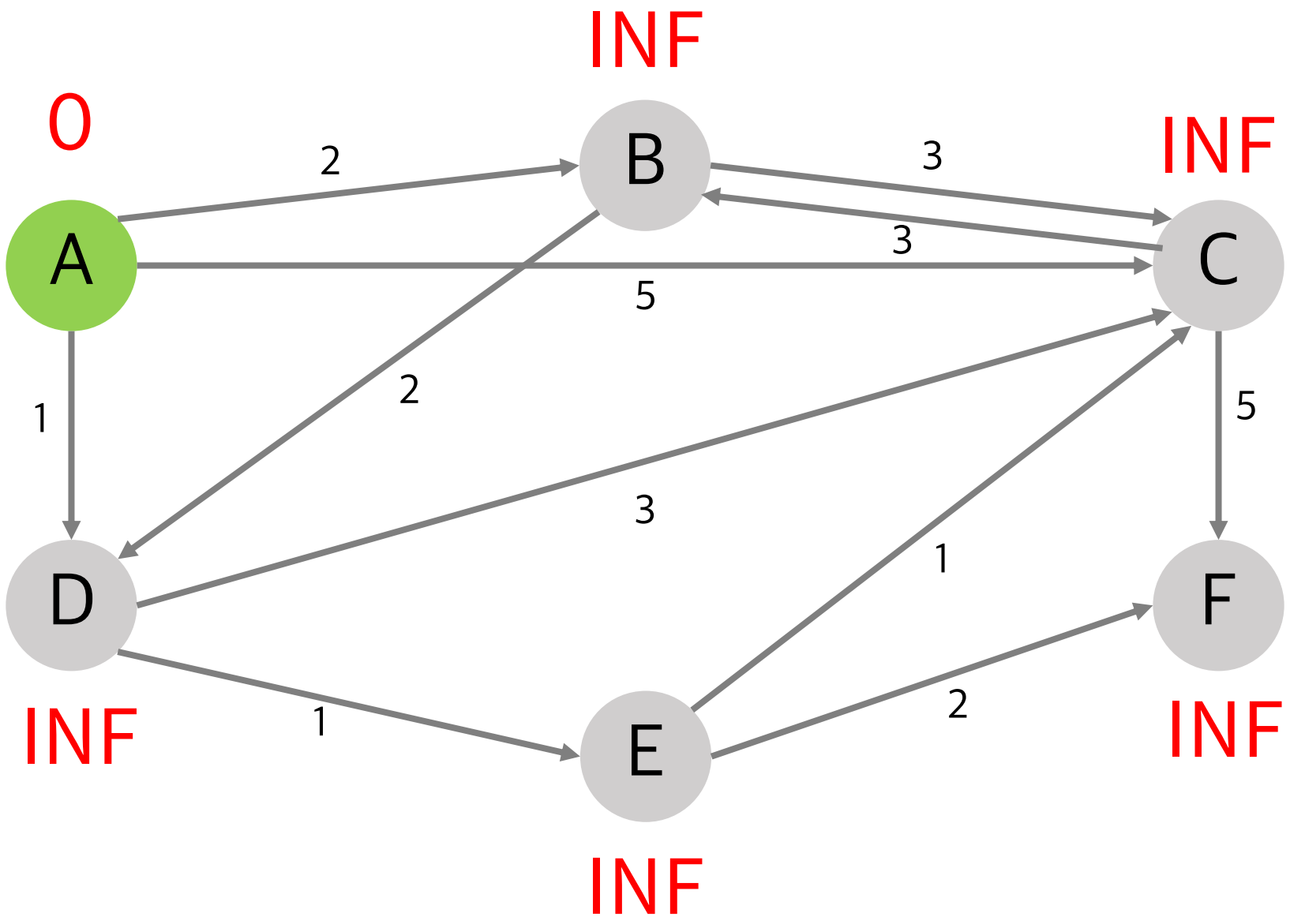
- **Output**
  from A to A : 0 from A to D : 1
  from A to B : 2 from A to E : 2
  from A to C : 3 from A to F : 4

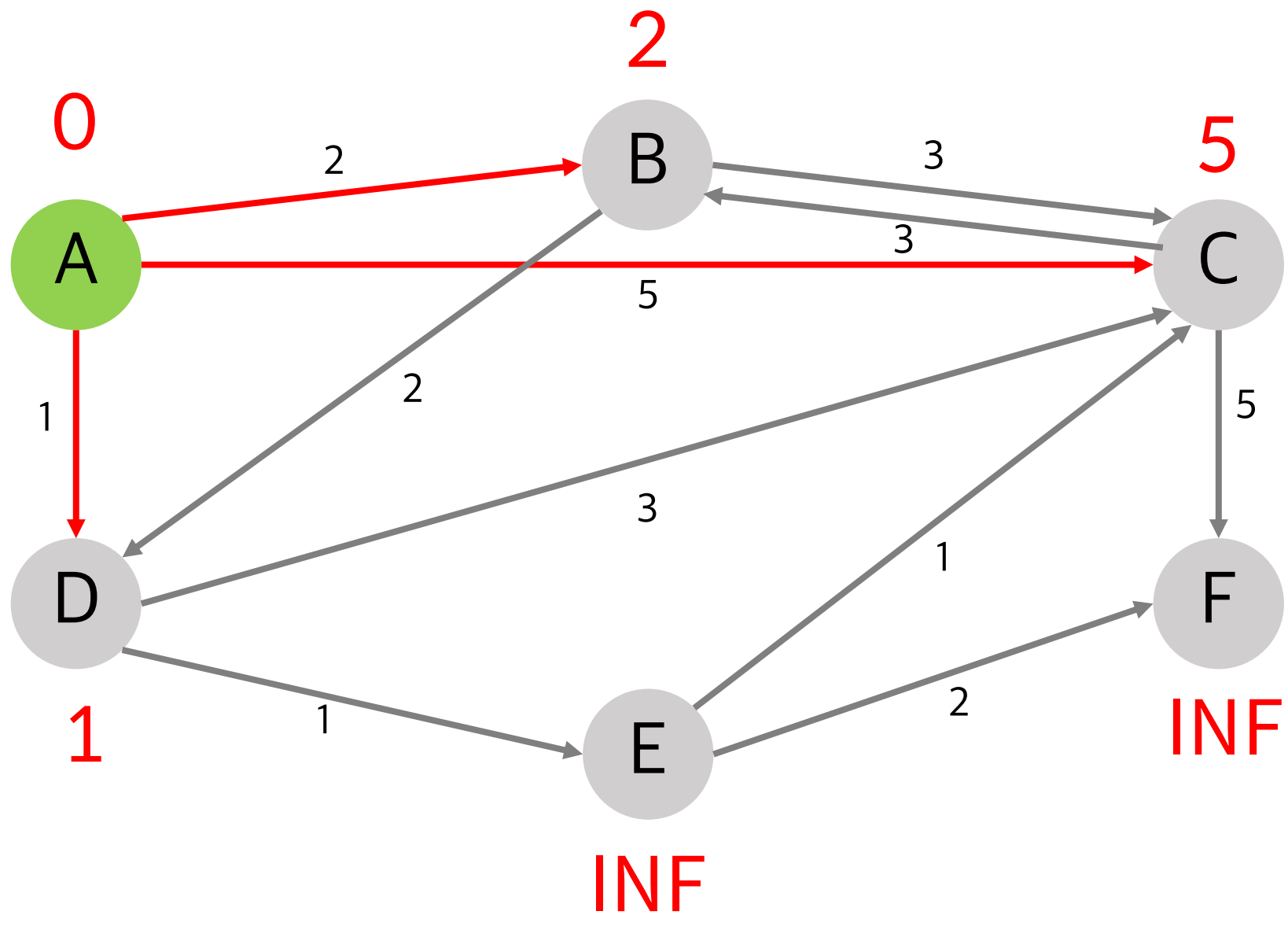
Dijkstra Algorithm

👉

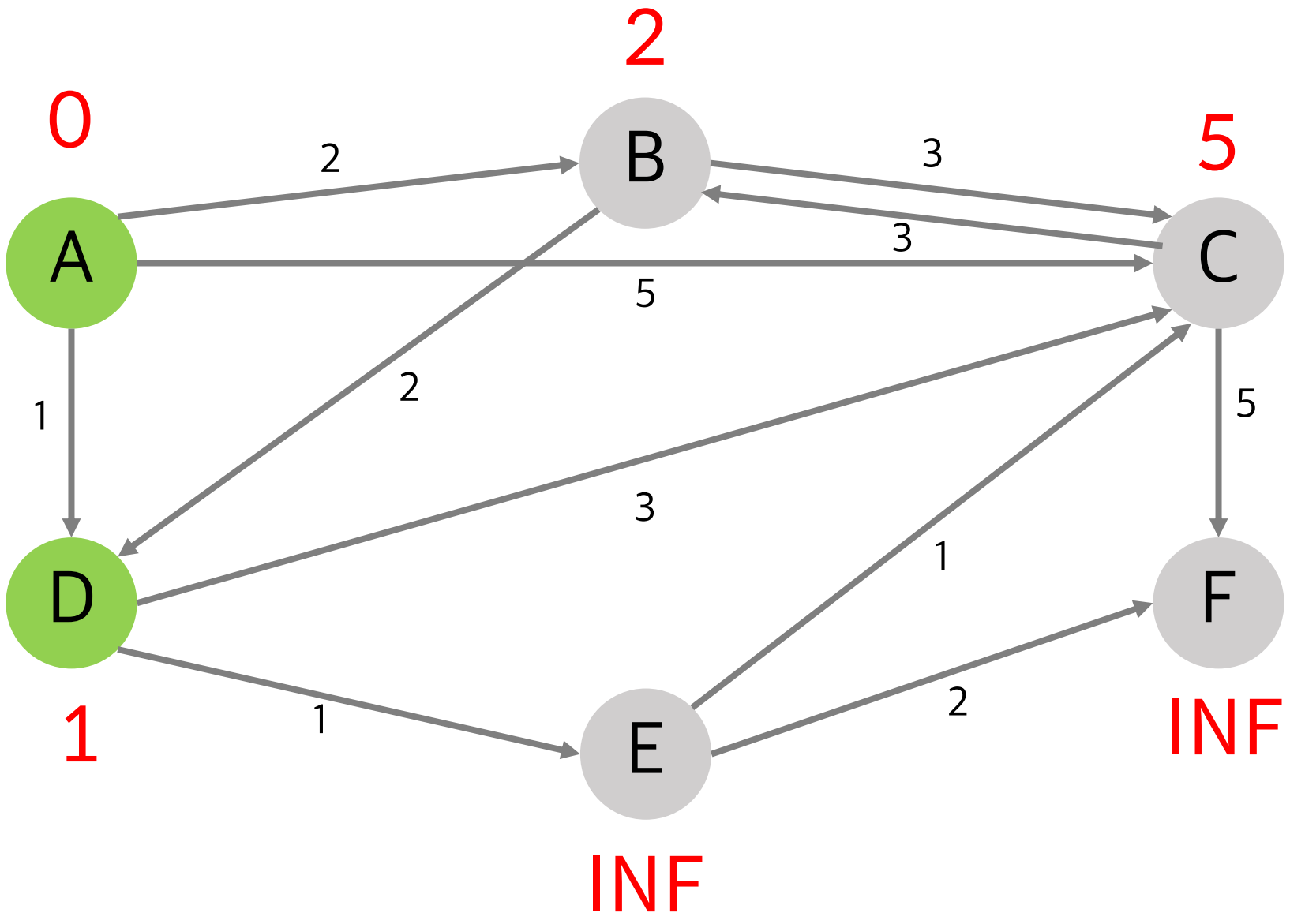
- **Process**
  1. **출발** 노드를 설정
  2. 최단 거리 테이블을 **초기화**
  3. 방문하지 않은 노드 중에서 최단 거리가 **최소**인 노드를 선택
  4. 해당 노드를 거쳐 다른 노드로 가는 비용을 계산하여 최단 거리 테이블을 **갱신**합니다
  5. 방문하지 않은 노드가 1개 남을 때까지 3,4번을 **반복**합니다.

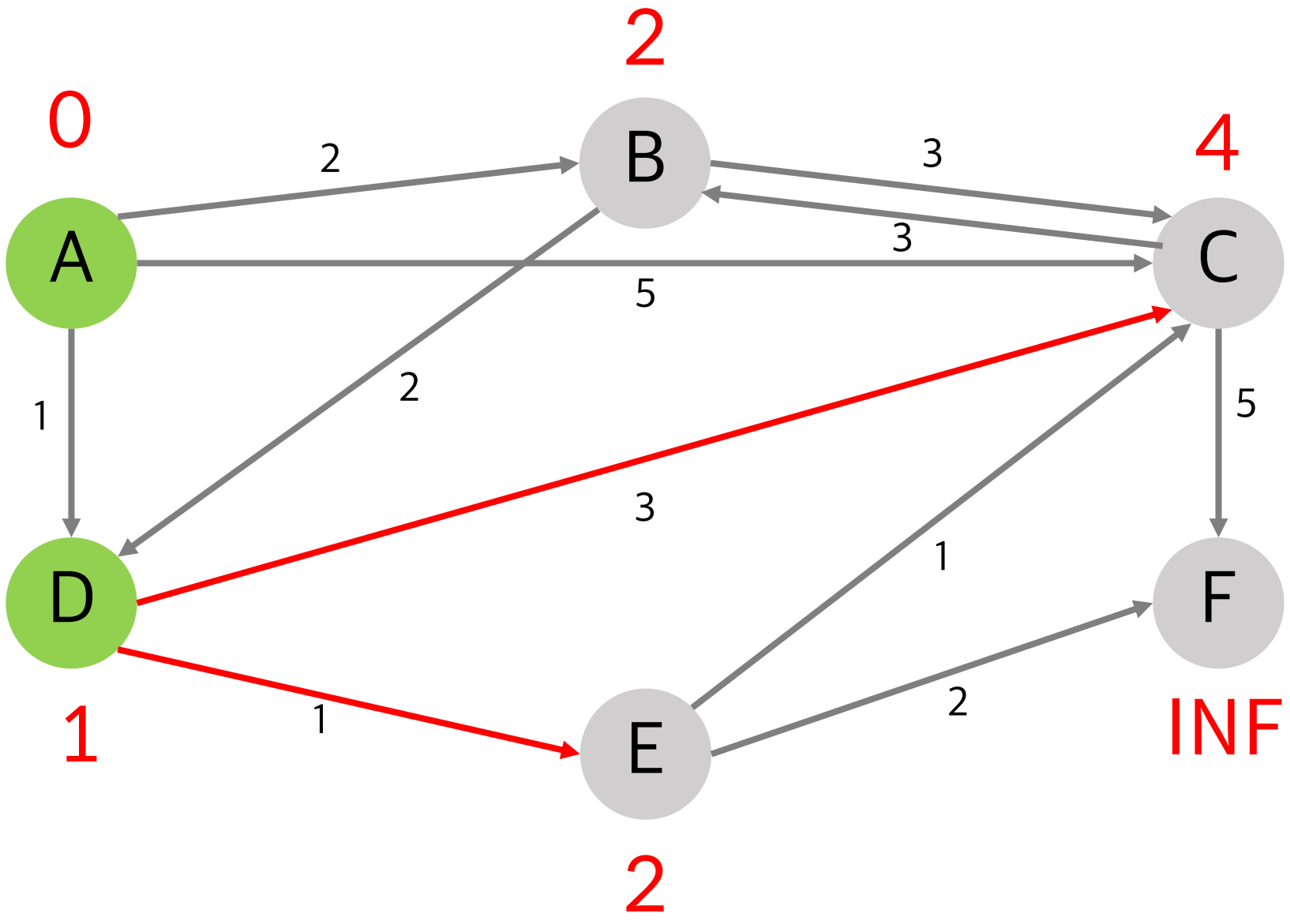
https://www.youtube.com/watch?v=acqm9mM1P6o&ab_channel=%EB%8F%99%EB%B9%88%EB%82%98

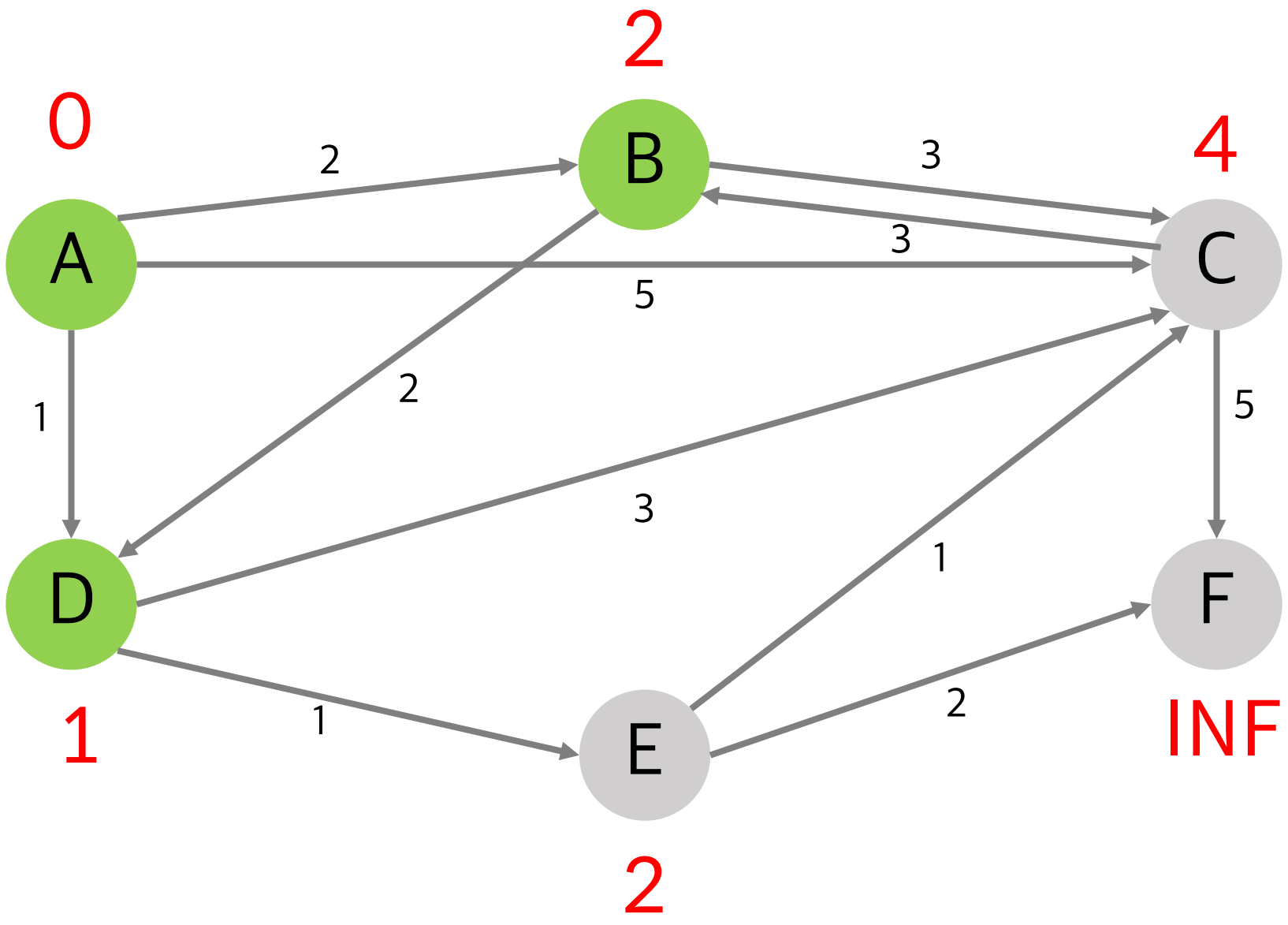

INF
0
B
INF
2
3
A
3
C
5
1
2
5
3
1
D
F
INF
1
2
INF
E
INF

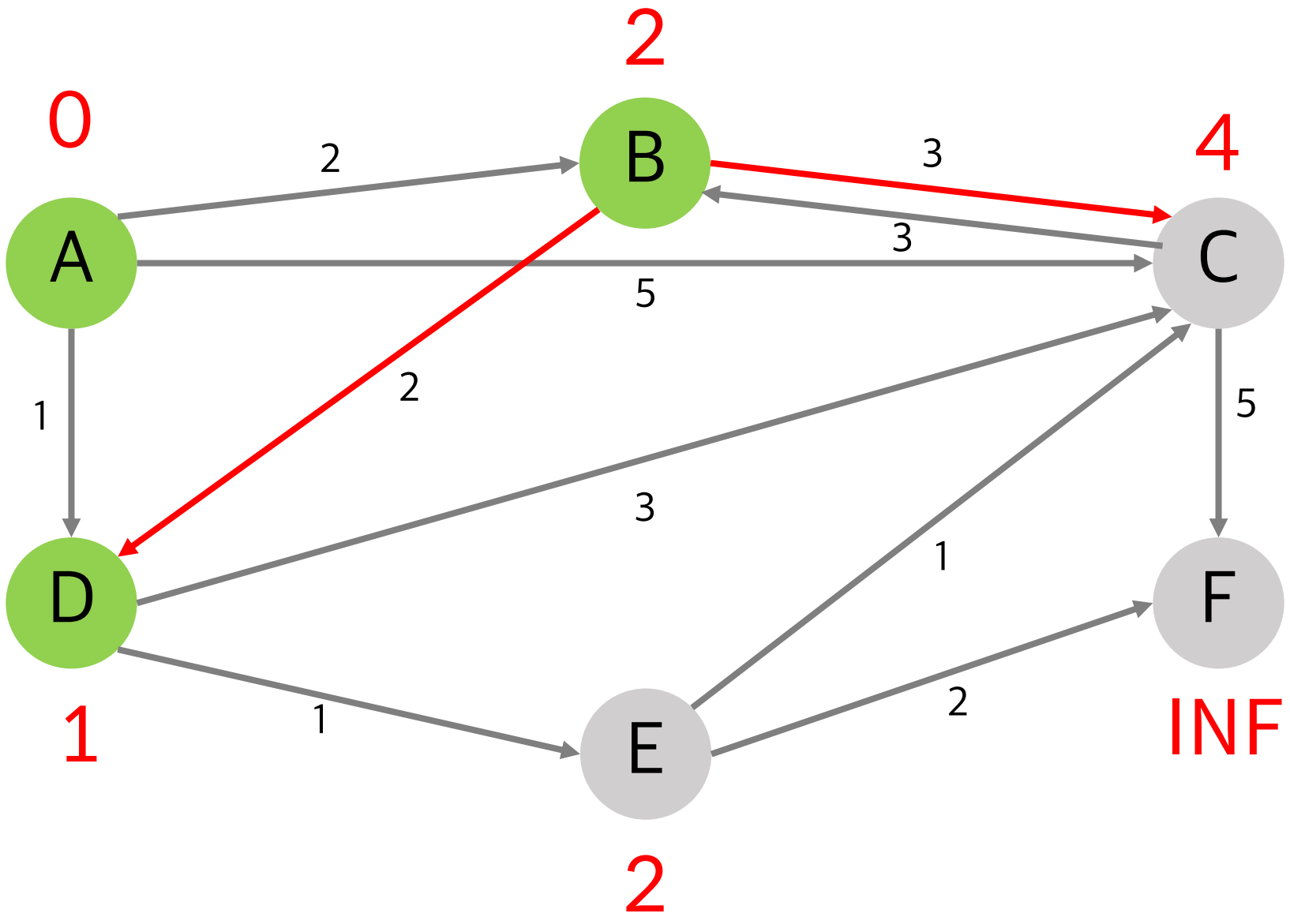


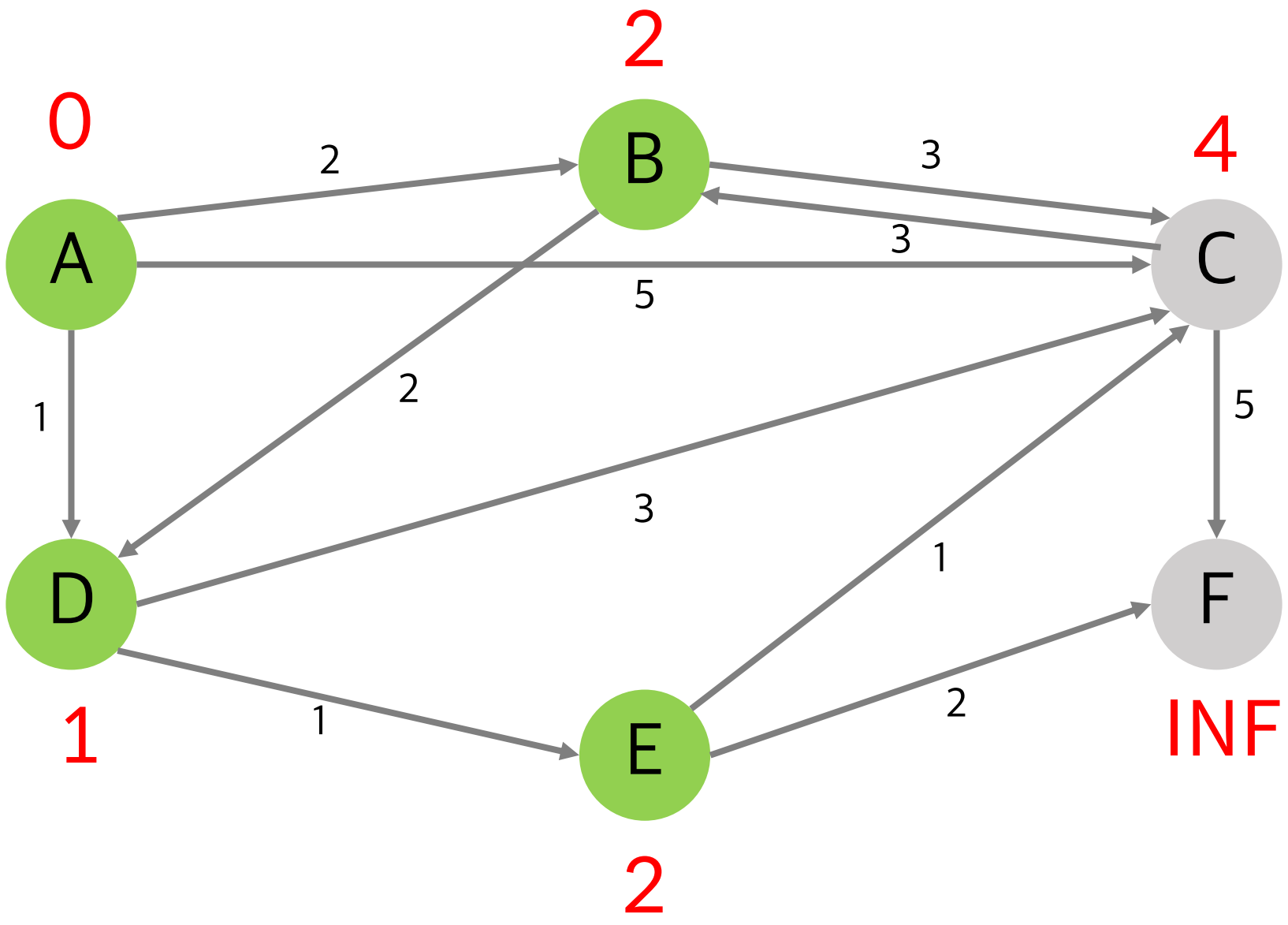
Graph	[0]	[1]	[2]
A	(B,2)	(D,1)	(C,5)
B	(D,2)	(C,3)	
C	(B,3)	(F,3)	
D	(C,3)	(E,1)	
E	(C,1)	(E,2)	
F			

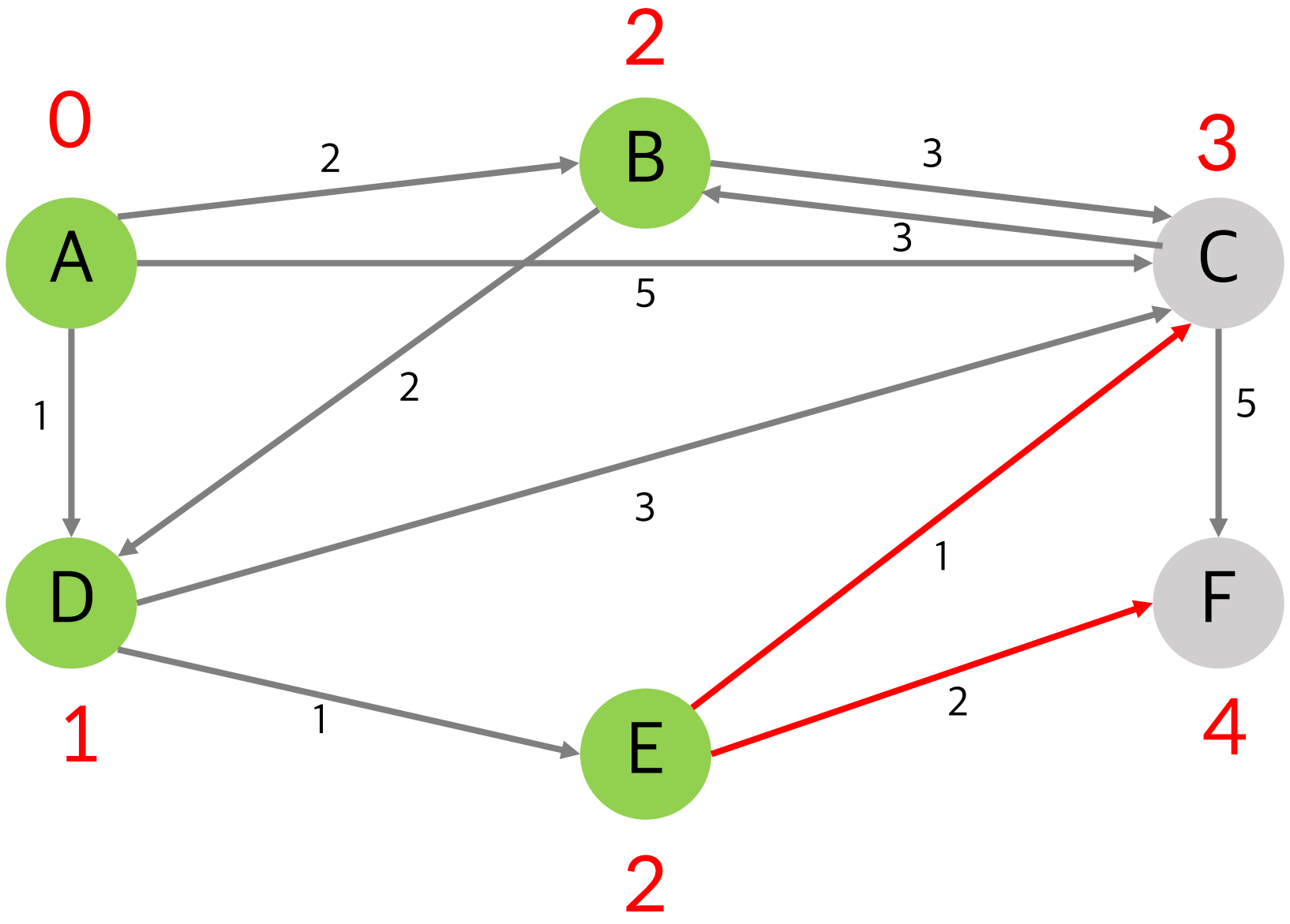

0
2
5
A
B
C
D
E
F
2
3
3
5
1
2
5
3
1
1
2
1
INF
INF

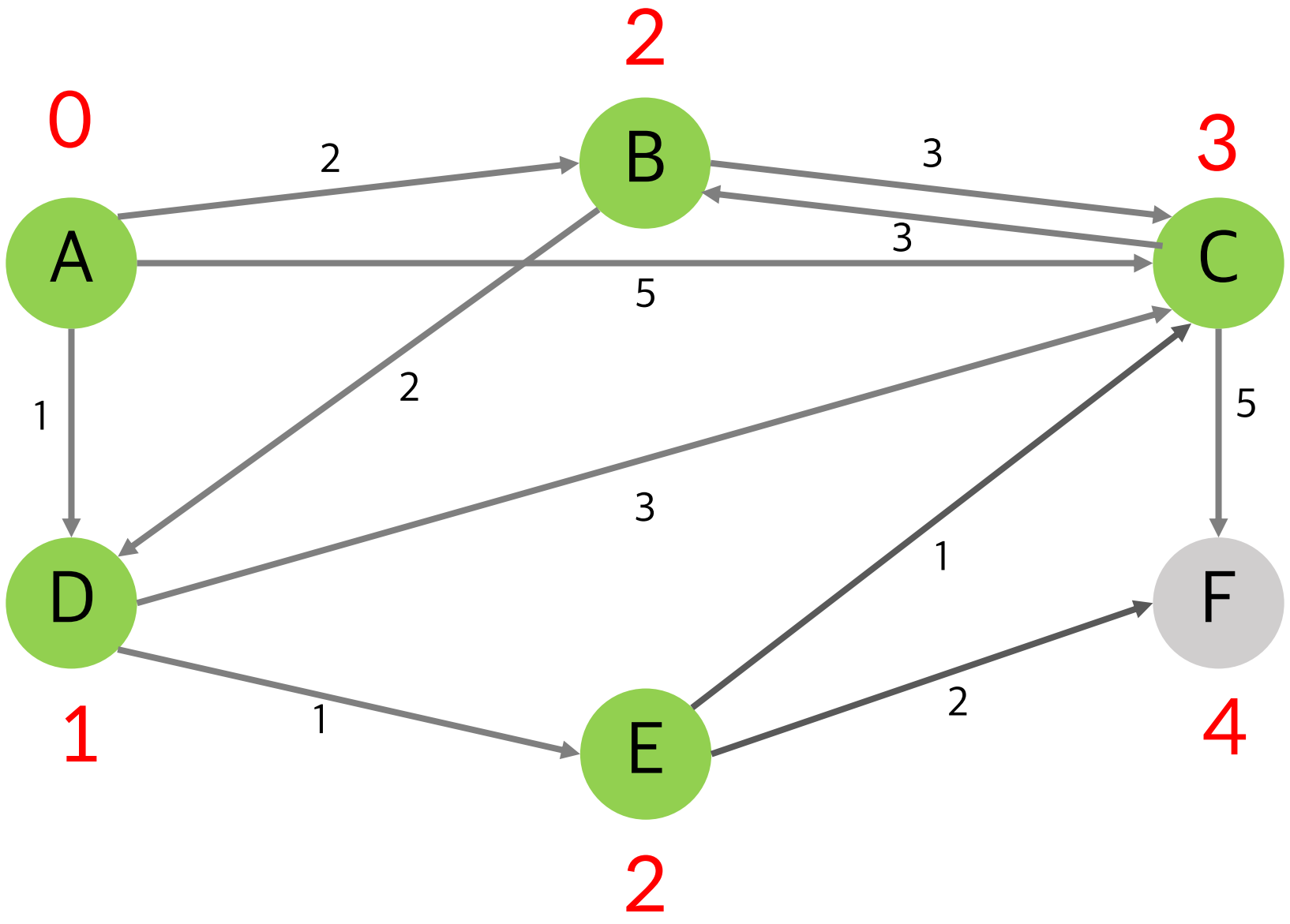

2
0
B
4
2
3
A
C
3
5
1
2
5
3
1
D
F
1
1
2
INF
E
2

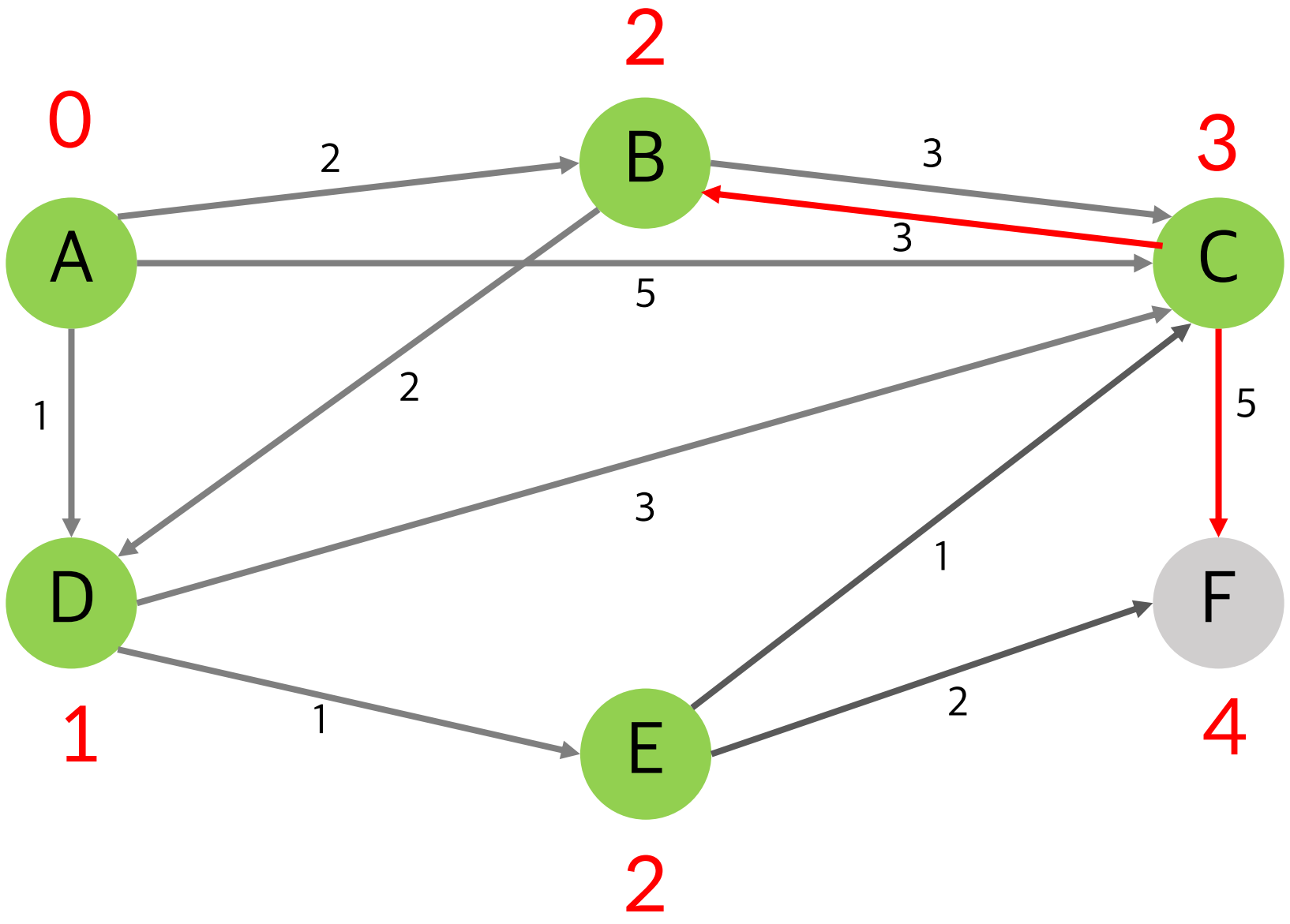

2
0
B
2
3
4
A
3
C
5
1
2
5
3
D
1
F
1
2
INF
E
2

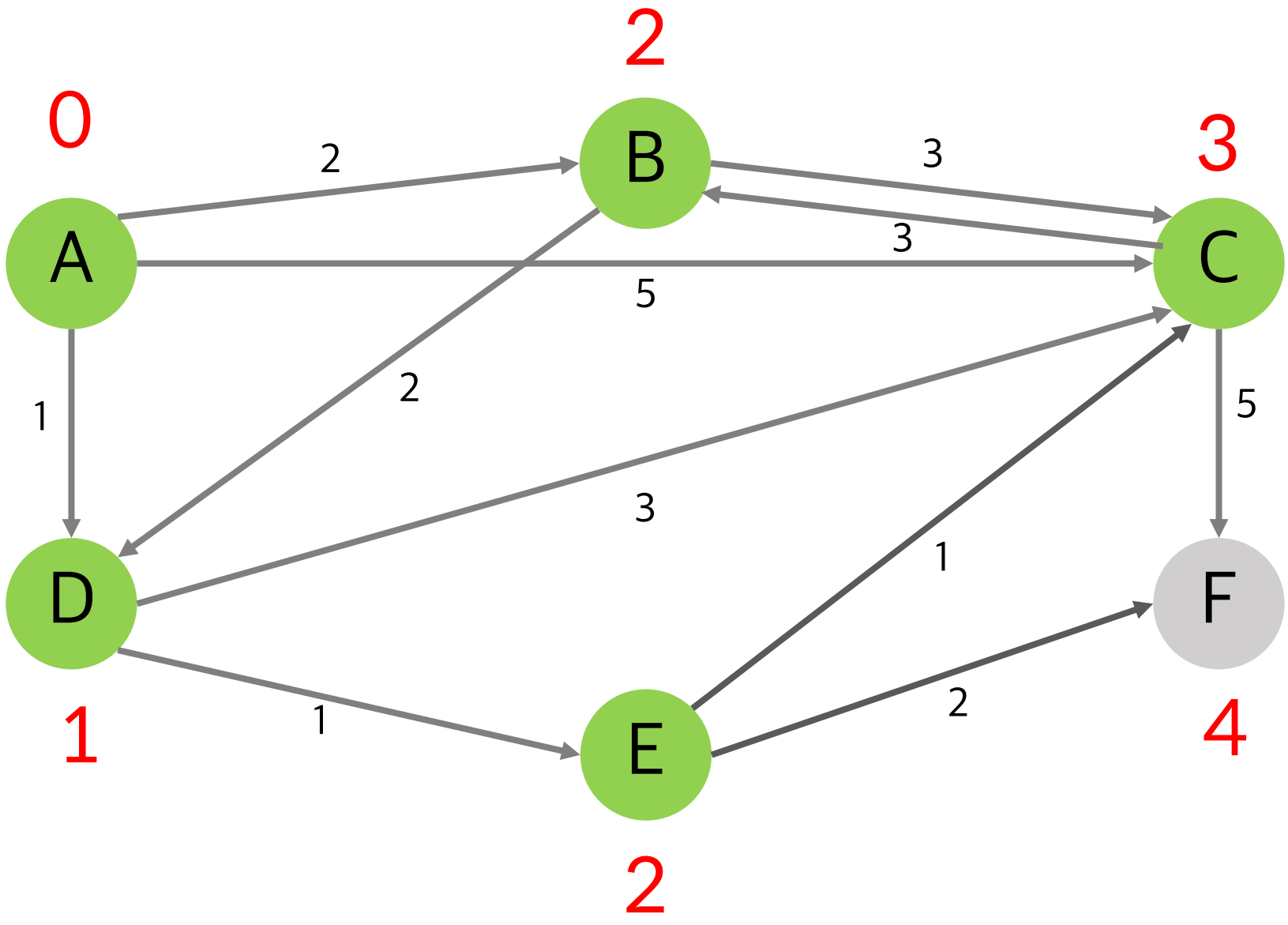

2
0
B
2
3
4
A
3
C
5
1
2
5
3
D
1
F
1
2
INF
E
2


2
0
B
4
2
3
A
C
3
5
1
2
5
3
1
D
F
1
1
2
INF
E
2

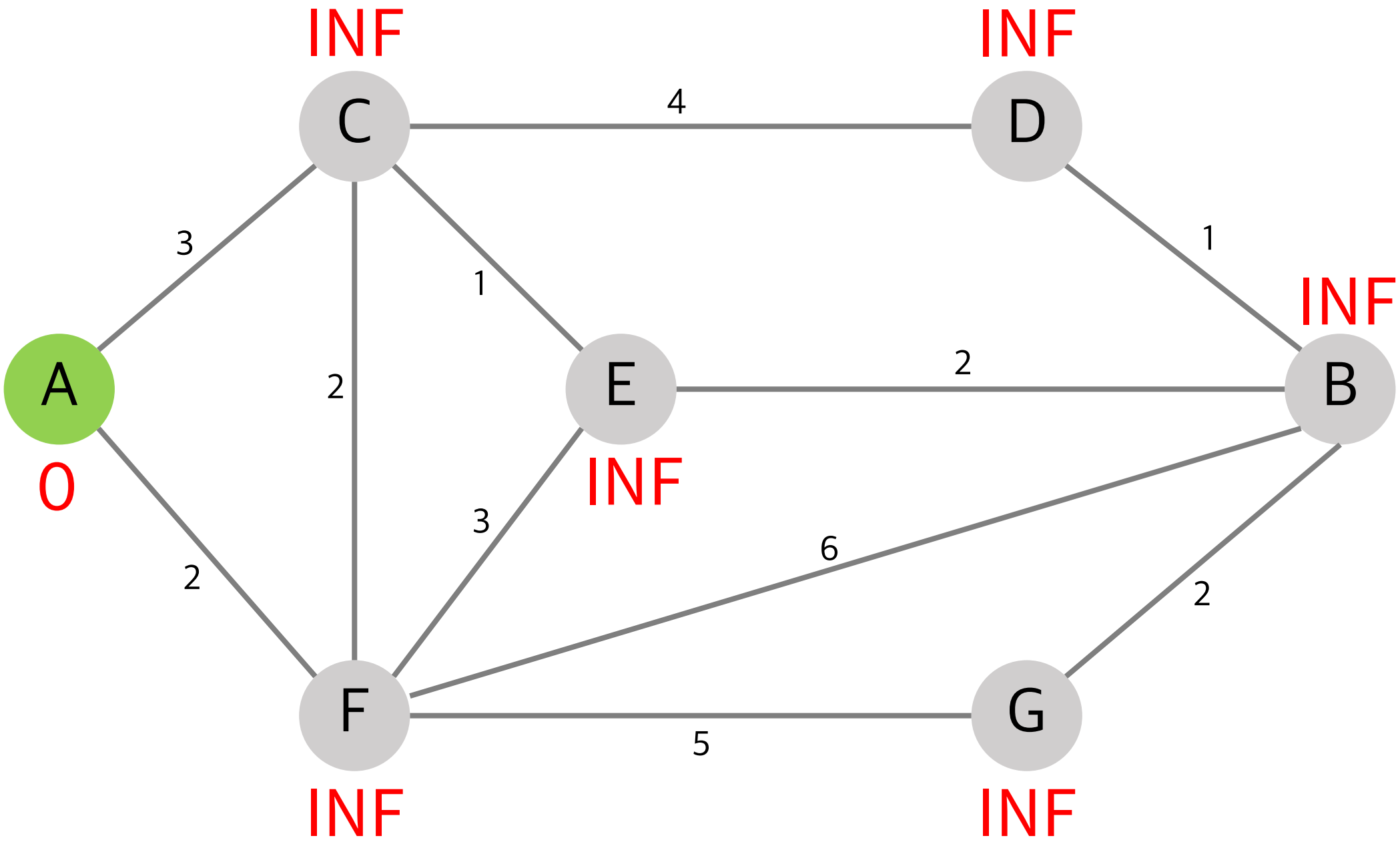

0
2
3
1
2
A
B
C
D
E
F
2
3
3
5
1
2
5
3
1
1
2

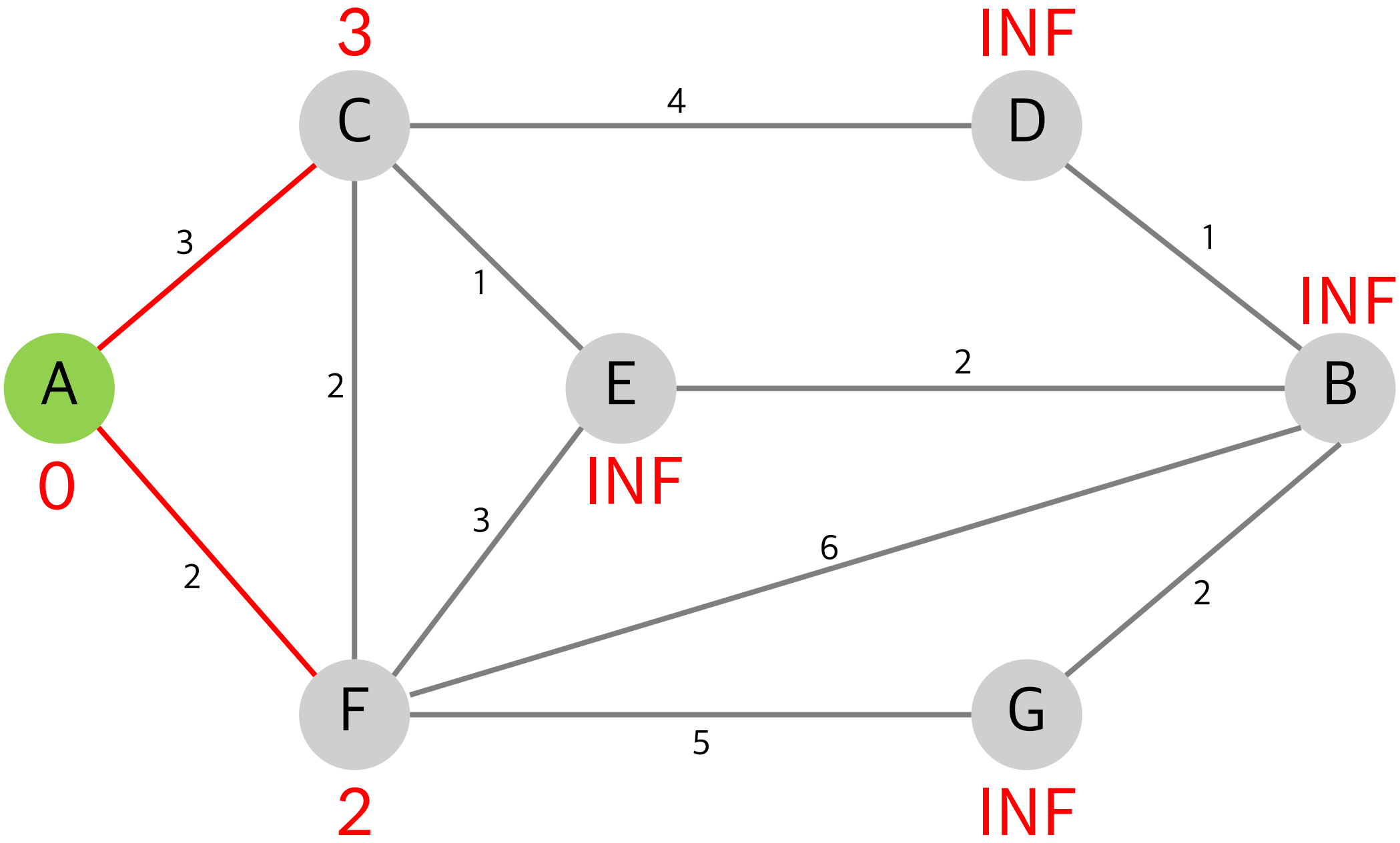

2
0
B
3
2
3
A
C
3
5
1
2
5
3
1
D
F
1
1
2
4
E
2

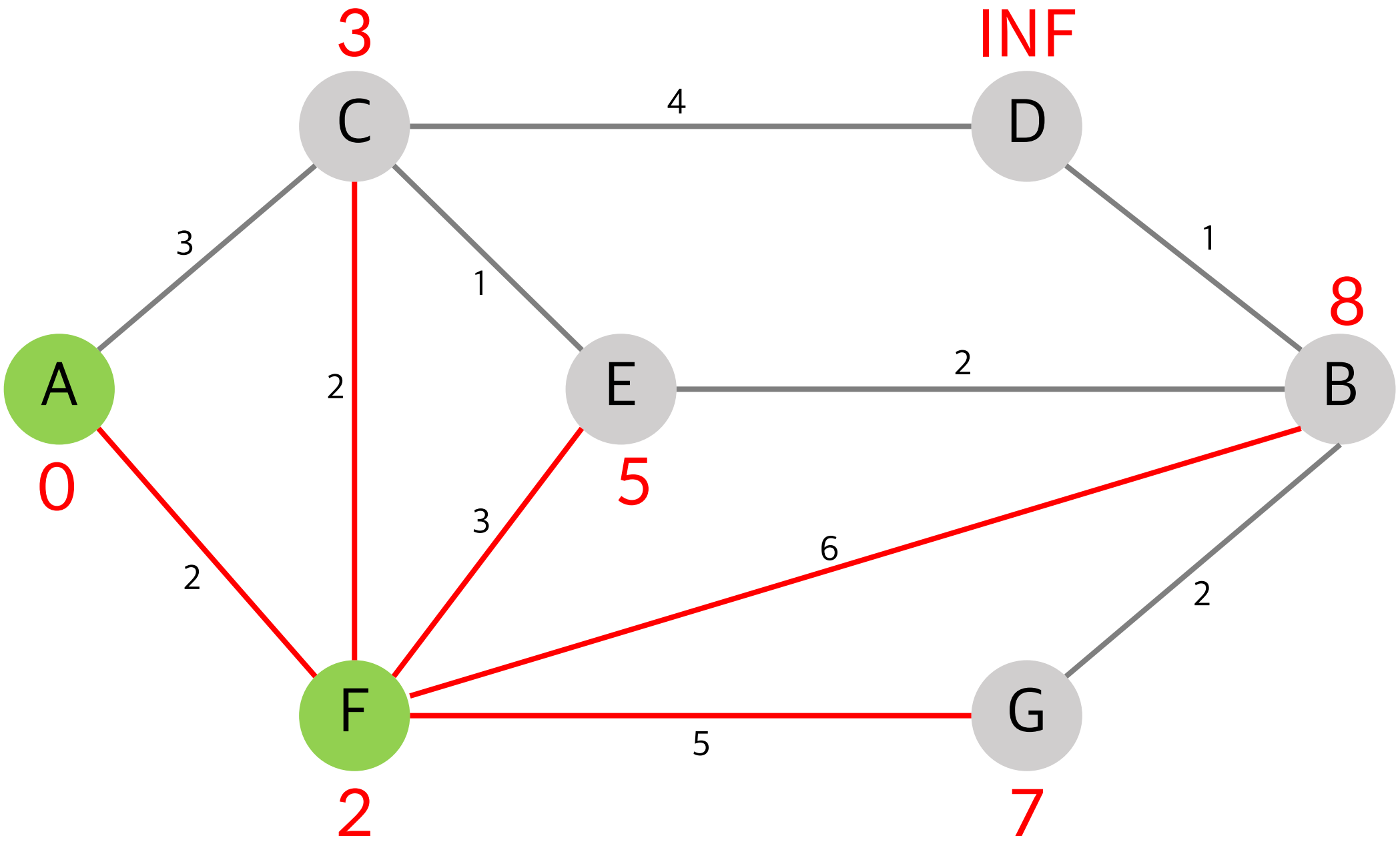

2
0
B
3
2
3
A
C
3
5
1
2
5
3
1
D
F
1
1
2
E
4
2

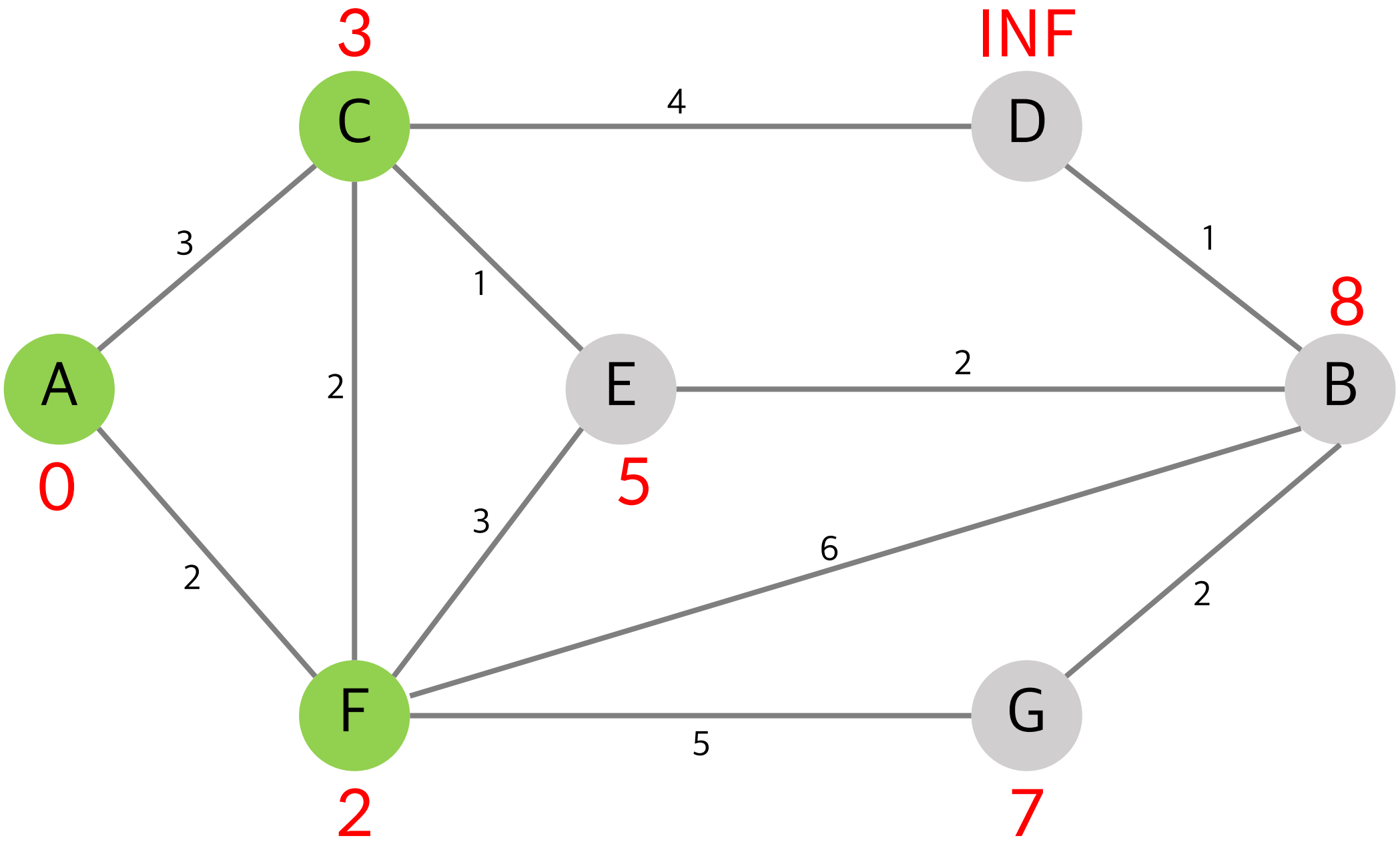


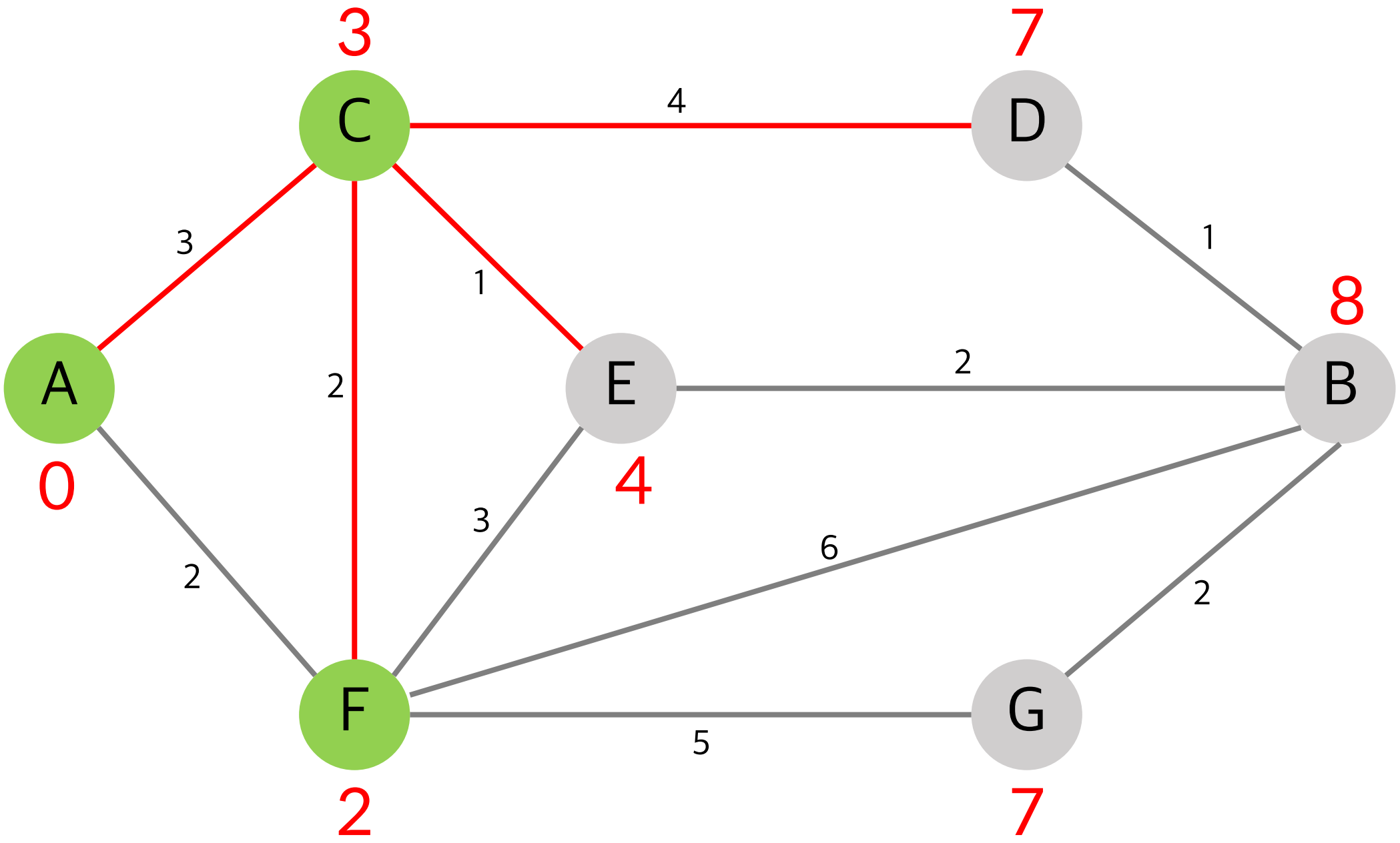
종료.

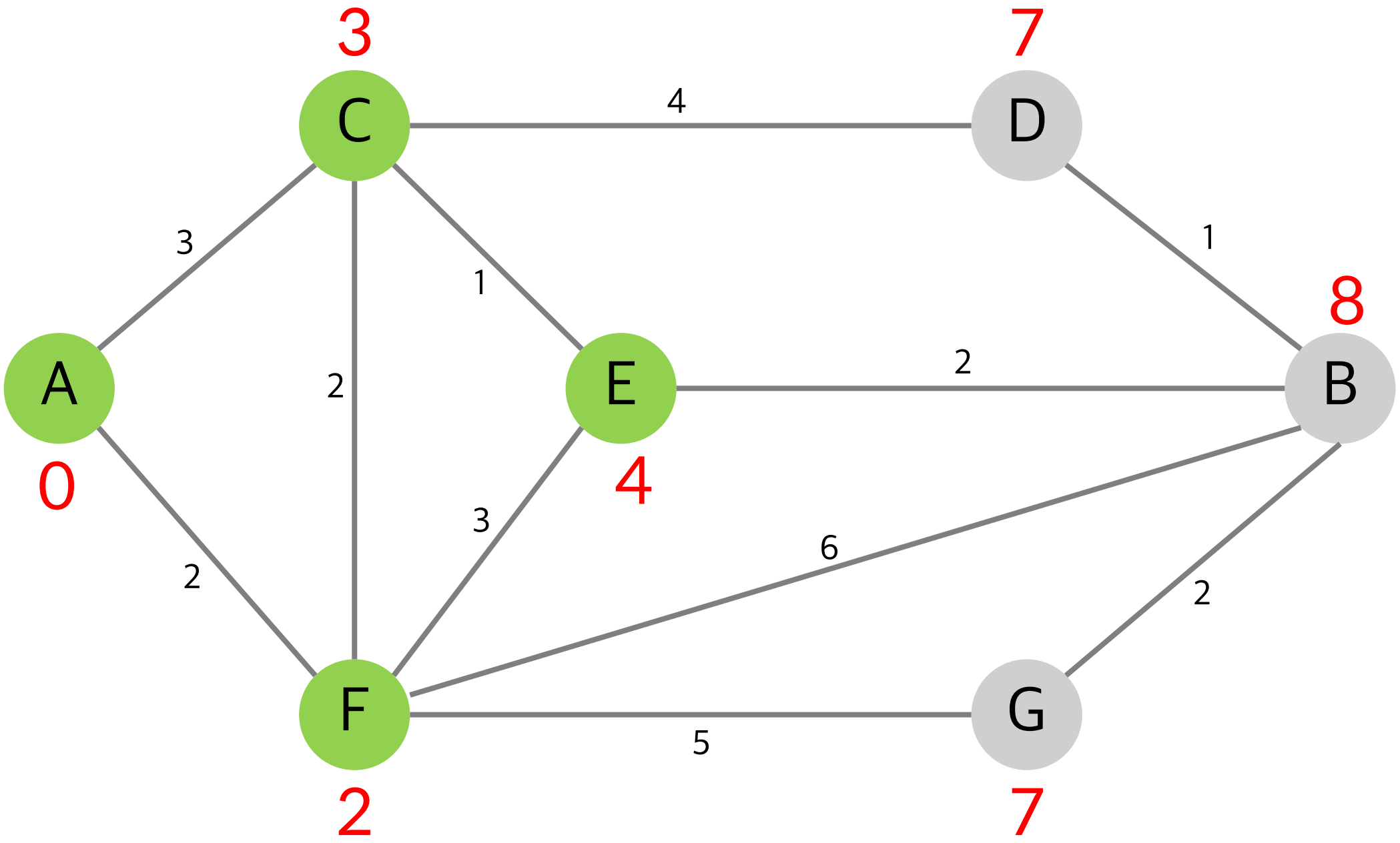

INF
INF
C
4
D
3
1
1
INF
A
2
E
2
B
0
INF
3
6
2
2
F
G
5
INF
INF

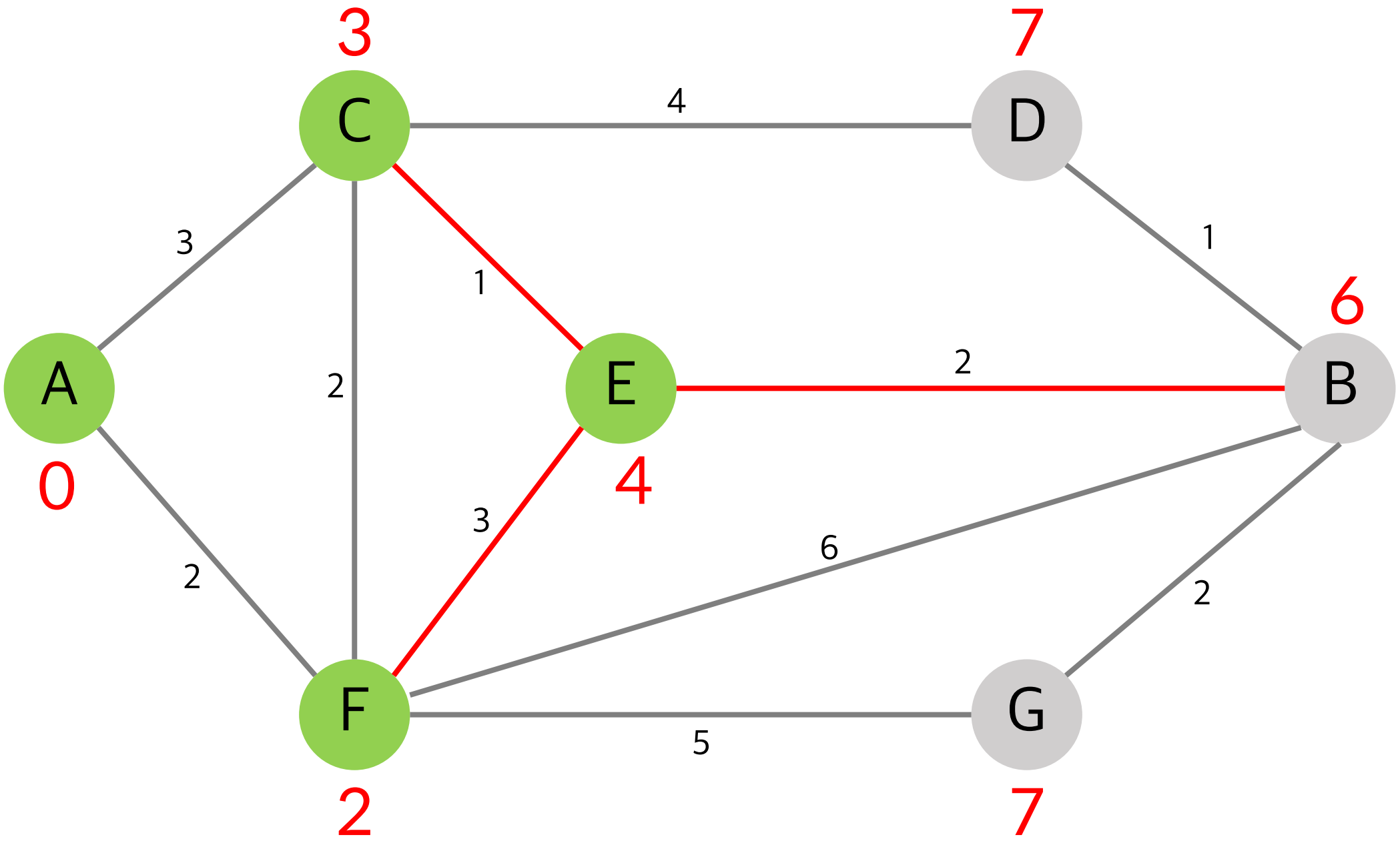

3
INF
C
4
D
3
1
1
INF
A
2
E
2
B
0
INF
3
6
2
2
F
G
5
2
INF

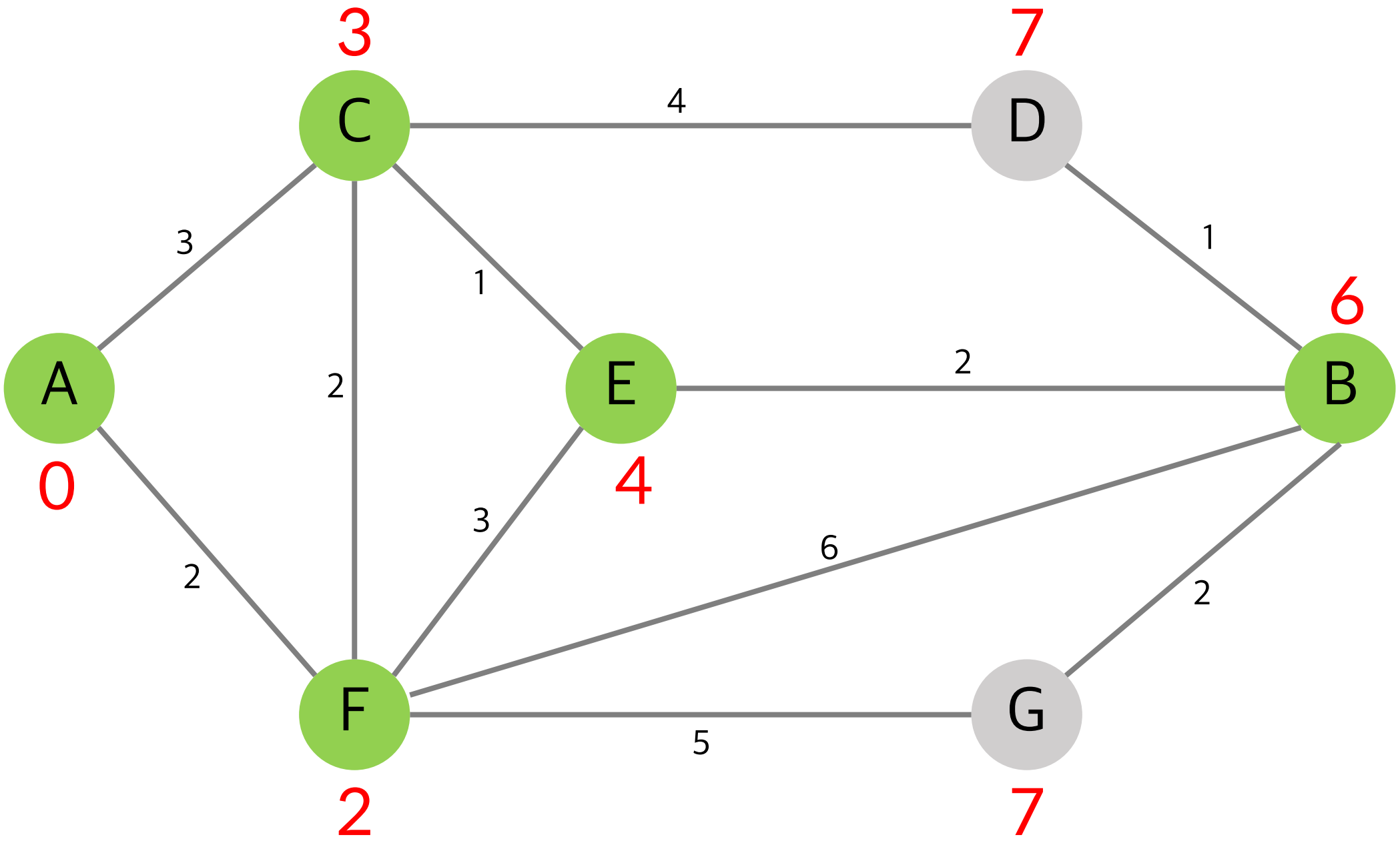

3
INF
C
4
D
3
1
1
8
A
2
E
2
B
0
5
3
6
2
2
F
G
5
2
7

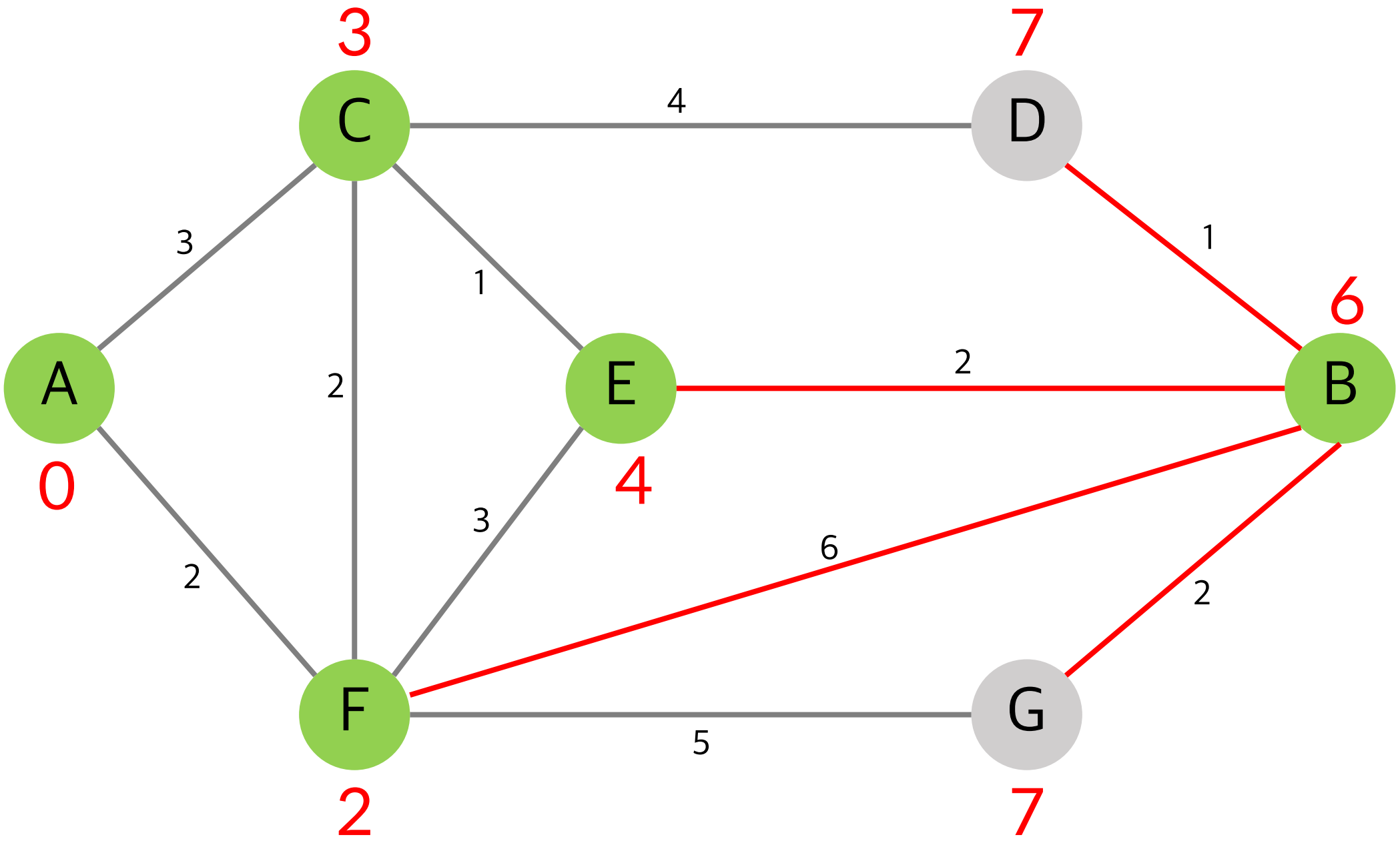

3
INF
C
4
D
3
1
1
8
A
2
E
2
B
0
5
3
6
2
2
F
G
5
2
7

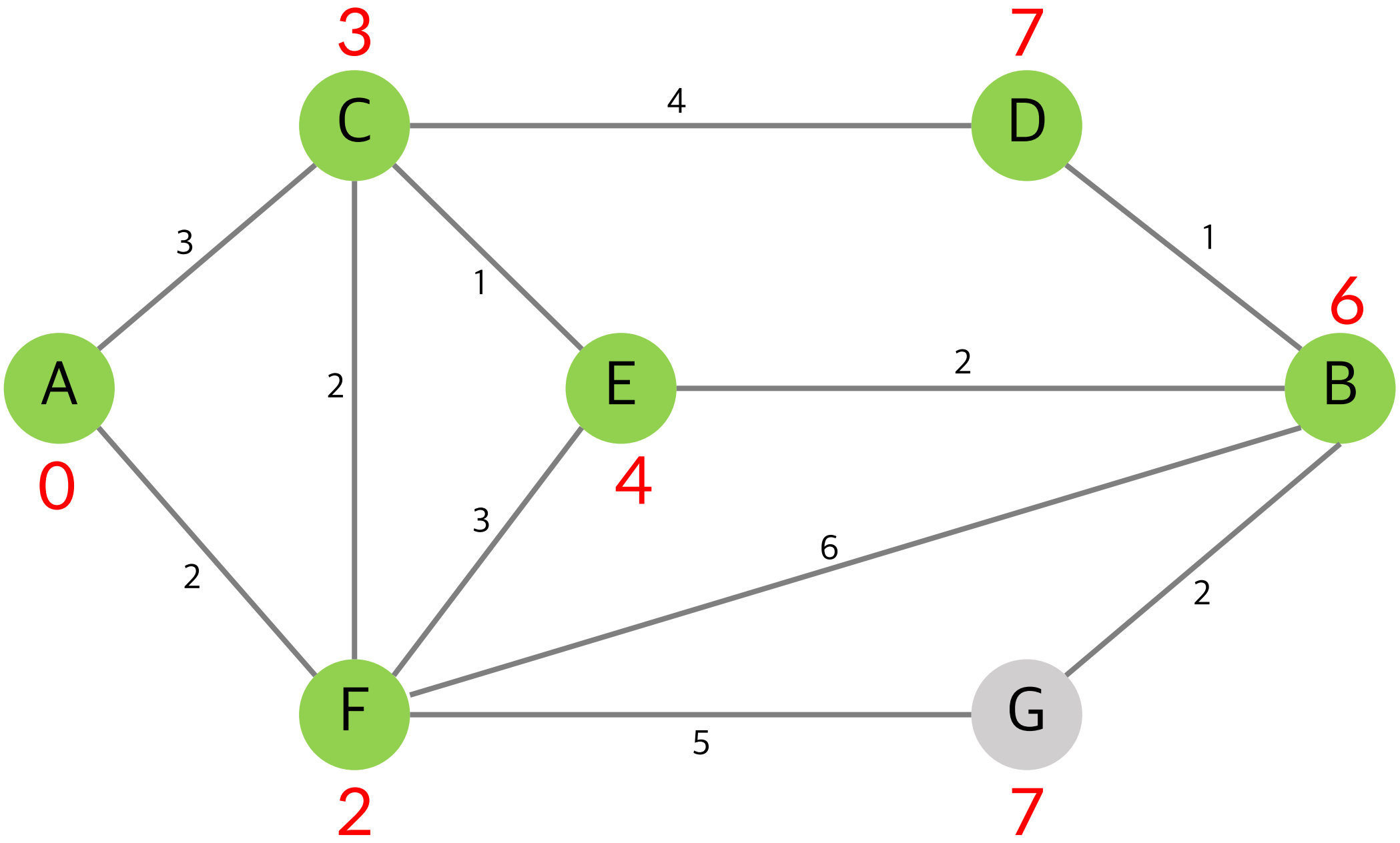

3
7
C
4
D
3
1
1
8
A
2
E
2
B
0
4
3
6
2
2
F
G
5
2
7

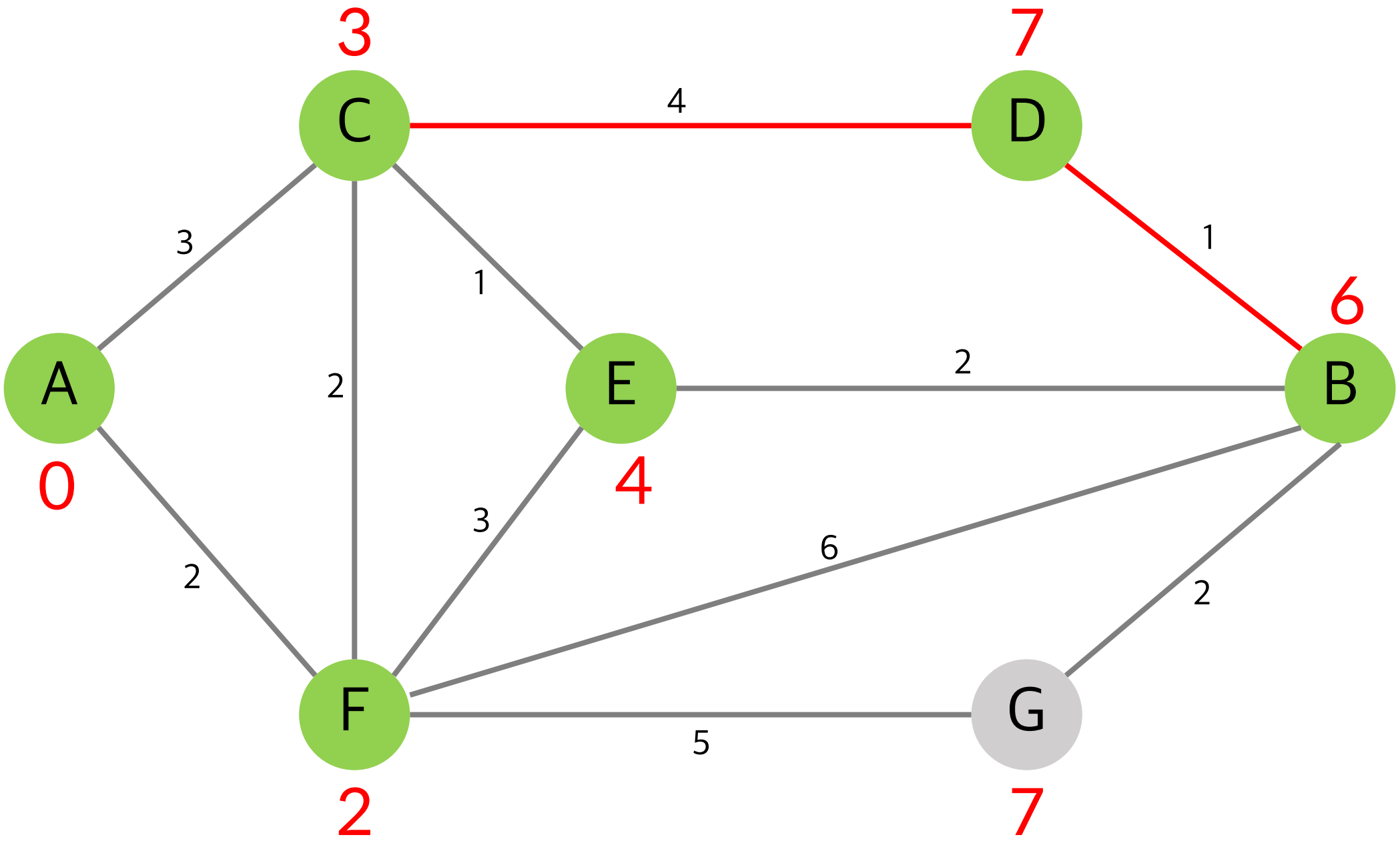

3
7
C
4
D
3
1
1
8
A
2
E
2
B
0
4
3
6
2
2
F
G
5
2
7

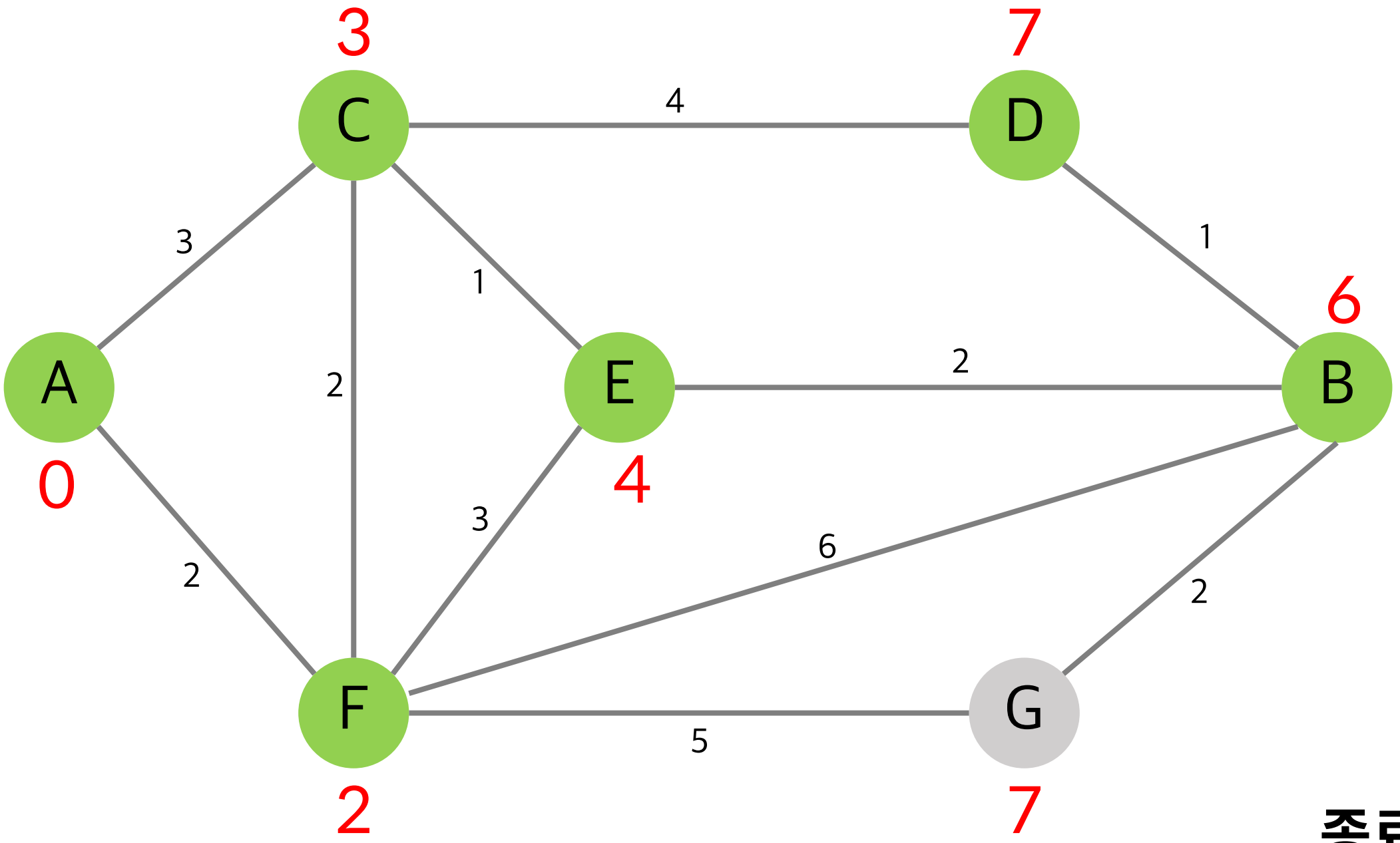

3
7
C
4
D
3
1
1
6
A
2
E
2
B
0
4
3
6
2
2
F
G
5
2
7


3
7
C
4
D
3
1
1
6
A
2
E
2
B
0
4
3
6
2
2
F
5
G
2
7

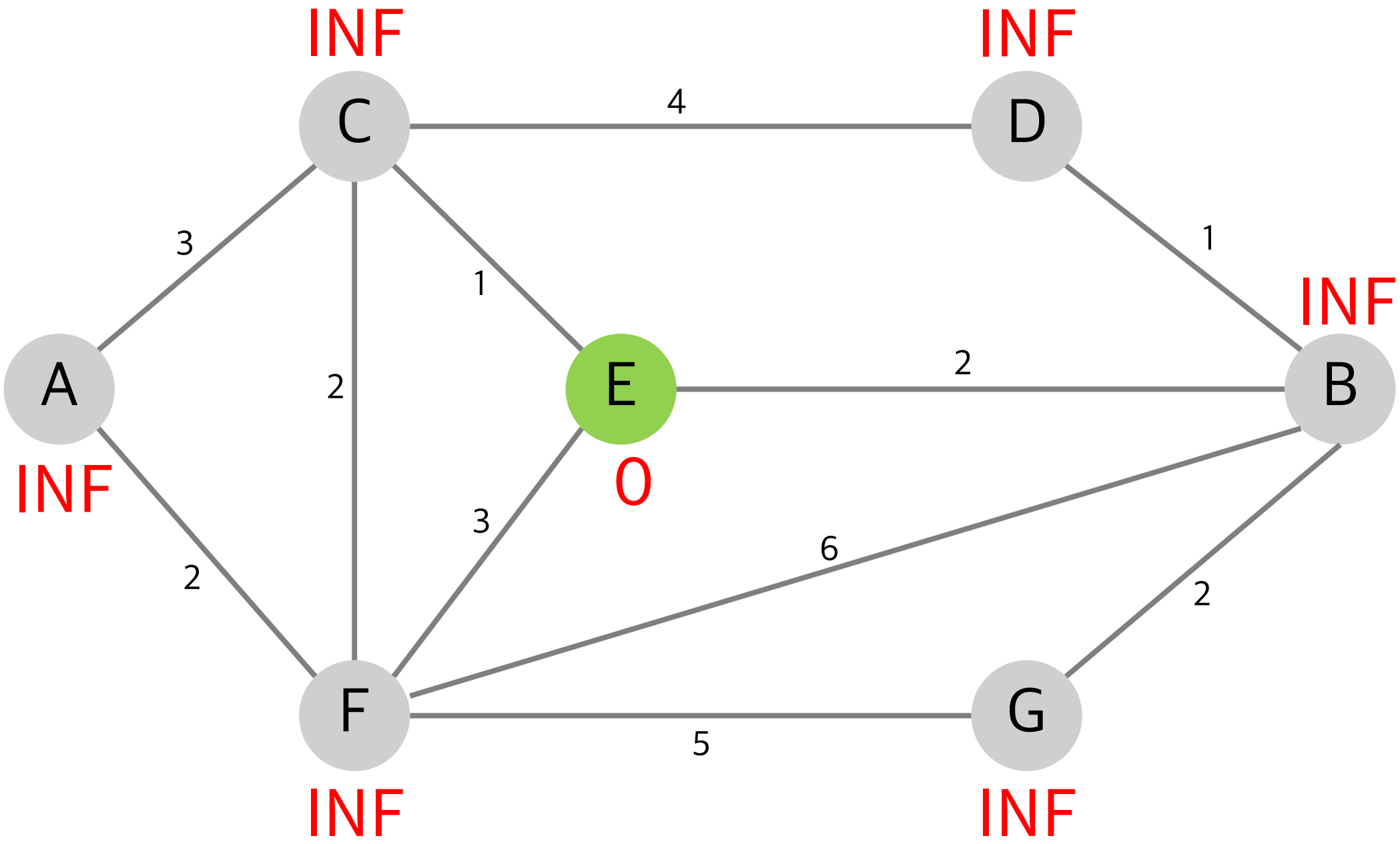

3
7
C
4
D
3
1
1
6
A
2
E
2
B
0
4
3
6
2
2
F
5
G
2
7

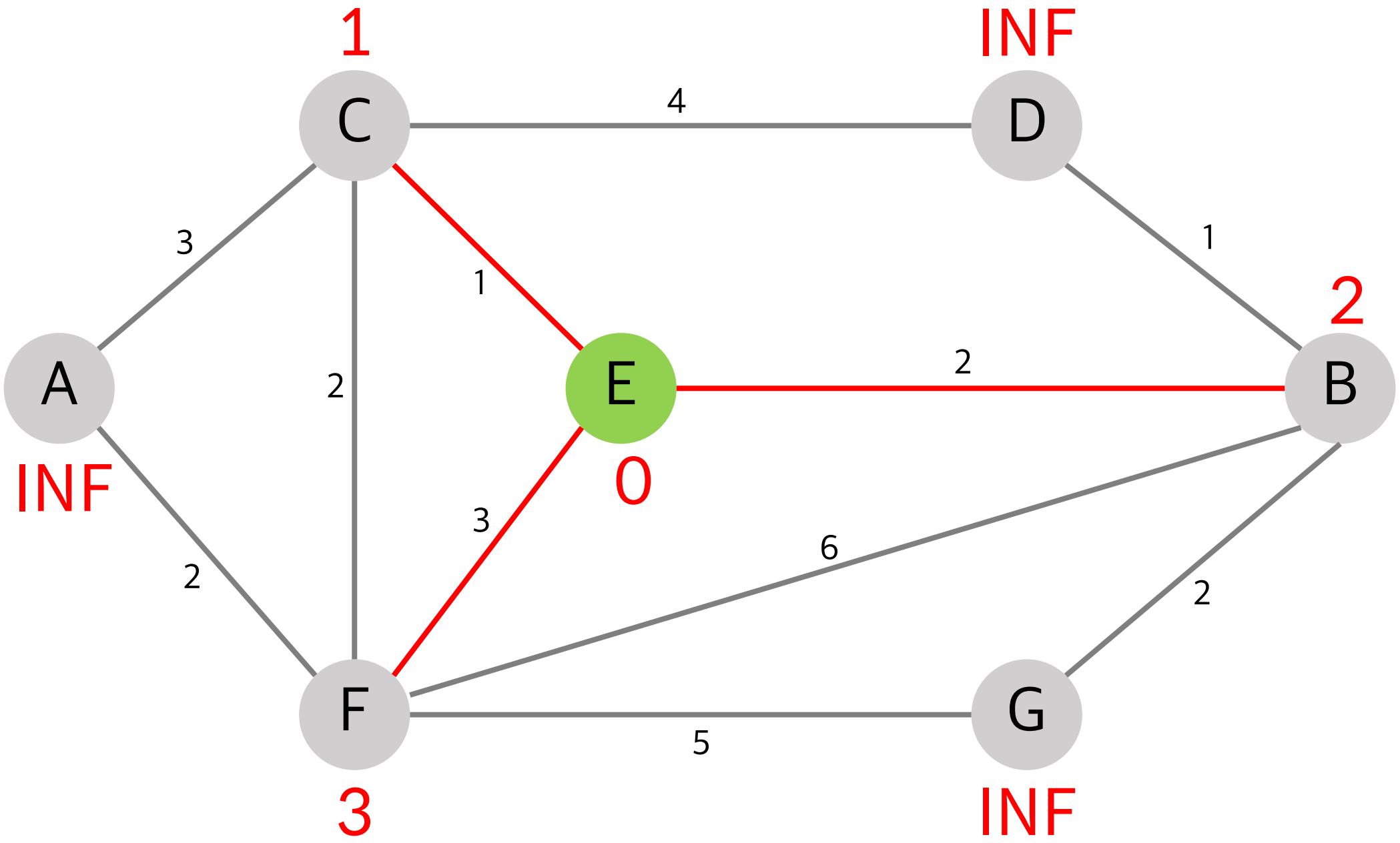

3
7
C
4
D
3
1
1
6
A
2
E
2
B
0
4
3
6
2
2
F
G
5
2
7

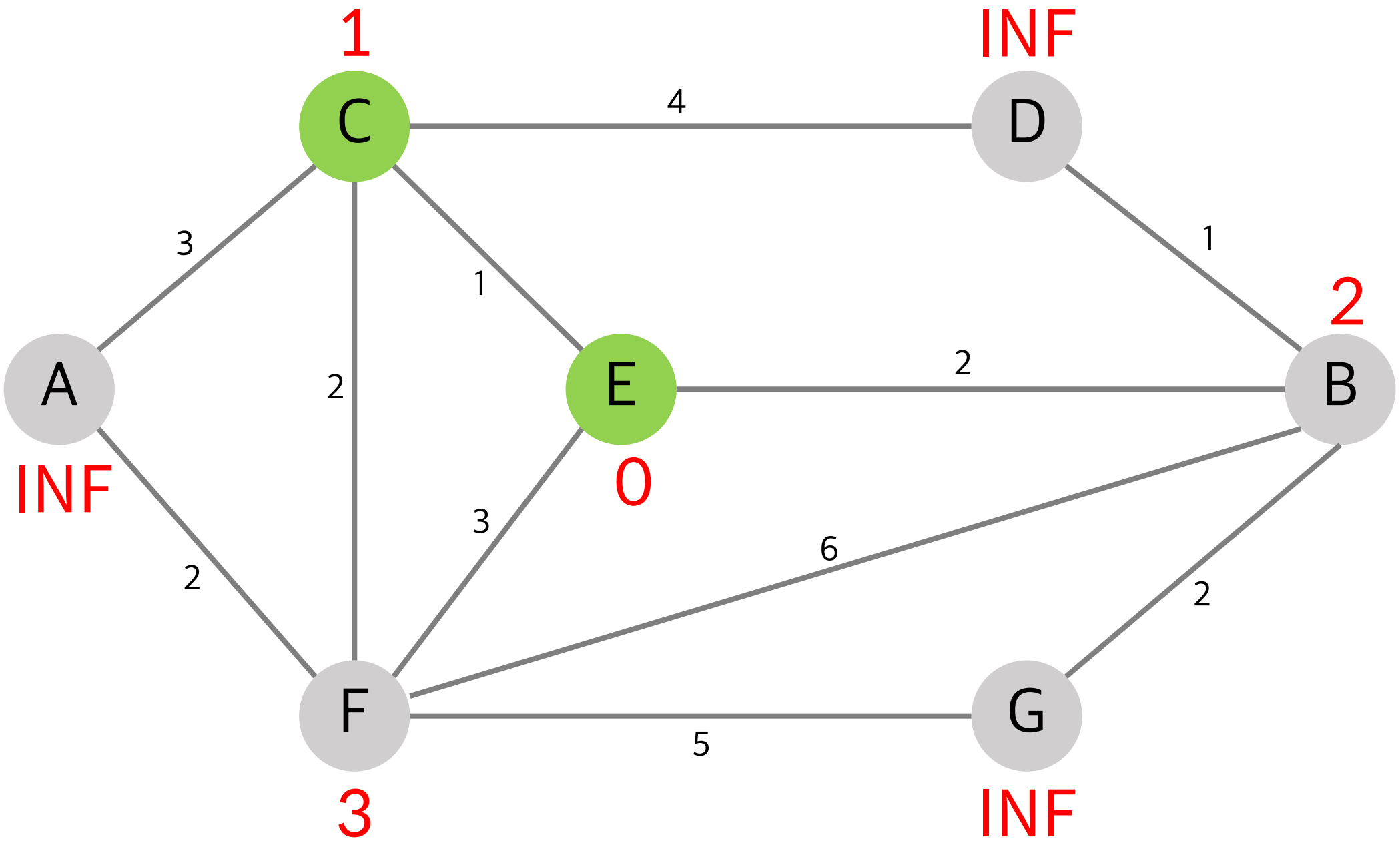

3
7
C
4
D
3
1
1
6
A
2
E
2
B
0
4
3
6
2
2
F
5
G
2
7

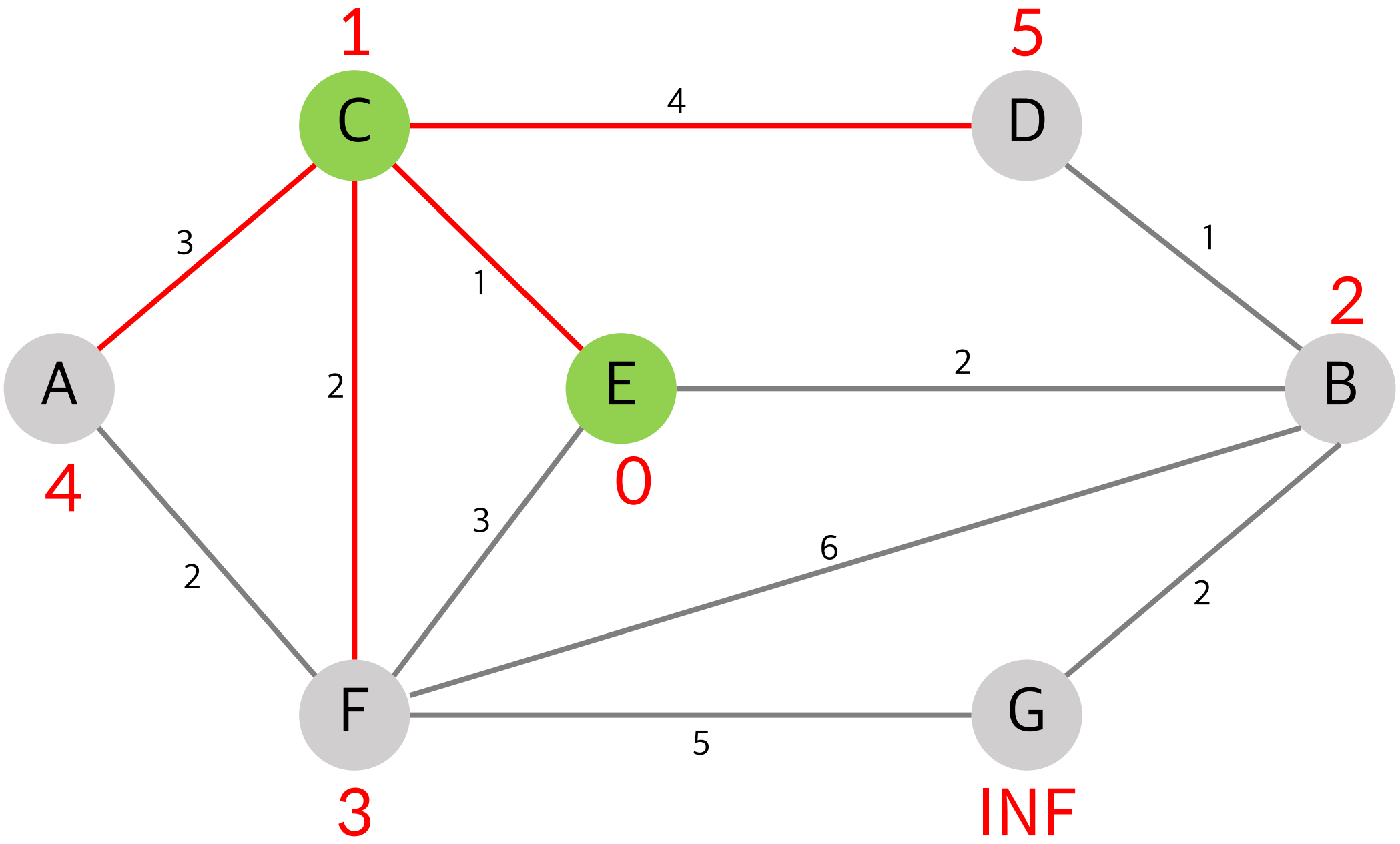


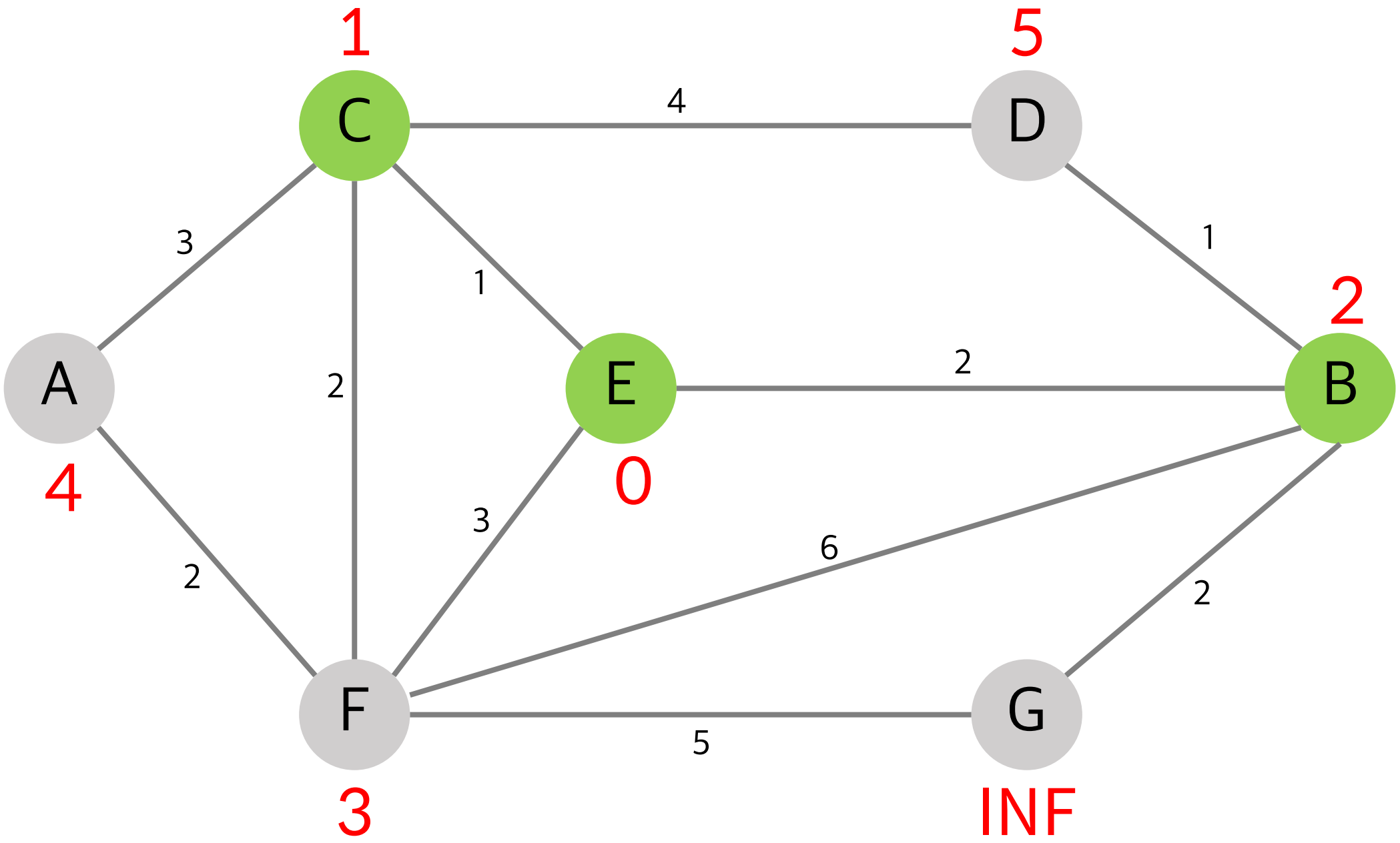
종료.

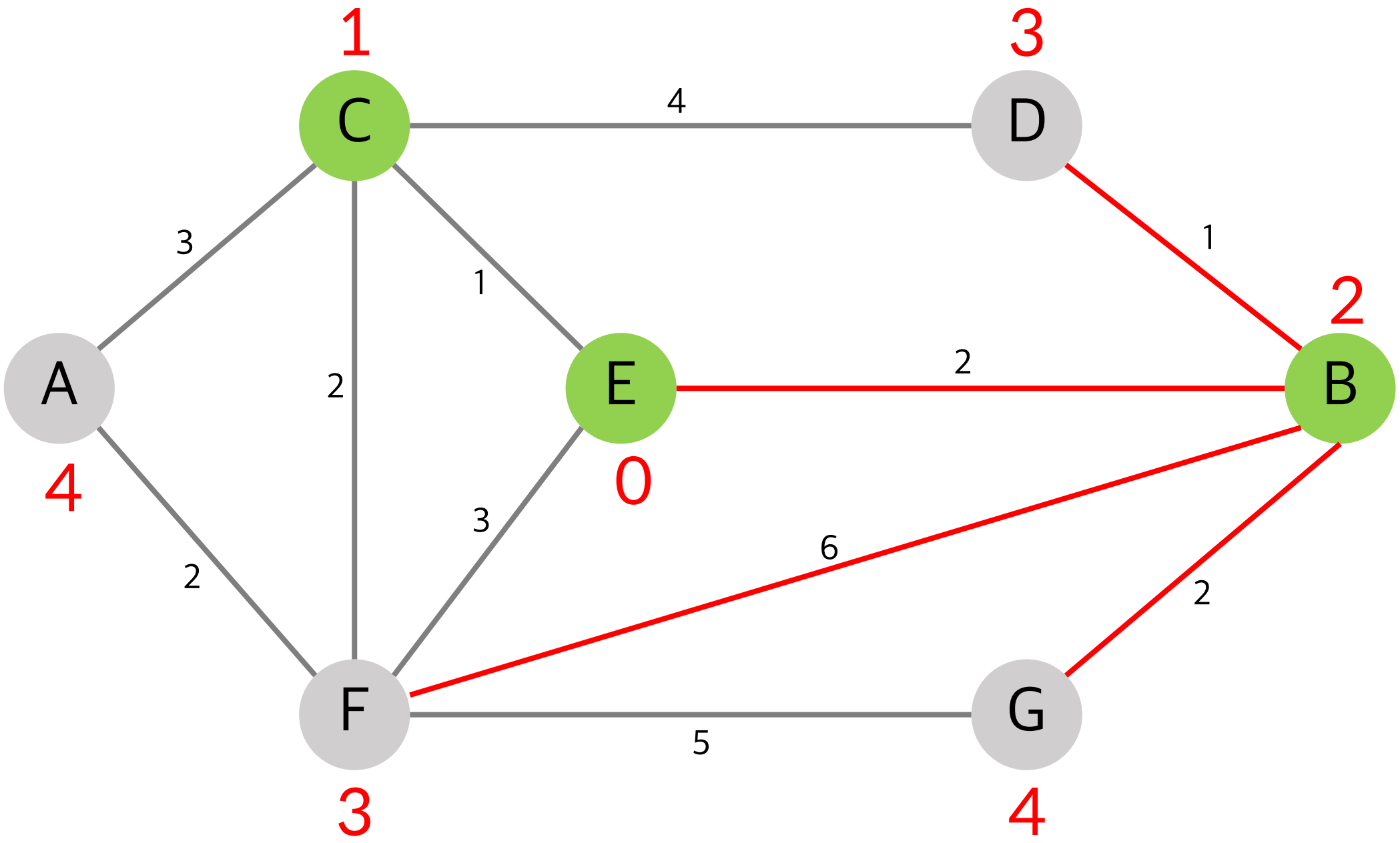

INF
INF
C
4
D
3
1
1
INF
A
2
E
2
B
INF
0
3
6
2
2
F
G
5
INF
INF

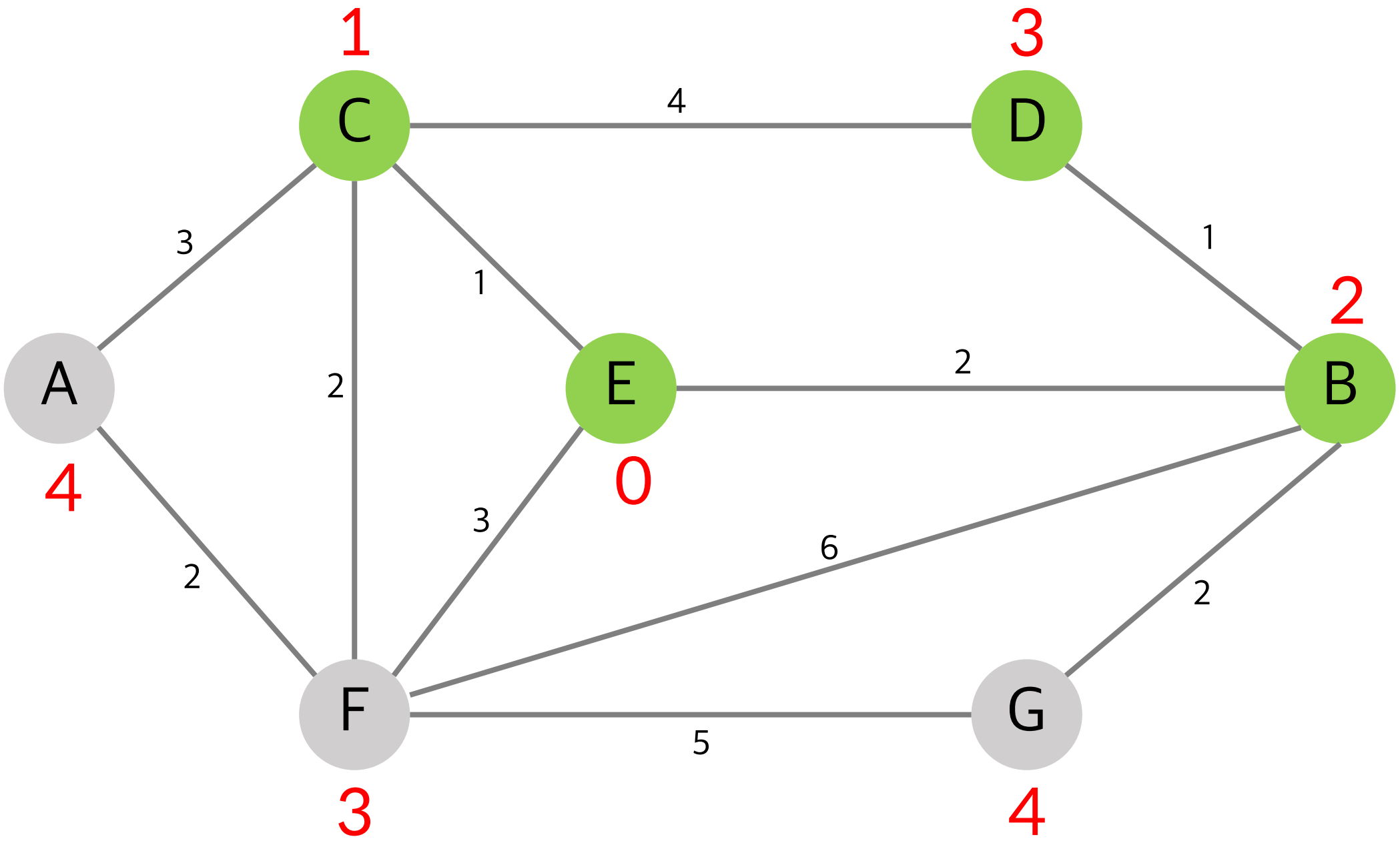

1
INF
C
4
D
3
1
1
2
A
2
E
2
B
INF
0
3
6
2
2
F
G
5
3
INF

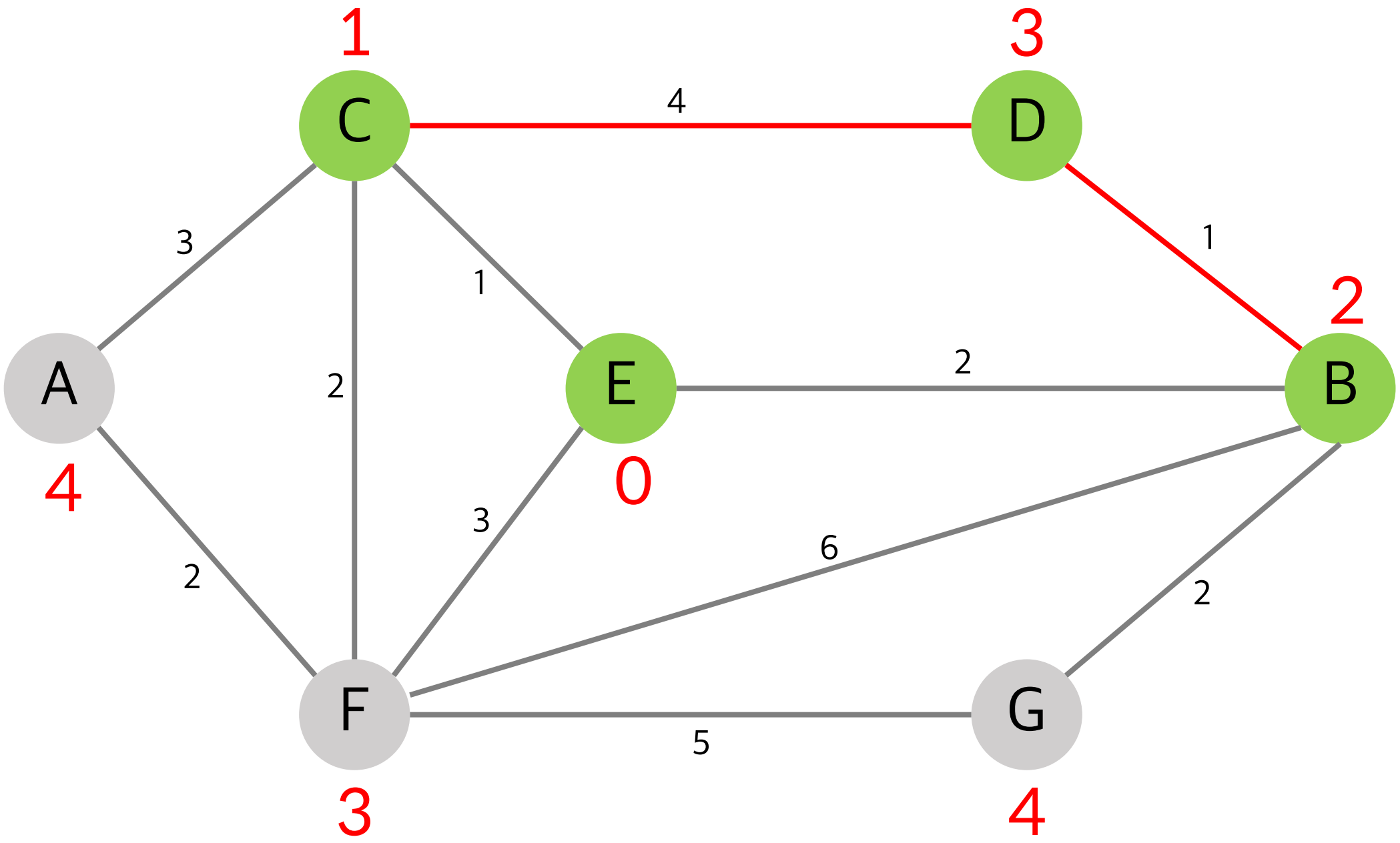

1
INF
C
4
D
3
1
1
2
A
2
E
2
B
INF
0
3
6
2
2
F
5
G
3
INF

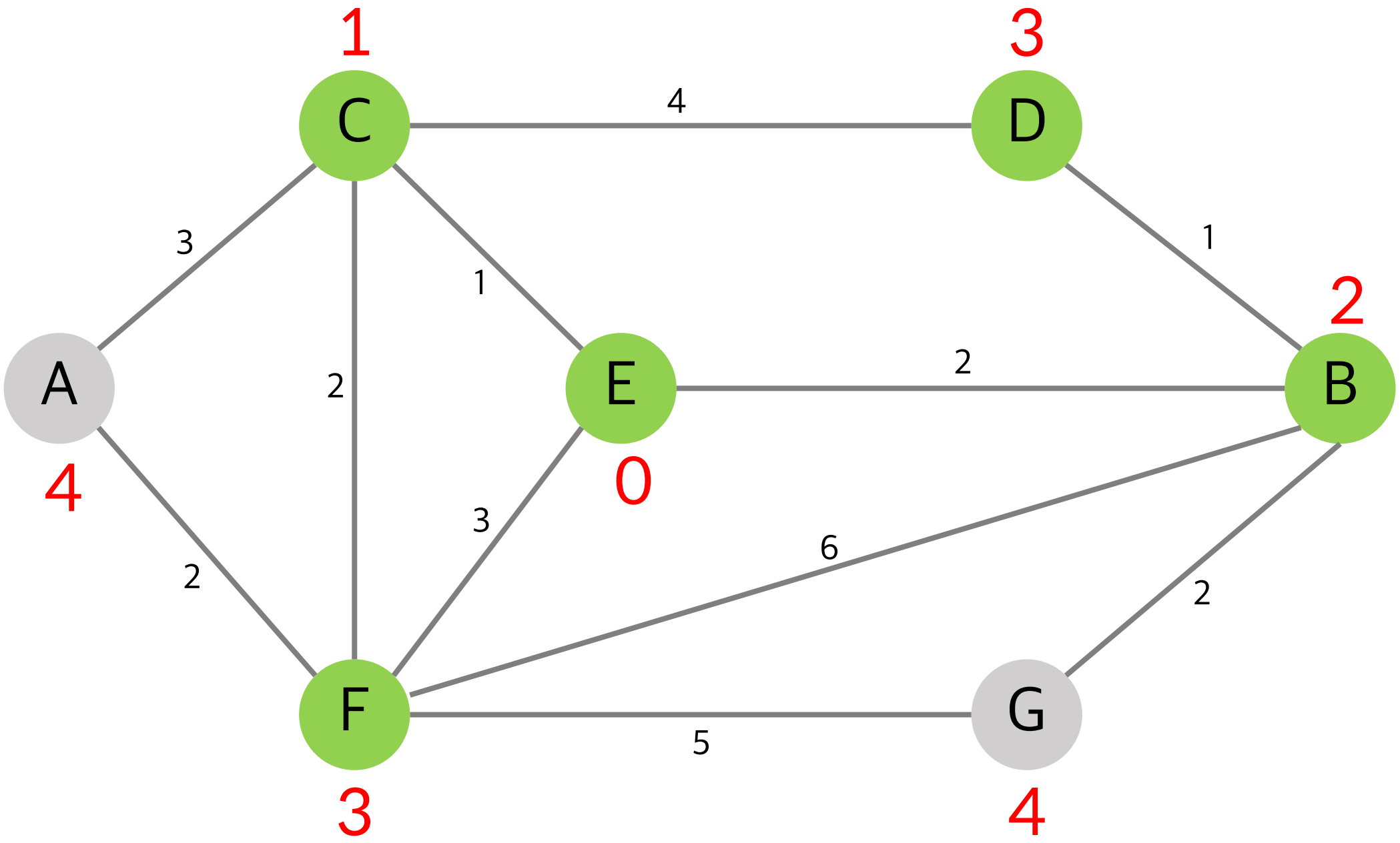

1
5
C
4
D
3
1
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
INF

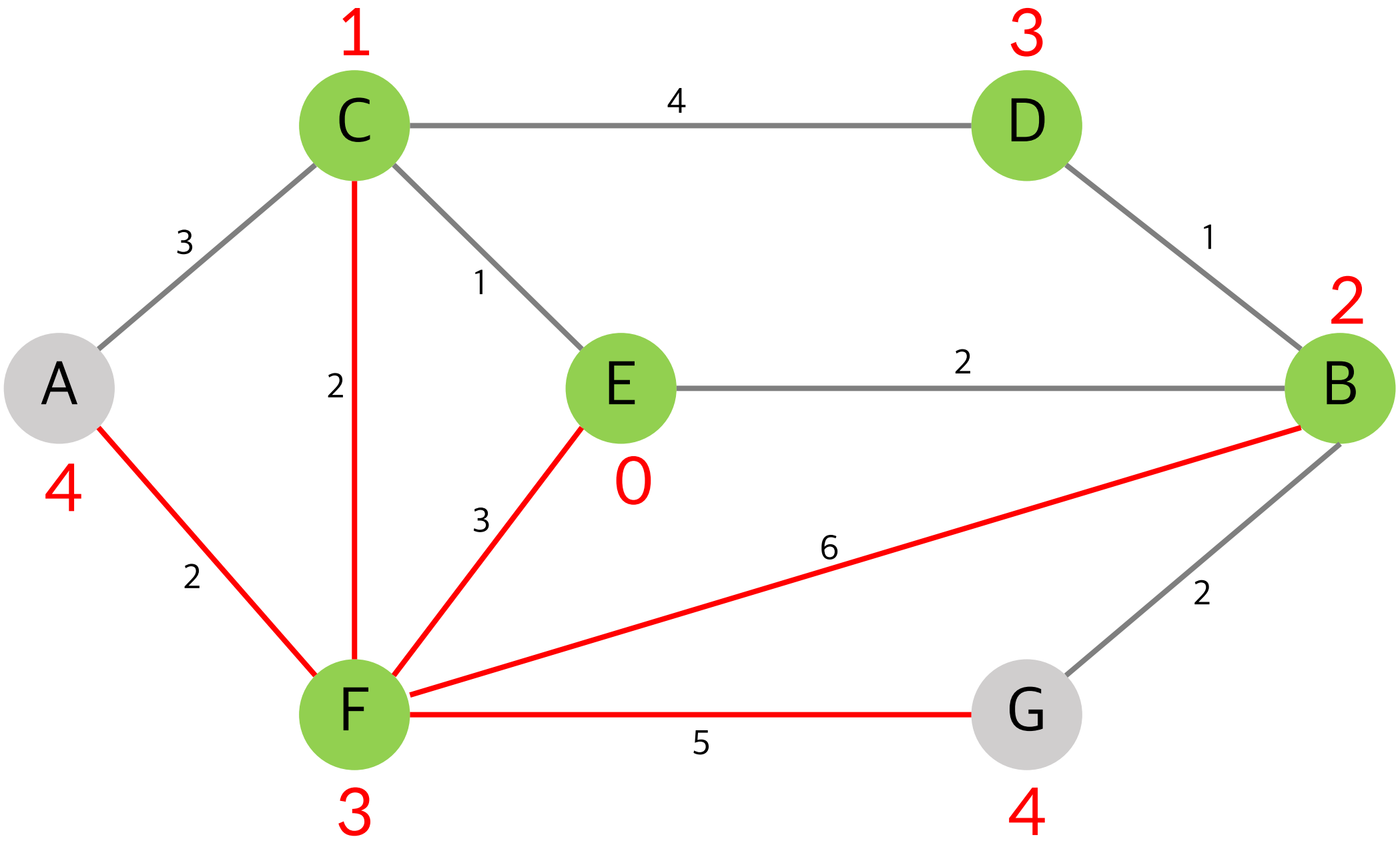

1
5
C
4
D
3
1
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
INF

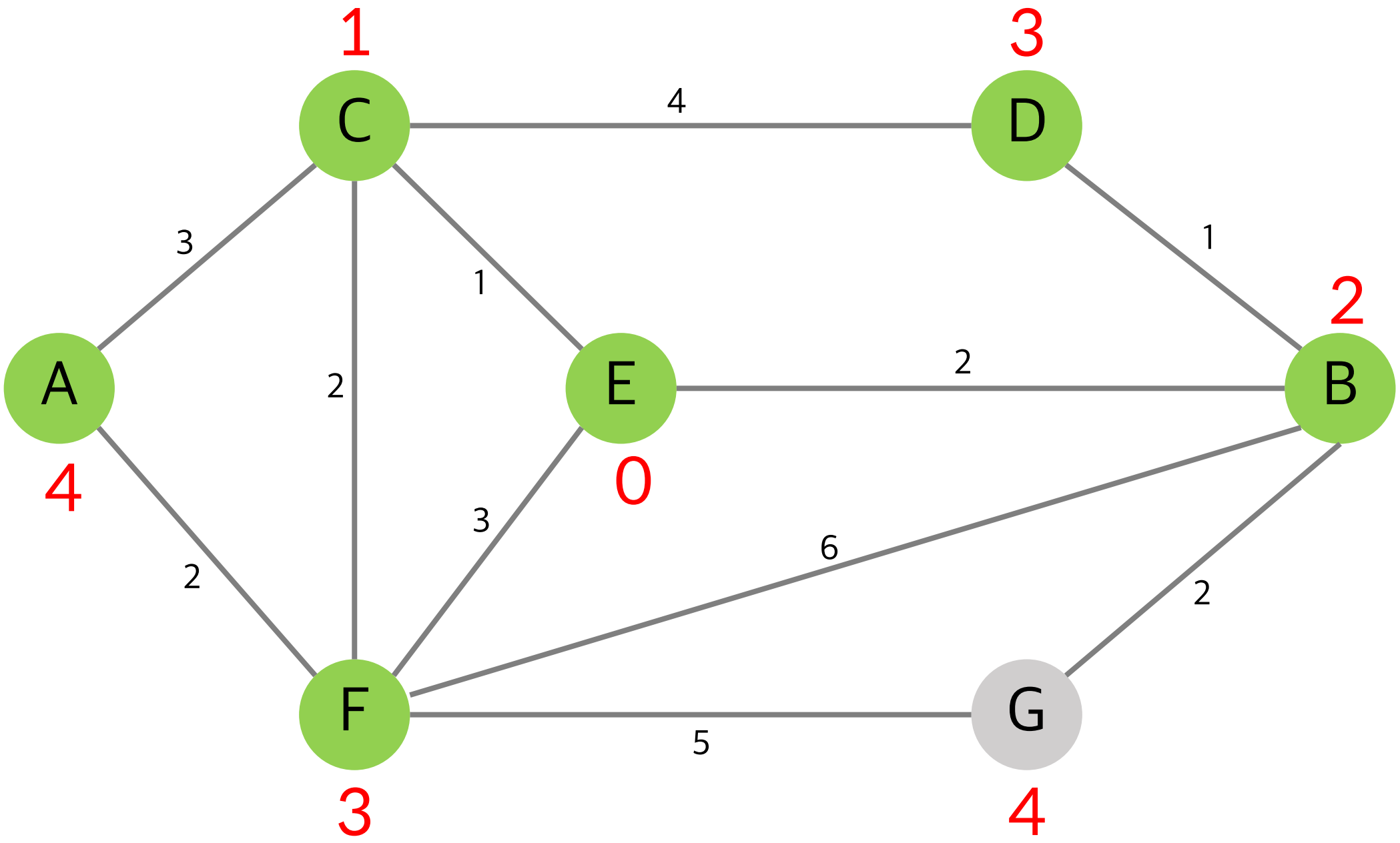

1
C
4
3
D
3
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
4

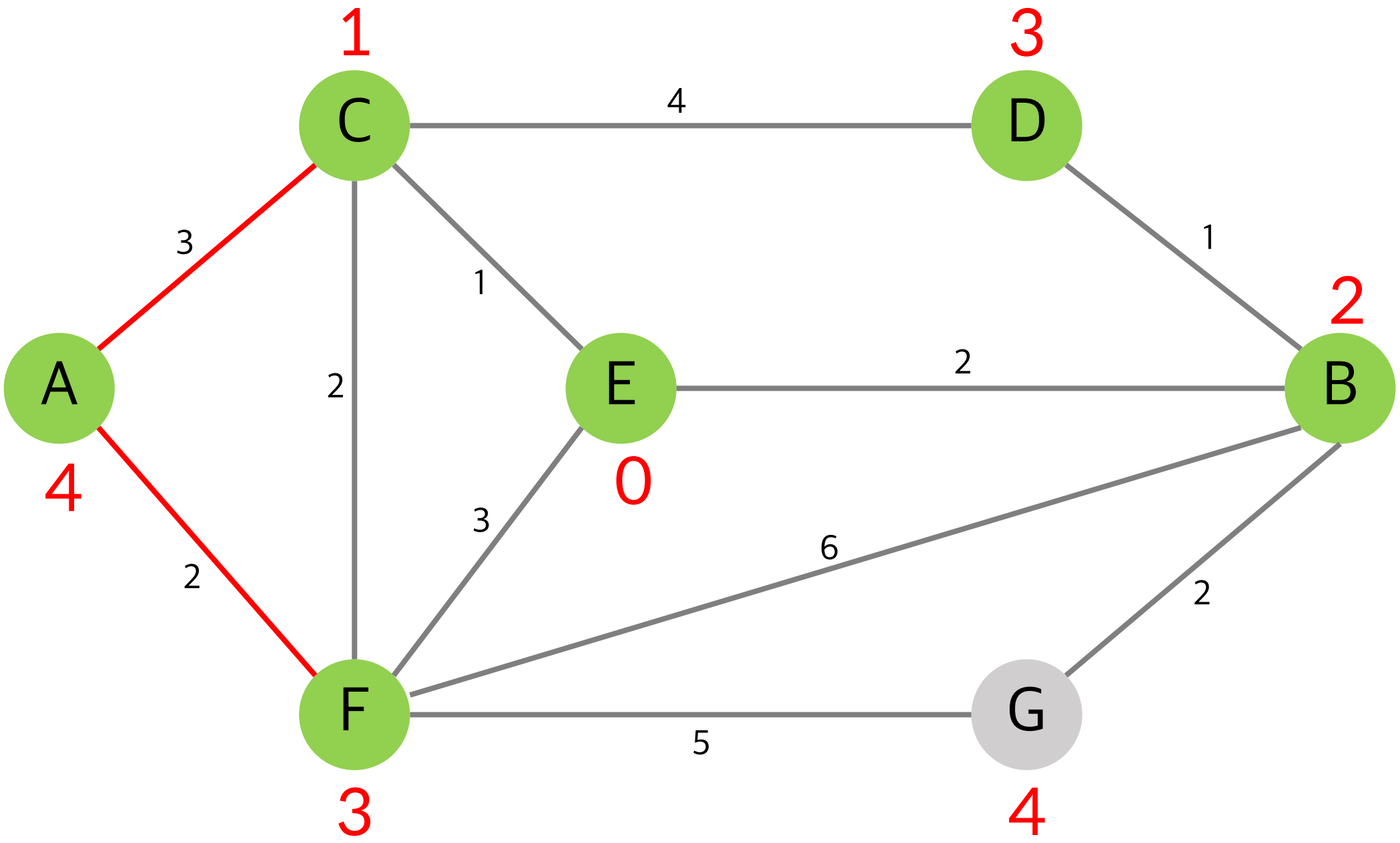

1
C
4
3
D
3
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
4

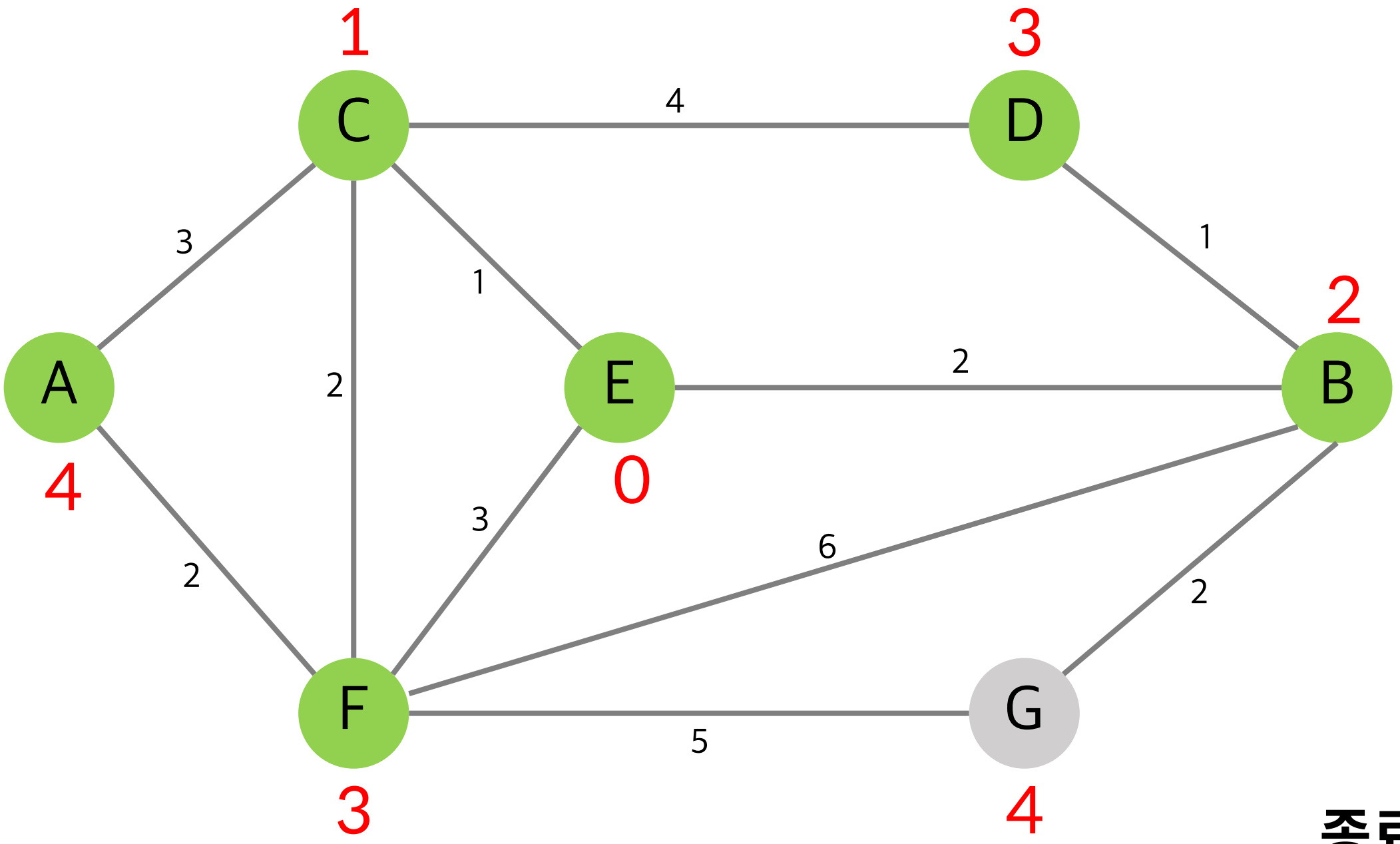

1
C
4
3
D
3
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
4


1
C
4
3
D
3
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
4


1
C
4
3
D
3
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
4


1
C
4
D
3
3
1
1
2
A
2
E
2
B
4
0
3
6
2
2
F
5
G
3
4


1
3
C
4
D
3
1
1
2
A
2
E
2
B
4
0
3
6
2
2
F
G
5
3
4



종료.

Dijkstra Algorithm

👉

- 총 O(V) 번에 걸쳐 최단 거리가 최소인 노드 선택
- 따라서, 시간 복잡도 O(V^2)
- 전체 노드의 개수가 5000개 이하라면 해결 가능
- 10000개가 넘으면?

https://gist.github.com/euije/84b5155422e90f82ace14bf37787d34b

https://github.com/ndb796/python-for-coding-test/tree/master/9

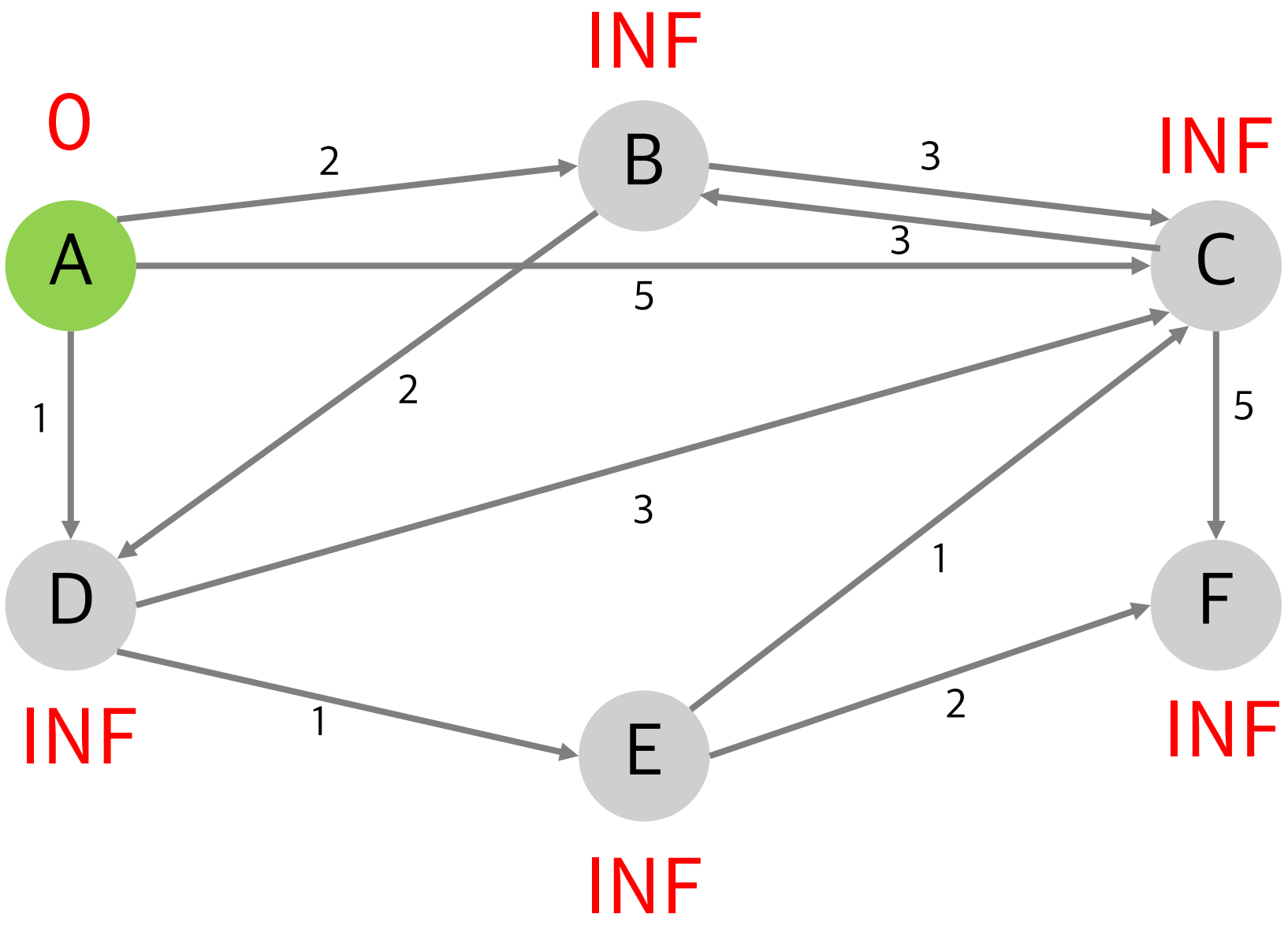
```cpp
#include <iostream>

#define INF 1e9

using namespace std;

const int n = 6;

int cost[n][n] ={
  {0,2,5,1,INF,INF},
  {2,0,3,2,INF,INF},
  {5,3,0,3,1,5},
  {1,2,3,0,1,INF},
  {INF,INF,1,1,0,2},
  {INF,INF,5,INF,2,0}
};

bool found[n];
int distance[n];

int getSmallIndex(){
  int min = INF;
  int index;
  for(int i = 0; i < n; i++){
    if(distance[i] < min && !found[i]){
      min = distance[i];
      index = i;
    }
  }
  return index;
}

void dijkstra(int start){
  int i, u, w;

  for(i = 0; i < n; i++){
    found[i] = false;
    distance[i] = cost[start][i];
  }

  found[start] = true;
  distance[start] = 0;

  for(i = 0; i < n-2; i++){
    u = getSmallIndex();
    found[u] = true;

    for(w = 0; w < n; w++){
      if(!found[w]){
        if(distance[u] + cost[u][w] < distance[w]) distance[w] = distance[u] + cost[u][w];
      }
    }
  }
}
```

Dijkstra Algorithm - Modified

👉

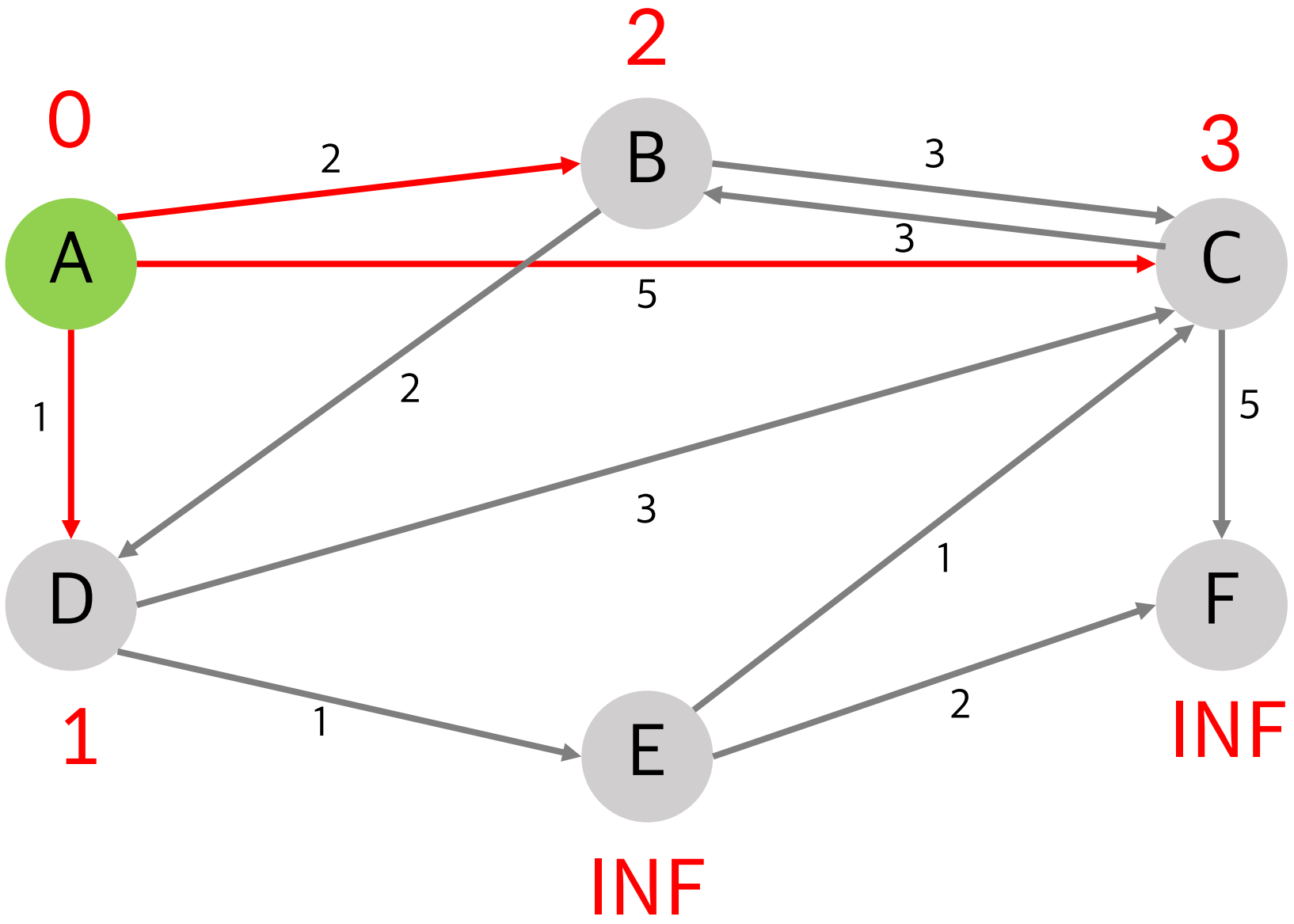
- 단계마다 방문하지 않는 노드 중에서 최단 거리가 가장 짧은 노드를 선택하기 위해 **우선 순위 큐**를 활용

- 다익스트라 알고리즘이 동작하는 기본 원리는 동일



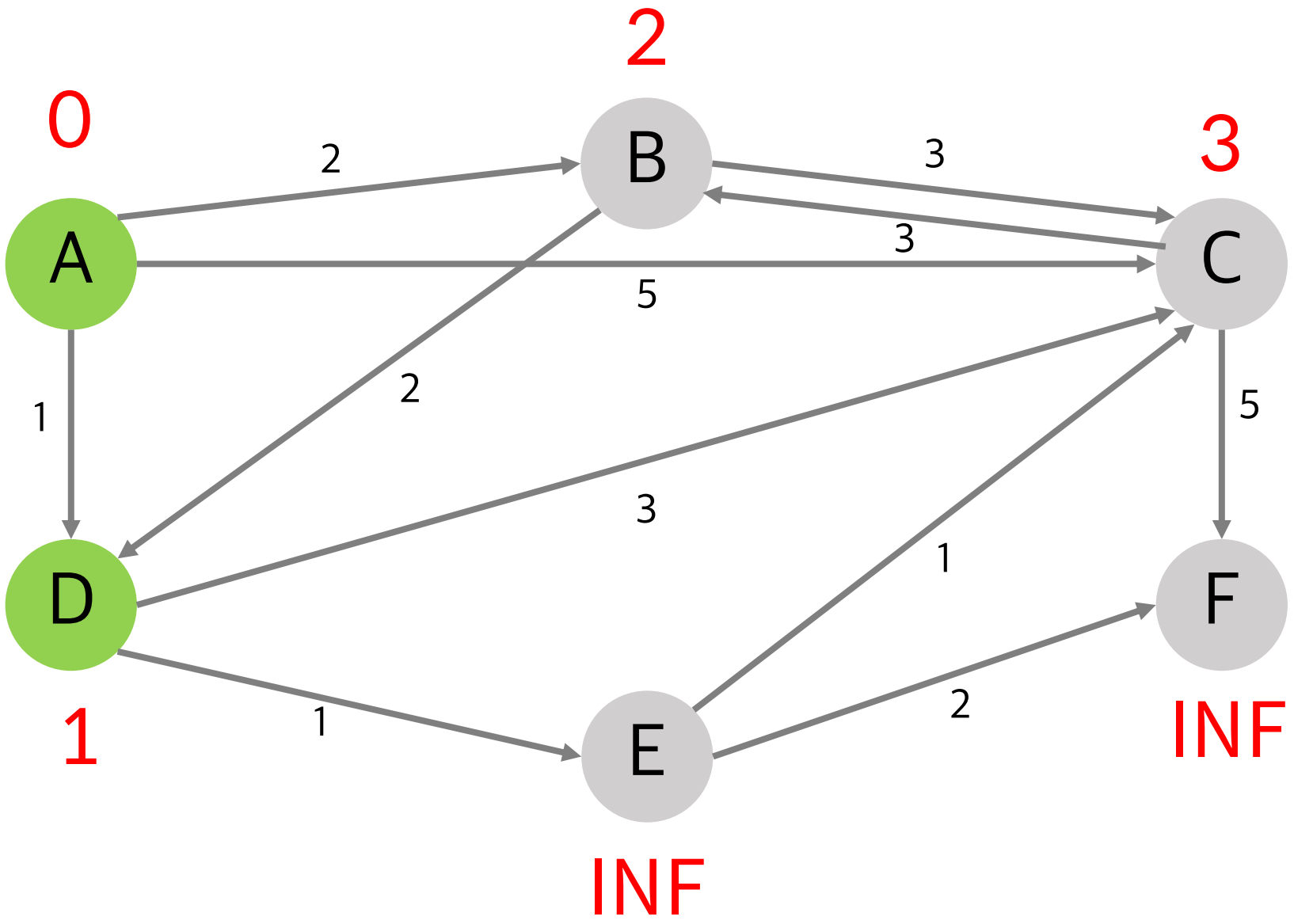
우선순위 큐
(노드, W)

(A, 0)



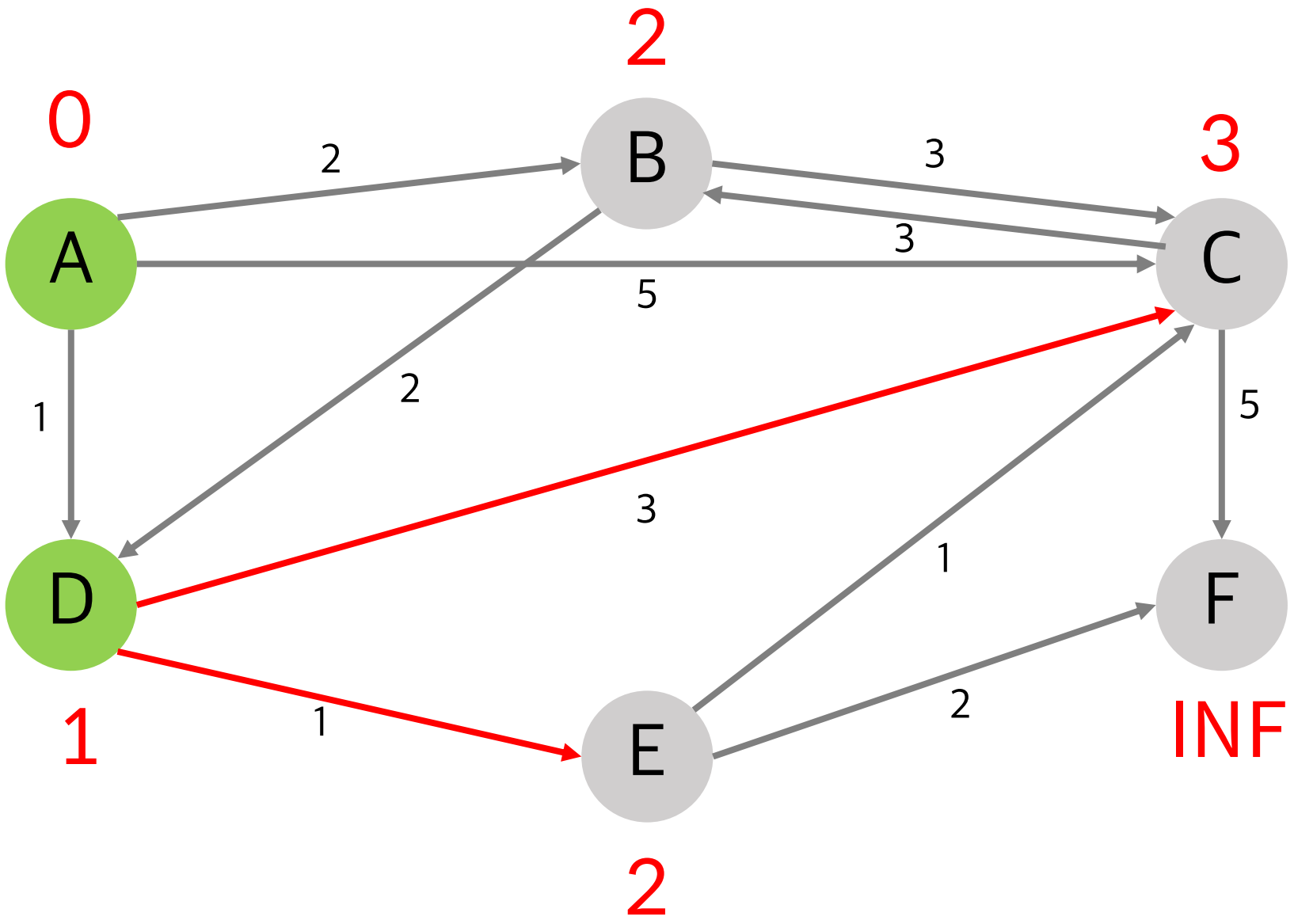
우선순위 큐 (노드, W)

(D, 1)
(B, 2)
(C, 5)



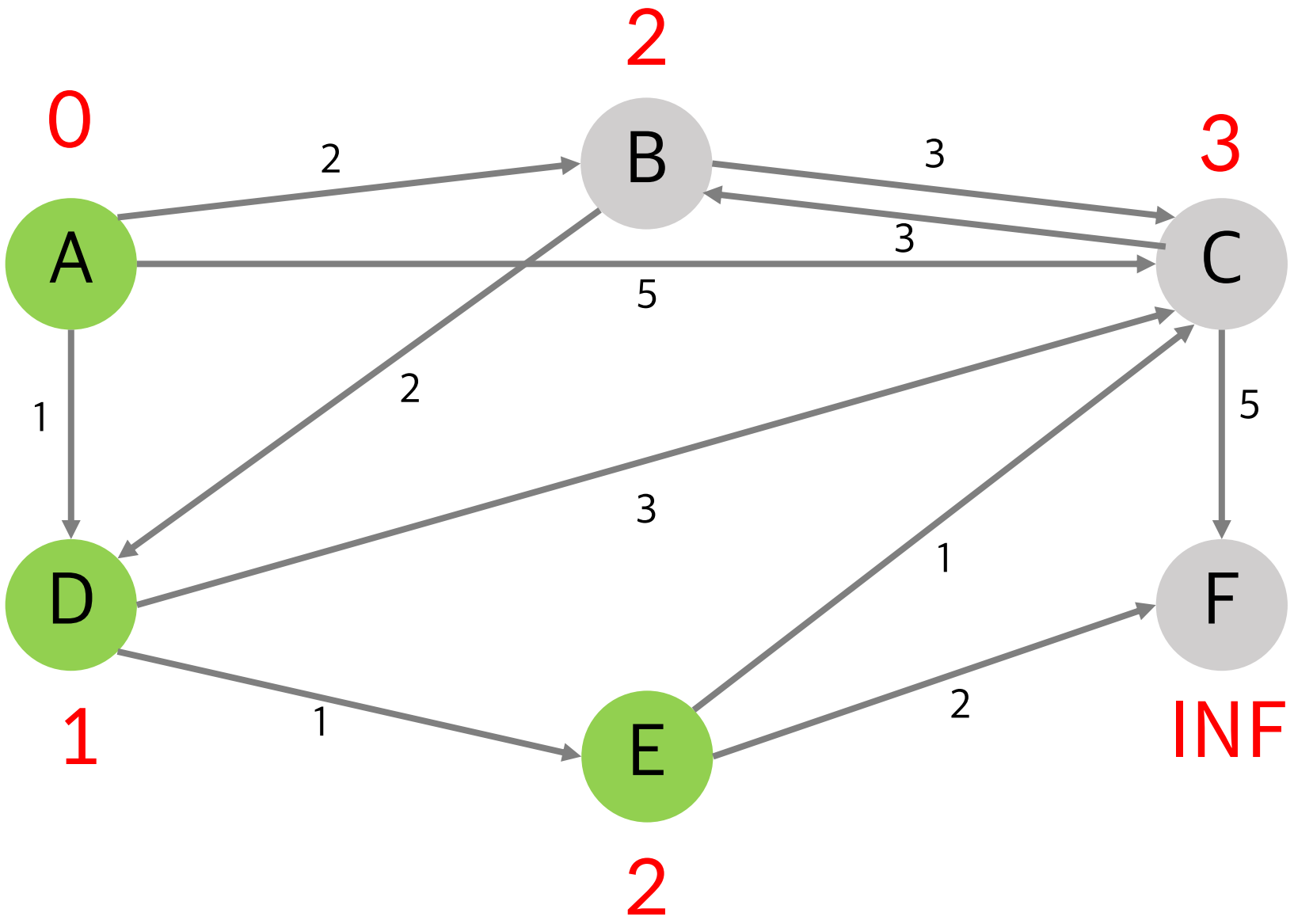
우선순위 큐
(노드, W)

(B, 2)
(C, 5)



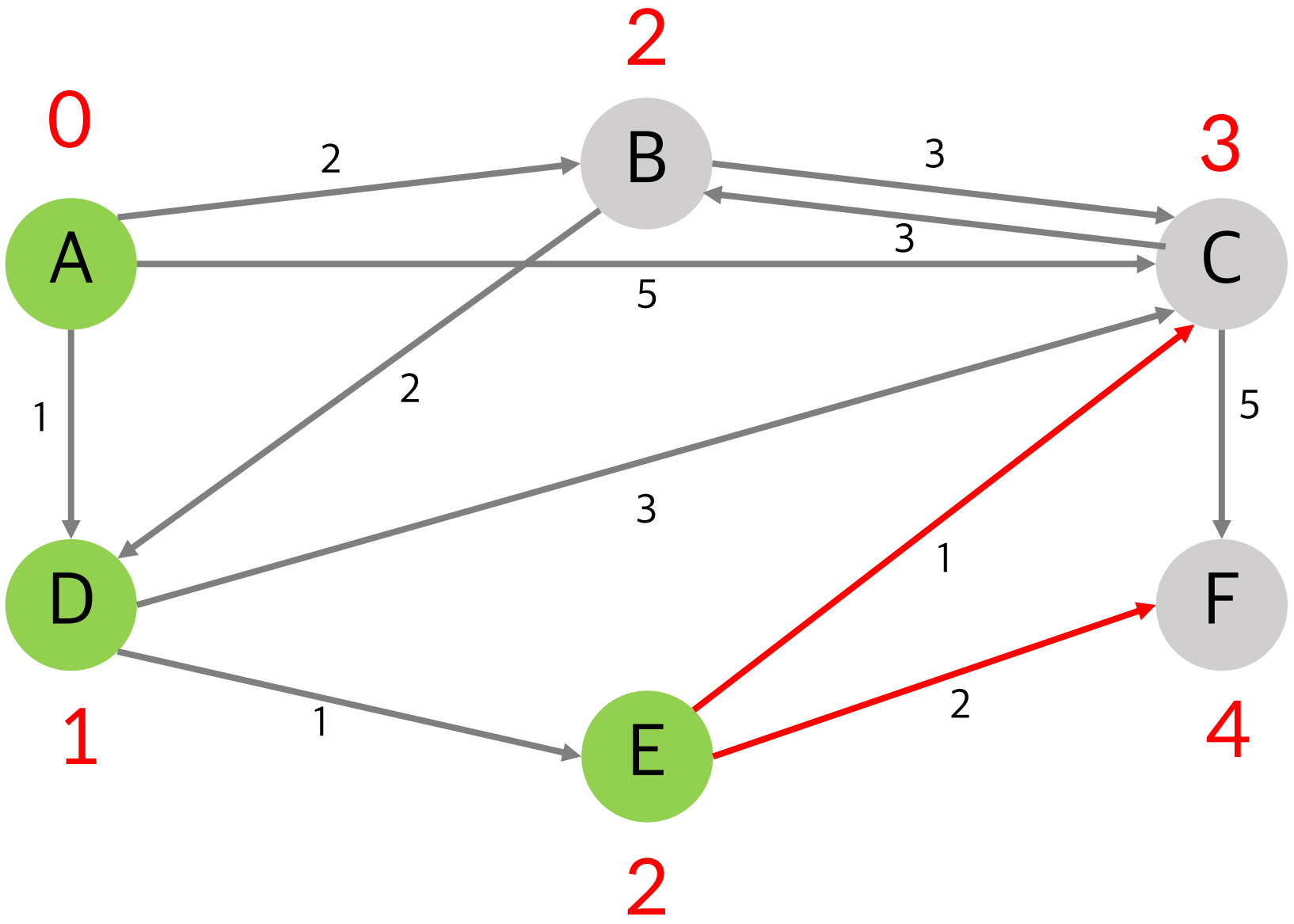
우선순위 큐
(노드, W)

(E, 1)
(B, 2)
(C, 5)



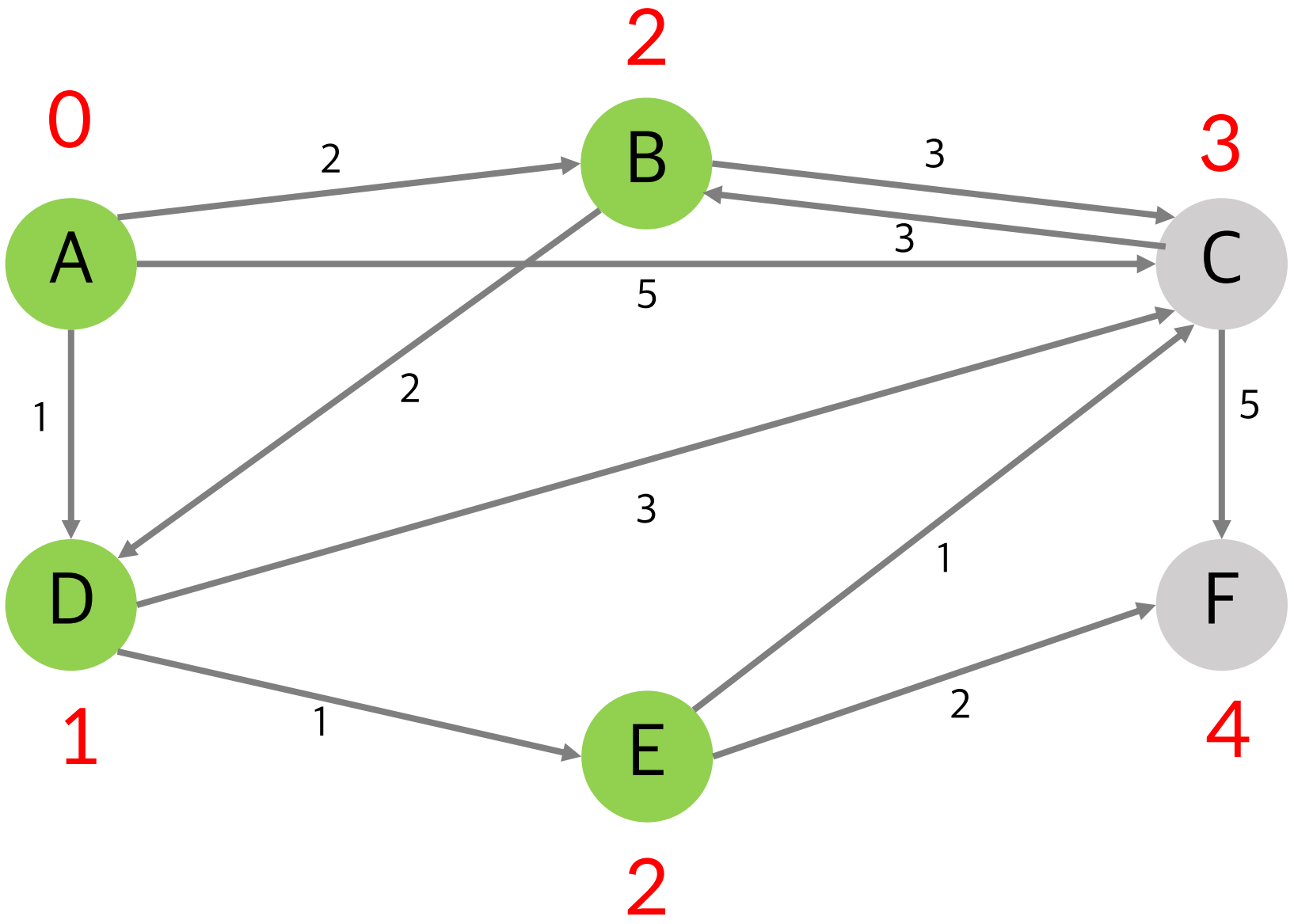
우선순위 큐
(노드, W)

(E, 1)
(B, 2)
(C, 5)



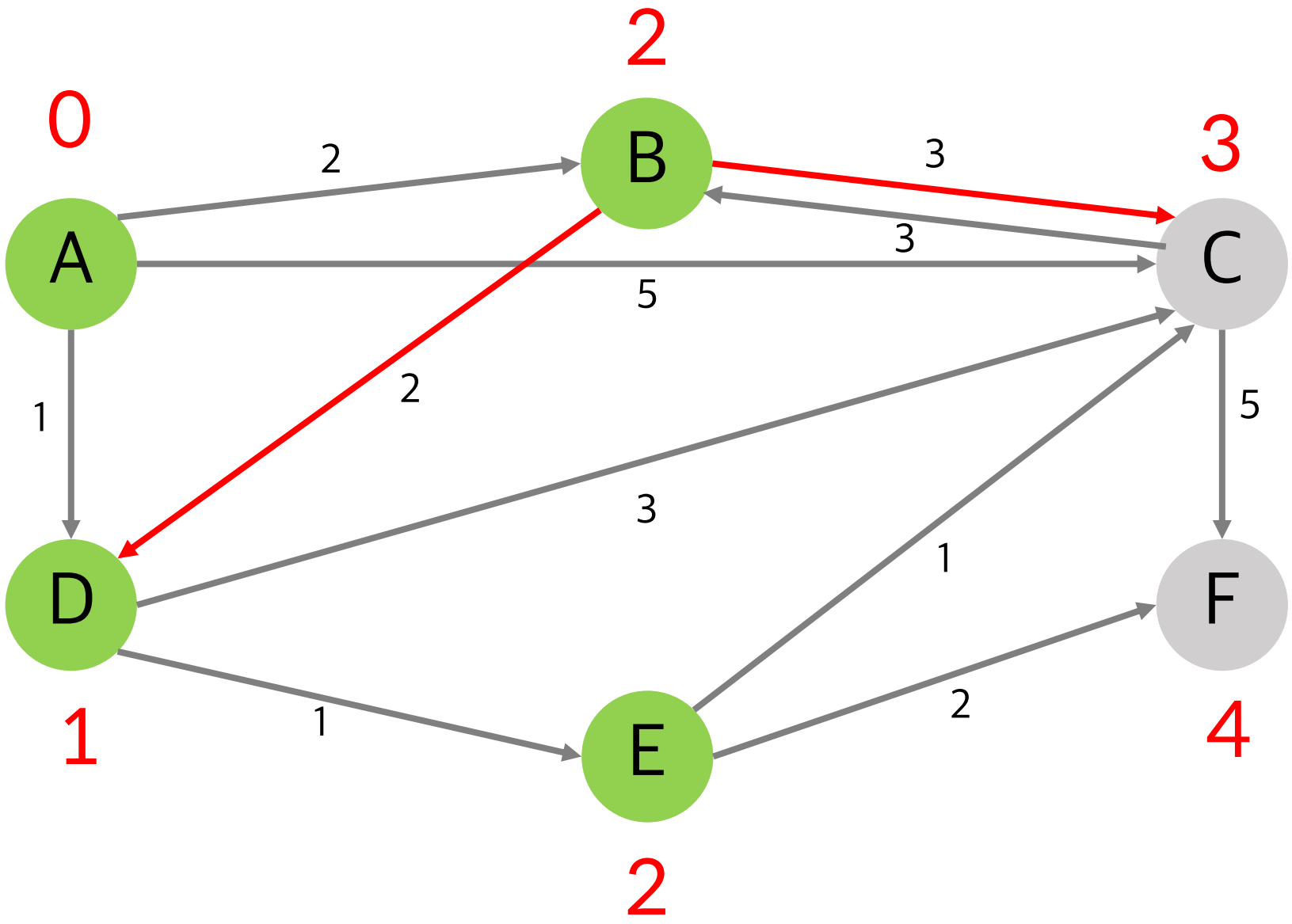
우선순위 큐
(노드, W)

(B, 2)
(F, 2)
(C, 5)



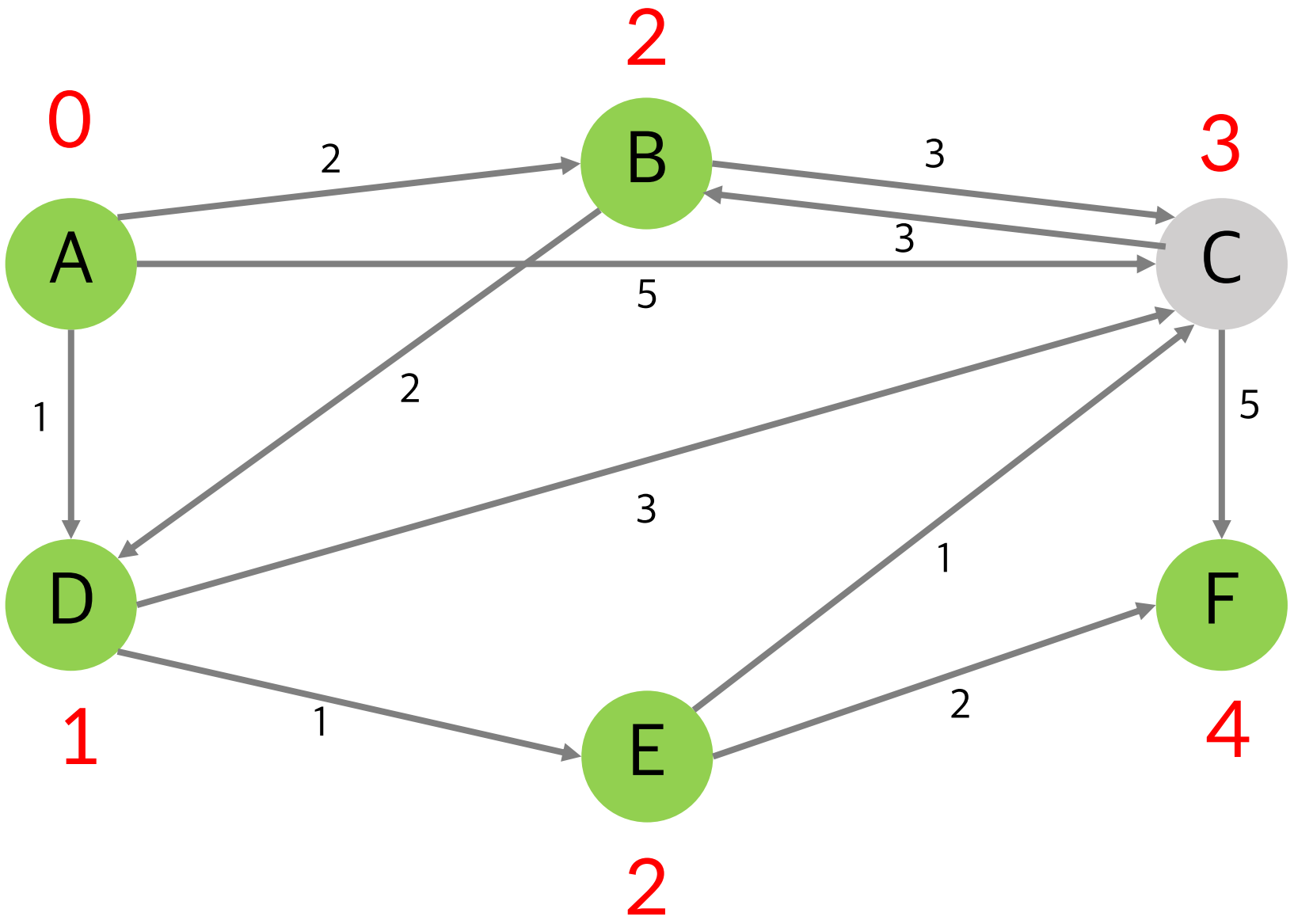
우선순위 큐
(노드, W)

(F, 2)
(C, 5)



우선순위 큐 (노드, W)

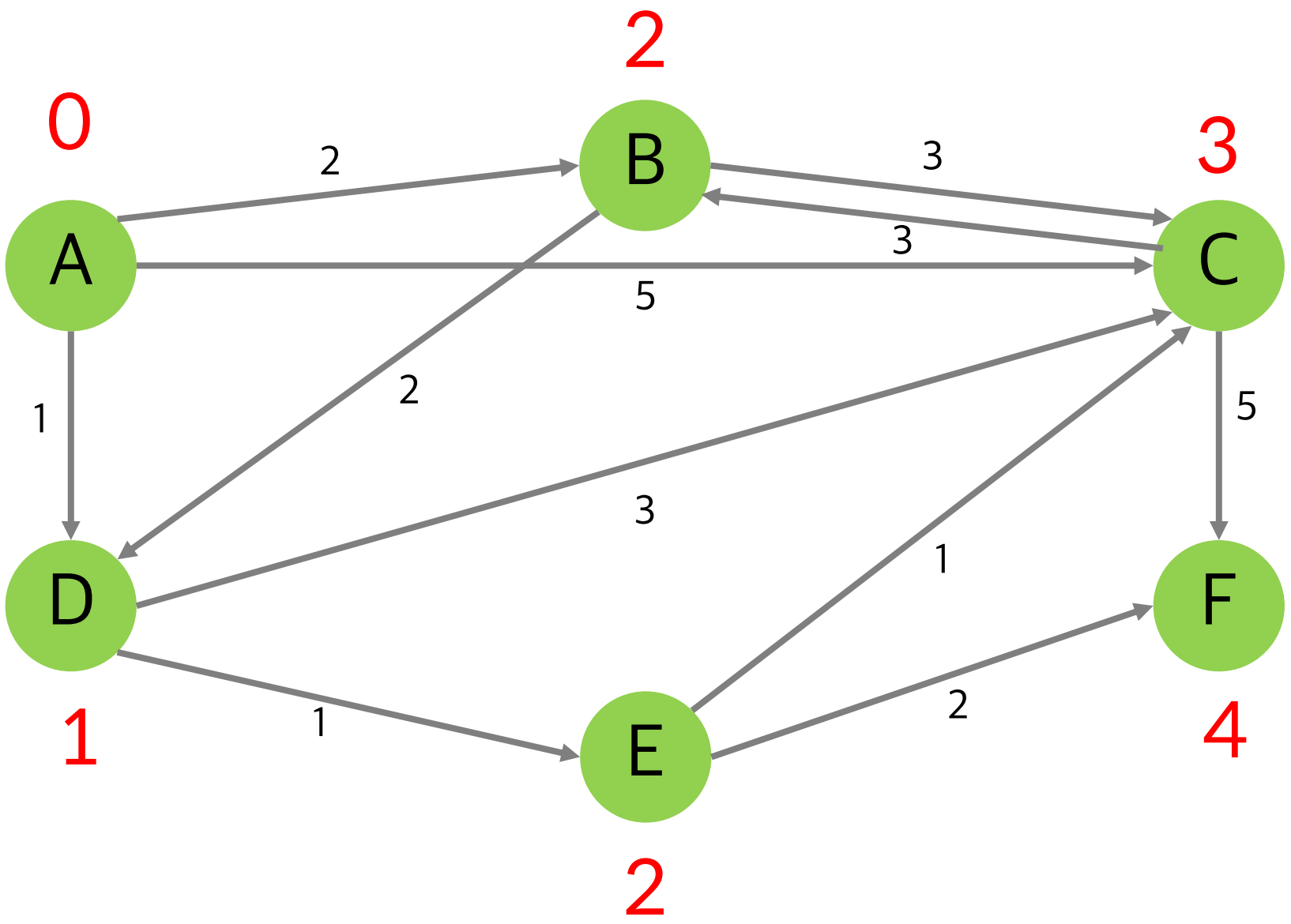
(F, 2)
(C, 5)

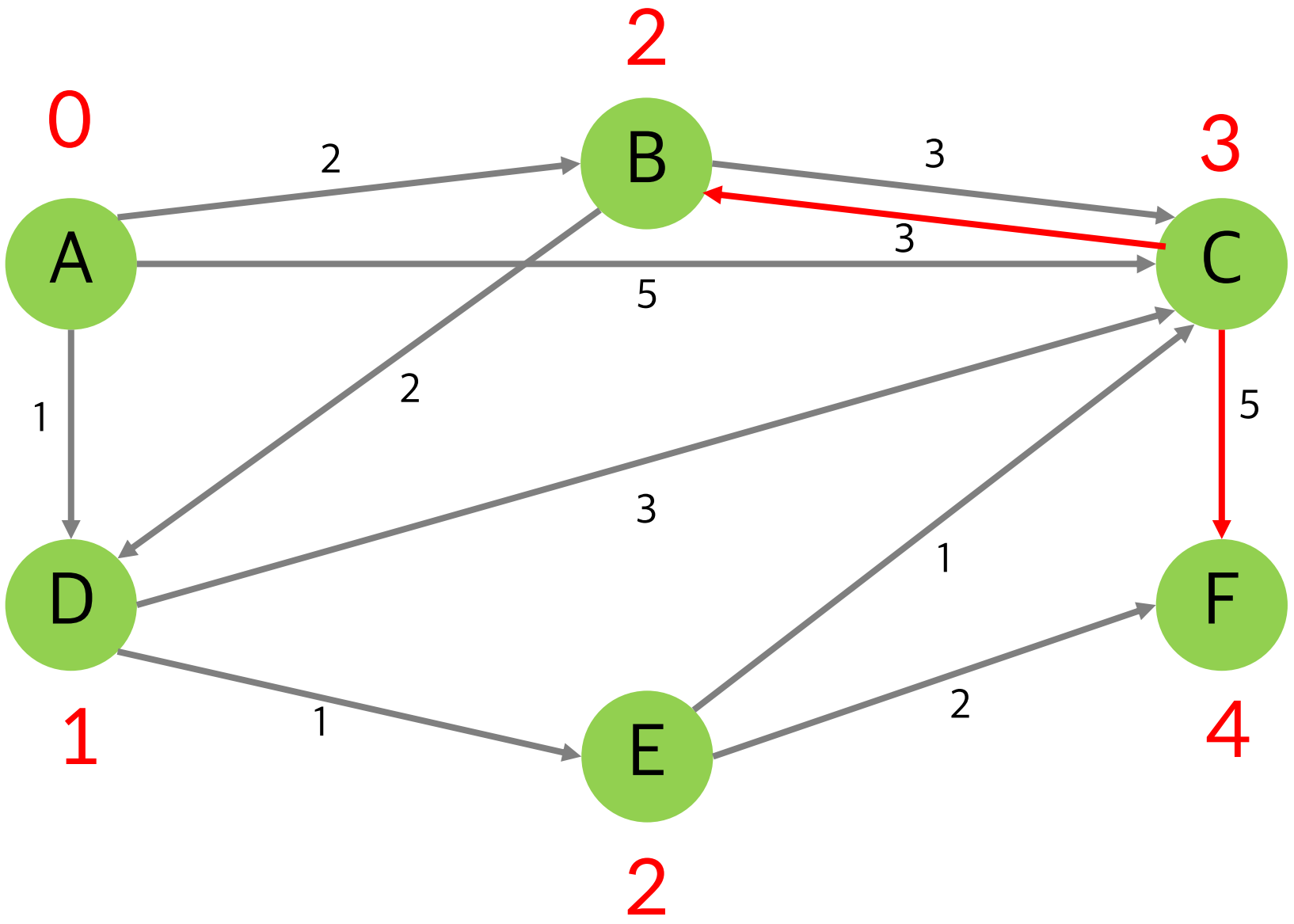


우선순위 큐
(노드, W)

(C, 5)

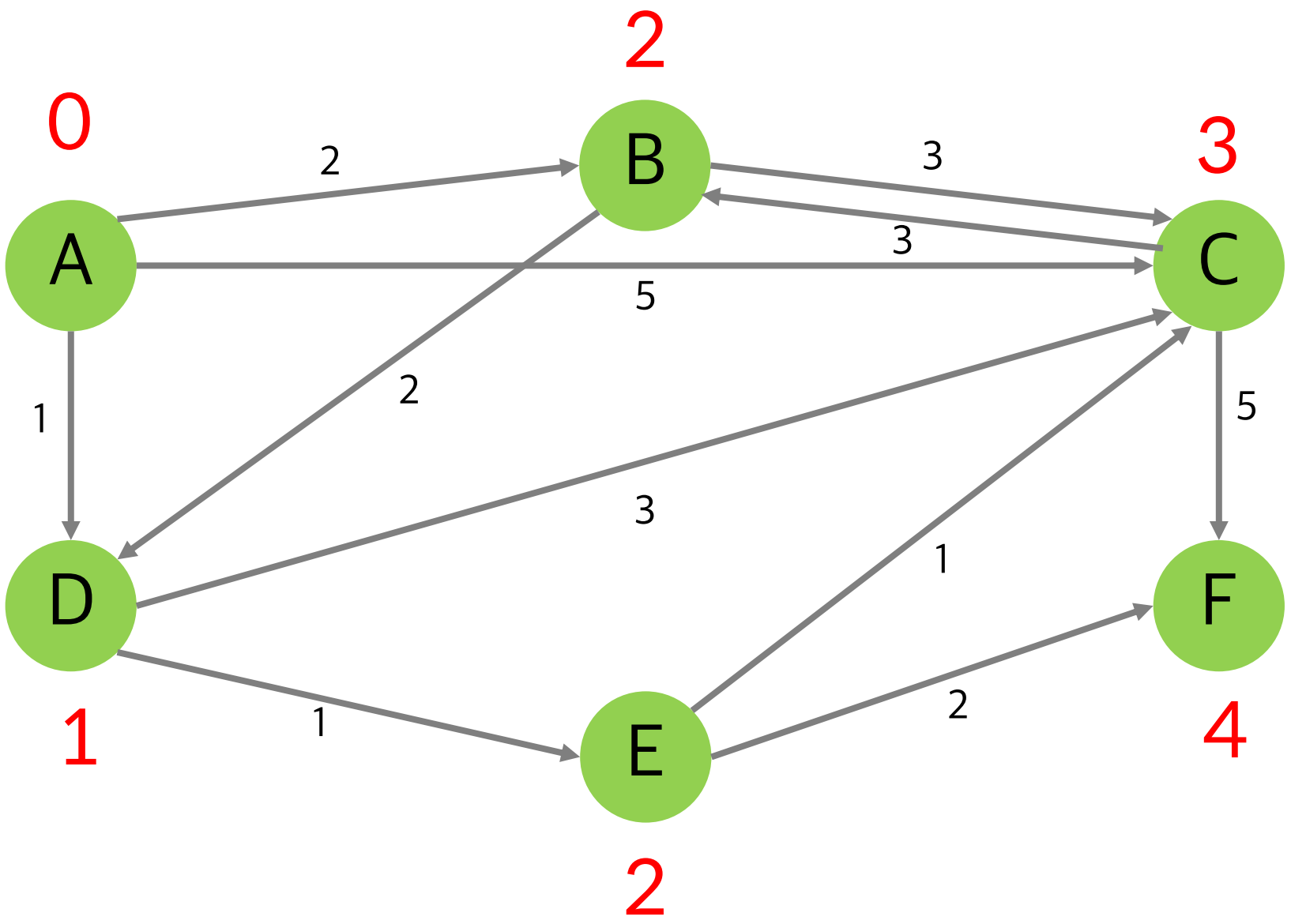
우선순위 큐
(노드, W)





우선순위 큐
(노드, W)

우선순위 큐
(노드, W)



Baekjoon - #11279

👉 오늘의 문제

⭐ #18352, ‘특정 거리의 도시 찾기’

⭐ #1753, ‘최단 경로’