

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**

- Acyclic
- 최적화 구조(Optimal Structure)가 존재
=> Greedy Algorithm 접근 가능

Dijkstra Algorithm

👉 정의

- **Input**

1 Weighted Graph 2 Starting Node

- **Output**

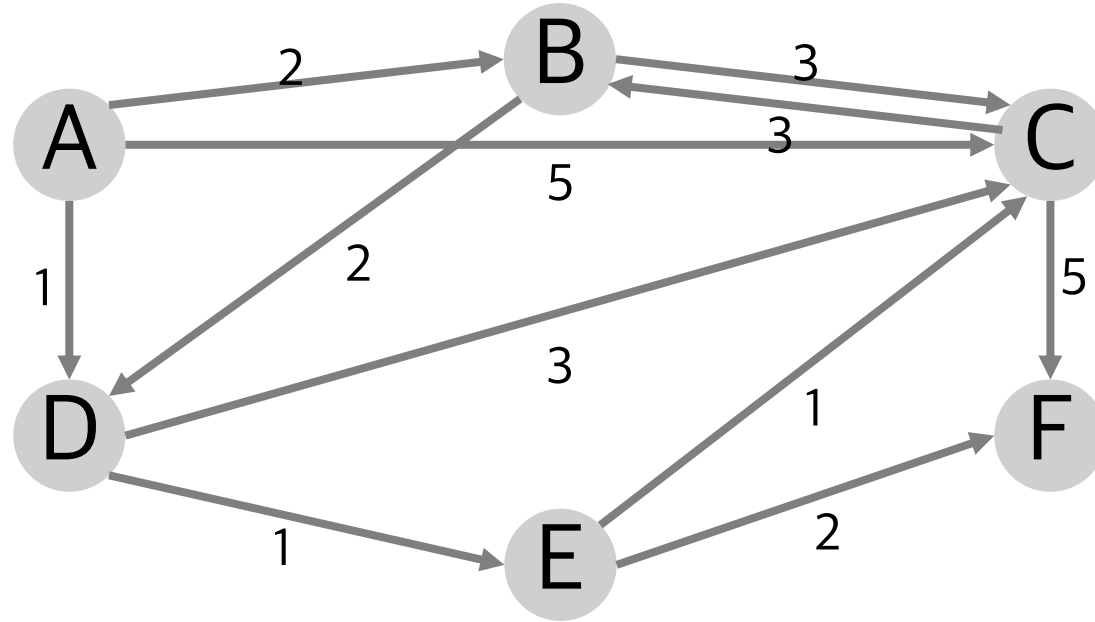
Shortest Paths from **Starting** to **each node**

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

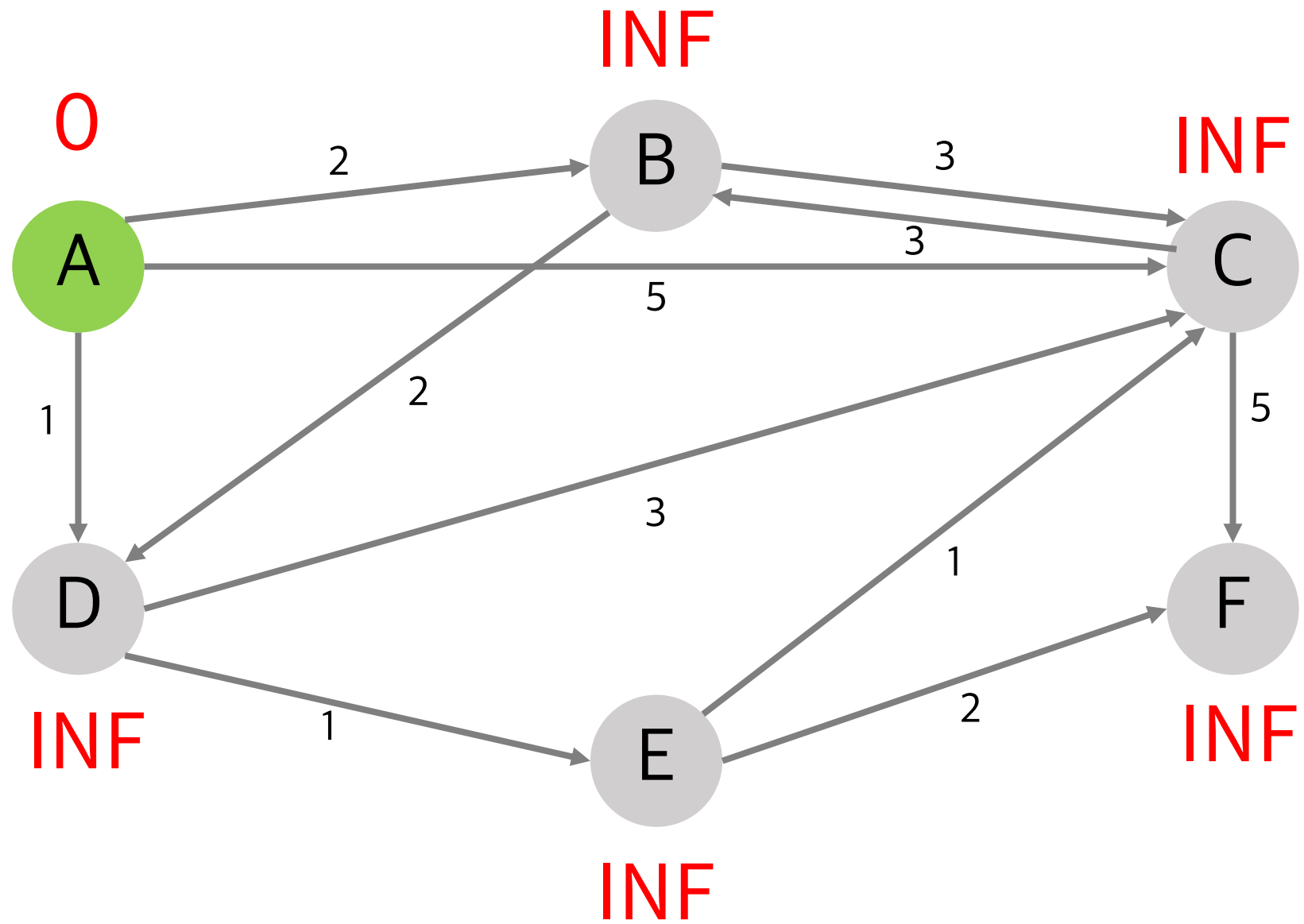
Dijkstra Algorithm

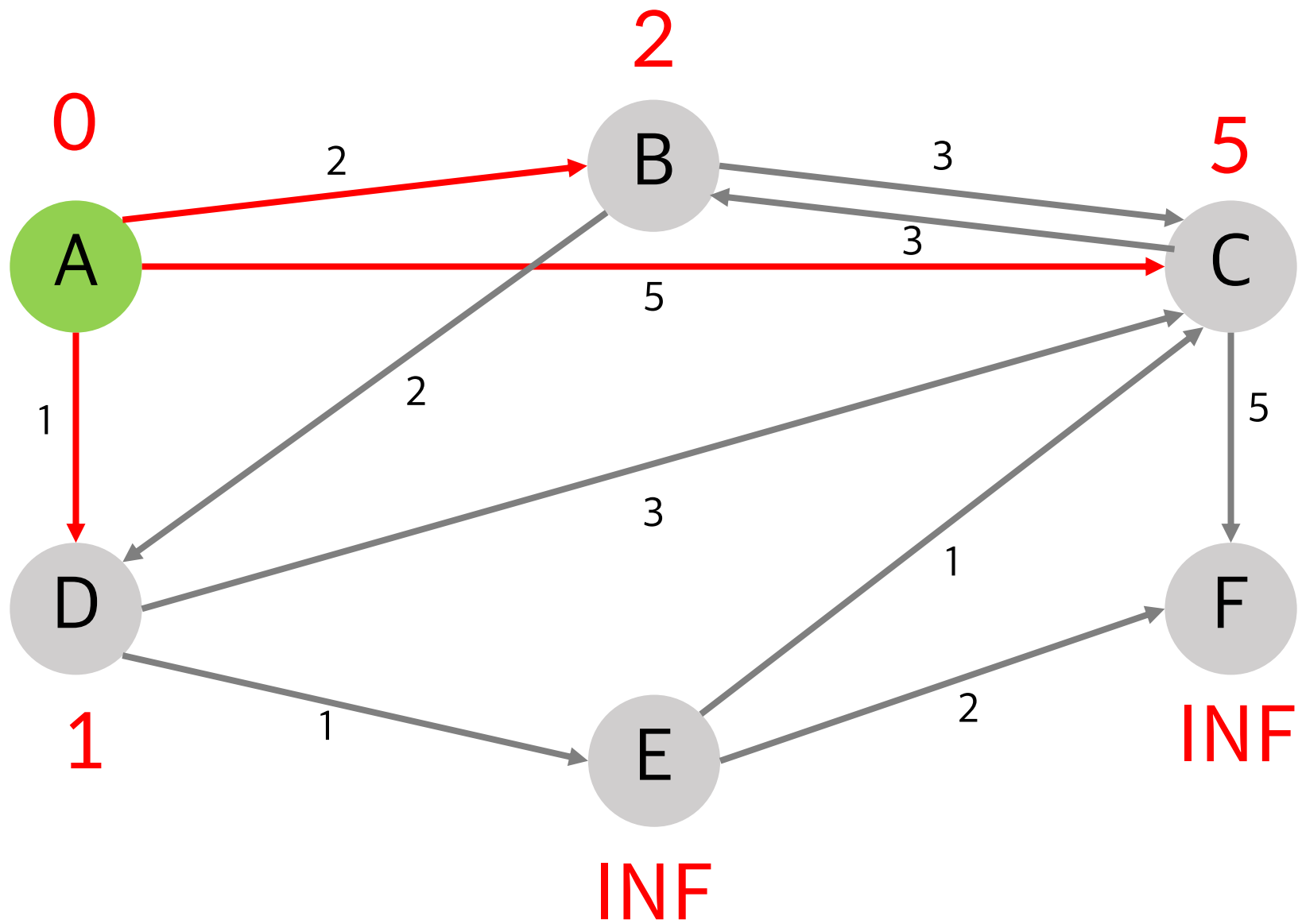


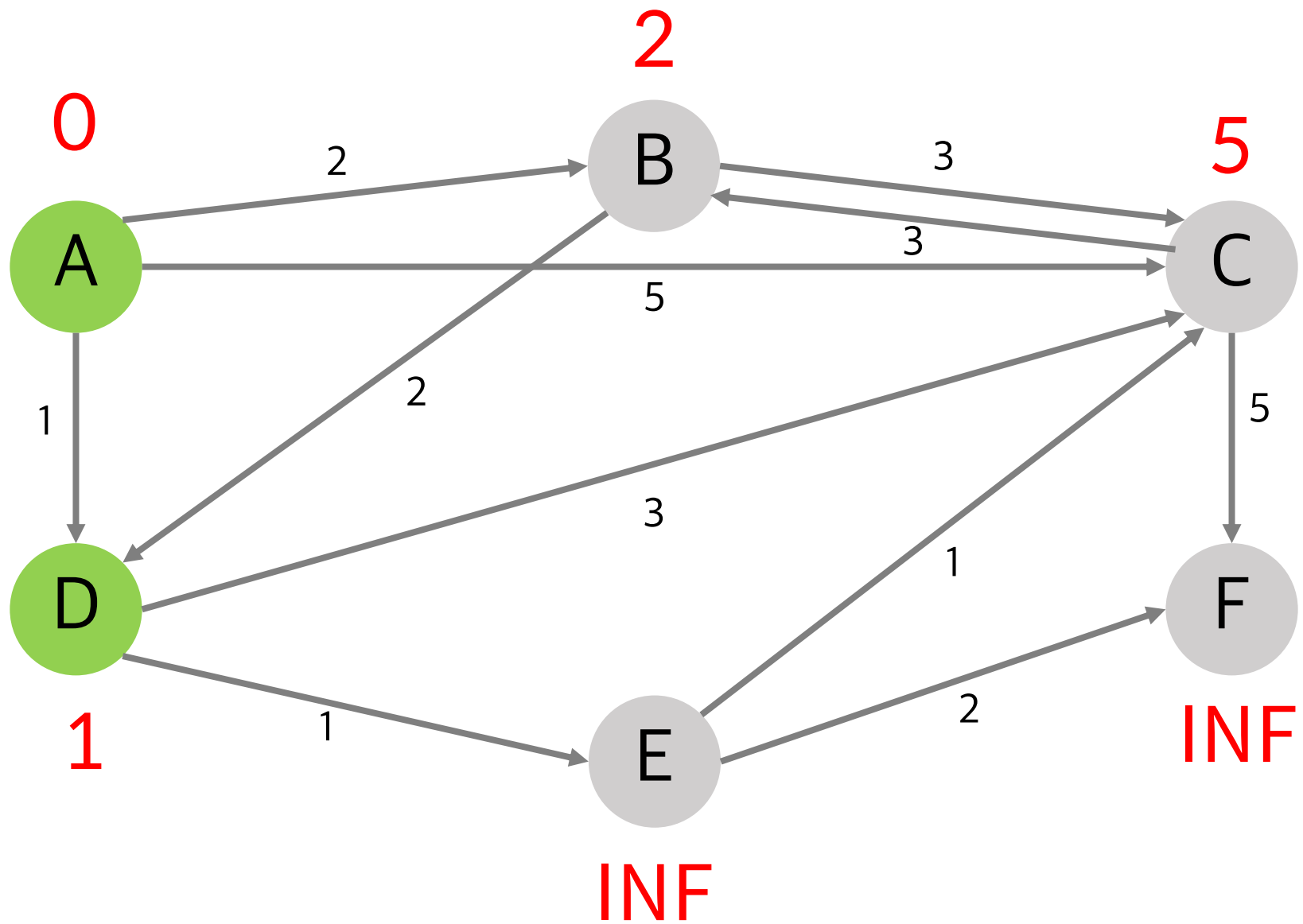
- Process

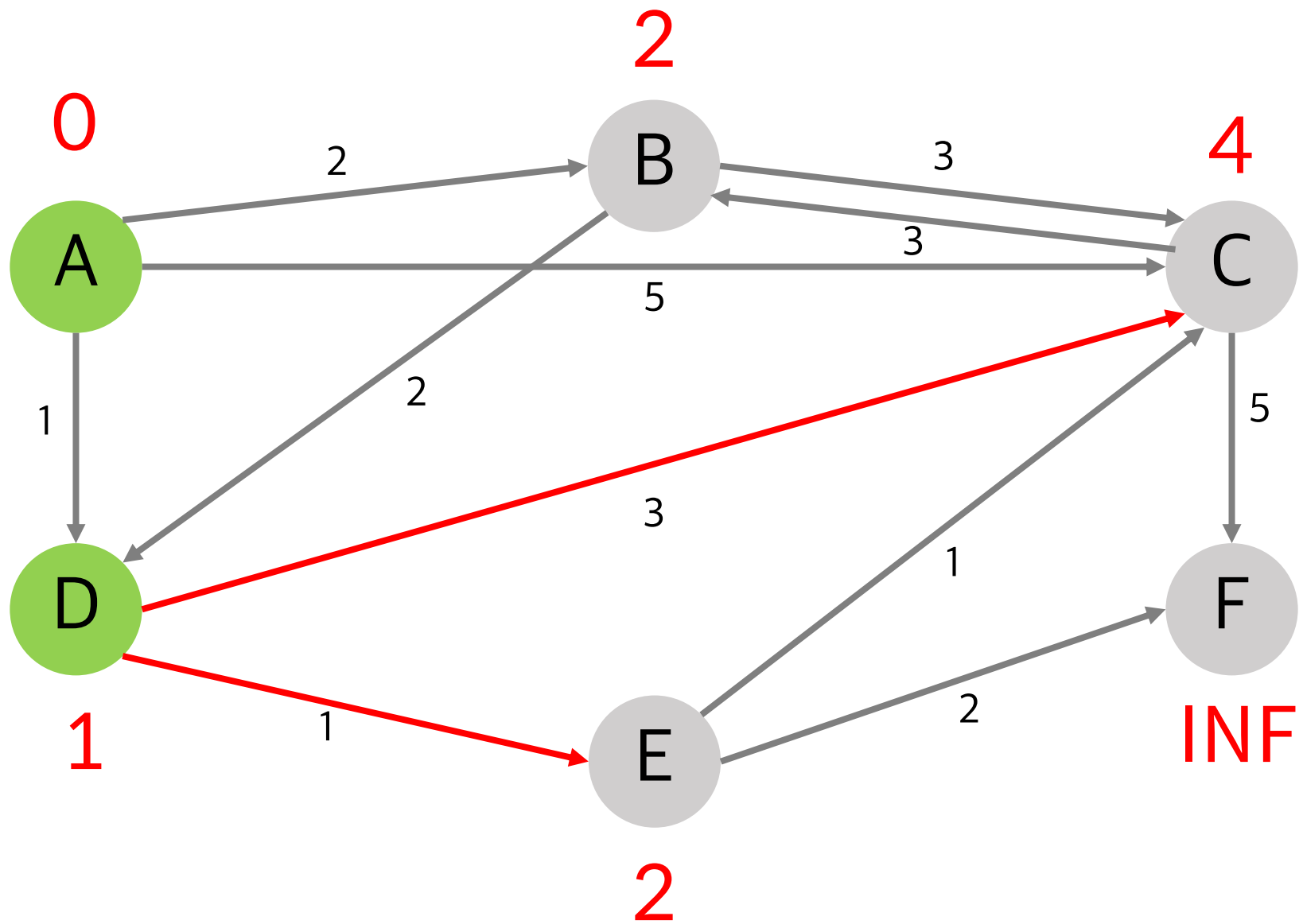
- 1 출발 노드를 설정
- 2 최단 거리 테이블을 초기화
- 3 방문하지 않은 노드 중에서 최단 거리가 최소인 노드를 선택
- 4 해당 노드를 거쳐 다른 노드로 가는 비용을 계산하여 최단 거리 테이블을 갱신합니다
- 5 방문하지 않은 노드가 1개 남을 때까지
3, 4번을 반복합니다.

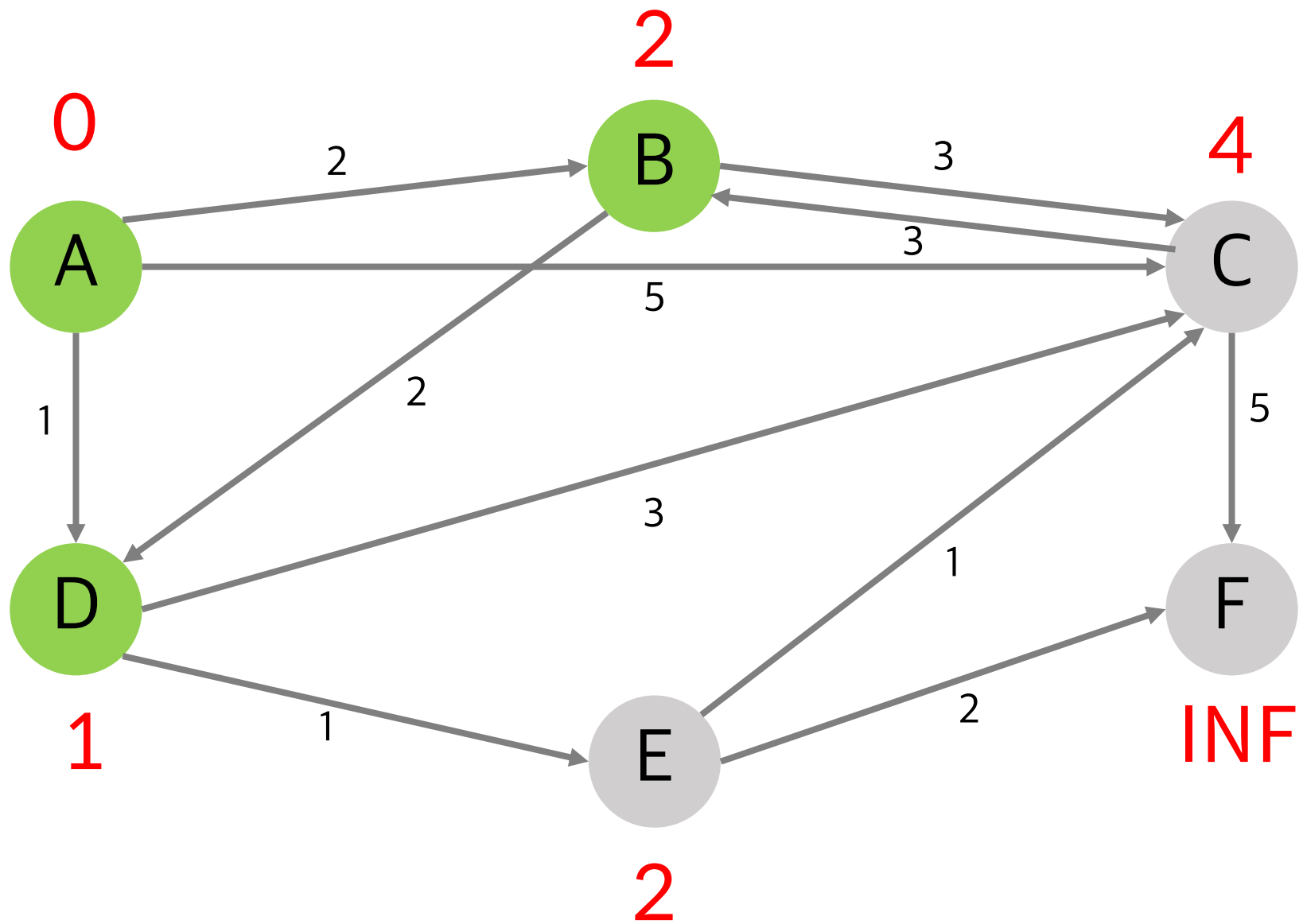
https://www.youtube.com/watch?v=acqm9mM1P6o&ab_channel=%EB%8F%99%EB%B9%88%EB%82%98

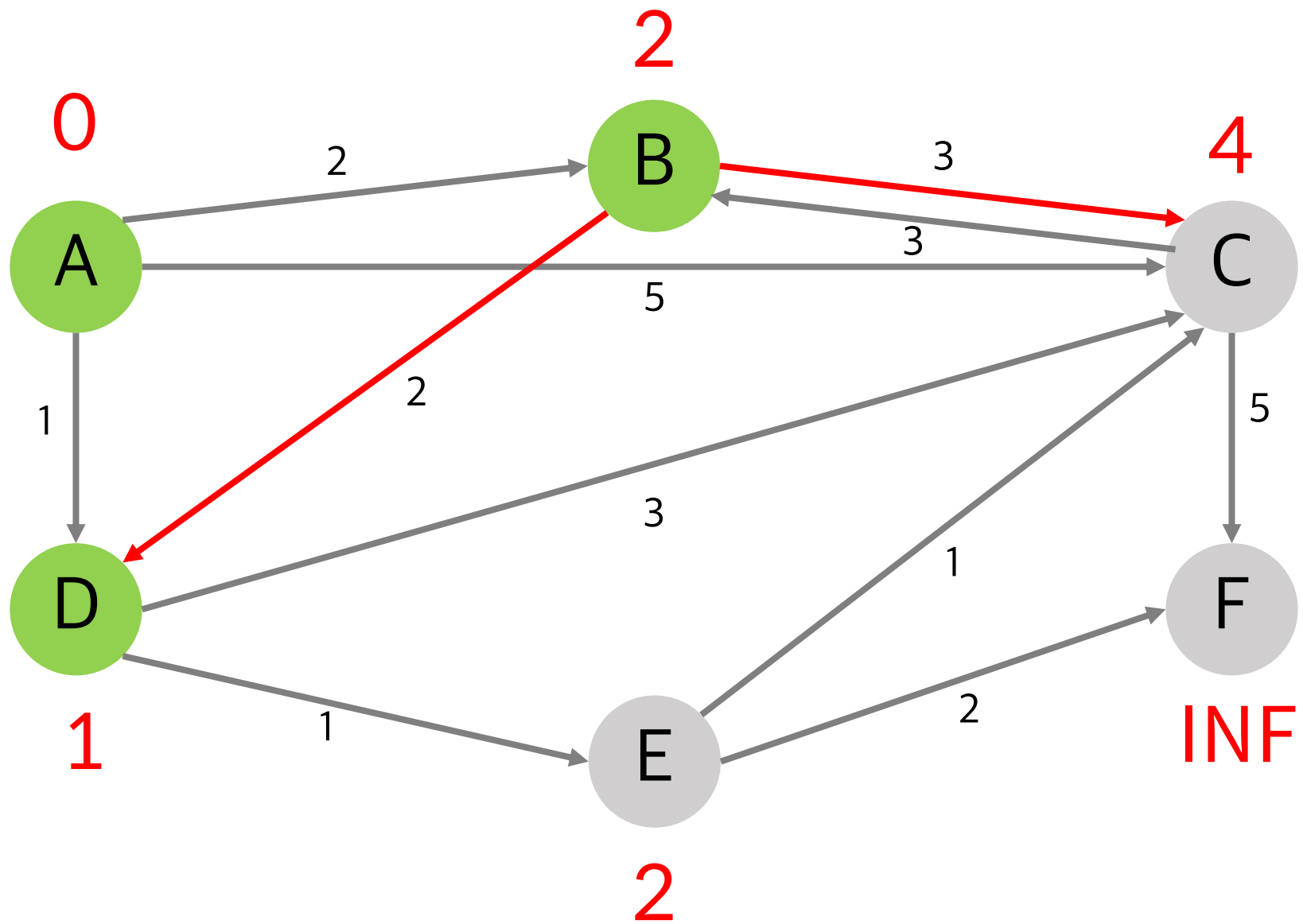


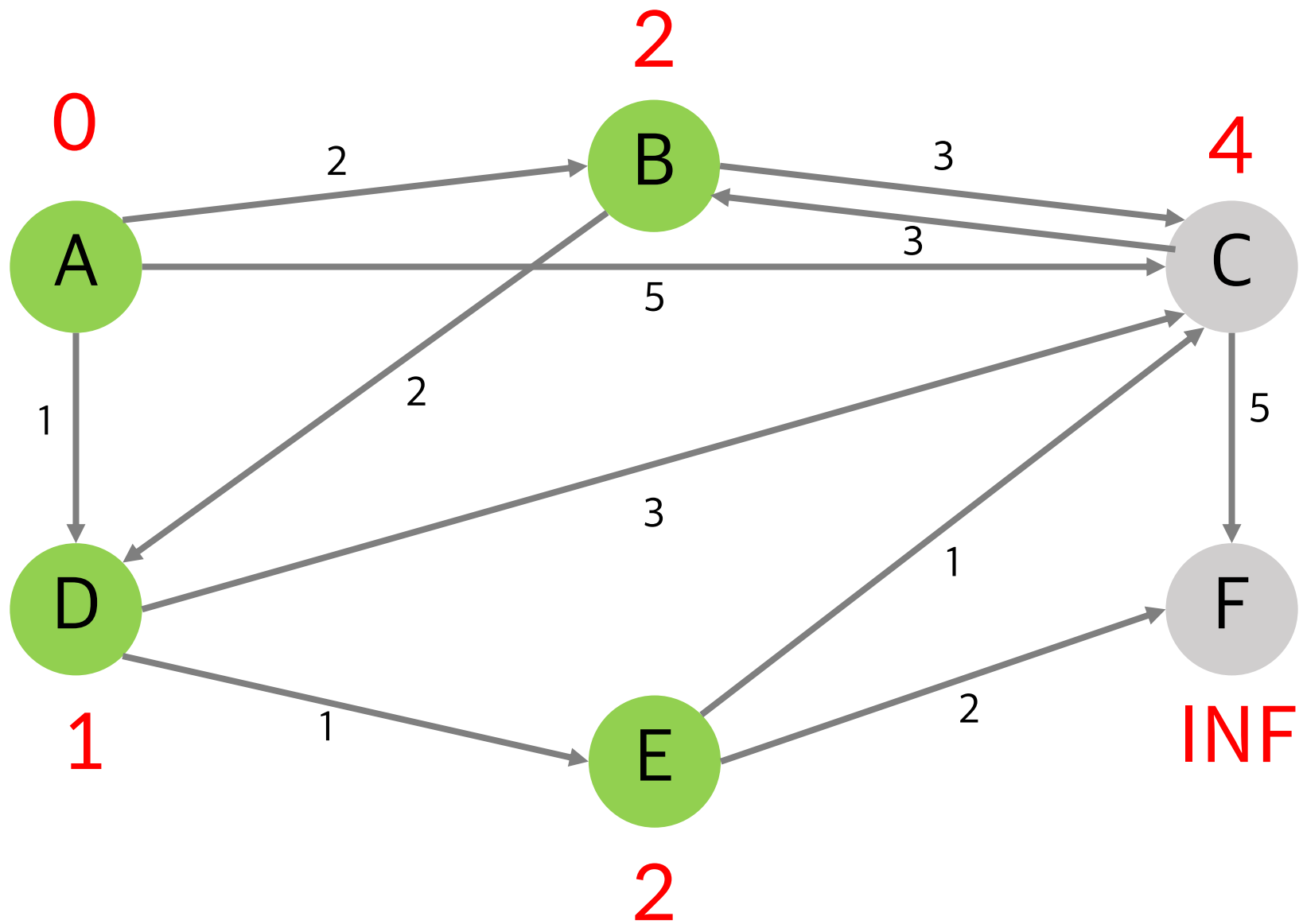


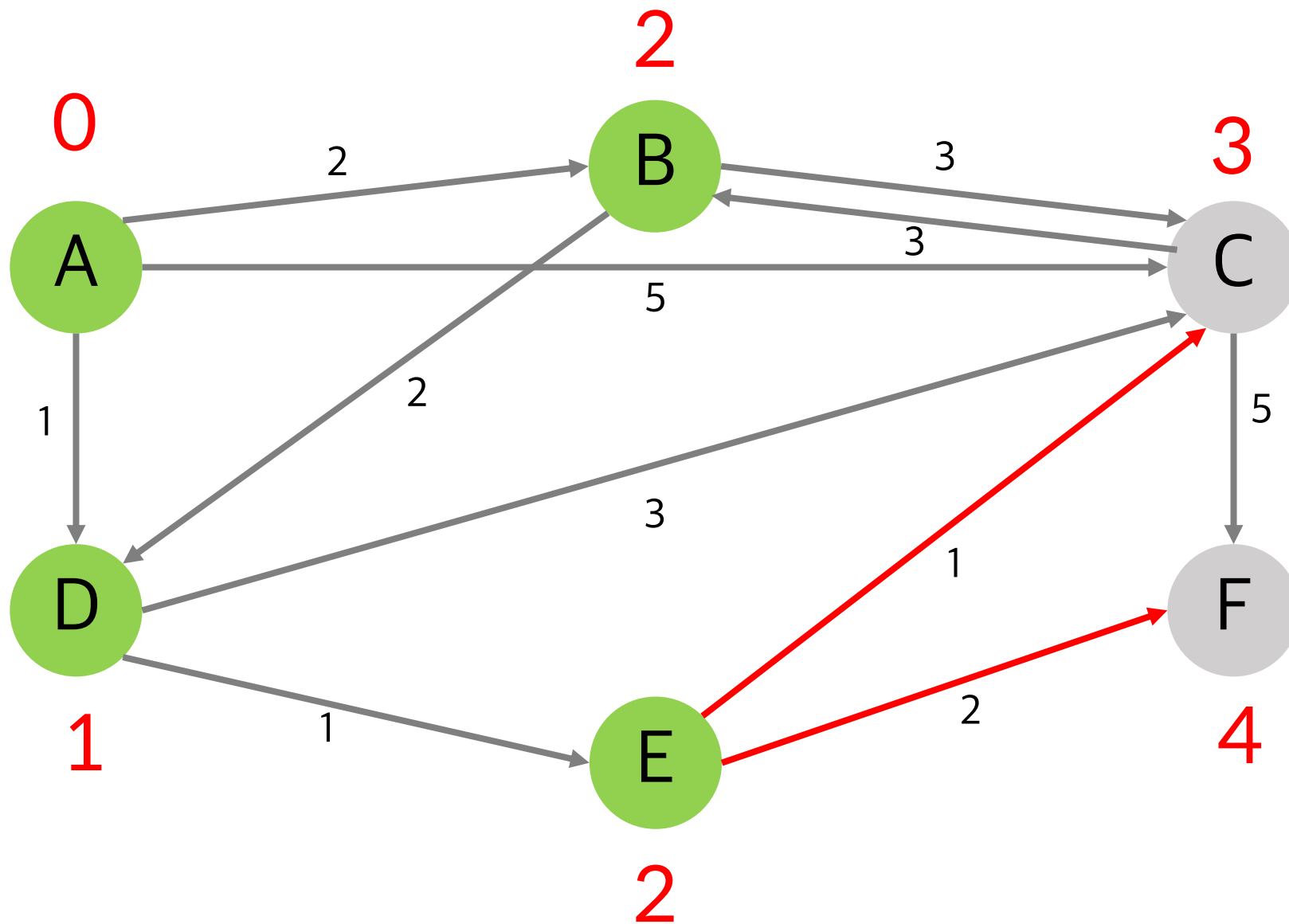


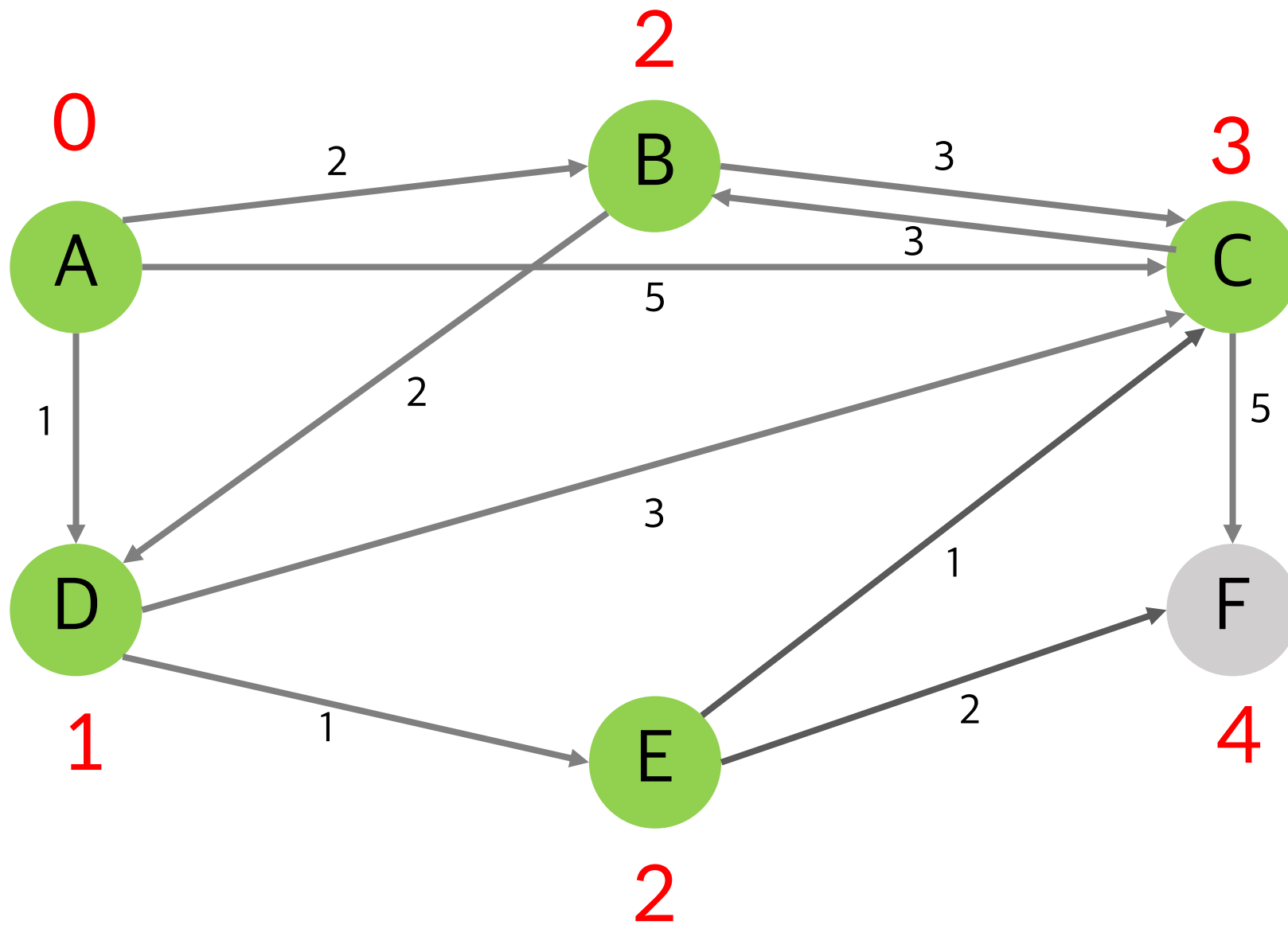


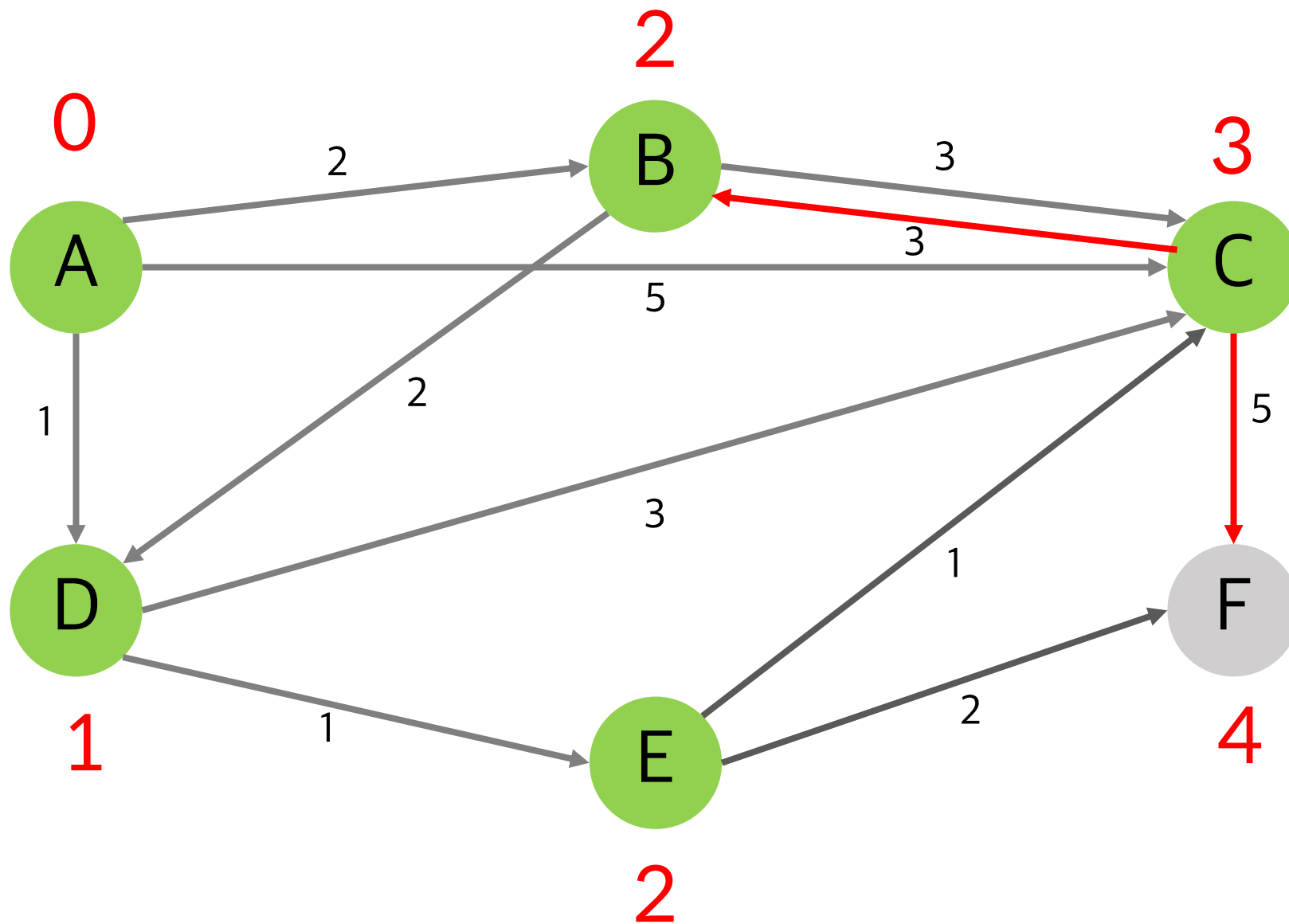


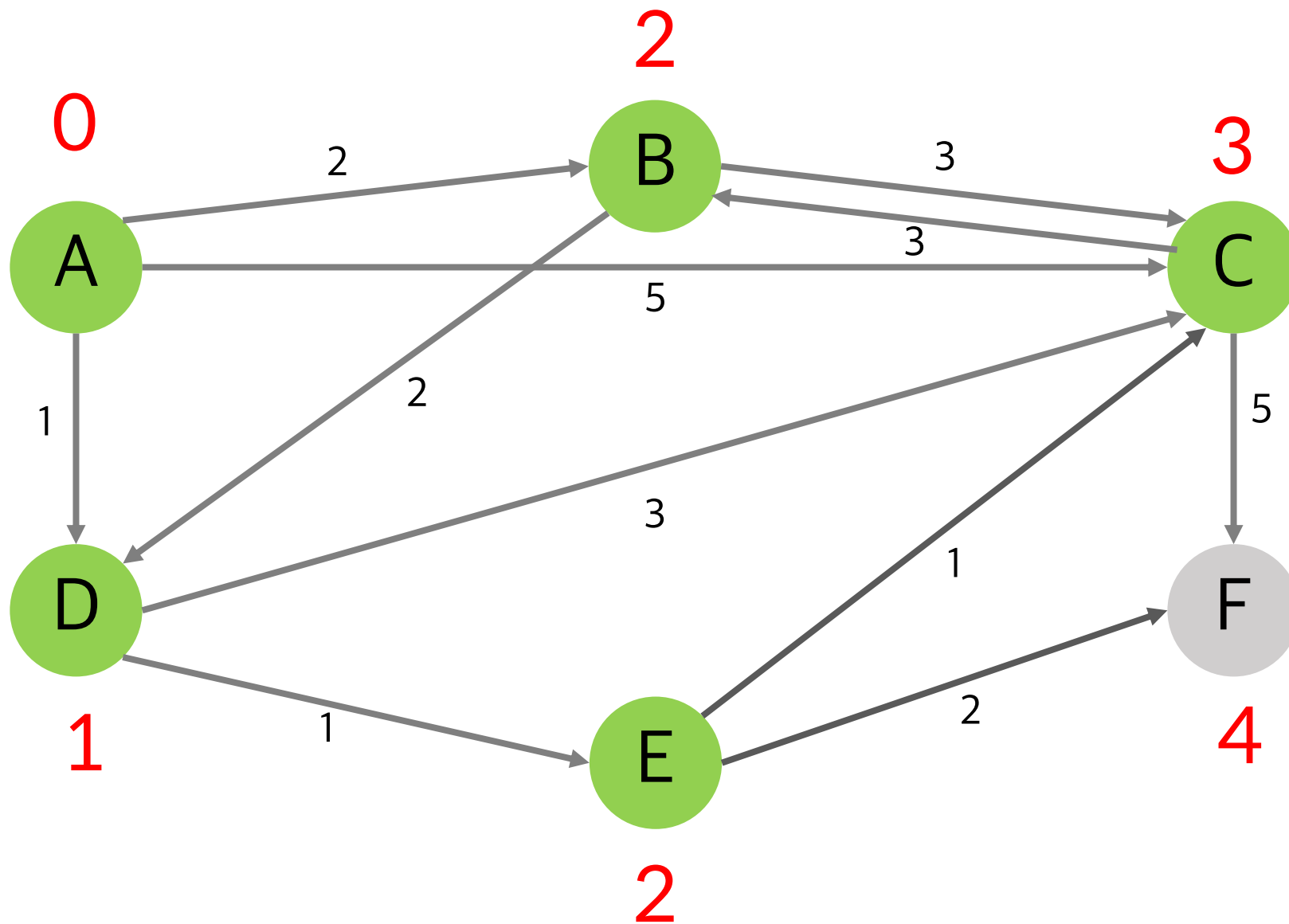




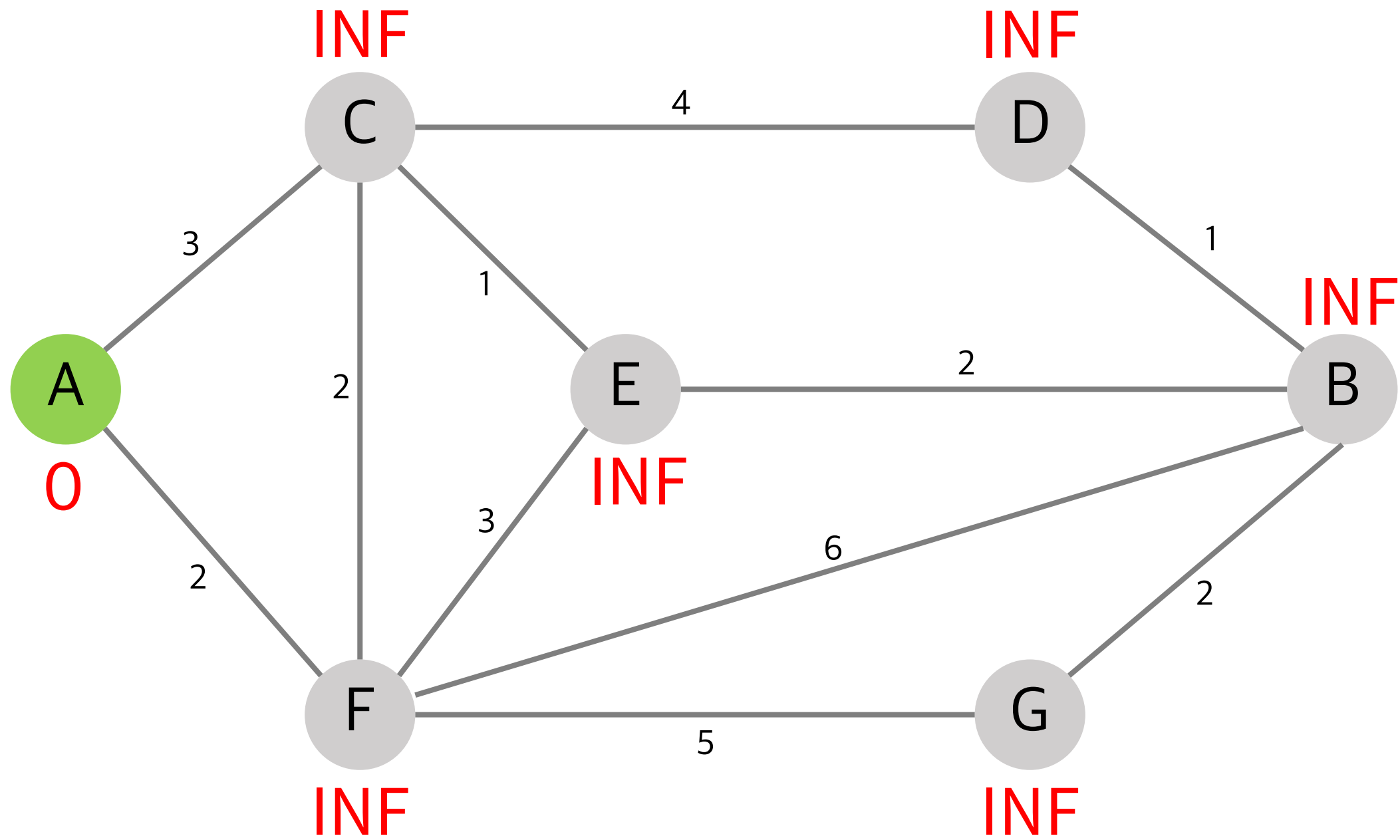


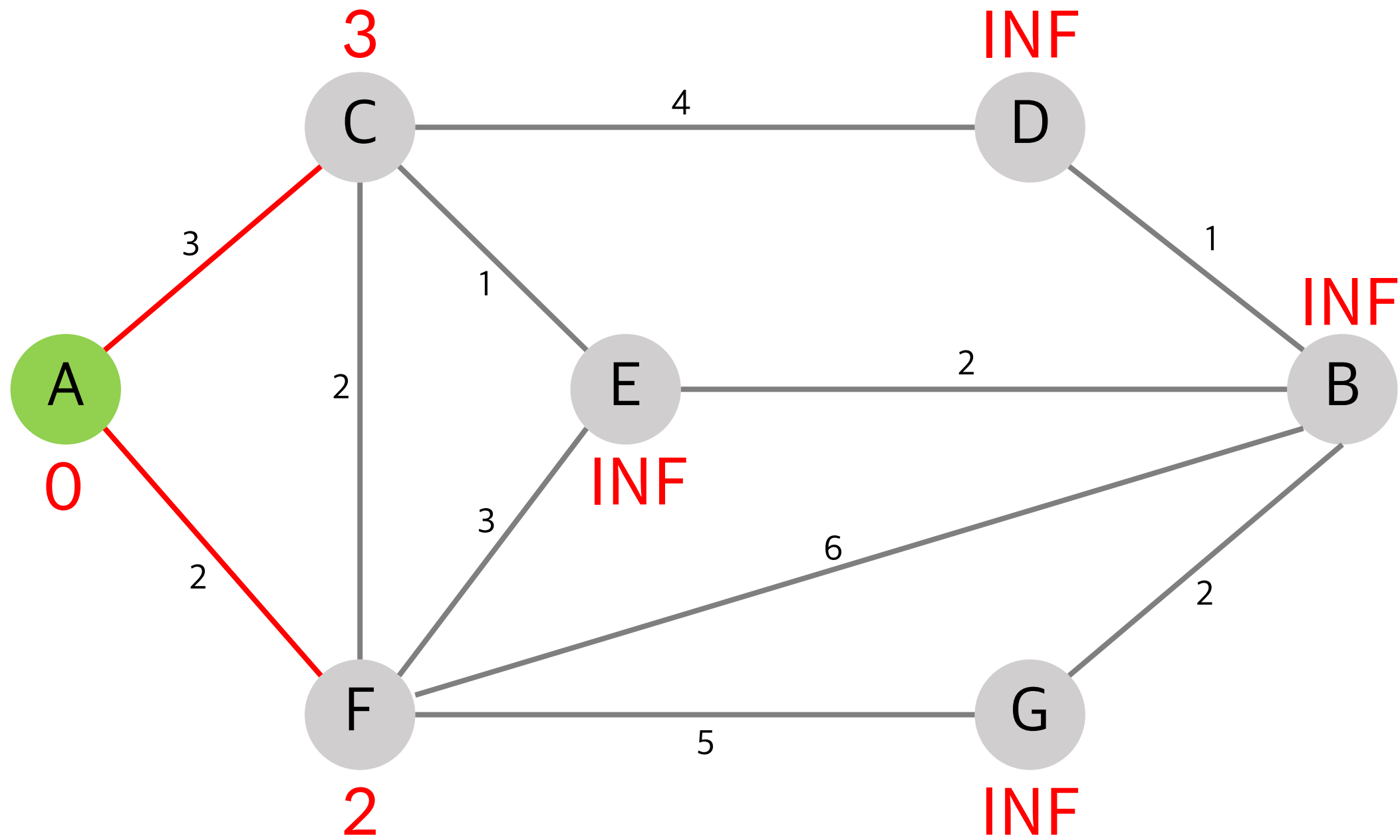


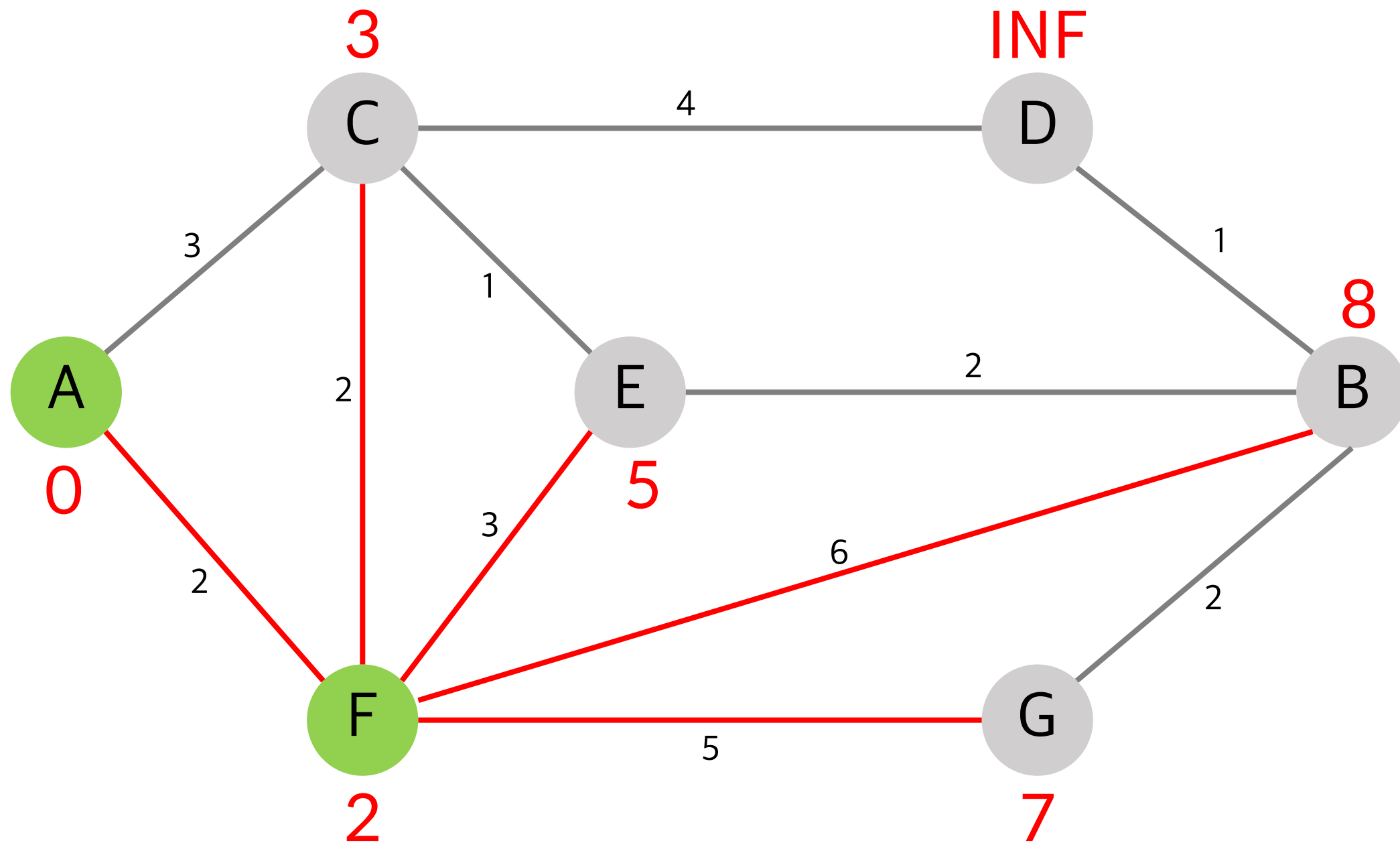


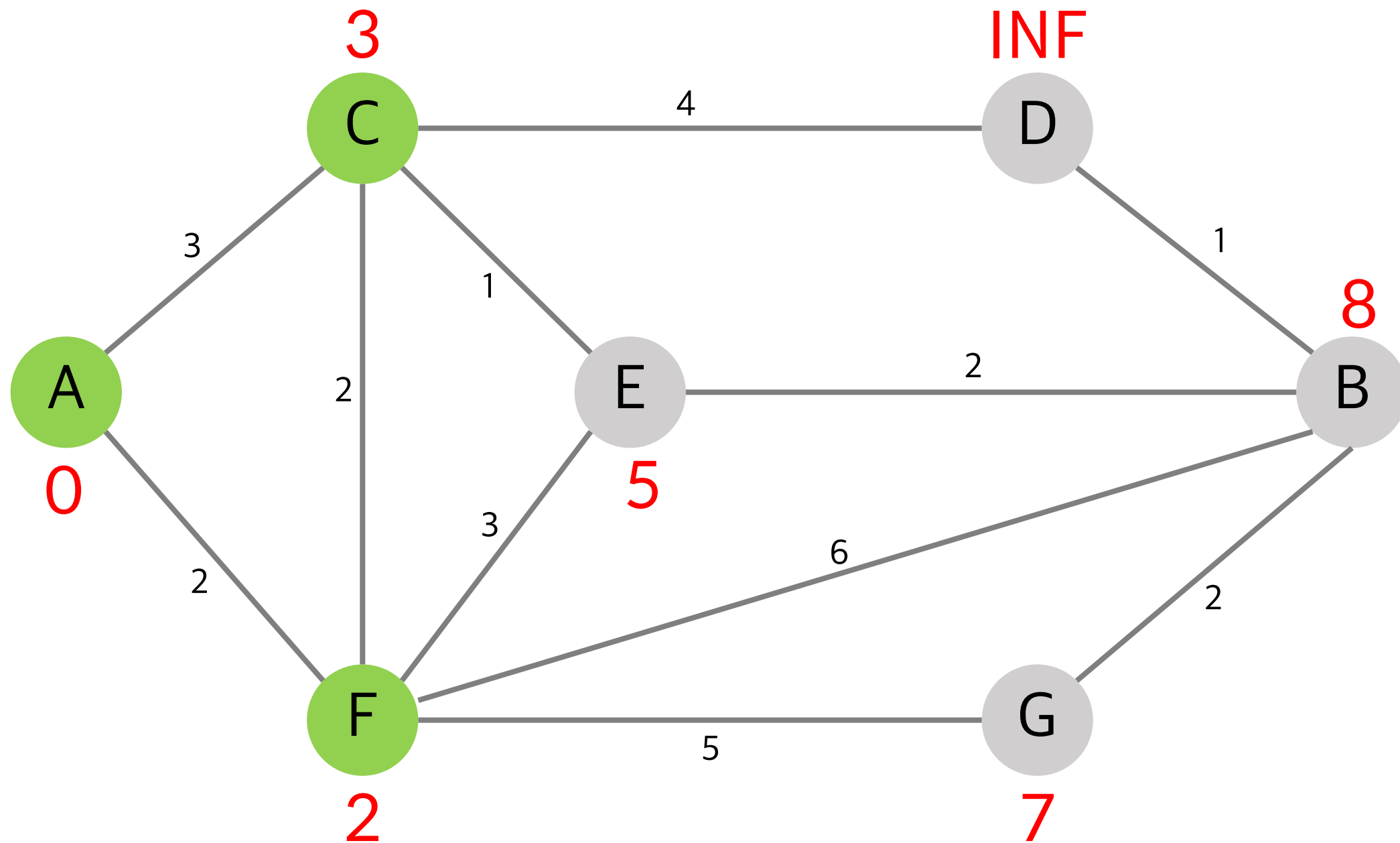


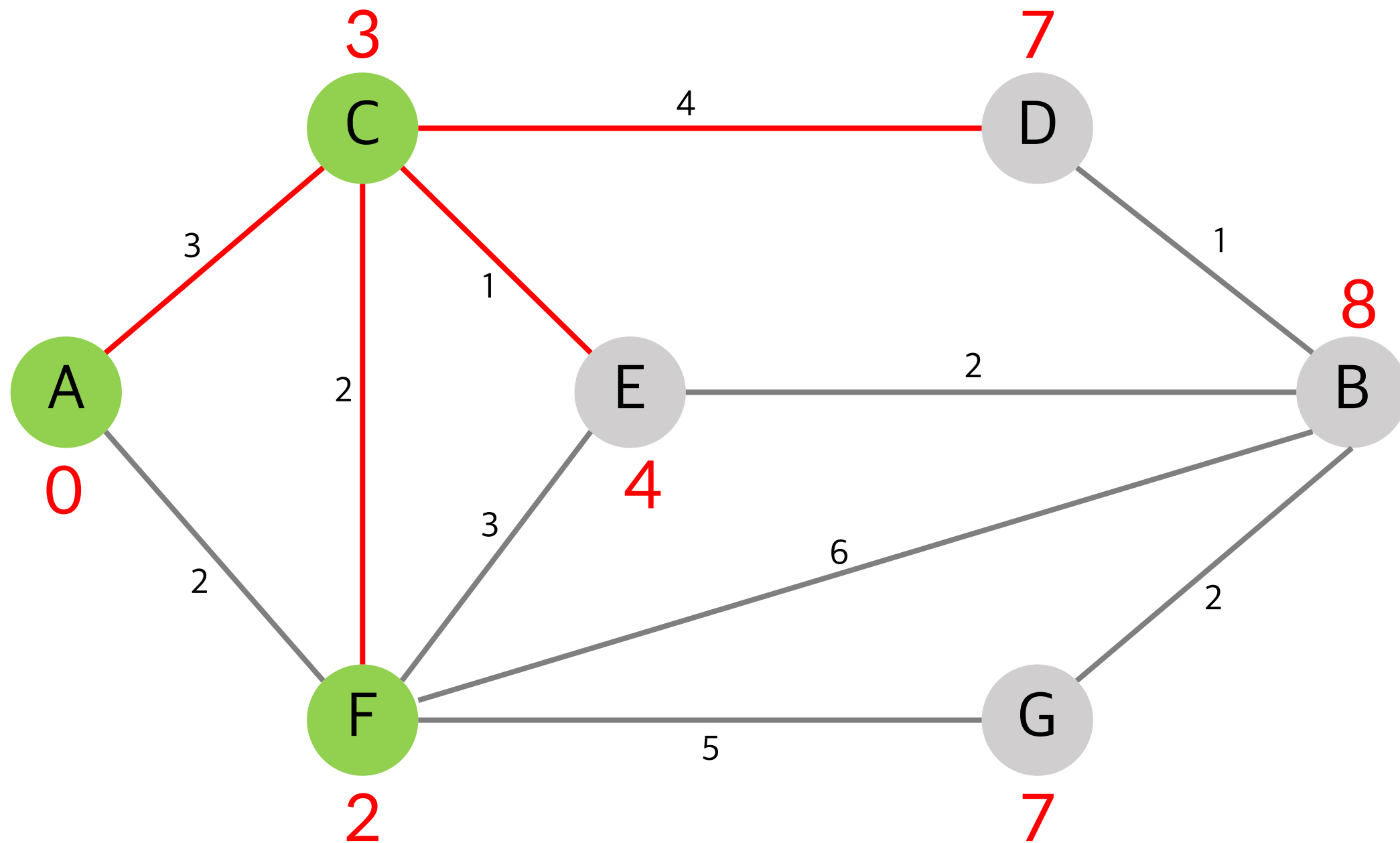
종료.

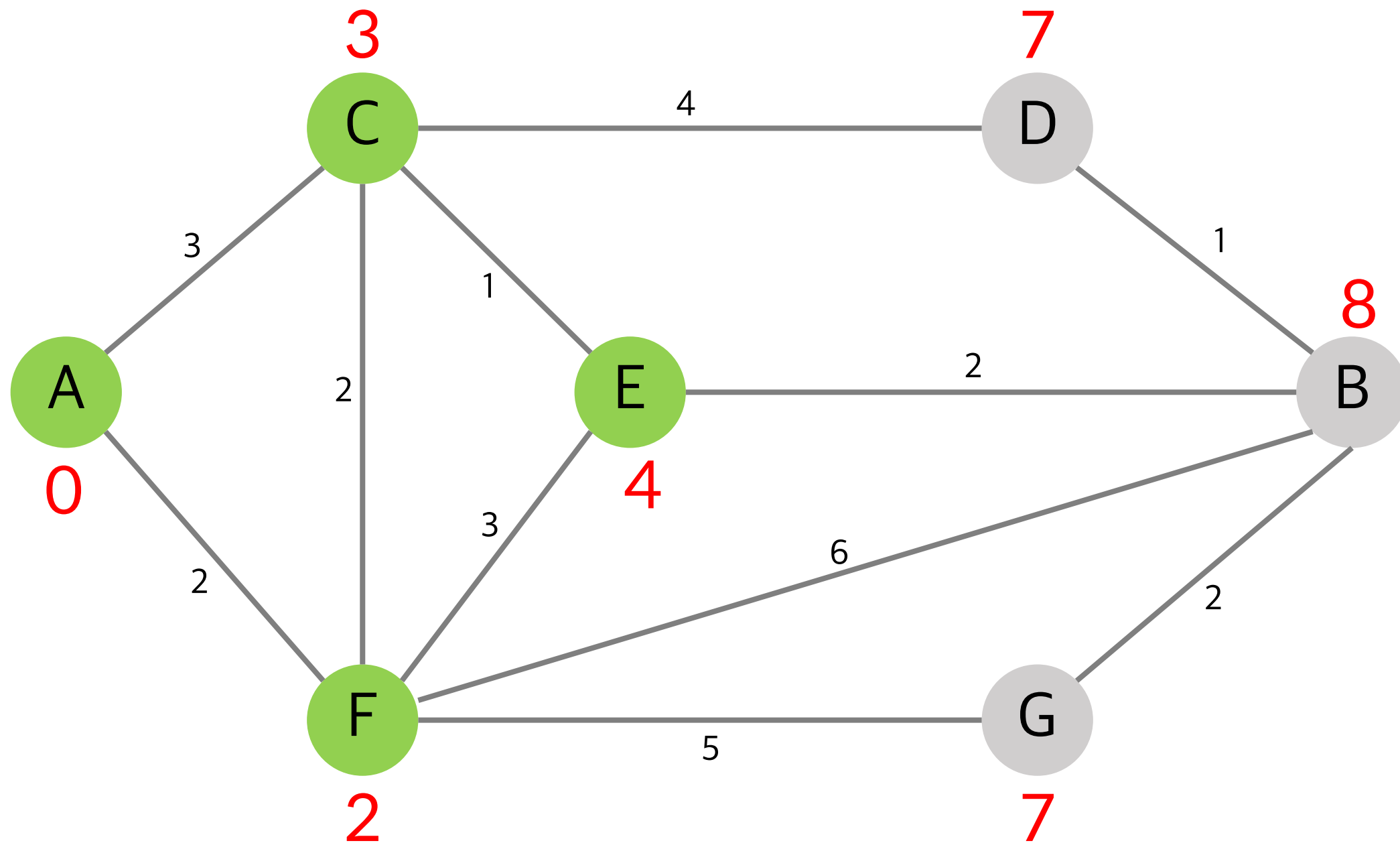


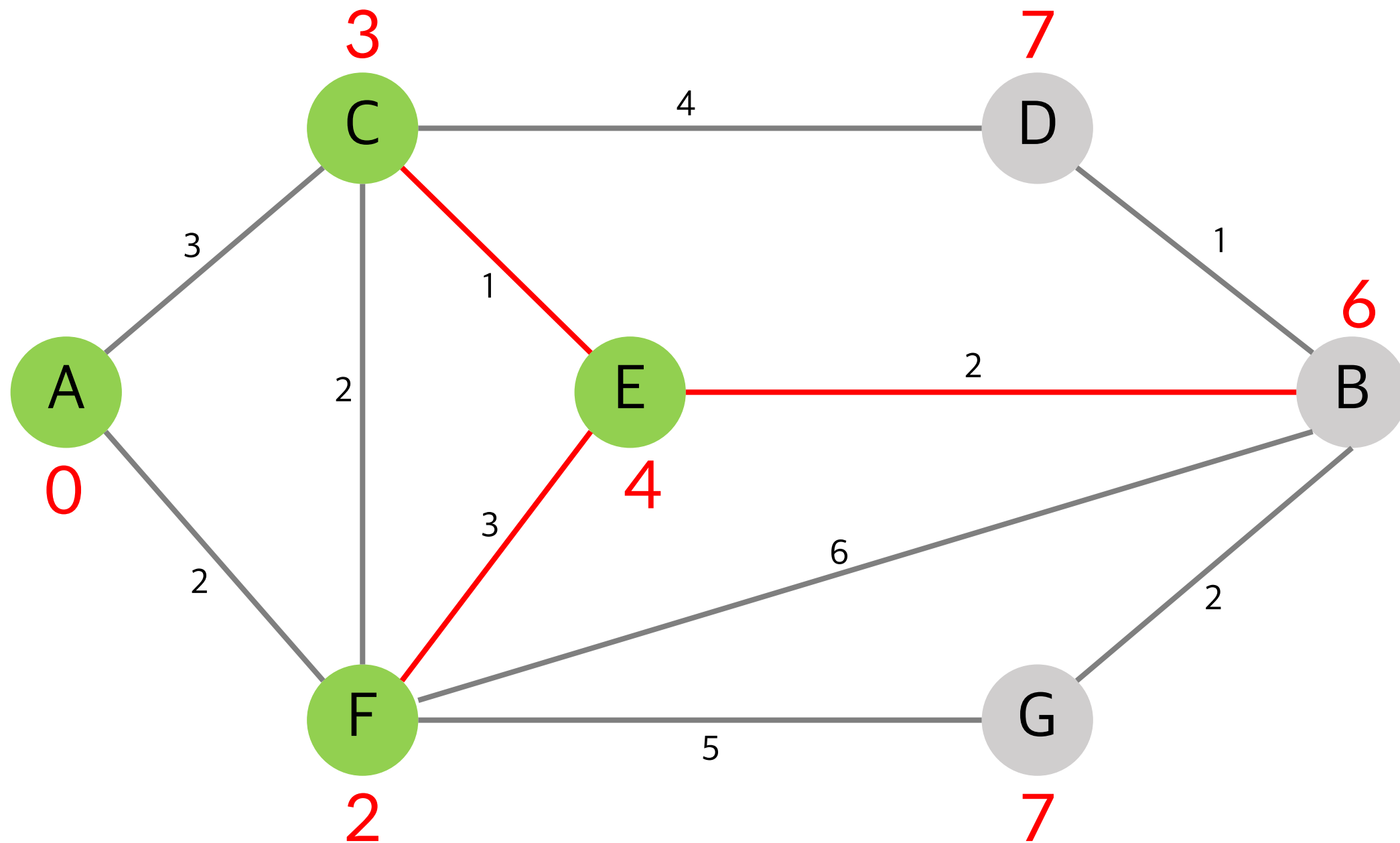


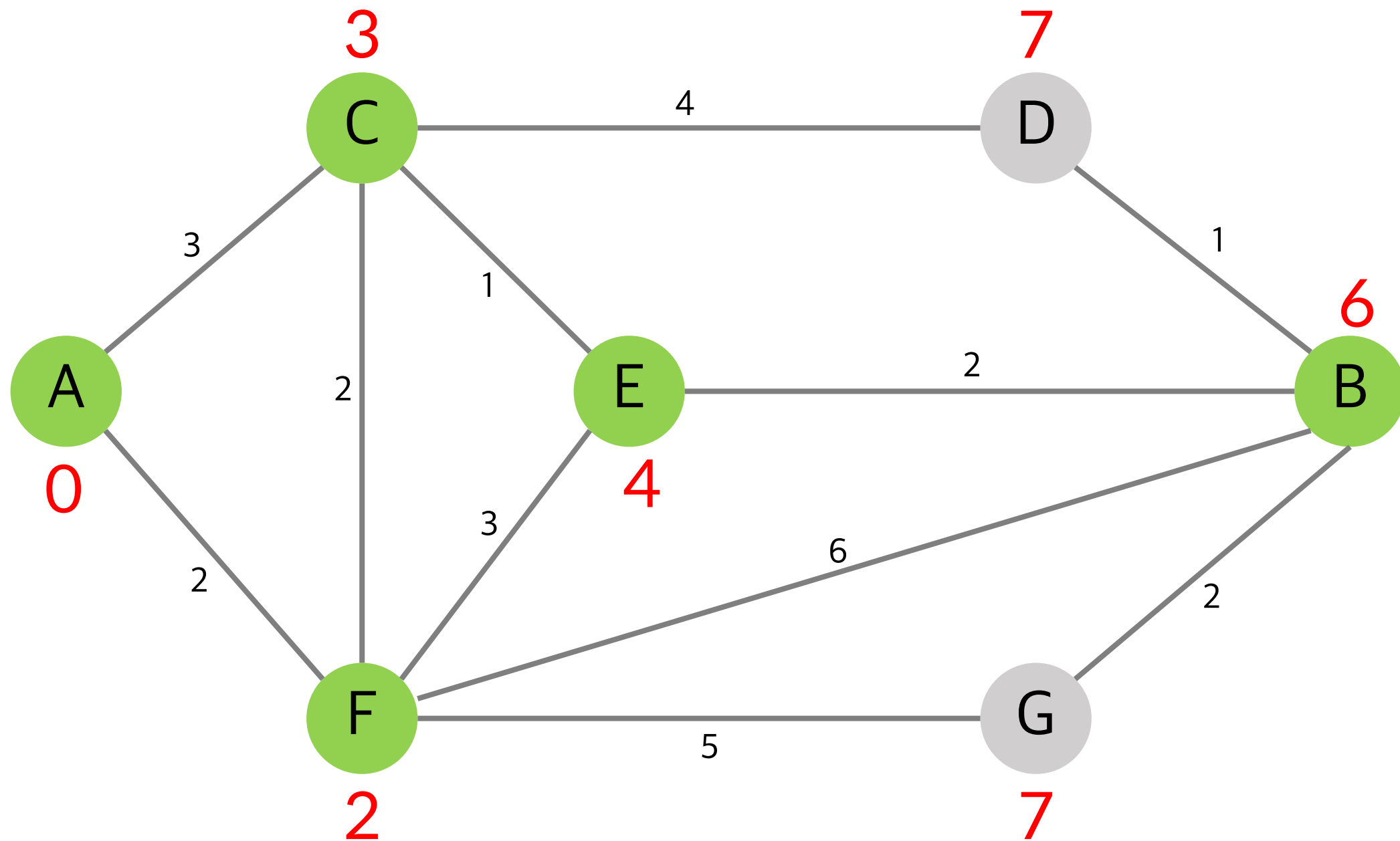


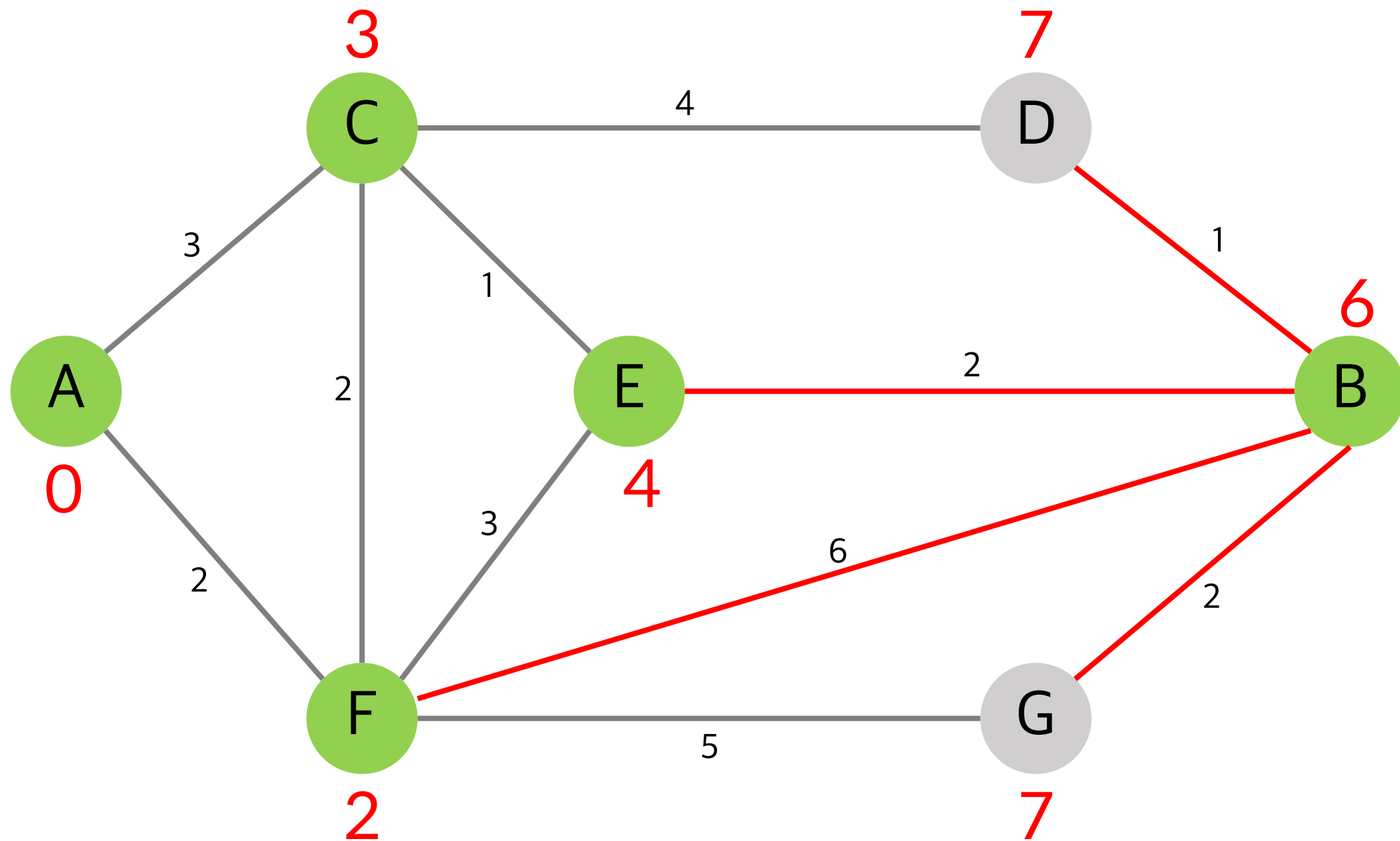


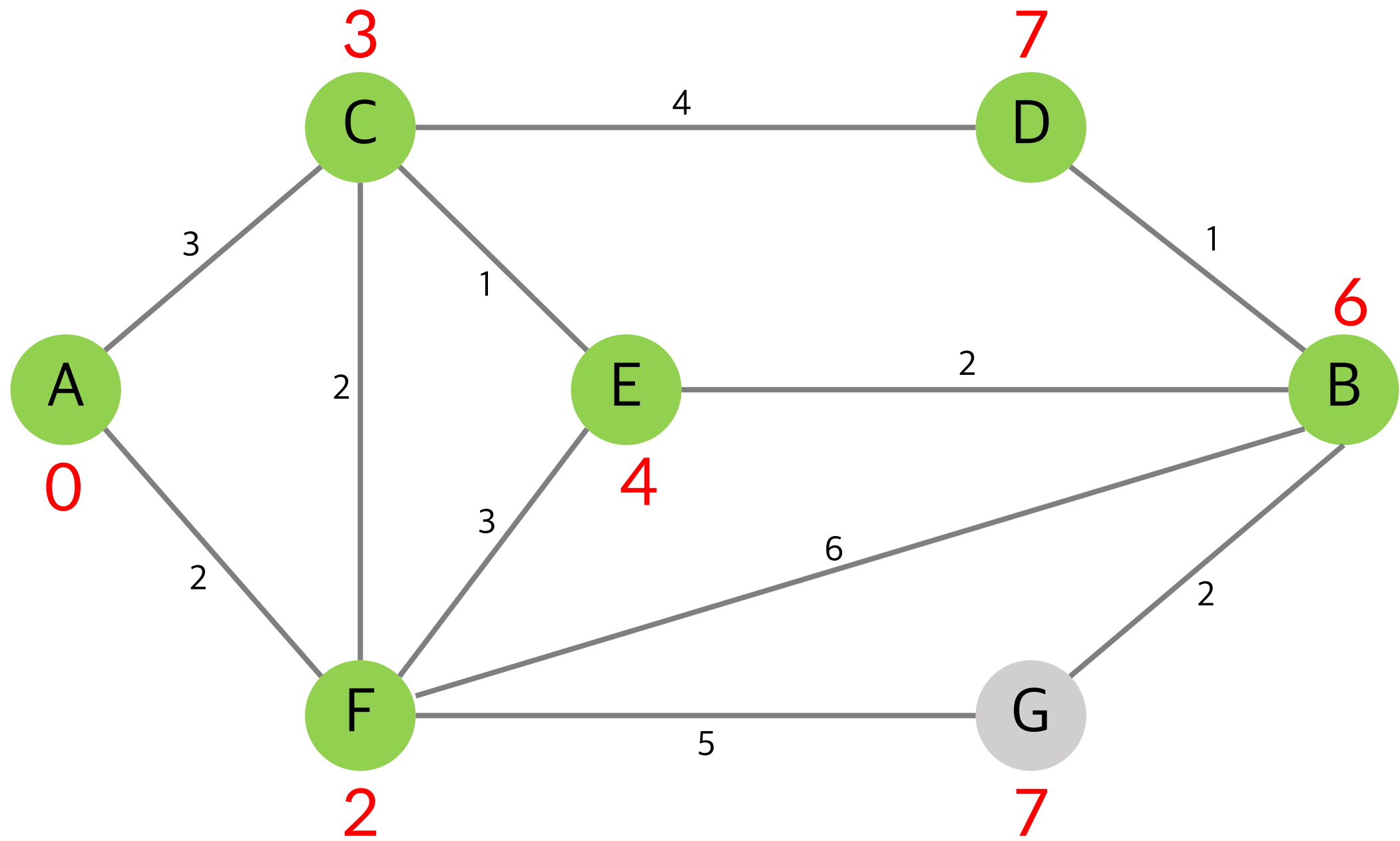


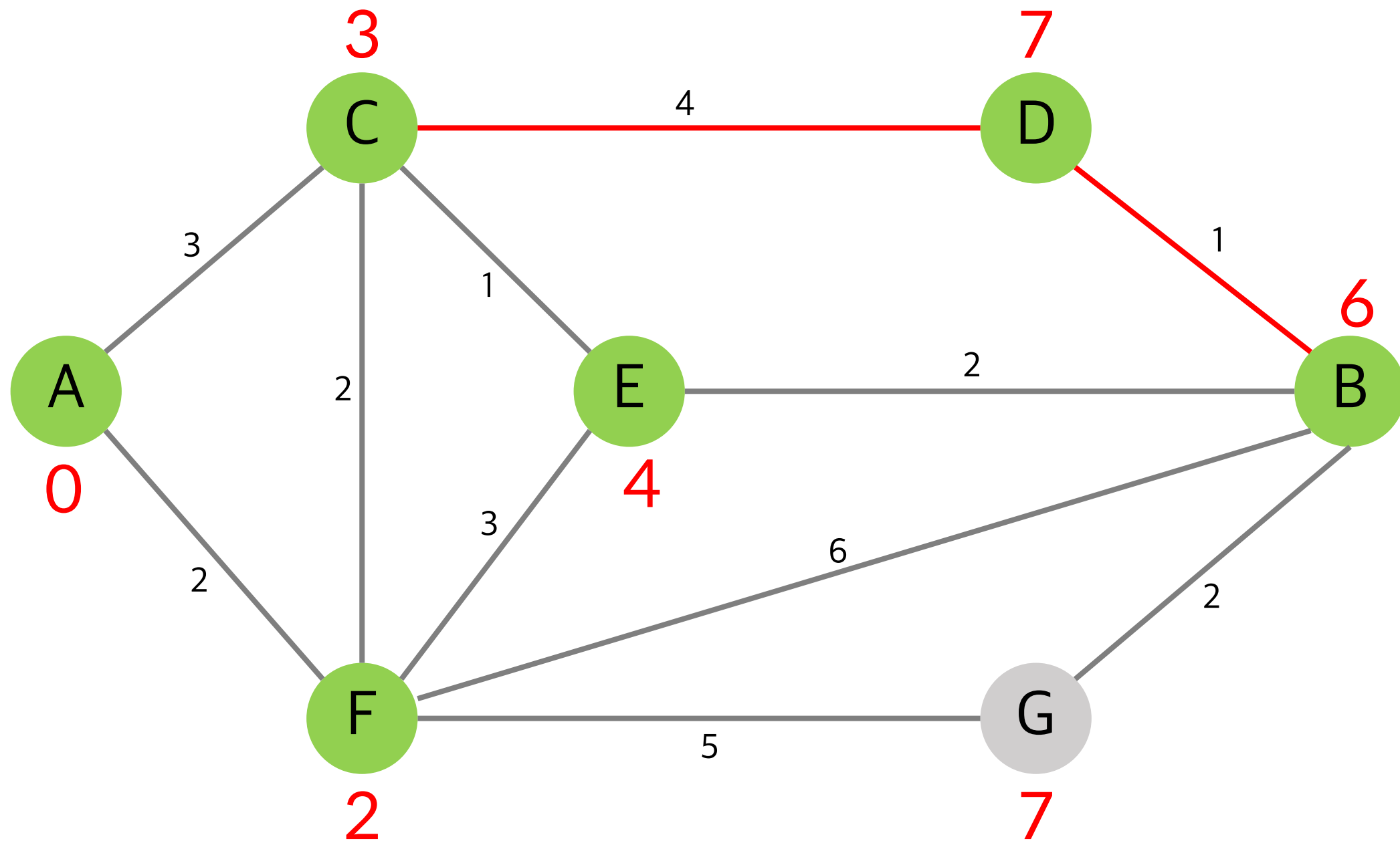


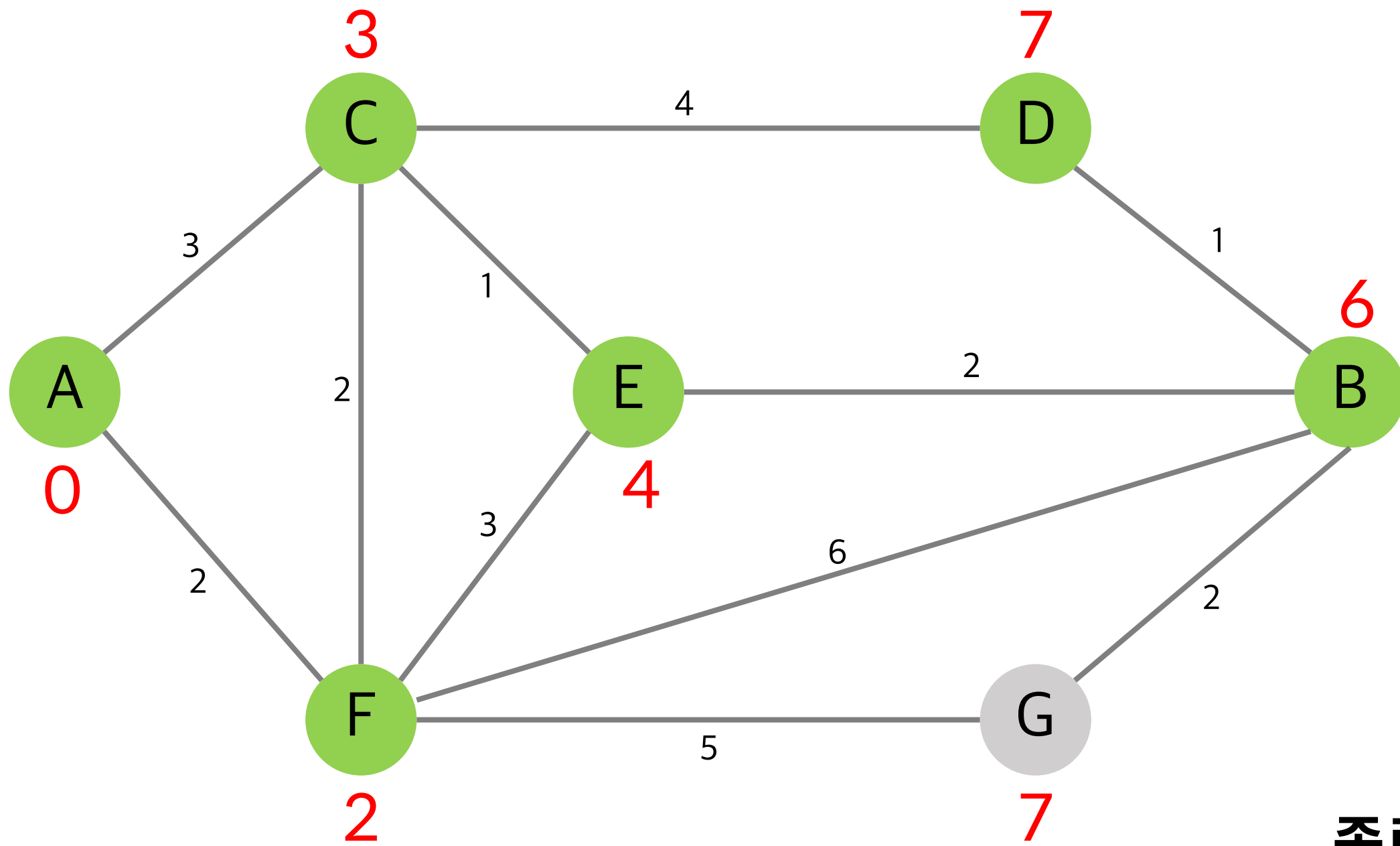




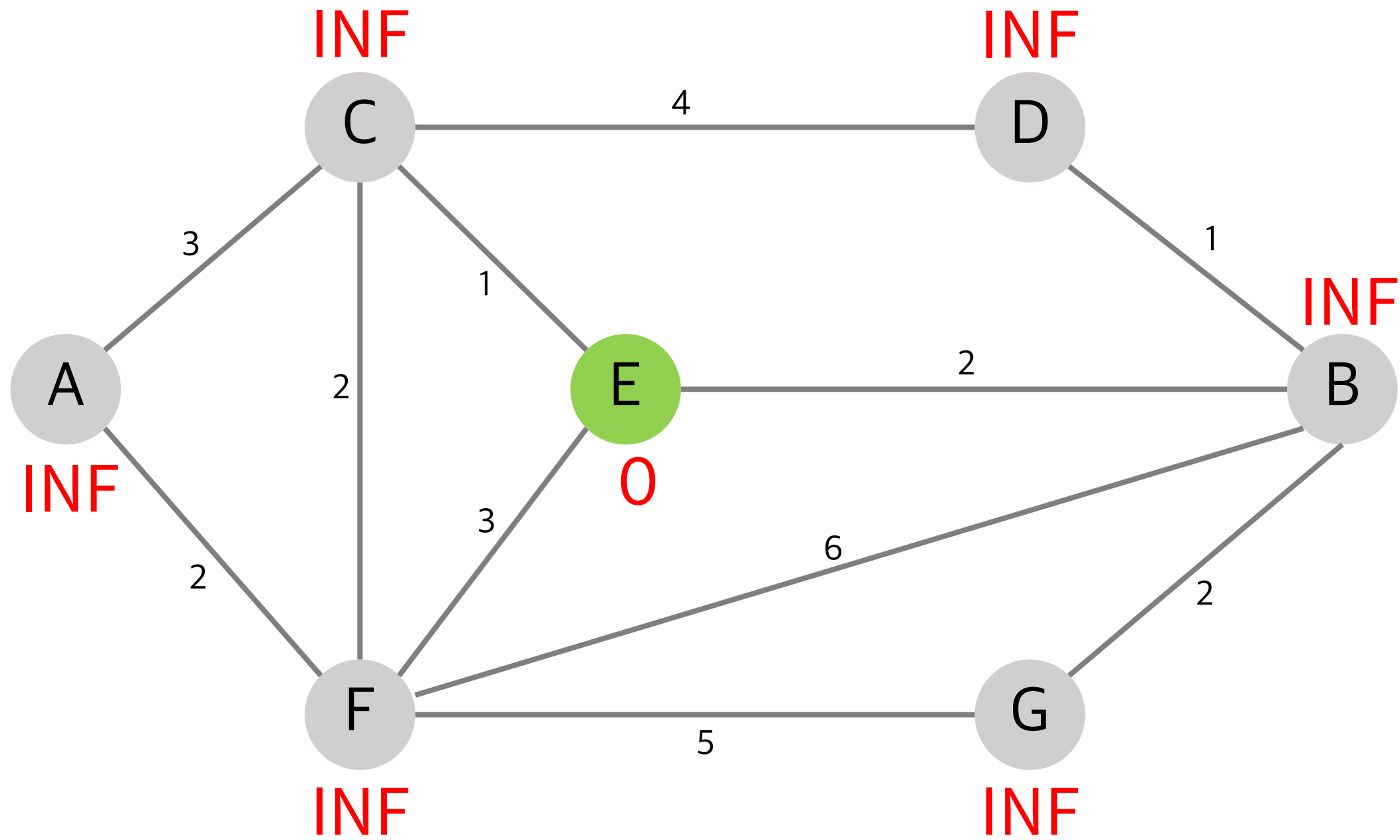


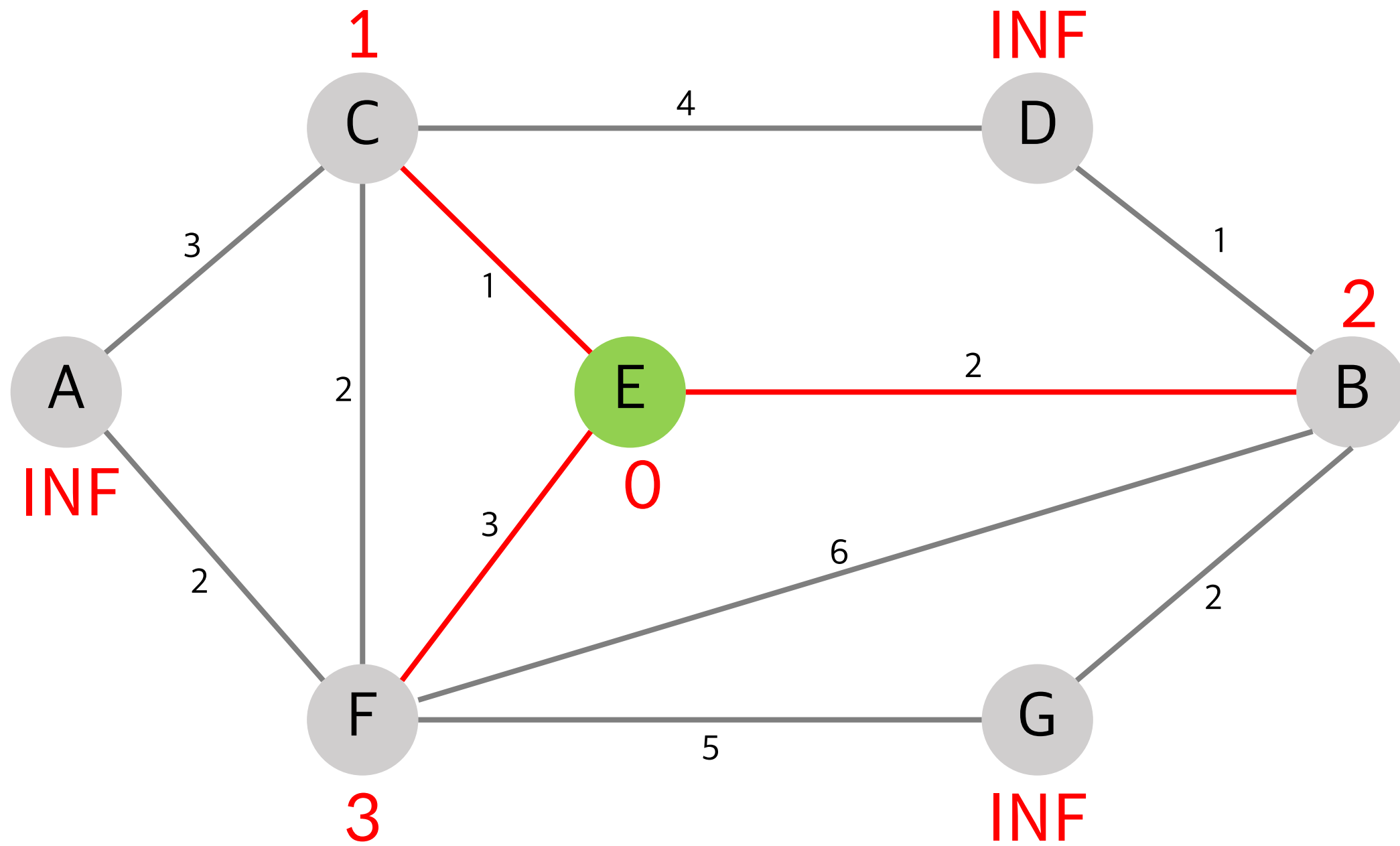


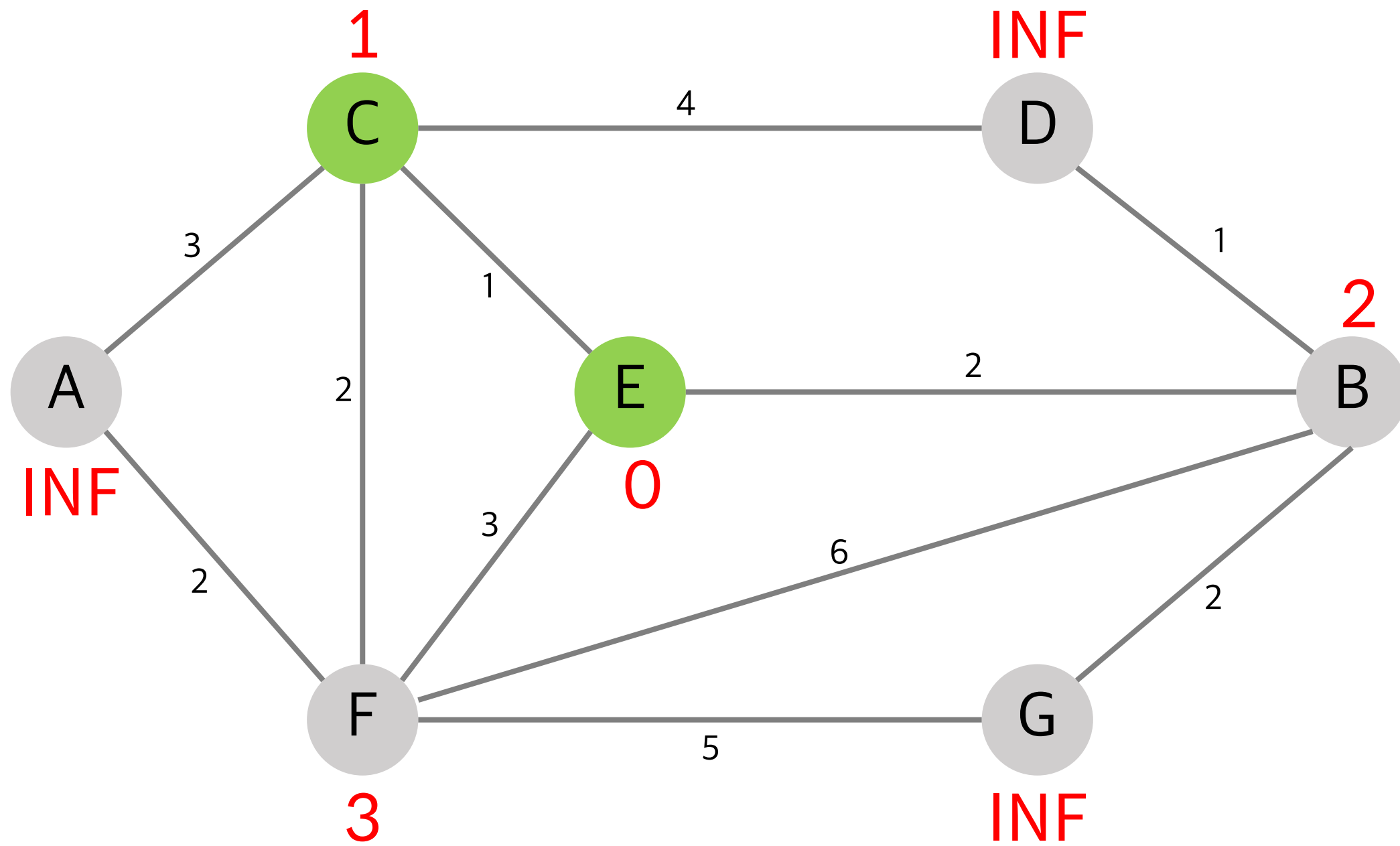


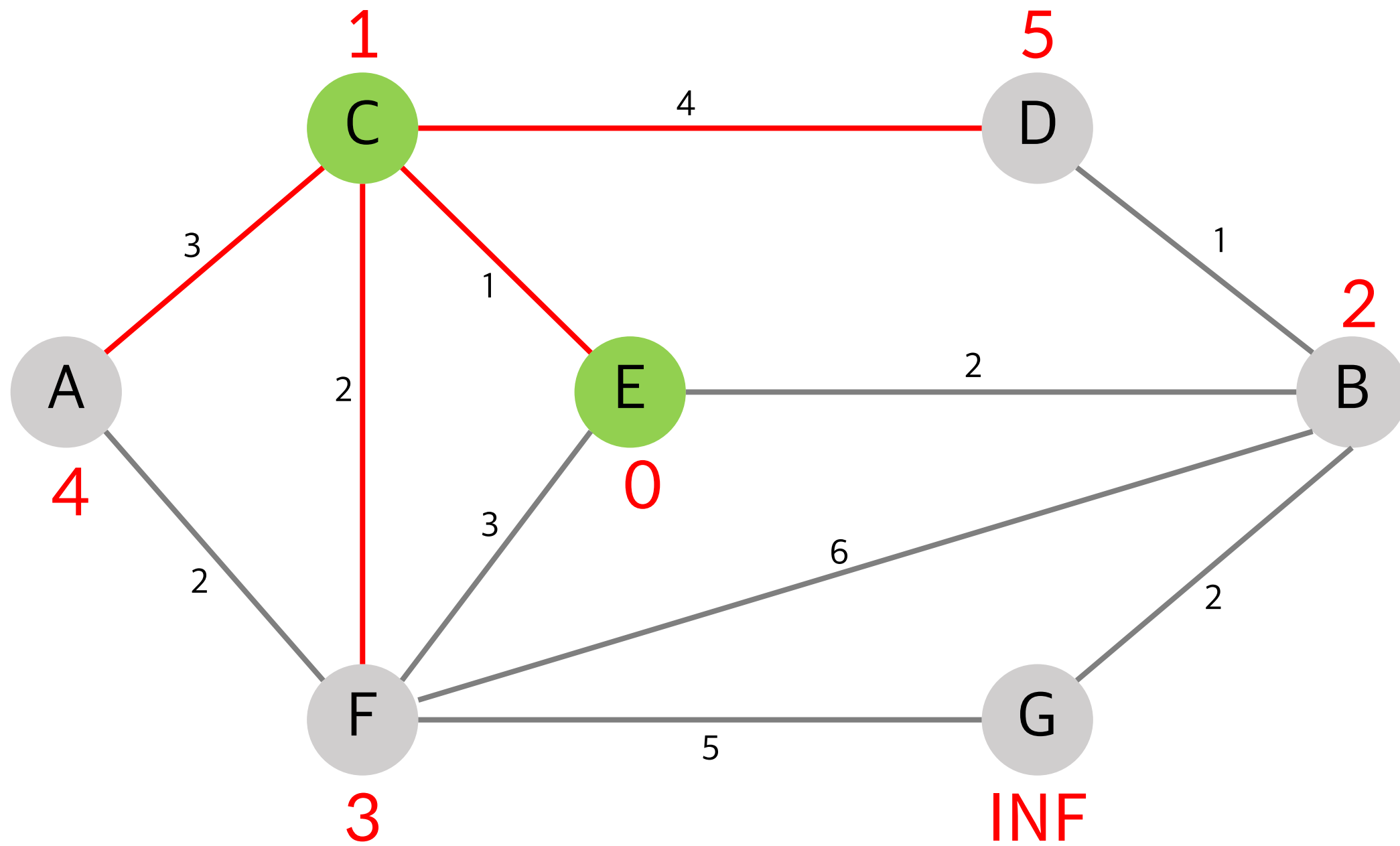


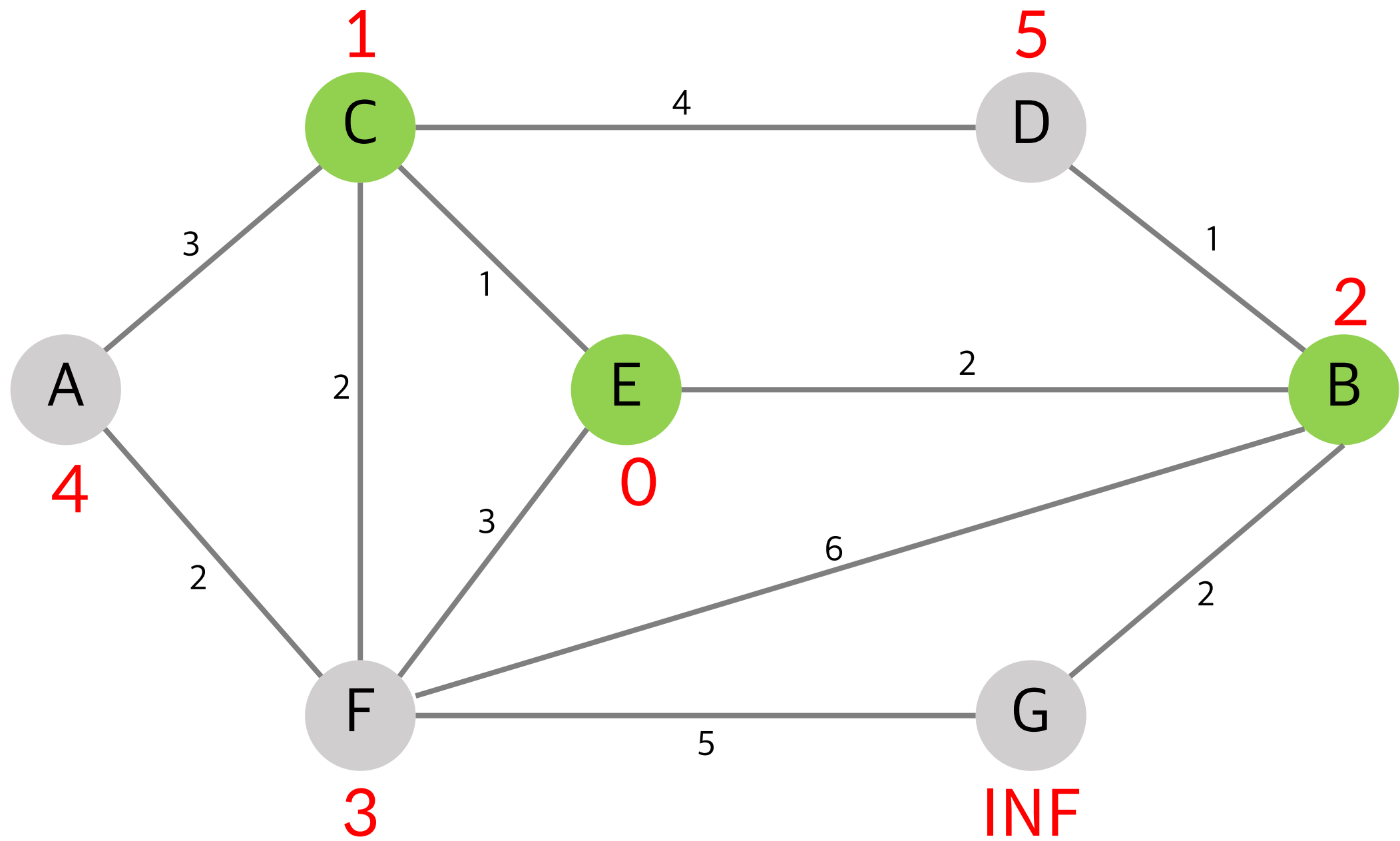
종료.

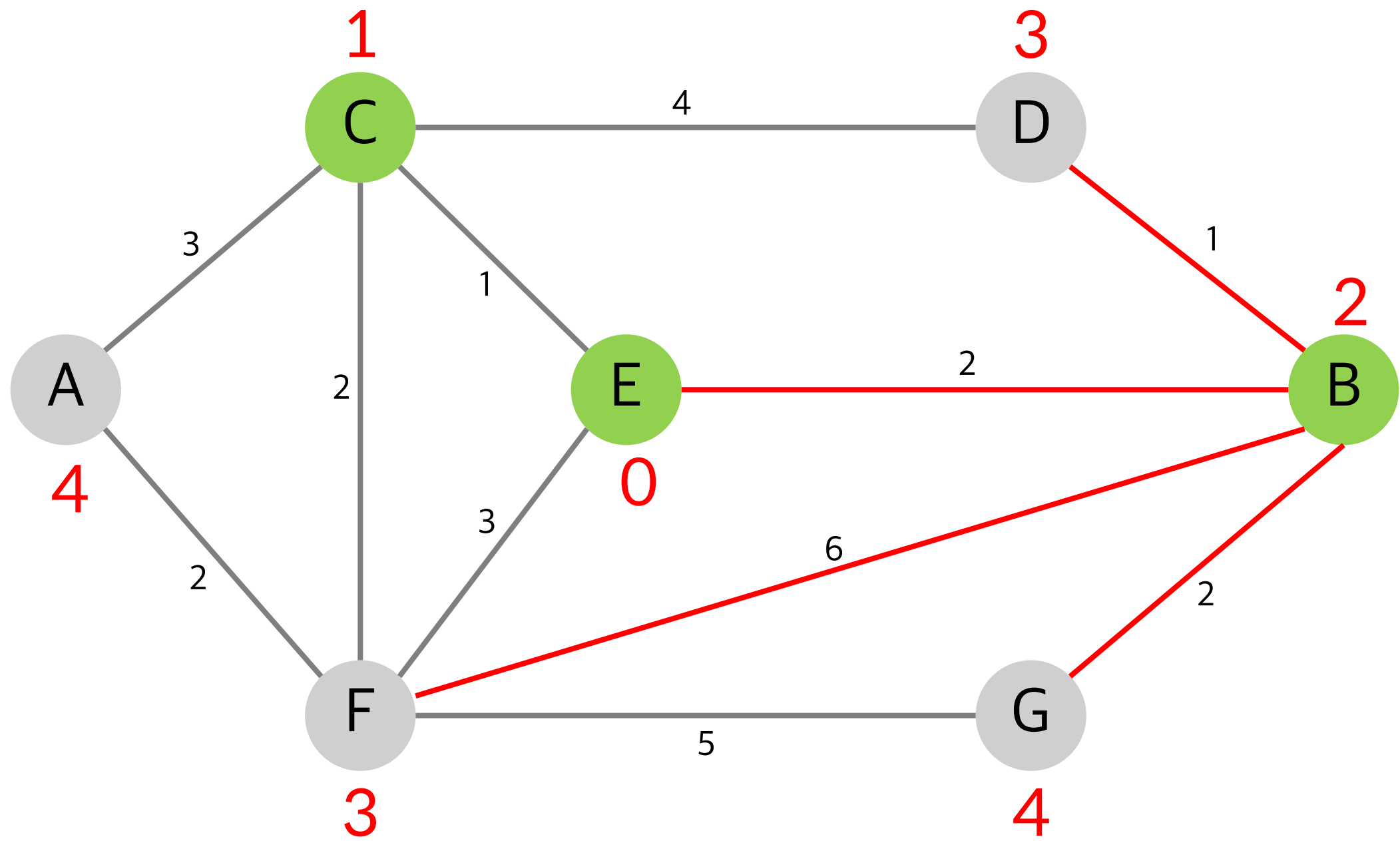


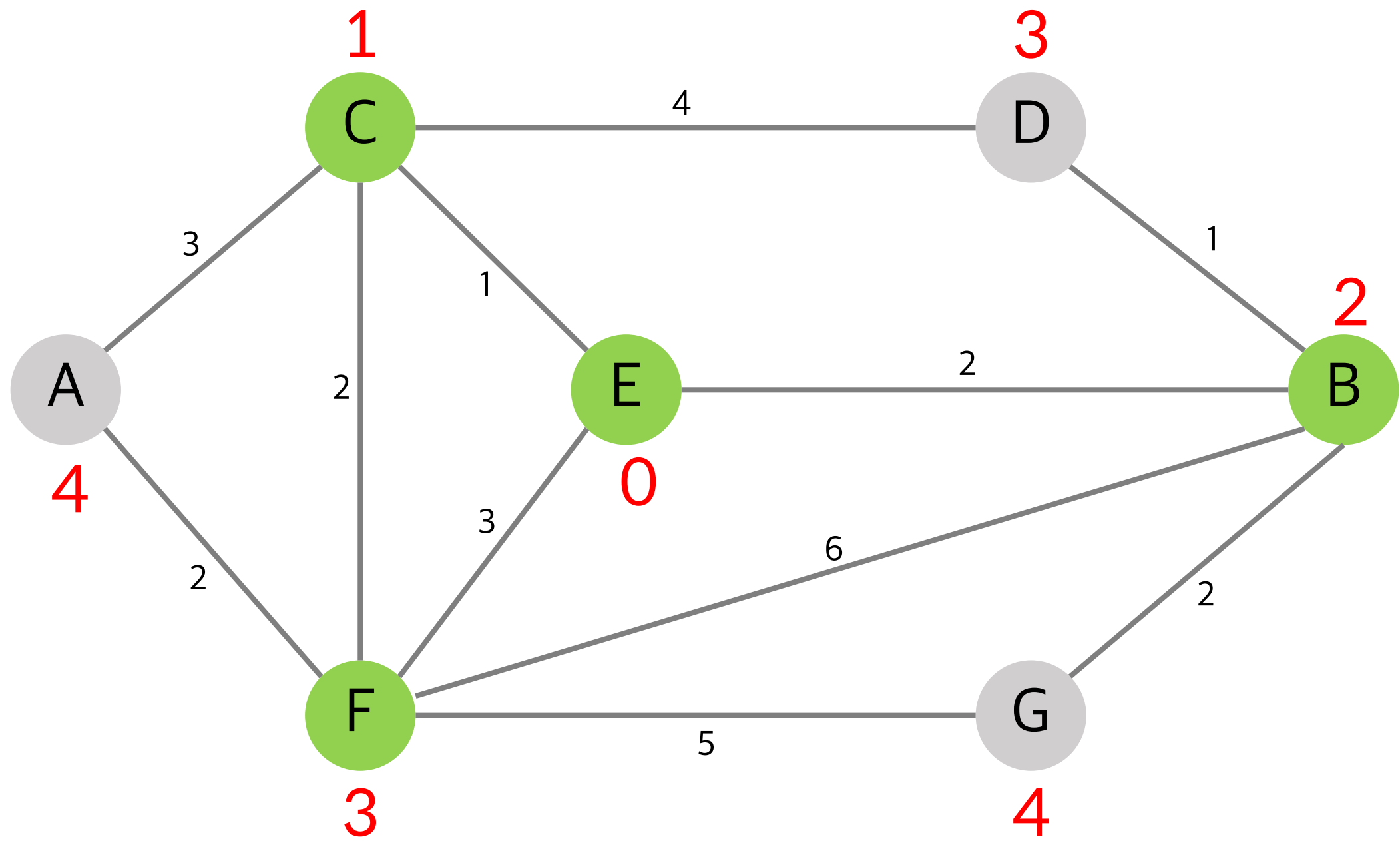


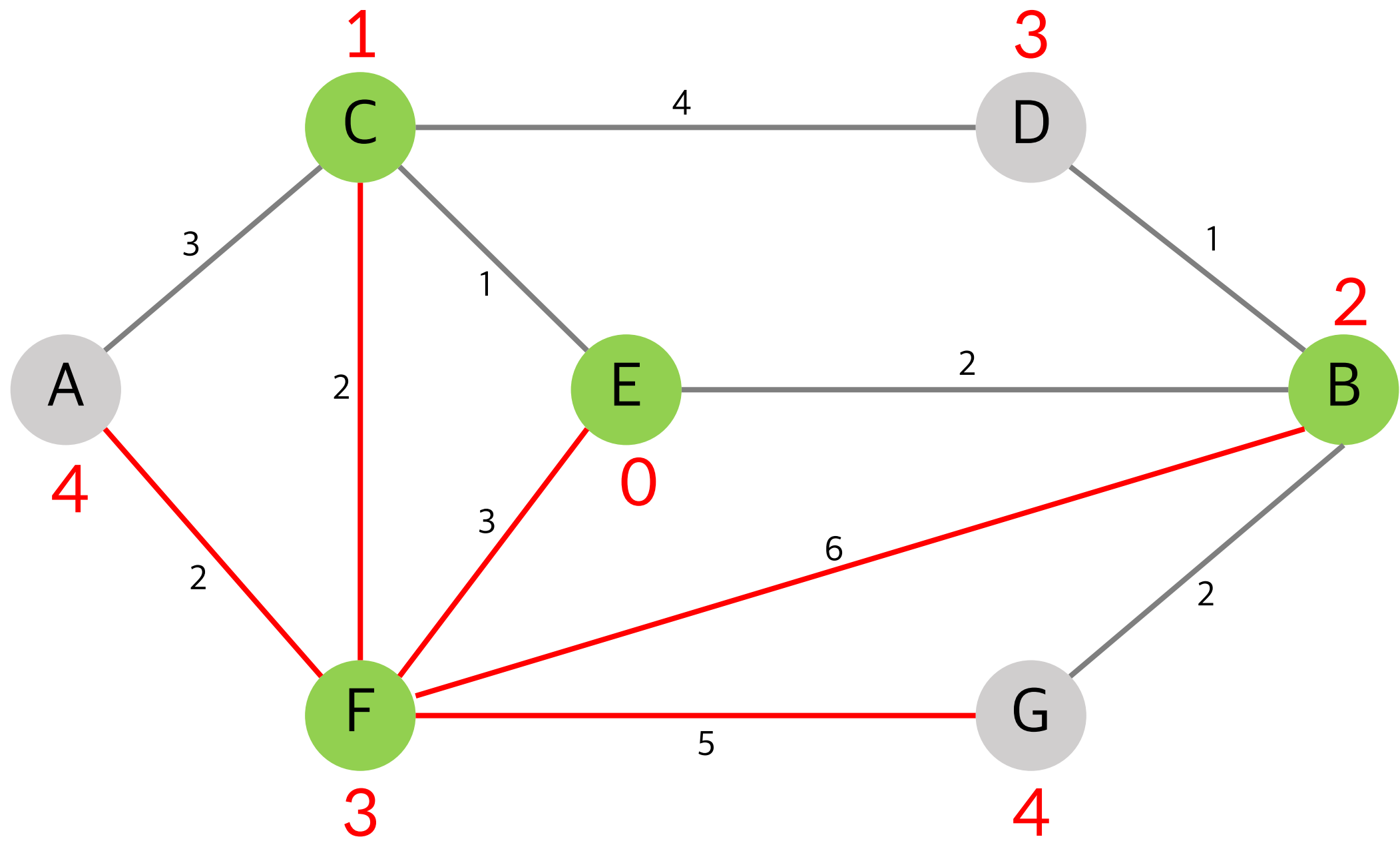


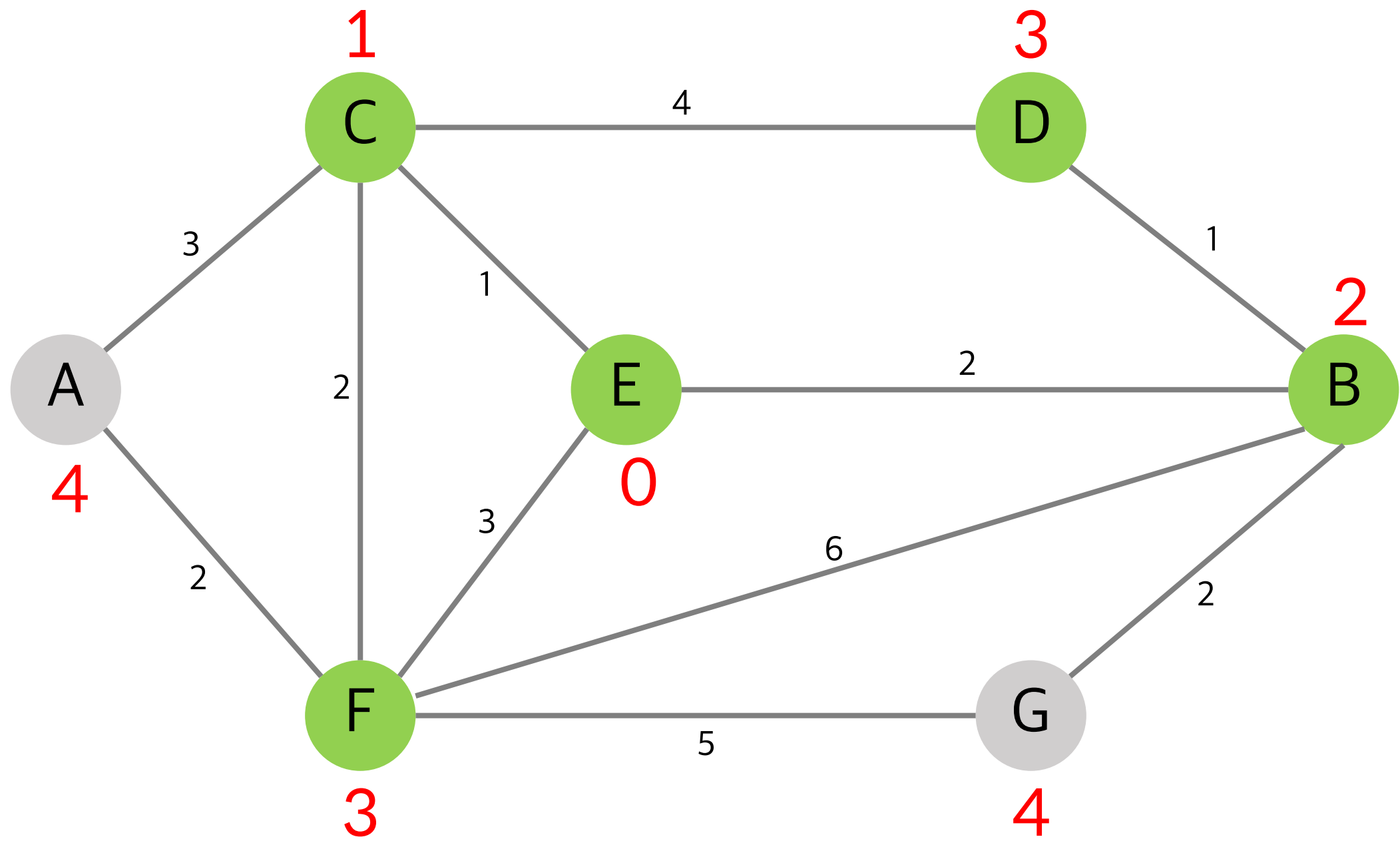


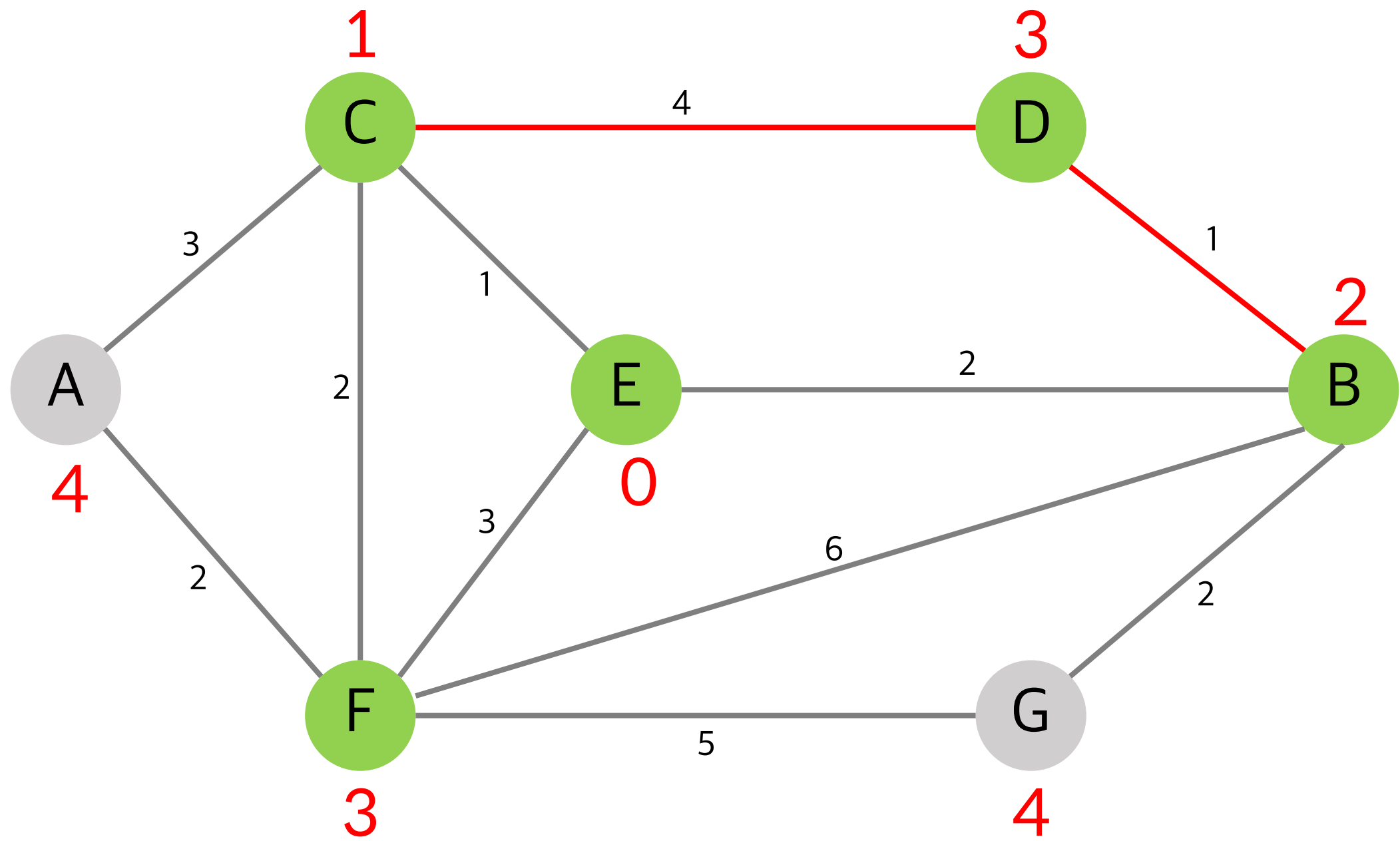


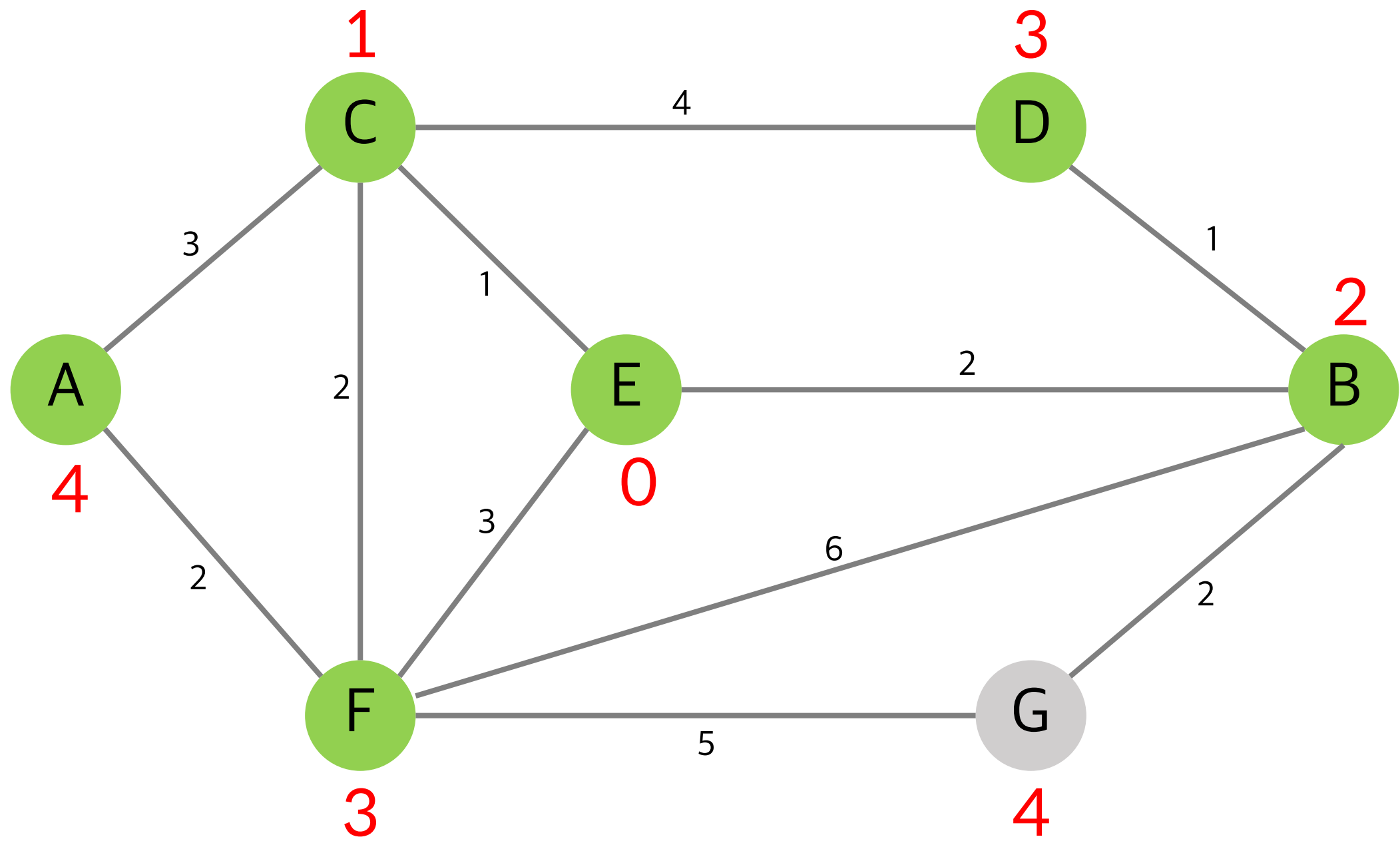


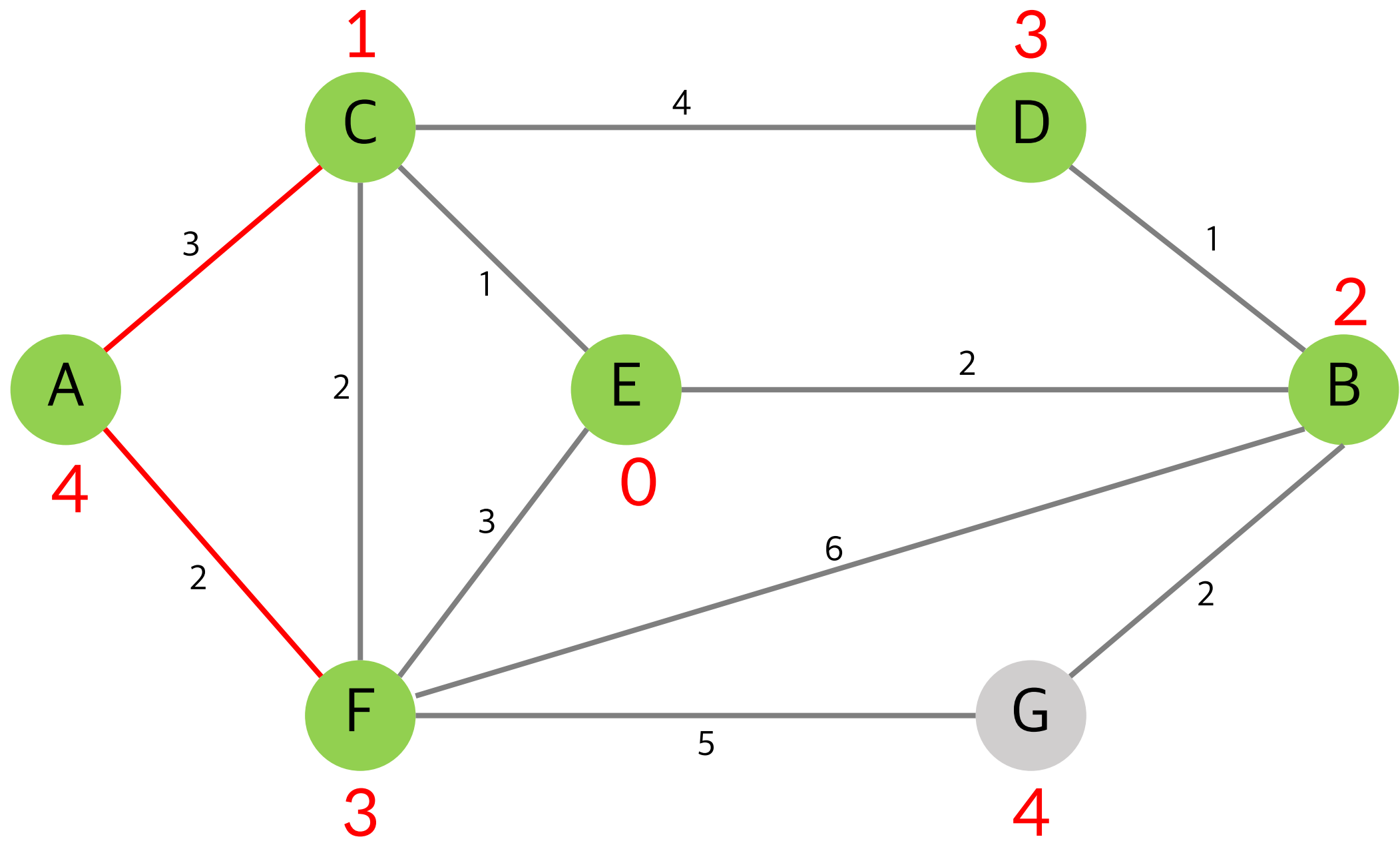


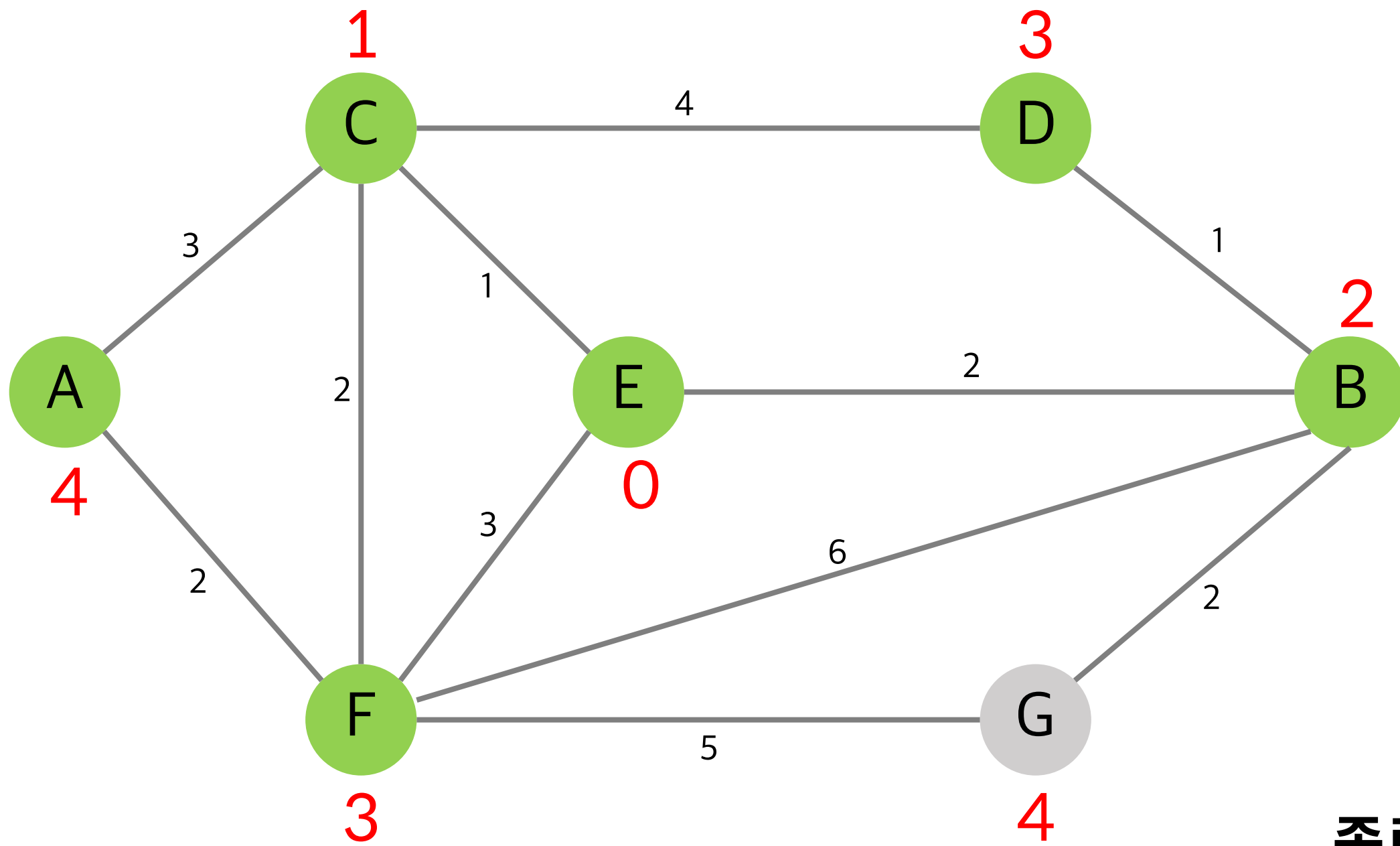












종료.

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>

```
1 #include <iostream>
2
3 #define INF 1e9
4
5 using namespace std;
6
7 const int n = 6;
8
9 int cost[n][n] = {
10     {0,2,5,1,INF,INF},
11     {2,0,3,2,INF,INF},
12     {5,3,0,3,1,5},
13     {1,2,3,0,1,INF},
14     {INF,INF,1,1,0,2},
15     {INF,INF,5,INF,2,0}
16 };
17
18 bool found[n];
19 int distance[n];
20
21 int getSmallIndex(){
22     int min = INF;
23     int index;
24     for(int i = 0; i < n; i++){
25         if(distance[i] < min && !found[i]){
26             min = distance[i];
27             index = i;
28         }
29     }
30     return index;
31 }
32
33 void dijkstra(int start){
34     int i, u, w;
35
36     for(i = 0; i < n; i++){
37         found[i] = false;
38         distance[i] = cost[start][i];
39     }
40
41     found[start] = true;
42     distance[start] = 0;
43
44     for(i = 0; i < n-2; i++){
45         u = getSmallIndex();
46         found[u] = true;
47
48         for(w = 0; w < n; w++){
49             if(!found[w]){
50                 if(distance[u] + cost[u][w] <
51 distance[w]) distance[w] = distance[u] + cost[u][w];
52             }
53         }
54     }
55 }
```

Baekjoon - #11279

👉 **오늘의 문제**

★ #18352, '특정 거리의 도시 찾기'

★ #1753, '최단 경로'

