8/20 - 1

최백준 choi@startlink.io

BFS

BFS

- BFS의 목적은 임의의 정점에서 시작해서, 모든 정점을 한 번씩 방문하는 것이다.

BFS

BFS

- BFS는 최단 거리를 구하는 알고리즘이다.

BFS

BFS

- BFS는 모든 가중치가 1일 때, 최단 거리를 구하는 알고리즘이다.

BFS

BFS

- BFS를 이용해 해결할 수 있는 문제는 아래와 같은 조건을 만족해야 한다

1. 최소 비용 문제이어야 한다
2. 간선의 가중치가 1이어야 한다
3. 정점과 간선의 개수가 적어야 한다. (적다는 것은 문제의 조건에 맞춰서 해결할 수 있다는 것을 의미한다)

BFS

BFS

- BFS를 이용해 해결할 수 있는 문제는 아래와 같은 조건을 만족해야 한다

1. **최소 비용** 문제이어야 한다
2. **간선의 가중치**가 1이어야 한다
3. 정점과 간선의 개수가 적어야 한다. (적다는 것은 문제의 조건에 맞춰서 해결할 수 있다는 것을 의미한다)

- 간선의 가중치가 문제에서 구하라고 하는 최소 비용과 의미가 일치해야 한다
- 즉, 거리의 최소값을 구하는 문제라면 가중치는 거리를 의미해야 하고, 시간의 최소값을 구하는 문제라면 가중치는 시간을 의미해야 한다

BFS

숨바꼭질

https://www.acmicpc.net/problem/1697

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 가장 빠른 시간을 구하는 문제

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: X+1 또는 X-1로 이동 (1초)
2. 순간이동: 2*X로 이동 (1초)

숨바꼭질

https://www.acmicpc.net/problem/1697

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 **가장 빠른 시간**을 구하는 문제

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: X+1 또는 X-1로 이동 (**1초**)
2. 순간이동: 2*X로 이동 (**1초**)

숨바꼭질

https://www.acmicpc.net/problem/1697

- 수빈이의 위치: 5
- 동생의 위치: 17

- 5-10-9-18-17 로 4초만에 동생을 찾을 수 있다.

숨바꼭질

https://www.acmicpc.net/problem/1697

- 큐에 수빈이의 위치를 넣어가면서 이동시킨다
- 한 번 방문한 곳은 다시 방문하지 않는 것이 좋기 때문에, 따로 배열에 체크하면서 방문

숨바꼭질

https://www.acmicpc.net/problem/1697

- check[i] = i를 방문했는지
- dist[i] = i를 몇 번만에 방문했는지

숨바꼭질

https://www.acmicpc.net/problem/1697

- now -> next를 갔다고 한다면

```
if (check[next] == false) {
    q.push(next);
    check[next] = true;
    dist[next] = dist[now] + 1;
}
```

숨바꼭질

https://www.acmicpc.net/problem/1697

```
check[n] = true;
dist[n] = 0;
queue<int> q;
q.push(n);
while (!q.empty()) {
    int now = q.front();
    q.pop();
    if (now-1 >= 0) {
        if (check[now-1] == false) {
            q.push(now-1);
            check[now-1] = true;
            dist[now-1] = dist[now] + 1;
        }
    }
    if (now+1 < MAX) {
        if (check[now+1] == false) {
            q.push(now+1);
            check[now+1] = true;
            dist[now+1] = dist[now] + 1;
        }
    }
    if (now*2 < MAX) {
        if (check[now*2] == false) {
            q.push(now*2);
            check[now*2] = true;
            dist[now*2] = dist[now] + 1;
        }
    }
}
```

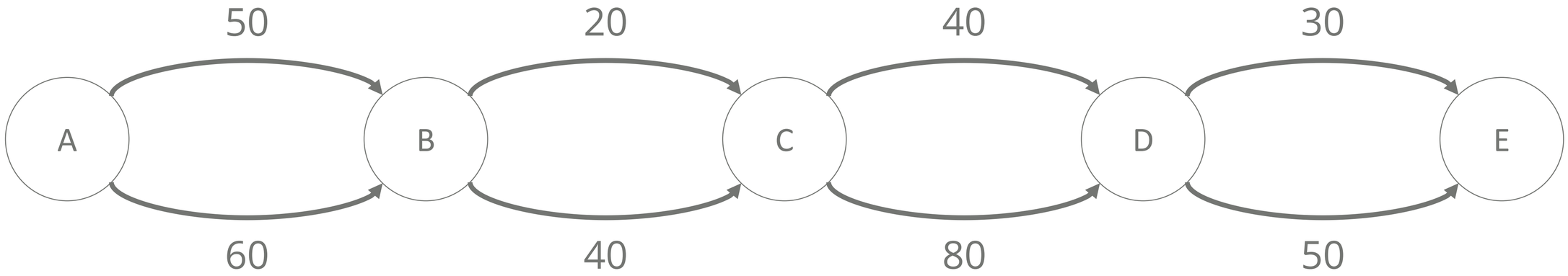
숨바꼭질

https://www.acmicpc.net/problem/1697

- 소스: http://codeplus.codes/136a408b93ec44578aba39c7e8a7db12

BFS

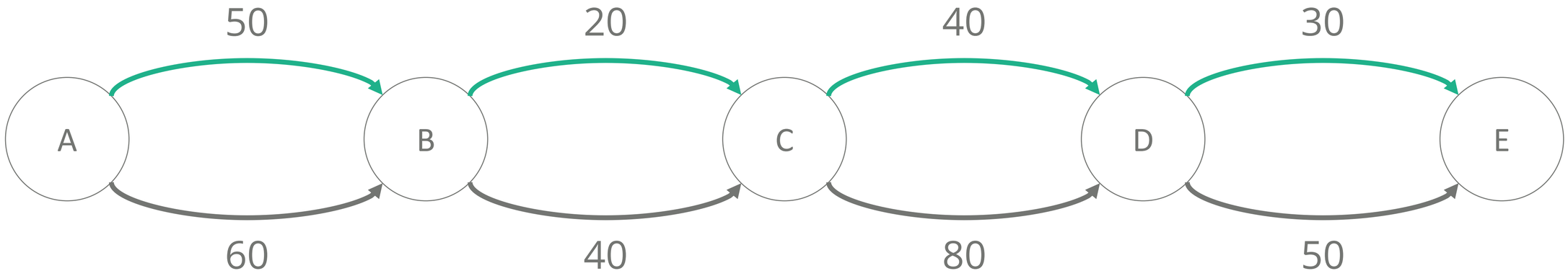
BFS



- A에서 E로 가는 가장 빠른 길은 무엇일까?

BFS

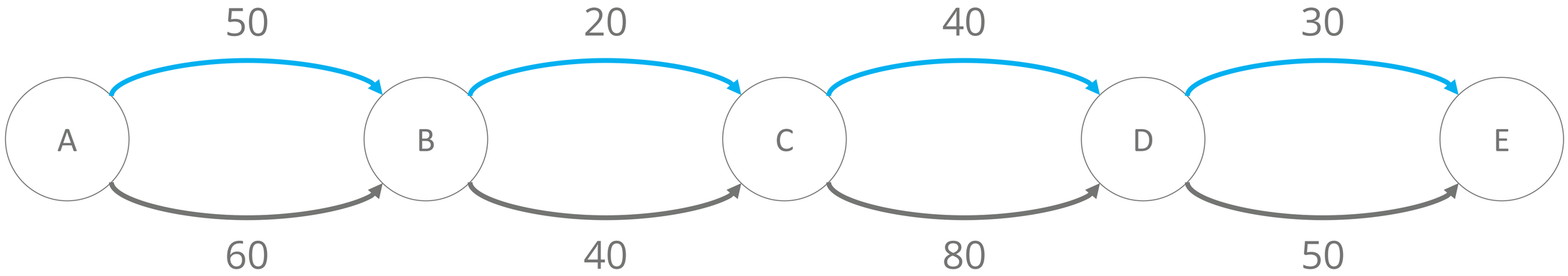
BFS



- A에서 E로 가는 가장 빠른 길은 무엇일까?
- A에서 B로 가는 가장 빠른 길 + B에서 E로 가는 가장 빠른 길

BFS

BFS

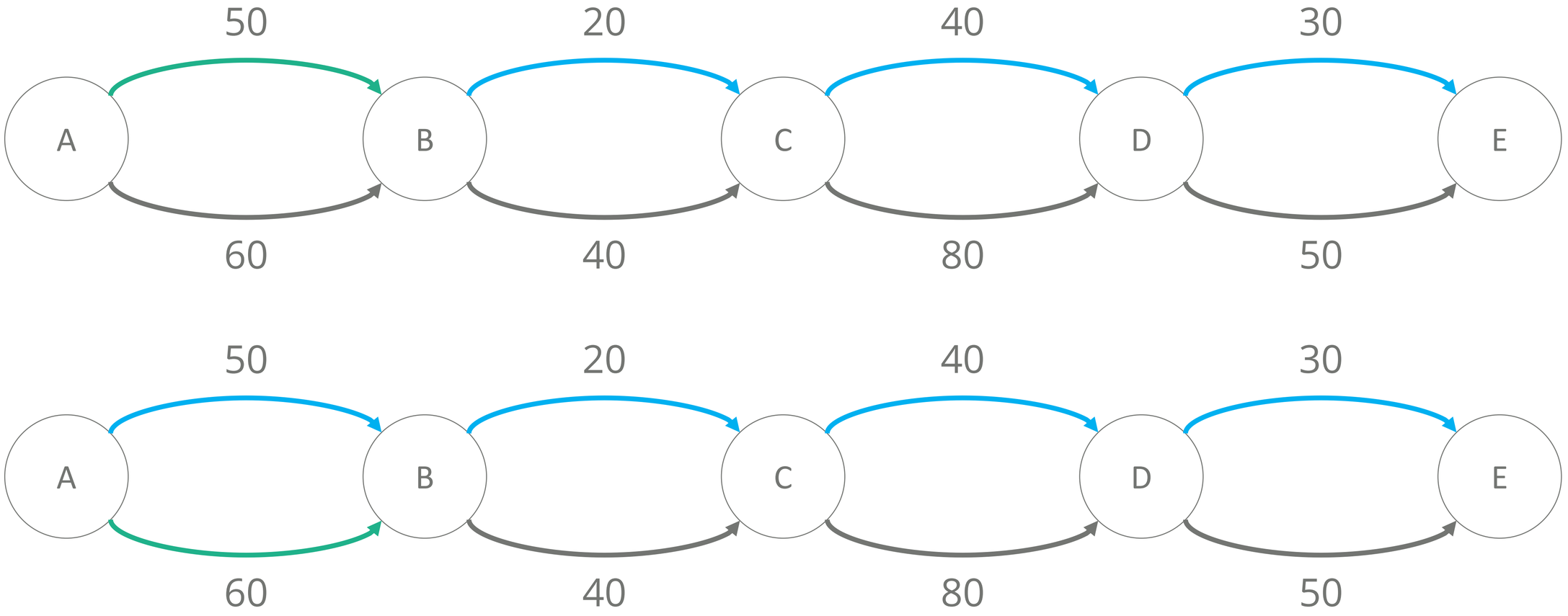


- A에서 E로 가는 가장 빠른 길은 무엇일까?
- 단, 파란 간선은 한 번만 사용할 수 있다.

BFS

BFS

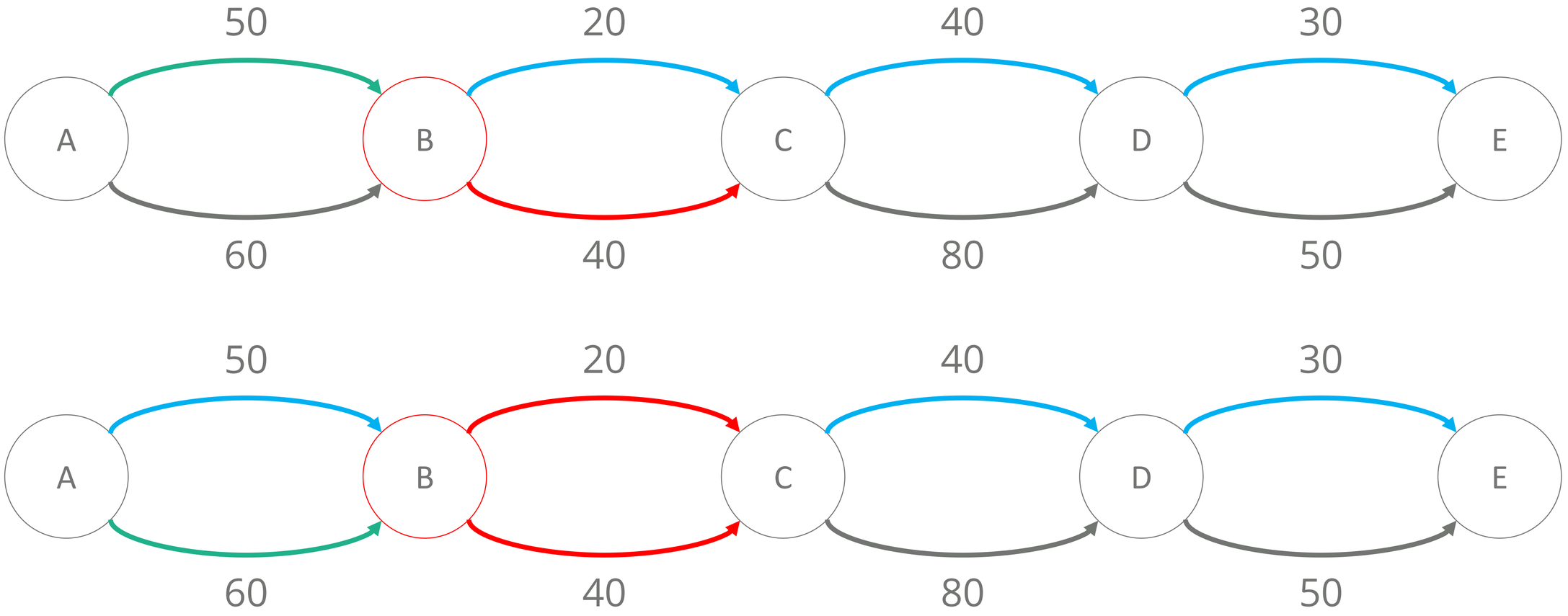
- 파란 간선을 한 번만 사용할 수 있다면, 위 B와 아래 B는 같은 정점이라고 할 수 없다



BFS

BFS

- 위 B와 아래 B에서 이동할 수 있는 방법이 다르기 때문에 같은 정점이 아니다



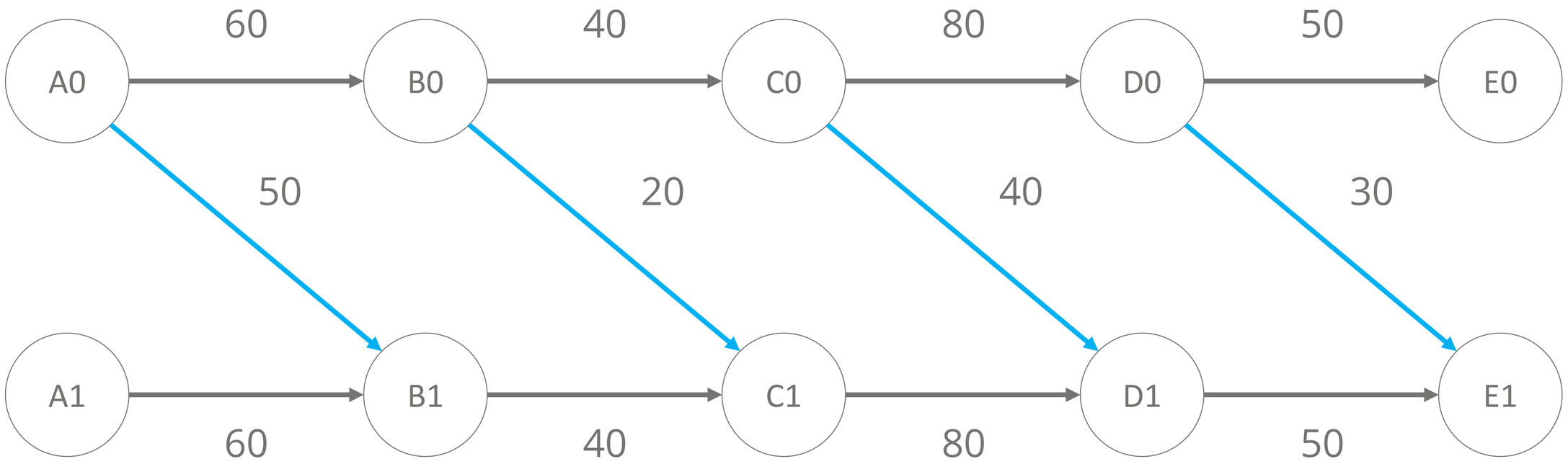
BFS

BFS

- 위 B와 아래 B를 다르다고 하는 기준은 파란 간선을 사용한 횟수이다
- 따라서, 정점을 파란 간선을 사용한 횟수를 기준으로 나눌 수 있다.

BFS

BFS



이모티콘

https://www.acmicpc.net/problem/14226

- 화면에 이모티콘은 1개다
- 할 수 있는 연산
  - 화면에 있는 이모티콘을 모두 복사해서 클립보드에 저장
  - 클립보드에 있는 모든 이모티콘을 화면에 붙여넣기
  - 화면에 있는 이모티콘 중 하나를 삭제
- S개의 이모티콘을 만드는데 걸리는 시간의 최소값을 구하는 문제

이모티콘

https://www.acmicpc.net/problem/14226

- BFS에서 하나의 정점이 서로 다른 두 개의 정보를 저장하고 있으면 안된다
- 화면에 있는 이모티콘의 개수가 5개인 경우
- 클립보드에 있는 이모티콘의 개수에 따라서, 클립보드에서 복사하기 연산의 결과가 다르다
- 즉, 화면에 이모티콘의 개수 s와 클립보드에 있는 이모티콘의 개수 c가 중요하다

이모티콘

https://www.acmicpc.net/problem/14226

- 복사: (s, c) -> (s, s)
- 붙여넣기: (s, c) -> (s+c, c)
- 삭제: (s, c) -> (s-1, c)
- $2 \leq S \leq 1{,}000$ 이기 때문에 BFS 탐색으로 가능하다.

이모티콘

https://www.acmicpc.net/problem/14226

- 소스: http://codeplus.codes/fd8b0ebe1c0e4b0a91ee5134ae7fb6b5