BFS

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/3e1de147337841668fe7af0dd257ae8e

숨바꼭질 4

https://www.acmicpc.net/problem/13913

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 가장 빠른 시간과 **이동하는 방법**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 4

https://www.acmicpc.net/problem/13913

- now -> next를 갔다고 한다면

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

숨바꼭질 4

https://www.acmicpc.net/problem/13913

- now -> next를 갔다고 한다면

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

숨바꼭질 4

https://www.acmicpc.net/problem/13913

- from[i] = 어디에서 왔는지
- 의미: from[i] -> i
- N에서 K를 가는 문제 이기 때문에
- K부터 from을 통해서 N까지 가야한다.
- 즉, 역순으로 저장되기 때문에, 다시 역순으로 구하는 것이 필요하다.

숨바꼭질 4

https://www.acmicpc.net/problem/13913

```
void print(int n, int m) {
    if (n != m) {
        print(n, from[m]);
    }
    cout << m << ' ';
}
```

숨바꼭질 4

https://www.acmicpc.net/problem/13913

```
stack<int> ans;
for (int i=m; i!=n; i=from[i]) {
    ans.push(i);
}
ans.push(n);
while (!ans.empty()) {
    cout << ans.top() << ' ';
    ans.pop();
}
cout << '\n';
```

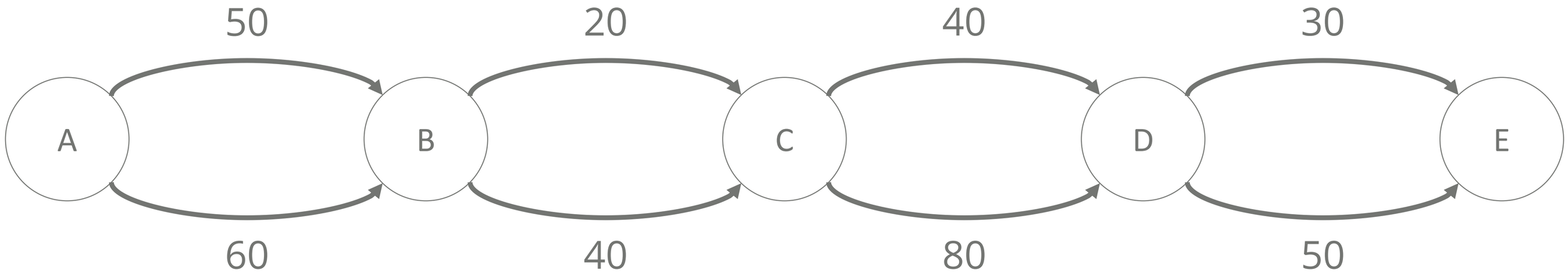
숨바꼭질 4

https://www.acmicpc.net/problem/13913

- 소스: http://codeplus.codes/74337f7a446944948d0a209ee4972ffa

BFS

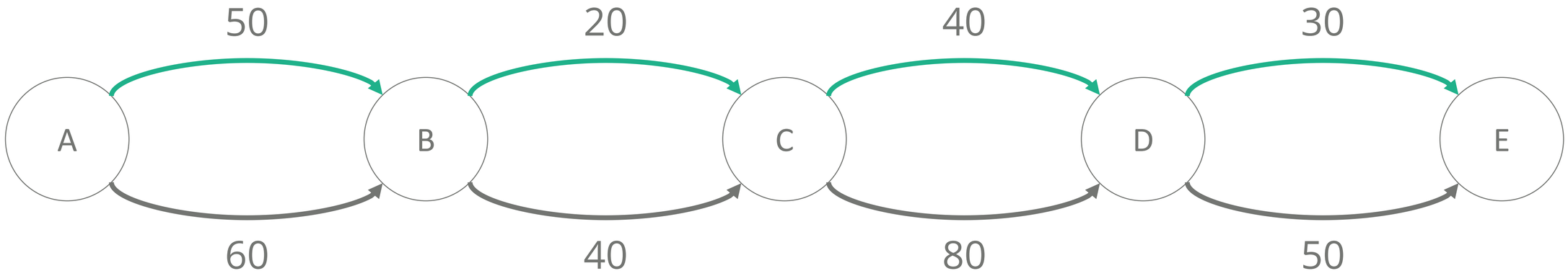
BFS



- A에서 E로 가는 가장 빠른 길은 무엇일까?

BFS

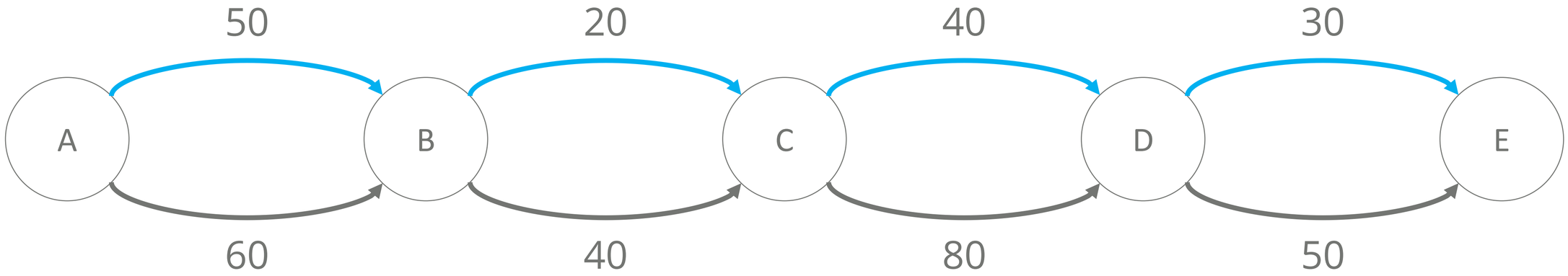
BFS



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

BFS

BFS

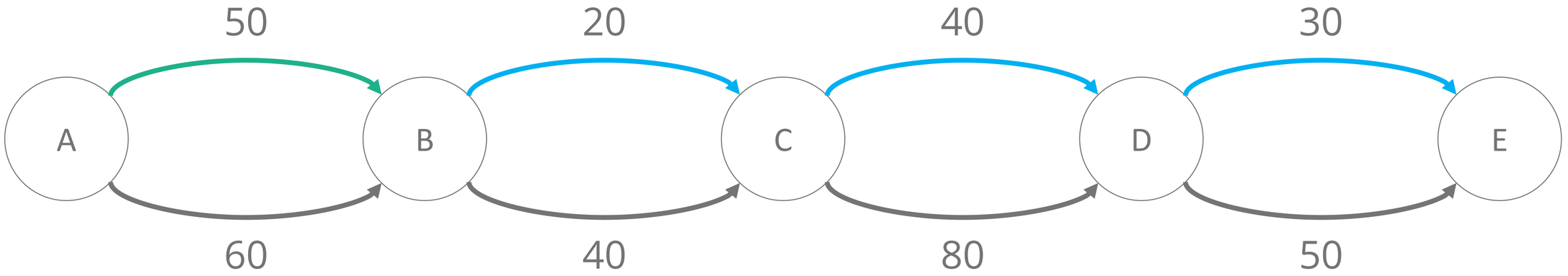


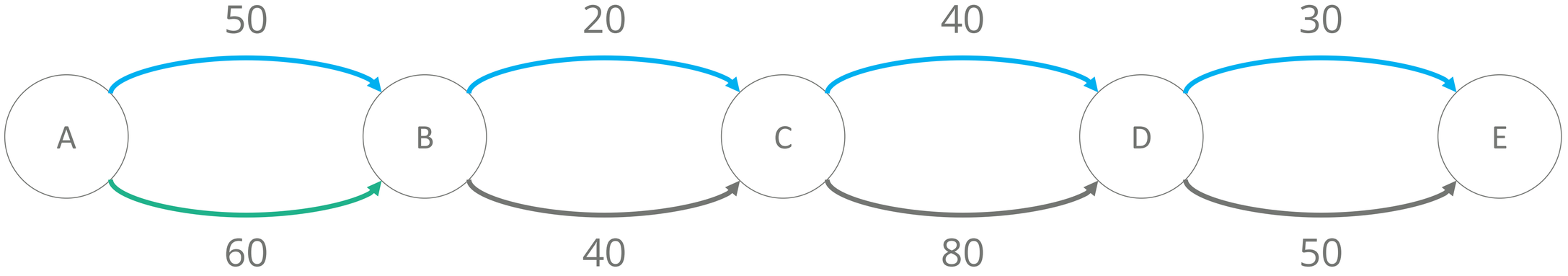
- A에서 E로 가는 가장 빠른 길은 무엇일까?
- 단, 파란 간선은 한 번만 사용할 수 있다.

BFS

BFS

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

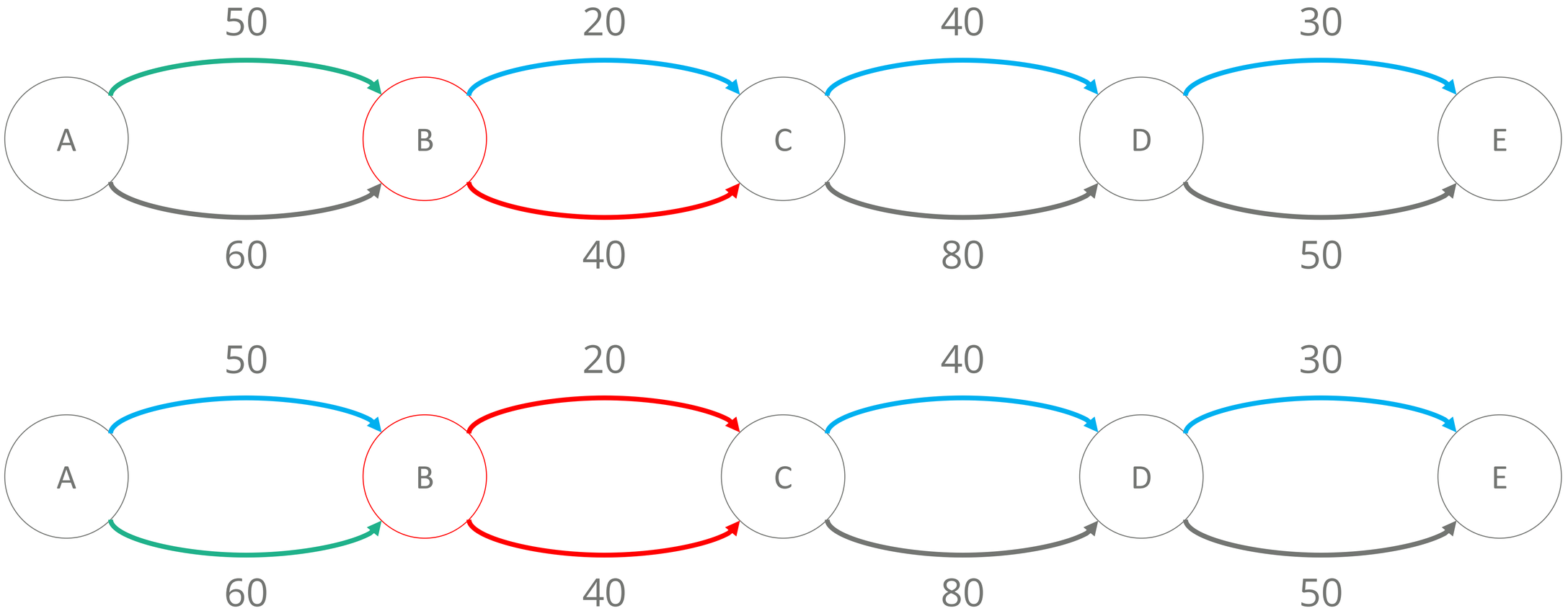




BFS

BFS

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



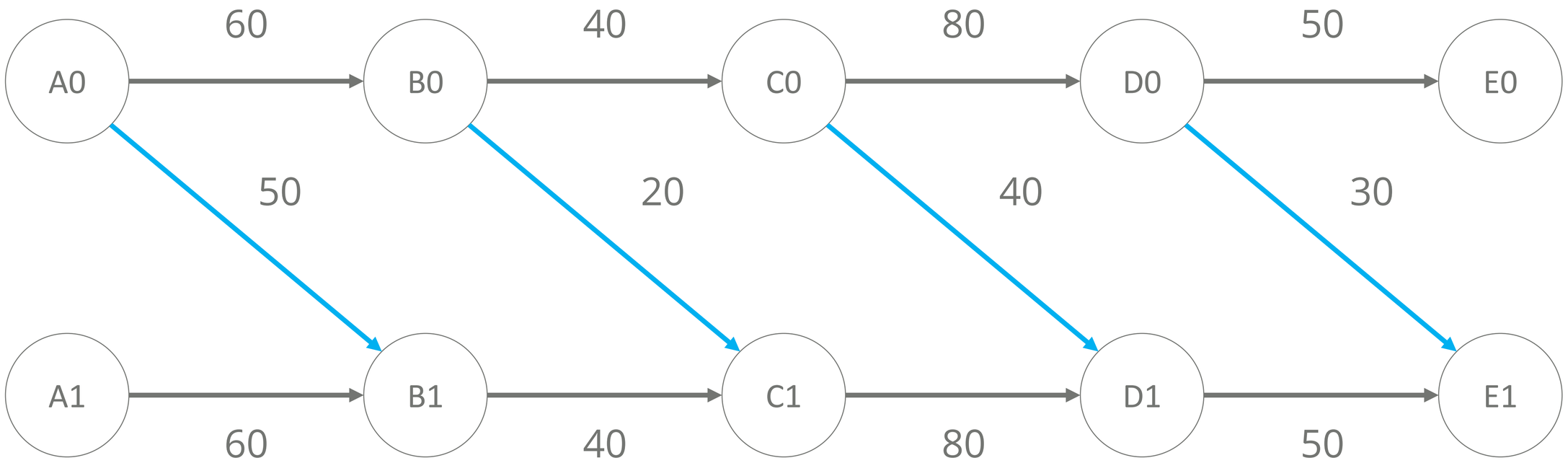
BFS

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/489a4b9a719c4958bbe6a02a7d860f74

덱 사용하기

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 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 (**0초**)

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5					

- 1초:

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10				

- 1초:

4	6				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10				

- 1초:

4	6				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11		

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11		

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	

- 2초:

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8

- 2초:

3					

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8						

- 2초:

3					

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12					

- 2초:

3	7				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12					

- 2초:

3	7				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18	16			

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 이런식으로 BFS를 진행한다.
- 큐는 총 2개가 필요하다.
- 현재 큐/다음 큐

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 덱을 사용해 순간 이동은 덱의 앞에, 걷기는 덱의 뒤에 넣는 방법도 있다.
- 현재 큐에 넣는 것을 덱의 앞에
- 다음 큐에 넣는 것을 덱의 뒤에

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 큐 소스: http://codeplus.codes/c2201205287445dd9fd2ded648e7b5d4
- 덱 소스: http://codeplus.codes/48d990040011464896774b54eb4bb56e

알고스팟

https://www.acmicpc.net/problem/1261

- 미로는 N*M크기이고, 총 1*1크기의 방으로 이루어져 있다
- 빈 방은 자유롭게 다닐 수 있지만, 벽은 부수지 않으면 이동할 수 없다
- (x, y)에 있을 때, 이동할 수 있는 방은 (x+1, y), (x-1, y), (x, y+1), (x, y-1) 이다
- (1, 1)에서 (N, M)으로 이동하려면 벽을 최소 몇 개 부수어야 하는지 구하는 문제

알고스팟

https://www.acmicpc.net/problem/1261

- 처음 상태

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

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 부수지 않고 이동할 수 있는 곳

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

0	0				
0					
0	0				

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 1개 부수고 이동할 수 있는 곳

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

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

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 2개 부수고 이동할 수 있는 곳

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

0	0	1	2	2	2
0	1	1	1	1	1
0	0	1	2	2	2
1	1	1	1	1	2
1	2	1	1	2	2
2	1	1	1	2	2

알고스팟

https://www.acmicpc.net/problem/1261

- BFS탐색을 벽을 부순 횟수에 따라서 나누어서 수행해야 한다.
- 소스: http://codeplus.codes/75306afddcfe49b4abc475cd216484f2

알고스팟

https://www.acmicpc.net/problem/1261

- 어차피 벽을 뚫는다와 안 뚫는다로 나누어지기 때문에, 덱을 사용한다
- 벽을 뚫는 경우에는 뒤에, 안 뚫는 경우에는 앞에 추가한다.
- 소스: http://codeplus.codes/b6650d4652c041f6a791d4ab1ef24eee

끝

코드 플러스

https://code.plus

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 codeplus@startlink.io 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.