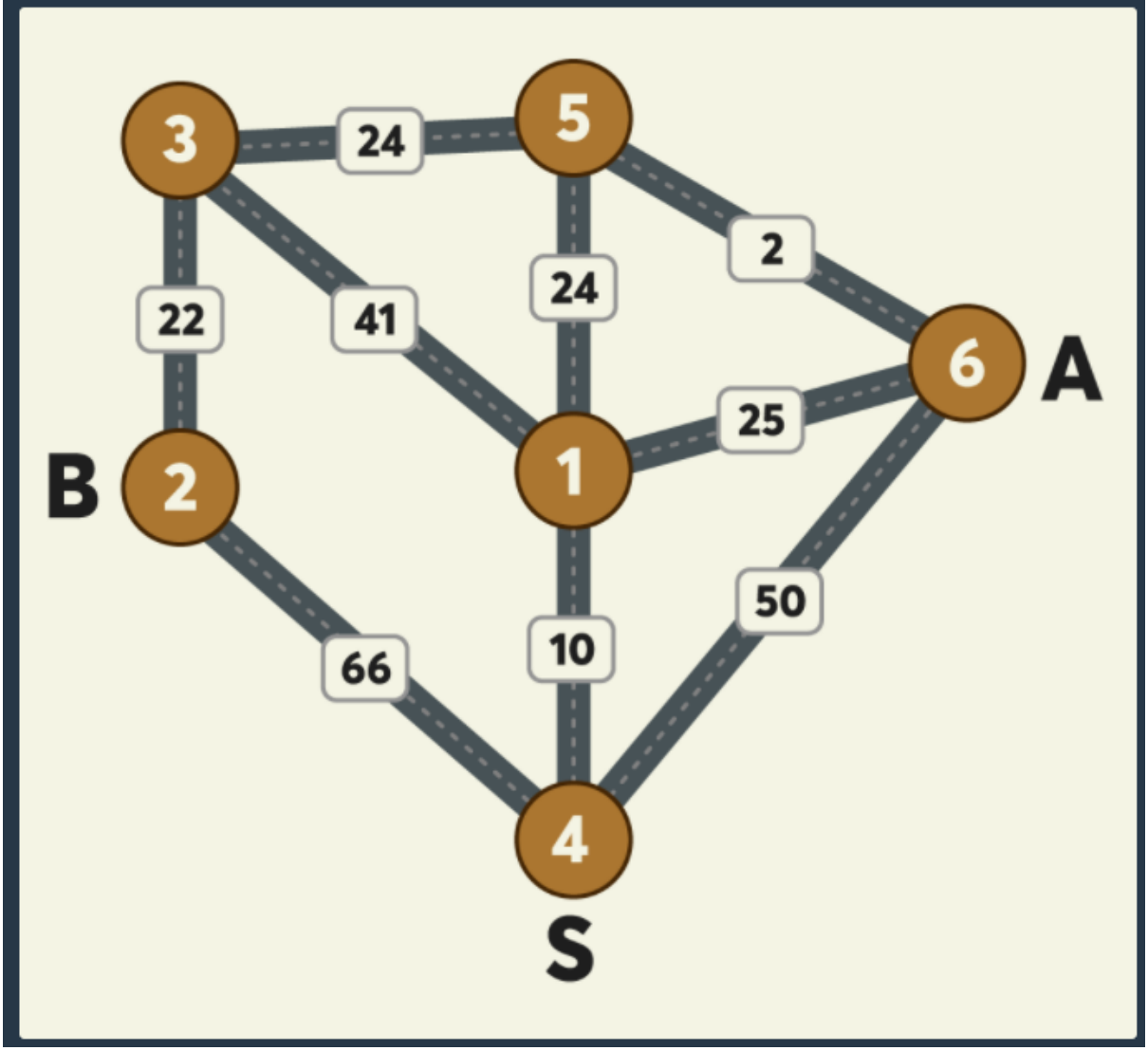
합승 택시 요금

문제 설명

[본 문제는 정확성과 효율성 테스트 각각 점수가 있는 문제입니다.]

밤늦게 귀가할 때 안전을 위해 항상 택시를 이용하던 무지 는 최근 야근이 잦아져 택시를 더 많이 이용하게 되어 택시비를 아낄 수 있는 방법을 고민하고 있습니다. "무지"는 자신이 택시를 이용할 때 동료인 어피치 역시 자신과 비슷한 방향으로 가는 택시를 종종 이용하는 것을 알게 되었습니다. "무지"는 "어피치"와 귀가 방향이 비슷하여 택시 합승을 적절히 이용하면 택시요금을 얼마나 아낄 수 있을 지 계산해 보고 "어피치"에게 합승을 제안해 보려고 합니다.



위 예시 그림은 택시가 이동 가능한 반경에 있는 6개 지점 사이의 이동 가능한 택시노선과 예상요금을 보여주고 있습니다.
그림에서 `A` 와 `B` 두 사람은 출발지점인 4번 지점에서 출발해서 택시를 타고 귀가하려고 합니다. `A` 의 집은 6번 지점에 있으며 `B` 의 집은 2번 지점에 있고 두 사람이 모두 귀가하는 데 소요되는 예상 최저 택시요금이 얼마인 지 계산하려고 합니다.

- 그림의 원은 지점을 나타내며 원 안의 숫자는 지점 번호를 나타냅니다.
  - 지점이 n개일 때, 지점 번호는 1부터 n까지 사용됩니다.
- 지점 간에 택시가 이동할 수 있는 경로를 간선이라 하며, 간선에 표시된 숫자는 두 지점 사이의 예상 택시요금을 나타냅니다.
  - 간선은 편의 상 직선으로 표시되어 있습니다.
  - 위 그림 예시에서, 4번 지점에서 1번 지점으로(4→1) 가거나, 1번 지점에서 4번 지점으로(1→4) 갈 때 예상 택시요금은 `10` 원으로 동일하며 이동 방향에 따라 달라지지 않습니다.
- 예상되는 최저 택시요금은 다음과 같이 계산됩니다.
  - 4→1→5 : `A` , `B` 가 합승하여 택시를 이용합니다. 예상 택시요금은 `10 + 24 = 34` 원 입니다.
  - 5→6 : `A` 가 혼자 택시를 이용합니다. 예상 택시요금은 `2` 원 입니다.
  - 5→3→2 : `B` 가 혼자 택시를 이용합니다. 예상 택시요금은 `24 + 22 = 46` 원 입니다.
  - `A` , `B` 모두 귀가 완료까지 예상되는 최저 택시요금은 `34 + 2 + 46 = 82` 원 입니다.

[문제]

지점의 개수 n, 출발지점을 나타내는 s, `A` 의 도착지점을 나타내는 a, `B` 의 도착지점을 나타내는 b, 지점 사이의 예상 택시요금을 나타내는 fares가 매개변수로 주어집니다. 이때, `A` , `B` 두 사람이 s에서 출발해서 각각의 도착 지점까지 택시를 타고 간다고 가정할 때, 최저 예상 택시요금을 계산해서 return 하도록 solution 함수를 완성해 주세요.
만약, 아예 합승을 하지 않고 각자 이동하는 경우의 예상 택시요금이 더 낮다면, 합승을 하지 않아도 됩니다.

[제한사항]

- 지점갯수 n은 3 이상 200 이하인 자연수입니다.
- 지점 s, a, b는 1 이상 n 이하인 자연수이며, 각기 서로 다른 값입니다.
  - 즉, 출발지점, `A` 의 도착지점, `B` 의 도착지점은 서로 겹치지 않습니다.
- fares는 2차원 정수 배열입니다.
- fares 배열의 크기는 2 이상 `n x (n-1) / 2` 이하입니다.
  - 예를들어, n = 6이라면 fares 배열의 크기는 2 이상 15 이하입니다. (`6 x 5 / 2 = 15`)
  - fares 배열의 각 행은 [c, d, f] 형태입니다.
  - c지점과 d지점 사이의 예상 택시요금이 `f` 원이라는 뜻입니다.
  - 지점 c, d는 1 이상 n 이하인 자연수이며, 각기 서로 다른 값입니다.
  - 요금 f는 1 이상 100,000 이하인 자연수입니다.
  - fares 배열에 두 지점 간 예상 택시요금은 1개만 주어집니다. 즉, [c, d, f]가 있다면 [d, c, f]는 주어지지 않습니다.
- 출발지점 s에서 도착지점 a와 b로 가는 경로가 존재하는 경우만 입력으로 주어집니다.

해결 방법

1. 문제를 해결할 알고리즘

=> A -> B 지점으로 이동하는 모든 최단거리가 필요함.

플로이드 와샬 알고리즘을 통해 구할 수 있다.

2. 이것 이외의 변수

-> 합승(X와 Y가 같은 방향으로 이동 할 경우 합승이 가능)

주어진 출발점에서 각자의 도착지까지 갈 때,

시작점 -> 중간지점(합승구간) -> X의 목적지
Y의 목적지

3. 최단 거리를 구하기 위해서는 ?

-> 각 정점을 중간지점이라고 가정,

중간 지점 까지의 최단거리 (합승 구간) + 중간지점에서 각 목적지 까지의 최단거리

코드

```
class Solution {
    public static int solution(int n, int s, int a, int b, int[][] fares) {
        int answer = 0;

        int[][] dp = new int[n + 1][n + 1];

        for(int i = 1; i <= n; i++) {
          for(int j = 1; j <= n; j++) {
            if(i == j) {
              dp[i][j] = 0;
              continue;
            }
            dp[i][j] = 20000000;
          }
        }

        for(int i = 0; i < fares.length; i++) {
          int st = fares[i][0];
          int ed = fares[i][1];
          int cost = fares[i][2];

          dp[st][ed] = cost;
          dp[ed][st] = cost;
        }

        for(int k = 1; k <= n; k++) {
          for(int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++) {
```

```
                dp[i][j] = Math.min(dp[i][j], dp[i][k] + dp[k][j]);
              }
            }
          }

          answer = Integer.MAX_VALUE;
          for(int i = 1; i <= n; i++) {
            answer = Math.min(answer, dp[s][i] + dp[i][a] + dp[i][b]);
          }
          return answer;
      }
}
```