이동조건

1. 임의로 주어진 정점은 무조건 통과

2. 같은 경로로 또 이동할 수 있지만 결론은
   최단거리 이동

최단거리 알고리즘

1. 다익스트라 → 에지가 모두 양수일 때

2. 벨만포드 → 에지가 음수여도 가능, 음수사이클을 찾을 수 있음

3. 플로이드 워셜 → 음수에지가 가능, 동적계획법 기반

1번정점에서 N번정점으로 이동

V_1, V_2를 무조건 거쳐야

1~V_1의 최소거리 + V_2에서 N의 최소거리 + V_1과 V_2 사이의 거리

⇒ 에지가 모두 양수이므로 다익스트라를 통해 해결

Sudo

Node

$V_1 = 2$

$V_2 = 3$

1 → {2, 3}, {3, 5}, {4, 4}

2 → {1, 3}, {3, 3}, {4, 5}

3 → {2, 3}, {4, 1}, {1, 5}

4 → {3, 1}, {2, 5}, {1, 4}

1	2	3	4
0	∞	∞	∞

다익스트라 를 사용해서

① 1 과 V_1 간의 최소거리를 구함

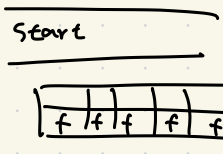
② V_2와 N 간의 최소거리를 구함

③ V_1과 V_2 간의 거리를 더함

다익스트라 메서드 (int start, int end)

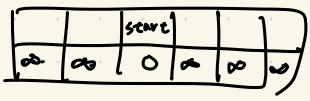
Queue 선언

queue.add(start)



visited [N+1]

distance [N+1]



플랄 지정 배제 Integer.MAX_VALUE

while(!queue.isEmpty())

int poll = queue.poll → start

if(!visited[poll])

visited[poll] = true ⇒ visited[start] = true

인접리스트 A 확인

for(int i = 0 ~ A[poll].size())

A
Node
1 → {2,3}, {3,5}, {4,4}
2 → {1,3}, {3,3}, {4,5}
3 → {2,3}, {4,1}, {1,5}
4 → {3,1}, {2,5}, {1,4}

Node = A[poll].get(i)

Node.getNode() = 다음노드

Node.getWeight() = 다음노드까지 거리

distance[다음노드] > distance[poll] + 다음노드까지 거리

⇒ distance[다음노드] 최신화 ⇒ 더 짧은 거리로

queue.add(다음노드번호, distance[다음노드])

↳ 이를 통해 계속 거리를 최소화 한다