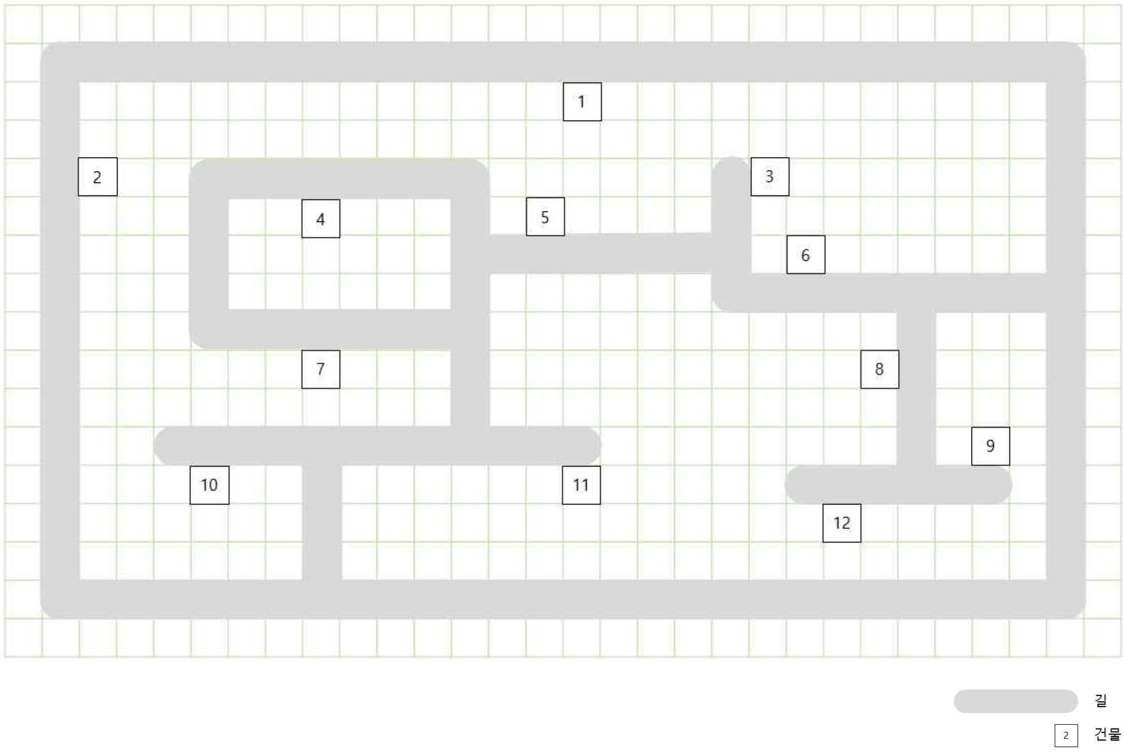
내비게이션

아래의 지도에서 두 번지수가 주어졌을 때, 경로의 좌회전/우회전 횟수를 구하는 문제입니다.



문제

- 번지수로 주어진 두 건물을 잇는 경로에서 좌회전과 우회전 횟수를 각각 출력하시오. (50)
- Dijkstra 알고리즘을 이용하여 최단거리를 구하는 과정을 포함하시오. (20점 추가)
  혹은 A* 알고리즘을 이용하시오. (50점 추가)
- 프로그램이 완성되지 않은 경우 코드를 보고 구현 정도에 따라 부분점수를 할당합니다.

조건

- 지도는 위에 주어진 지도를 소스에 하드코딩해서 이용하면 됩니다. 지도를 표현하는 방법, 길을 찾는 방법 등에 대한 제한은 없습니다.
- 지도의 길은 직선과 직각으로만 이루어져 있습니다. 교차로 여부와 상관없이 우회전과 좌회전 횟수를 계산해서 출력해야 합니다.
- 소스에 주석으로 어떤 관점에서 어떤 방식으로 코딩했는지에 대한 서술을 포함해야 합니다. Mac을 제외하고 한글 주석도 무방합니다. 가능하면 파일의 인코딩을 UTF-8로 설정하세요.
- 지도를 표현하는 방법, 도로를 표현하는 방법, 교차로에 대한 처리 등으로 인해 정답임에도 돔저지에서 Wrong-Answer가 뜰 수 있습니다. 소스에 어떤 관점에서 어떤 방식으로 코딩했는지 주석으로 설명되어 있고 오류가 없으면 정답 처리가 가능합니다.
- 두 번지수의 입력은 한 번만 들어옵니다. 출력 후 프로그램을 바로 종료하면 됩니다.

입력/출력

두 번지수가 입력되면 좌회전과 우회전 횟수를 좌 우 순서로 출력하면 됩니다.

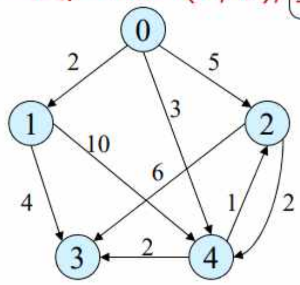
2 10 3 0 // 해설 : 2에서 10으로 갈 때, 좌회전 3번이면 10 앞에 도착할 수 있습니다.	4 8 2 3 // 해설 : 4에서 8로 갈 때, 우-좌-우-좌-우 순서로 각각 좌회전 2번, 우회전 3번만에 갈 수 있습니다.

힌트
Dijkstra 알고리즘이 수록되어 있습니다.

하나의 정점에서 다른 모든 정점까지의 최단경로(1)

- 시작점 v에서 G의 나머지 모든 정점까지의 최단 경로
- 시작점 v와 목표점 t까지의 경로 중, 경로를 구성하는 간선들의 가중치 합이 최소가 되는 경로
- 방향 그래프 G=(V, E), **weight[i, j] ≥ 0**

음이 아닌 가중치



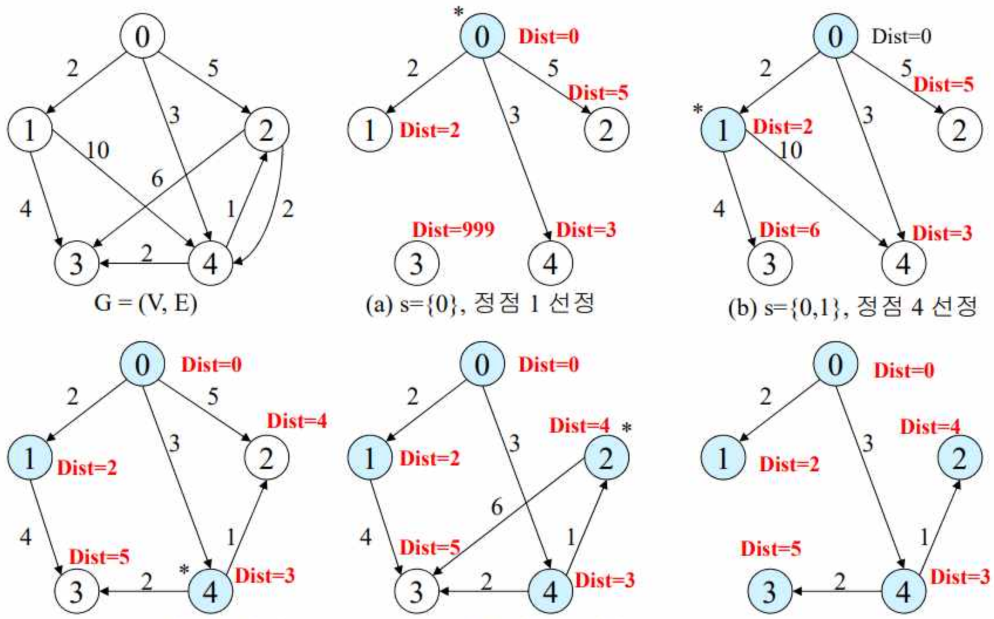
(a) G = (V, E)

경로	거리
0, 1	2
0, 4	3
0, 4, 2	4
0, 4, 3	5

(b) 최단 경로

방향 그래프와 최단 경로

하나의 정점에서 다른 모든 정점까지의 최단경로(2)



G = (V, E)

(a) s={0}, 정점 1 선정

(b) s={0,1}, 정점 4 선정

(c) s={0,1,4}, 정점 2 선정

(d) s={0,1,4,2}, 정점 3 선정

(e) s={0,1,4,2,3}

최단 경로 계산의 예

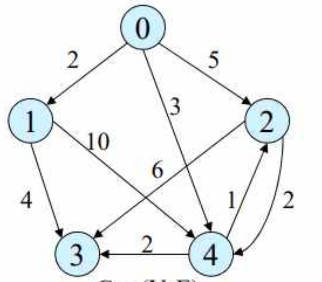
하나의 정점에서 다른 모든 정점까지의 최단경로(3)

- (Dijkstra) 최단 경로 알고리즘의 원리
  - S : 최단 경로가 발견된 정점들의 집합
  - weight[i, j] : 아크 <i, j>의 가중치.
  - Dist[i] : S에 속하지 않은 i에 대해서, v에서 시작하여 S에 있는 정점만을 거쳐 정점 i에 이르는 최단 경로의 길이

1. 처음 S에는 시작점 v만 포함, Dist[v]=0
2. 가장 최근에 S에 첨가한 정점을 u로 설정
3. u의 모든 인접 정점 중에서 S에 포함 되지 않은 w에 대해 Dist[w]를 다시 계산
   - Dist[w]=min{Dist[w], Dist[u] + weight[u, w]}
4. S에 포함되지 않은 모든 정점 w중에서 dist[w]가 가장 작은 정점 w를 S에 첨가
5. 모든 정점에 대한 최단 경로가 결정될 때까지 단계 2~4 를 반복

하나의 정점에서 다른 모든 정점까지의 최단경로(4)

- 최단 경로 알고리즘
  - G의 n개의 정점을 0에서 n-1까지 번호를 붙임
  - S[] : 정점 i 가 S에 포함되어 있으면 S[i] = true, 아니면 S[i]=false로 표현하는 불리언 배열
  - weight[n, n] : 가중치 인접행렬
    - weight[i, j] : 아크 < i, j>의 가중치.
      아크 <i, j>가 그래프에 포함되어 있지 않은 경우에는 아주 큰 값으로 표현



G = (V, E)

	[0]	[1]	[2]	[3]	[4]
[0]	0	2	5	999	3
[1]	999	0	999	4	10
[2]	999	999	0	6	2
[3]	999	999	999	0	999
[4]	999	999	1	2	0

weight[5, 5]

그래프 G와 가중치 인접 행렬

하나의 정점에서 다른 모든 정점까지의 최단경로(5)

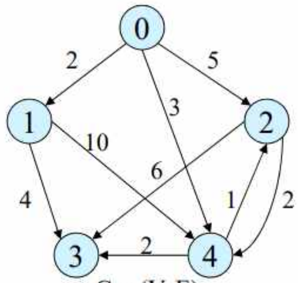
❖ Dijkstra의 최단 경로 알고리즘

```
shortestPath(v, weight, n)
   // v는 시작점, weight는 가중치 인접 행렬, n은 정점수
   // create S[n], Dist[n]
  for (i←0; i<n; i←i+1) do {
      S[i] ← false;                // S를 초기화
      Dist[i] ← weight[v, i];      // Dist를 초기화
  }
  S[v] ← true;
  Dist[v] ← 0;
  for (i←0; i<n-2; i←i+1) do {   // n-2번 반복
      select u such that                    // 새로운 최단 경로를 선정
          Dist[u] = min{Dist[j] | S[j] = false and 0≤j<n};
      S[u] ← true;
      for (w←0; w<n; w←w+1) do {          // 확정이 안된 경로들에 대해 다시 계산
          if (S[w] = false) then {
              if (Dist[w] > (Dist[u] + weight[u, w])
                  then Dist[w] ← Dist[u] + weight[u, w];
  }}}
end shortestPath
```


하나의 정점에서 다른 모든 정점까지의 최단경로(6)

초기화 : 시작 정점[0], Dist[0] ← 0;

정점	[0]	[1]	[2]	[3]	[4]
S	T	F	F	F	F
Dist	0	2	5	999	3



G = (V, E)

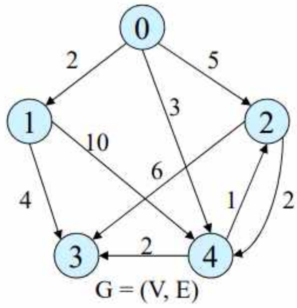
for 루프 1 : 정점 [1]을 선정

Dist[2] ← min { Dist[2], Dist[1]+weight[1,2] }
Dist[3] ← min { Dist[3], Dist[1]+weight[1,3] }
Dist[4] ← min { Dist[4], Dist[1]+weight[1,4] }

정점	[0]	[1]	[2]	[3]	[4]
S	T	T	F	F	F
Dist	0	2	5	6	3

그래프 G에 대한 shortestPath 수행 내용(1/2)

하나의 정점에서 다른 모든 정점까지의 최단경로(7)



for 루프 2 : 정점 [4]를 선정

Dist[2] ← min { Dist[2], Dist[4]+weight[4,2] }
Dist[3] ← min { Dist[3], Dist[4]+weight[4,3] }

정점	[0]	[1]	[2]	[3]	[4]
S	T	T	F	F	T
Dist	0	2	4	5	3

for 루프 3 : 정점 [2]를 선정

Dist[3] ← min { Dist[3], Dist[2]+weight[2,3] }

정점	[0]	[1]	[2]	[3]	[4]
S	T	T	T	F	T
Dist	0	2	4	5	3

남은 정점 [3]의 거리는 최단 거리임

그래프 G에 대한 shortestPath 수행 내용(2/2)