Algorithm Study

06.23.2024 김홍주

#1504. 특정한 최단 경로

/Gold 4

Q. 특정한 최단 경로

방향성이 없는 그래프가 주어진다.

세준이는 1번 정점에서 N번 정점으로 최단 거리로 이동하려고 한다

세준이는 두 가지 조건을 만족하면서 이동하는 특정한 최단 경로를 구하고 싶은데,

[1] 임의로 주어진 두 정점은 반드시 통과해야 한다는 것이다.

[2] 세준이는 한번 이동했던 정점은 물론, 한번 이동했던 간선도 다시 이동할 수 있다.

1번 정점에서 N번 정점으로 이동할 때,
주어진 두 정점을 반드시 거치면서 최단 경로로 이동하는 프로그램을 작성하시오.

[예제 입출력]

```
4 6
1 2 3
2 3 3
3 4 1
1 3 5
2 4 5
1 4 4
2 3
```

```
7
```

<입력 >

- 1줄: 정점의 개수 N과 간선의 개수 E (2 ≤ N ≤ 800, 0 ≤ E ≤ 200,000)
- 2 ~ E+1 : 세 개의 정수 a, b, c로, 양방향 a번 에서 b번 정점까지 거리가 c이다. (1 ≤ c ≤ 1,000)
- E+2줄 : 반드시 거쳐야 하는 두 개의 서로 다른 정점 번호 v1과 v2. (v1 ≠ v2, v1 ≠ N, v2 ≠ 1)
- 임의의 두 정점 u와 v사이에는 간선이 최대 1개 존재한다.

<출력 조건>

- 두 정점(v1 ,v2) 지나는 최단 경로 출력(경로가 없으면 -1 출력)

Hint

1. 문제 유형 : 다익스트라

문제 조건
(1) 간선의 cost가 자연수 (1<= c <= 1000)
(2) 지정된 시작점으로 부터 최단 거리 구하기

- 그래프에서 한 정점(X)에서 다른 정점(A,B,C..) 까지 가는 각각의 최단 경로 구하는 방법
- **동작 과정**

1. 출발 노드 설정 : X
2. 최단 거리 테이블 초기화

노드 번호 N	1(start)	2	3	4	…
최단 거리	0	INF	INF	INF	

*각 노드에 대한 최단 거리를 담은 1차원 리스트

3. 노드들 중
   (1) 방문 하지 않았고
   (2) 현재 가장 거리가 짧은 노드를 선택
   - 구현 방법 : 완전 탐색(O(V^2)) / **우선순위 큐 by heapque(O(ElogV))**
     : min(Cost(X -> a,b,c…)) & not visited => a 선택
4. 출발점(X)에서 위 노드(a) 를 경유해서 다른(인접) 노드 (Y) 로 가는 거리 계산 => 최단 거리 테이블 업데이트
   : *Cost(X->a -> Y) < Cost(X->Y) => update!*

Hint

1-2. 다익스트라 (Dijkstra) *# 최단거리 , # 그리디 , # 우선순위 큐*

```python
import heapq

def dijkstra(s):
    #2. 최단거리 테이블 초기화
    D = [float('inf')] * (N+1)
    D[s] = 0
    q = []                      # 최단 거리 테이블을 heap으로 구현
    heapq.heappush(q, (0, s))       # heap에 (가중치, 노드) 형식으로 삽입

    #3. 현재 출발점과 가장 가깝고 ,방문 안한 노드(경유지) 찾기
    while q:
        dist, now = heapq.heappop(q)    # 최소힙이므로 가중치가 가장 작은 값이 pop
        if D[now] >= dist:          # 이미 최솟값 구했는지 확인(방문여부확인)

        #4. 인접한 노드 중 now를 경유할 때 더 작은 값이면 최단거리 테이블 갱신 & 큐 삽입
            for v, val in city[now]:    # 연결된 노드(Y)들 확인
                if dist + val < D[v]:   # 경유 방법이 가중치가 더 작은 값이면 갱신
                    D[v] = dist + val
                    heapq.heappush(q, (dist + val, v)) # 큐에 삽입
    return D

dijkstra(start) # 1.출발점 설정
```

- 그래프에서 한 정점(X)에서 다른 정점(A,B,C..) 까지 가는 각각의 최단 경로 구하는 방법
- **동작 과정**

1. 출발 노드 설정 : X *distance([1,2,3 .. n]) = INF
2. 최단 거리 테이블 초기화

노드 번호 N	1(start)	2	3	4	...
최단 거리	0	INF	INF	INF	

*각 노드에 대한 최단 거리를 담은 1차원 리스트

3. 현재 노드와 연결된 노드 중
   (1) 방문 하지 않았고
   (2) 가장 거리가 짧은 노드를 선택
   - 구현 방법 : **우선순위 큐 by heapque(O(ElogV))**
4. 현재 노드(X)가 위 노드(a) 를 경유해서 다른(인접) 노드 (Y) 로 가는 거리 계산
   => 최단 거리 테이블 업데이트

Hint

2. 풀이 flow

1. 인접 리스트 graph 정의 : 양방향 edge 저장하기
2. 우선순위 큐(heaqp)을 사용한 다익스트라 함수 정의하기
3. **경로 1 (1- > v1 -> v2 -> N) 과 경로2(1-> V2 -> v1 -> N) 중 최단 경로 구하기**

   min(path1 , path2)

[1] 다익스트라 시작점을 start (1번 정점) 과 end(N번 정점) 두기#3번

min(s_v1 + v2_e , s_v2+v1_e) + v1_v2

(1) 시작점(1)이 START 인 distance 테이블 구하기

→ start ~ v1 , start ~ v2 최단 거리 구함

(2) 도착점 (N) 이 START 인 distance 테이블 구하기

→ end - v1 , end -v2 최단 거리 구하기

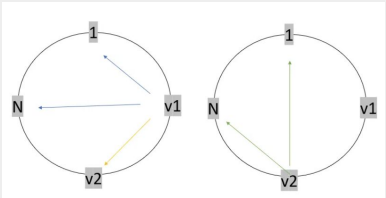
(3) 시작점이 v1 (or v2)인 distance 테이블 구하기

→ v1 - v2 최단 거리 구하기

[2] 시작점을 v1 ,v2 로 두기#2번

dist_from_v1 = dijkstra(v1)

dist_from_v2 = dijkstra(v2)



 # 1 -> v1 -> v2 -> N

path_1 = dist_from_v1[1] + dist_from_v1[v2] + dist_from_v2[N]

 # 1 -> v2 -> v1 -> N

path_2 = dist_from_v2[1] + dist_from_v2[v1] + dist_from_v1[N]

My Solution

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이
95585873	zaqquum01	1504	맞았습니다!!	71428 KB	452 ms	Python 3 / 수정	1499 B

```python
import sys
import heapq

INF = 1e9
input = sys.stdin.readline
answer = -1
# 0. 입력 변수
N ,E = map(int, input().split())
graph =[[] for _ in range(N+1)] #인접리스트 초기화
for i in range(E) :
  a, b , cost = map(int, input().split())
  graph[a].append([b , cost ]) ; graph[b].append([a, cost])
v1 ,v2 = map(int, input().split())

#1. 다익스트라 초기 설정 - 시작점
start =1 ; target = N

#2. 다익스트라 함수 정의
def dikstra(start) :

  # 최단 거리 테이블 초기화
  distance =[INF]*(N+1)
  q = []
  heapq.heappush(q, (0,start))
  distance[start] = 0
  #[2] 경유지 선택
  while q :
    dist , now  = heapq.heappop(q)
    #3-1 현재 노드가 이미 업데이트 완료되면 무시
    if distance[now] <dist :
      continue
    #3-2. 현재 노드와 인접한 노드 확인 , 방문
    for i in graph[now] :
      cost = dist + i[1]
      # 4. 현재 노드 now을 경유해서 , 다른 노드(i)로 이동하는 거리가 더 짧은 경우
      if cost <distance[i[0]] :
        distance[i[0]] = cost # 업데이트
        heapq.heappush(q , (cost ,i[0]))
  return distance
```

```python
# 3. v1,v2 지나는 1- > N 의 최단 거리 고르기
s_distance = dikstra(start)
e_distance = dikstra(target) # 양방향
s_v1 = s_distance[v1]
s_v2 = s_distance[v2]
v1_e = e_distance[v1]
v2_e = e_distance[v2]
v1_v2 = dikstra(v1)[v2] # v1 - v2 거리(공통)

answer =min(s_v1 + v2_e , s_v2+v1_e ) + v1_v2
if answer >= INF :
  print(-1)
else :
  print(answer)
```

1. 인접 리스트 graph 정의 : 양방향 edge 저장하기

2. 우선순위 큐(heaqp)을 사용한 **다익스트라** 함수 정의하기

3. min(s_v1 + v2_e , s_v2+v1_e) + v1_v2 구하기

다른 Solution 1

제출 번호	아이디	문제	문제 제목	결과	메모리	시간	언어	코드 길이
91100847	hcsksy3	1504	특정한 최단 경로	맞았습니다!!	116424 KB	180 ms	PyPy3	1275 B

다익스트라 함수 2회 & 시작점 v1 ,v2

```python
import sys
import heapq

INF = sys.maxsize

def dijkstra(start, graph, N):
    distance = [INF] * (N+1)
    distance[start] = 0
    pq = [(0, start)]

    while pq:
        dist, now = heapq.heappop(pq)

        if distance[now] < dist:
            continue

        for next_node, weight in graph[now]:
            cost = dist + weight
            if cost < distance[next_node]:
                distance[next_node] = cost
                heapq.heappush(pq, (cost, next_node))

    return distance
```

```python
N, E = map(int, sys.stdin.readline().split())
graph = [[] for _ in range(N+1)]

for _ in range(E):
    u, v, w = map(int, sys.stdin.readline().split())
    graph[u].append((v, w))
    graph[v].append((u, w))

v1, v2 = map(int, sys.stdin.readline().split())

dist_from_v1 = dijkstra(v1, graph, N)
dist_from_v2 = dijkstra(v2, graph, N)

path_1 = dist_from_v1[1] + dist_from_v1[v2] + dist_from_v2[N]  # 1 -> v1 -> v2 -> N
path_2 = dist_from_v2[1] + dist_from_v2[v1] + dist_from_v1[N]  # 1 -> v2 -> v1 -> N

result = min(path_1, path_2)

if result >= INF:
    print(-1)
else:
    print(result)
```

다른 Solution 2

제출 번호	아이디	문제	문제 제목	결과	메모리	시간	언어	코드 길이	제출한 시간
95485852	jelly7777	1504	특정한 최단 경로	맞았습니다!!	126188 KB	244 ms	PyPy3	1048 B	3일 전

다익스트라 함수 **6회 &** 다익스트 함수 사용시, 지정한 **end** 정점 최단 거리 구하면 종료

```python
import sys
import heapq

input = sys.stdin.readline
INF = float('inf')

def dijkstra(start, end, graph):
    distance = [INF] * (n + 1)
    distance[start] = 0
    heap = [(0, start)]

    while heap:
        dist, now = heapq.heappop(heap)

        if now == end:
            return dist

        if distance[now] < dist:
            continue

        for neighbor, cost in graph[now]:
            new_cost = dist + cost
            if new_cost < distance[neighbor]:
                distance[neighbor] = new_cost
                heapq.heappush(heap, (new_cost, neighbor))

    return distance[end]
```

```python
n, e = map(int, input().split())

graph = [[] for _ in range(n + 1)]
for _ in range(e):
    a, b, c = map(int, input().split())
    graph[a].append((b, c))
    graph[b].append((a, c))

v1, v2 = map(int, input().split())
case1 = dijkstra(1, v1, graph) + dijkstra(v1, v2, graph) + dijkstra(v2, n, graph)
case2 = dijkstra(1, v2, graph) + dijkstra(v2, v1, graph) + dijkstra(v1, n, graph)
result = min(case1, case2)
print(result if result < INF else -1)
```

Assignment

[백준#1753. 최단경로] 골드4

- 문제 : https://www.acmicpc.net/problem/1753

#다익스트라 , # 그래프