자료구조 14주차 과제

- 이름: 안찬웅
- 학번:
- 과제: 각 10점

1. P11.5
2. P11.6

** 풀지 못한 문제 - 만일 과제의 문제를 다 풀지 못한 경우, 여기에 풀지 못한 번호를 적으시오.

과제는 문제에 대한 코딩이 완성되고 테스트를 통해 적절성이 검증된 경우만 점수가 부여되며, 이외 사항에 대해서는 0점 처리. 코드에 에러가 있음에도 불구하고, 과제 앞 부분 미완성 부분에 적시하지 않은 경우 전체 과제를 0점 처리합니다.

11.5

아래에 코드셀을 만들고, 셀에 인접행렬로 표현된 그래프를 인자로 받아 Maximal Spanning Tree를 구하는 하는 함수 maxSpanningTree(...) 를 정의하시오. 함수는 visit 되는 407 페이지에 있는 그림처럼 추가되는 간선을 (x, y, weight) 형태로 출력할 것.

```
def maxSpanningTree(graph):
    vertex, weight = graph
    n = len(vertex)
    visited = [False] * n
    visited[0] = True
    total = 0
    for _ in range(n-1):
        maxWeight = 0
        maxVertex = 0
        for i in range(n):
            if visited[i]:
                for j in range(n):
                    if not visited[j] and weight[i][j] is not None and weight[i][j] > maxWeight:
                        maxWeight = weight[i][j]
                        maxVertex = j
        visited[maxVertex] = True
        total += maxWeight
        print(f'간선 추가 : {vertex[maxVertex]} ({maxWeight})')

    return total
```

더블클릭 또는 Enter 키를 눌러 수정

아래 코드셀은 11.5 을 테스트 하는 코드이다. 주어진 데이터를 이용하여 테스트를 실행하시오.

```
# 교재 407 페이지
vertex =  ['A',   'B',   'C',   'D',   'E',   'F',   'G' ]
weight = [ [None, 29,   None, None, None,   10,   None],
           [29, None, 16,   None, None, None, 15  ],
           [None, 16,   None, 12,   None, None, None],
           [None, None,   12,   None, 22,   None, 18  ],
           [None, None, None,   22,   None, 27,   25  ],
           [10, None, None, None,   27,   None, None],
           [None,  15,    None,   18,   25,   None, None]]

graph = (vertex, weight)

maxSpanningTree(graph)
```

```
간선 추가 : B (29)
간선 추가 : C (16)
간선 추가 : G (15)
간선 추가 : E (25)
간선 추가 : F (27)
간선 추가 : D (22)
134
```

11.6

아래에 코드셀을 만들고, 문제 11.6 의 솔루션을 제공하는 함수 dijkstra_SP_with_path_print(...)을 작성하시오. 교재 428 문제 11.6에 보이는 바와 같이 출력하도록 작성하시오.

```
INF = int(1e9)

graph = (vertex, weight)
start = 0

def dijkstra_SP_with_path_print(start, graph):
    vertex, weight = graph
    n = len(vertex)
    visited = [False] * n
    distance = weight[start]
    path = [start] * n

    visited[start] = True
    for _ in range(n-1):
        print('Step%2d : ' % (_+1), distance)
        minDistance = INF
        minVertex = 0
        for i in range(n):
            if not visited[i] and distance[i] < minDistance:
                minDistance = distance[i]
                minVertex = i
        visited[minVertex] = True

        for i in range(n):
            if not visited[i] and weight[minVertex][i] != INF:
                if distance[minVertex] + weight[minVertex][i] < distance[i]:
                    distance[i] = distance[minVertex] + weight[minVertex][i]
                    path[i] = minVertex


    for i in range(n):
        if i != start:
            print(f'{vertex[start]} -> {vertex[i]} : {distance[i]} (', end='')
            p = i
            print(vertex[p], end='')
            while p != start:
                print(f' <- {vertex[path[p]]}', end='')
                p = path[p]
            print(')')
```

아래 코드셀은 11.6 을 테스트 하는 코드이다. 주어진 데이터를 이용하여 테스트를 실행하시오.

```
vertex =  ['A',   'B',   'C',   'D',   'E',   'F',   'G'  ]
weight = [ [0,     7,    INF,   INF,   3,     10,   INF ],
           [7,    0,     4,    10,    2,     6,     INF ],
           [INF,  4,    0,     2,    INF,   INF,   INF ],
           [INF,  10,   2,     0,     11,   9,    4   ],
           [3,     2,    INF,   11,   0,     13,   5   ],
           [10, 6,     INF,   9,     13,   0,    INF ],
           [INF,  INF,  INF,   4,    5,    INF,   0   ] ]

graph = (vertex, weight)
start = 0
dijkstra_SP_with_path_print(start, graph) # modified 12.06
```

```
Step 1 :  [0, 7, 1000000000, 1000000000, 3, 10, 1000000000]
Step 2 :  [0, 5, 1000000000, 14, 3, 10, 8]
Step 3 :  [0, 5, 9, 14, 3, 10, 8]
Step 4 :  [0, 5, 9, 12, 3, 10, 8]
Step 5 :  [0, 5, 9, 11, 3, 10, 8]
Step 6 :  [0, 5, 9, 11, 3, 10, 8]
A -> B : 5 (B <- E <- A)
A -> C : 9 (C <- B <- E <- A)
A -> D : 11 (D <- C <- B <- E <- A)
A -> E : 3 (E <- A)
A -> F : 10 (F <- A)
A -> G : 8 (G <- E <- A)
```

✓ 0초 오후 8:17에 완료됨