자료구조 13주차 과제

- 이름: 안찬웅
- 학번: 32162566
- 과제: 각 5점

1. P10.1
2. P10.2
3. P10.3
4. P10.4
5. P10.6

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

4. P10.4

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.

# 10.1

아래에 코드셀을 만들고, 셀에 인접행렬로 표현된 그래프를 인자로 받아 깊이우선탐색을 하는 함수 matGraphDFS(...) 를 정의하시오. 함수는 visit 되는 vertex 값을 출력하여야 한다.

```
visited = set()
def matGraphDFS(vertex, graphAM, start): # 'A' ~ 'H'를 0 ~ 7로 변환해서 사용
                                         # start가 문자니까 start의 ASCII에서 'A'의 ASCII를 빼면 0
    idx = ord(start) - ord('A')
    if idx not in visited:               # visited에 넣는 것도 숫자
        visited.add(idx)                 # start가 문자 start를 바로 출력
        print(start, end = ' ')

       # graphAM에서 idx가 가지는 배열이니 이를 순회하면서
        for v in range(len(graphAM[idx])):
            if graphAM[idx][v] == 1:
                # 호출할 때 인자는 숫자가 아니라 문자 형태여야 하므로 vertex를 이용해 다시 숫자 ->
                matGraphDFS(vertex, graphAM, vertex[v])
```

아래 코드셀은 10.1 를 테스트 하는 코드이다. 주어진 데이터를 이용하여 테스트를 실행하시오.

```
# 교재 366 페이지
vertex = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H' ]
graphAM = [ [ 0,  1,  1,  0,  0,  0,  0,  0 ],
            [ 1,  0,  0,  1,  0,  0,  0,  0 ],
            [ 1,  0,  0,  1,  1,  0,  0,  0 ],
            [ 0,  1,  1,  0,  0,  1,  0,  0 ],
            [ 0,  0,  1,  0,  0,  0,  1,  1 ],
            [ 0,  0,  0,  1,  0,  0,  0,  0 ],
            [ 0,  0,  0,  0,  1,  0,  0,  1 ],
            [ 0,  0,  0,  0,  1,  0,  1,  0 ] ]

matGraphDFS(vertex, graphAM, 'A')
```

```
A B D C E G H F
```

## 10.2

아래에 코드셀을 만들고, 셀에 인접행렬로 표현된 그래프를 인자로 받아 너비우선탐색을 하는 함수 matGraphBFS(...) 를 정의하시오. 함수는 visit 되는 vertex 값을 출력하여야 한다.

```
import collections as cols

def matGraphBFS(vertex, graphAM, start):
    idx = ord(start) - ord('A')

    queue = cols.deque()       # 빈 덱에서 시작
    queue.append(idx)

    visited = set()
    visited.add(idx)

    while len(queue) > 0:      # 공백이 아닐 때까지
        idx = queue.popleft()  # 큐에서 하나의 vertex 빼냄
        print(vertex[idx], end=' ')
        for v in range(len(graphAM[idx])):
            if graphAM[idx][v] == 1 and v not in visited:
                visited.add(v)
                queue.append(v)
```

아래 코드셀은 10.2 을 테스트 하는 코드이다. 데이터는 10.1 문제에 사용한 vertex 와 graphAM 을 사용하시오.

```
matGraphBFS(vertex, graphAM, 'A')
```

```
A B C D E F G H
```

## 10.3

아래에 코드셀을 만들고, 셀에 인접행렬로 표현된 그래프를 인자로 받아 연결성분검사를 하는 함수 matGraphFindConnectedGrapth(...)와 너비우선 탐색을 이용한 matGraphBFS_cc(...) 를 정의하시오. 교재 379 페이지에 형식으로 연결성분을 출력하여야 한다.

```
def matGraphFindConnectedGrapth(adj_matrix):

    N, stack, BCClist = len(adj_matrix), [], []

    def dfs(v):
        visited[v] = True
        for adj_v in range(N):
            if adj_matrix[v][adj_v] == 1 and not visited[adj_v]:
                dfs(adj_v)
        stack.append(v)

    def rev_dfs(v):
        visited[v] = True
        for adj_v in range(N):
            if adj_matrix[adj_v][v] == 1 and not visited[adj_v]:
                rev_dfs(adj_v)
        bcclist.append(v)

    visited = [False] * N
    for v in range(N):
        if not visited[v]:
            dfs(v)

    visited = [False] * N
    while stack:
        v = stack.pop()
        if not visited[v]:
            bcclist = []
            rev_dfs(v)
            BCClist.append(bcclist)

    return BCClist
```

아래 코드셀은 10.3 을 테스트 하는 코드를 작성하시오. 데이터는 10.1 문제에 사용한 vertex 와 graphAM 을 사용하시오.

```
print('matGraphFindConnectedGrapth: ')
matGraphFindConnectedGrapth(graphAM)
```

```
matGraphFindConnectedGrapth:
[[7, 6, 4, 2, 5, 3, 1, 0]]
```

더블클릭 또는 Enter 키를 눌러 수정

# 10.4

아래에 코드셀을 만들고, 셀에 인접행렬로 표현된 그래프를 인자로 받아 너비우선탐색을 이용하여 신장트리를 구하는 함수 matGraphSpanningTreeBFS(...) 정의하시오. 교재 380 페이지에 있는 예제 형식으로 연결되는 간선들을 출력하여야 한다.

더블클릭 또는 Enter 키를 눌러 수정

아래 코드셀은 10.4 을 테스트 하는 코드를 작성하시오. 데이터는 10.1 문제에 사용한 vertex 와 graphAM 을 사용하시오.

# 10.6

아래에 코드셀을 만들고, 셀에 인접행렬로 표현된 그래프를 인자로 받아 브리지를 찾는 함수 matGraphFindBridge(...) 를 정의하시오.

```
def matGraphFindBridge(vertex, graph):
    n = len(vertex)
    inDeg = [0] * n

    for i in range(n):
        for j in range(n):
            if graph[i][j] > 0:
                inDeg[j] += 1

    vlist = []
    for i in range(n):
         if inDeg[i] == 0:
             vlist.append(i)

    while len(vlist) > 0:
        v = vlist.pop()
        print(vertex[v], end=' ')

        for u in range(n):
            if v != u and graph[v][u] > 0:
                inDeg[u] -= 1
                if inDeg[u] == 0:
                    vlist.append(u)
```

아래 코드셀은 10.6 을 테스트 하는 코드를 작성하시오. 데이터는 문제 10.6 에 나와 있는 그래프를 인접행렬로 표현하여 사용하시오.

```
vertex = ['A', 'B', 'C', 'D', 'E', 'F']
graphAM = [[0, 0, 1, 1, 0, 0],
           [0, 0, 0, 1, 1, 0],
           [0, 0, 0, 1, 0, 1],
           [0, 0, 0, 0, 0, 1],
           [0, 0, 0, 0, 0, 1],
           [0, 0, 0, 0, 0, 0]]

matGraphFindBridge(vertex, graphAM)
print()
```

```
     B E A C D F
```