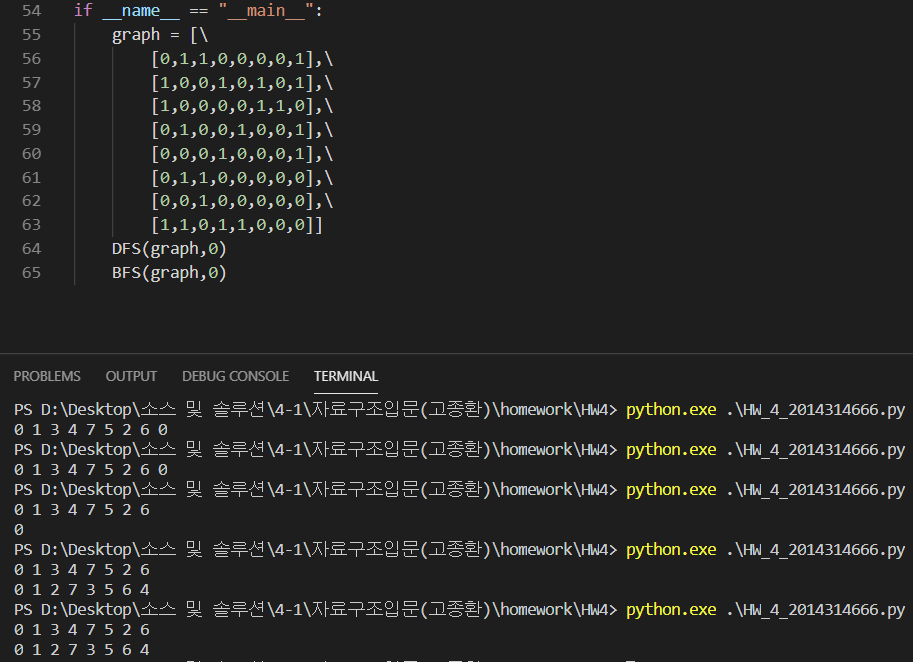
1.

Selection Sort로 stable하면서 in-place하게 만들 수 있다.

4a 3 4b 1 5 를 sorting 할 때, 처음에는 4a와 1의 자리를 바꾼다. 하지만 이렇게 하지않고, 4a부터 1까지 index를 하나씩 미뤄서 1 4a 3 4b 5 이런 방식으로 진행하면 stable하다.

2.



input이 list가 아닌 graph의 형태이다. 위처럼 진행했을 때, 결과가 잘 나온 것을 확인할 수 있었다.

DFS 코드는 다음과 같다.  
def DFSUtil(graph, v, visited):

    visited[v] = True               # Mark the current node as visited and print it

    print(v, end = ' ')

    for i in graph[v]:              # Recur for all the vertices adjacent to this vertex

        if visited[i] == False:

            DFSUtil(graph, i, visited)

def DFS(graph, v):

    List = []

    for i in range(len(graph)):

        tmp = []

        for j in range(len(graph[i])):

            if graph[i][j] == 1:

                tmp.append(j)

        List.append(tmp)

    graph = List

    visited = [False] \* (len(graph))     # Mark all the vertices as not visited

    DFSUtil(graph, v, visited)

    print()

위 코드에서 바꾼 점은 DFS 함수가 실행됨과 동시에 matrix 형태를 List 형태로 바꿔주는 것이다.

다음은 BFS 함수이다.  
def BFS(graph, s):

    List = []

    for i in range(len(graph)):

        tmp = []

        for j in range(len(graph[i])):

            if graph[i][j] == 1:

                tmp.append(j)

        List.append(tmp)

    graph = List

    visited = [False] \* (len(graph))         # Mark all the vertices as not visited

    queue = []                   # Create a queue for BFS

    queue.append(s)              # Mark the source node as visited and enqueue it

    visited[s] = True

    while queue:

        s = queue.pop(0)             # Dequeue a vertex from queue and print it

        print (s, end = " ")

        for i in graph[s]:          # Get all adjacent vertices of the dequeued vertex s.

            if visited[i] == False:         # If a adjacent has not been visited,

                    queue.append(i)         # then mark it visited and enqueue it

                    visited[i] = True

BFS도 동일하게 matrix 형태를 list 형태로 바꾸어주었다.