Breadth First Search\_ adjacency list

* Time complexity:



연결 리스트를 바탕으로 구현된 DFS의 시간 복잡도를 구할 때와 비슷한 논리로 시간 복잡도를 구할 수 있다. BFS가 끝난 뒤를 생각해 보면, 해당 라인은 모든 간선에 대해서 한 번씩 비교할 것이기 때문에 총 E번 반복할 것이고, 각 정점을 한 번씩 모두 방문하여 V의 시간이 걸릴 것이다.

따라서, 인접 리스트로 구현한 BFS의 시간 복잡도는 O(V+E) 이다.

def bfs(start):

    check = [False] \* (n+1)

    q = deque()

    q.append(start)

    check[start] = True

    while q:

        x = q.popleft()

        print(x, end=' ')

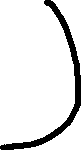
        for y in a[x]:

            if check[y] == False:

                check[y] = True

                q.append(y)

EXAMPLE 1.



|  |  |
| --- | --- |
| 1 | [2,3,4] |
| 2 | [1,4] |
| 3 | [1,4] |
| 4 | [1,2,3] |

* bfs(1):

check = [False] \* 5

q = [1]

check[1] = True

while [1]:

x = 1 🡪 q = []

print(1, end = ‘ ‘)

for y in [2,3,4]:

if not check[2]:

check[2] = True

q.append(2) 🡪 q = [2]

if not check[3]:

check[3] = True

q.append(3) 🡪 q = [2,3]

if not check[4]:

check[4] = True

q.append(4) 🡪 q = [2,3,4]

while [2,3,4]:

x = 2 🡪 q = [3,4]

print(2, end = ‘ ‘)

for y in [1,4]:

while [3,4]:

x = 3 🡪 q = [4]

print(3, end = ‘ ‘)

for y in [1,4]:

while [4]:

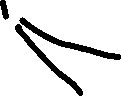
x = 4 🡪 q = []

print(4, end = ‘ ‘)

for y in [1,2,3]:

while []:

EXAMPLE 2.



|  |  |
| --- | --- |
| 1 | [2,3,4] |
| 2 | [1] |
| 3 | [1] |
| 4 | [1] |

* bfs(1):

check = [False] \* 5

q = [1]

check[1] = True

while [1]:

x = 1 🡪 q = []

print(1, end = ‘ ‘)

for y in [2,3,4]:

if not check[2]:

check[2] = True

q.append(2) 🡪 q = [2]

if not check[3]:

check[3] = True

q.append(3) 🡪 q = [2,3]

if not check[4]:

check[4] = True

q.append(4) 🡪 q = [2,3,4]

while [2,3,4]:

x = 2 🡪 q = [3,4]

print(2, end = ‘ ‘)

for y in [1]:

while [3,4]:

x = 3 🡪 q = [4]

print(3, end = ‘ ‘)

for y in [1]:

while [4]:

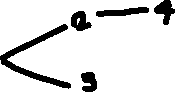
x = 4 🡪 q = []

print(4, end = ‘ ‘)

for y in [1]:

while []:

EXAMPLE 3.



|  |  |
| --- | --- |
| 1 | [2,3] |
| 2 | [1,4] |
| 3 | [1] |
| 4 | [2] |

* bfs(1):

check = [False] \* 5

q = [1]

check[1] = True

while [1]:

x = 1 🡪 q = []

print(1, end = ‘ ‘)

for y in [2,3]:

if not check[2]:

check[2] = True

q.append(2) 🡪 q = [2]

if not check[3]:

check[3] = True

q.append(3) 🡪 q = [2,3]

while [2,3]:

x = 2 🡪 q = [3]

print(2, end = ‘ ‘)

for y in [1,4]:

if not check[4]:

check[4] = True

q.append(4) 🡪 q = [3,4]

while [3,4]:

x = 3 🡪 q = [4]

print(3, end = ‘ ‘)

for y in [1]:

while [4]:

x = 4 🡪 q = []

print(4, end = ‘ ‘)

for y in [1]:

while []:

BFS에 관하여.