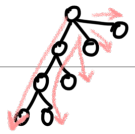
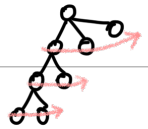
## < DFS와 BFS >

* 그래프 검색방법

DFS : 깊이 우선검색 / BFS : 넓이 우선 검색



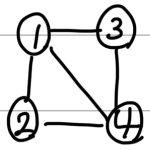


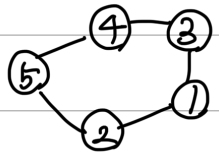
끝까지 노드까지 탐색 — 같은 레벨의 노드부터 탐색

→ stack 이용 — → Queue

( 재귀로도 구현가능 )

### 1. DFS와 BFS



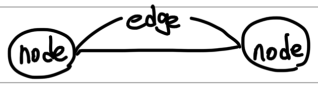




```python
import sys
from collections import deque

# node, branch, first node
n, m, v = map(int, sys.stdin.readline().split())
graph = [[] for _ in range(n+1)]
visited = [False] * (n + 1)

# make adjacency list
for _ in range(m):
    a, b = map(int, sys.stdin.readline().split())
    graph[a].append(b)
    graph[b].append(a)
# sort adjacency list
for i in range(1, n+1):
    graph[i].sort()

dfs(v)
# initialize check list
visited = [False] * (n + 1)
print()
bfs(v)
```




→ 각 노드마다 어떤 edge와 이어져있는지 기록

→ 노드를 방문했는지 check

graph = [[], [2,3,4], [1,4], [1,4], [1,2,3]]

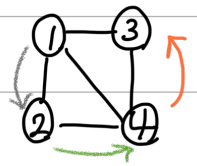
빈공간 / 1번노드와 연결된 노드들

```python
# Depth First Search
def dfs(n):
    print(n, end=' ')
    visited[n] = True
    for i in graph[n]:
        if not visited[i]:
            dfs(i)

# Breadth First Search
def bfs(n):
    visited[n] = True
    queue = deque([n])
    while queue:
        v = queue.popleft()
        print(v, end= ' ')
        for i in graph[v]:
            if not visited[i]:
                queue.append(i)
                visited[i] = True
```

1부터 방문

for i in [2,3,4]:




dfs(2)

↓

for i in [1,4]:

dfs(1) → 건너뜀

dfs(4)

↓

for i in [1,2,3]:

dfs(1) → 건너뜀

dfs(2) → 건너뜀

dfs(3)

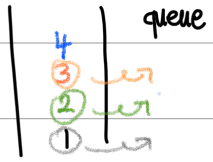
∴ print(1, 2, 4, 3)

1부터 방문

v=1

for i in [2,3,4]:

append(2), append(3), append(4)




v=2

for i in [1,4]:

append(1), append(4) ⇒ 건너뛰기

v=1

for i in [2,3,4]:

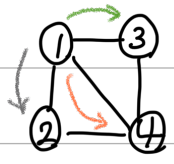
append(1), append(2), append(3) ⇒ 건너뛰기

v=3

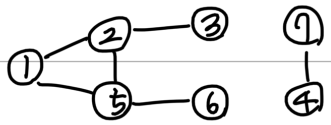
for i in [1,4]:

append(1), append(4) ⇒ 건너뛰기

print(1,2,3,4)



## 2. 바이러스



```python
import sys
n = int(sys.stdin.readline())
_n = int(sys.stdin.readline())
graph = [[] for _ in range(n+1)]
visited = [False] * (n + 1)
ans = []

for i in range(_n):
  a, b = map(int, sys.stdin.readline().split())
  graph[a].append(b)
  graph[b].append(a)

for i in range(1, n+1):
  graph[i].sort()


def dfs(n):
    ans.append(n)
    visited[n] = True
    for i in graph[n]:
        if not visited[i]:
            dfs(i)

dfs(1)
print(len(ans)-1)
```

## 3. 단지번호 붙이기


| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |

<그림 1>

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 2 | 0 | 0 |
| 0 | 1 | 1 | 0 | 2 | 0 | 2 |
| 1 | 1 | 1 | 0 | 2 | 0 | 2 |
| 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 3 | 3 | 3 | 3 | 0 |
| 0 | 3 | 3 | 3 | 0 | 0 | 0 |

<그림 2>

```python
n = int(input())
graph = []
num = []
for i in range(n):
    graph.append(list(map(int, input())))

dx = [0, 0, 1, -1]
dy = [1, -1, 0, 0]
def DFS(x, y):
    if x < 0 or x >= n or y < 0 or y >= n:
        return False

    if graph[x][y] == 1:
        global count
        count += 1
        graph[x][y] = 0
        for i in range(4):
            nx = x + dx[i]
            ny = y + dy[i]
            DFS(nx, ny)
        return True
    return False

count = 0
result = 0

for i in range(n):
    for j in range(n):
        if DFS(i, j) == True:
            num.append(count)
            result += 1
            count = 0

num.sort()
print(result)
for i in range(len(num)):
    print(num[i])
```

[ [ 1, 1, 0, 0]

[ 1, 0, 0, 1]

[ 0, 1, 1, 1]

[ 1, 0, 0, 1] ]

DFS (0,0)

DFS (0,1) — DFS (0,2) F, DFS (0,0) F, DFS (1,1) F

DFS (0,-1) F — DFS (-1,1) F

DFS (1,0)

DFS (-1,0) F