# 개념 ) BFS/DFS

💡 **DFS [ Depth-First Search ; 깊이 우선 탐색 ]**

## 정의

- 깊게 먼저 들어간 후 내려갈 곳이 없으면 옆으로 이동

## 특징

- 모든 노드를 방문하는 경우 사용
- BFS보다 간단
- BFS보다 느림
- 자기 자신을 호출하는 순환 알고리즘

## 구현

- 스택, 재귀 이용
- C언어 / Python

```
void DFS(int v,int N){
    //현재 위치 방문 처리
    DFSvisit[v]=1;

    //현재 노드 값 출력
    printf("%d ",v);

    //연결된 노드 방문
    for(int i=1;i<=N;i++){
        if(Graph[v][i]==1 && DFSvisit[i]==0){
            DFS(i,N);
        }
    }
    return;
}
```

```
def dfs(arr,n,hn):
    dfsvisit[hn] = 1
    print(f"{hn+1}", end=" ")

    for i in range(n):
        if arr[hn][i] ==1 and dfsvisit[i]==0:
            dfs(arr,n,i)
    return
```

💡 **BFS [ Breadth-First Search ; 너비 우선 탐색 ]**

## 정의

- 넓게 먼저 이동 후 옆으로 이동할 곳이 없으면 아래로 이동

## 특징

- 시작 정점으로부터 가까운 정점 먼저 방문
- 최단 경로 문제에서 사용

## 구현

- 큐 이용
- C언어 / Python

```
void BFS(int v,int N){
    int front,rear,pop;
    front = rear = 0;

    //현재 노드 방문처리
    BFSvisit[v]=1;

    //현재 노드 출력
    printf("%d ",v);

    //큐 마지막에 노드 값 넣기
    queue[rear]=v;
```

```
    rear++;

    //큐가 비어 있을 때 까지 반복
    while(front<rear){
        //큐의 첫번째 값 출력
        pop=queue[front];
        front++;

        //큐의 노드를 방문하며 연결된 노드 큐에 삽입
        for(i=1;i<=N;i++){
            if(Graph[Pop][i]==1 && BFSvisit[i]==0){
                printf("%d ",i);
                queue[rear]=i;
                rear++;
                BFSvisit[i]=1;
            }
        }
    }

    return;
}
```

```
def bfs(arr,n,hn):
    start, end, pop =0,0,0


    bfsvisit[hn] = 1

    S.append(hn)
    end = end+1
    print(f"{hn+1}", end=" ")

    while(start<end):
        pop = S[start]
        start = start+1

        for i in range(n):
            if arr[pop][i] ==1 and bfsvisit[i]==0:
                print(f"{i+1}", end=" ")
                S.append(i)
                end = end+1
                bfsvisit[i] =1

    return
```

💡 **서로 비교**

## 시간 복잡도

- 인접 리스트 : O(N+E)
- 인접 행렬 : O(N^2)

⇒ 인접 리스트가 효율적

## 문제 유형

| DFS (스택 / 재귀) | BFS (큐) |
|---|---|
| 모든 정점 방문 | 모든 정점 방문 |
| 경로의 특징 저장 | 최단 거리 |
| 검색 대상 그래프가 크면 | 규모가 작으면 |

💡 문제 리스트

| | 간단 풀 | 링크 |
|---|---|---|
| B | BFS,DFS ) 기본 노드의 이동을 확인 할 수 있는 문제 | 백준_1260 |
| B | BFS,DFS) 2차원 맵에서 이동 (방향 2개) [출발지 ~ 목적] | 백준_16173 |
| B | BFS ) 3차원 맵에서 이동 (방향 6개) [전체 방문] | 백준_7569 |