



DFS / BFS

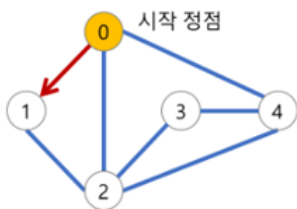
DFS : stack, 재귀 이용

BFS : 최단거리/ 촌수 계산 → Queue 이용

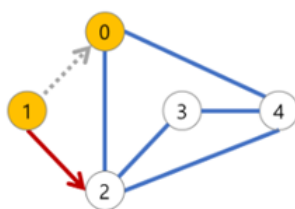
DFS

깊이 우선 탐색(DFS)의 과정

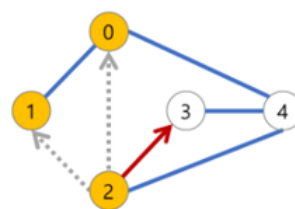
(1) 정점 1 방문



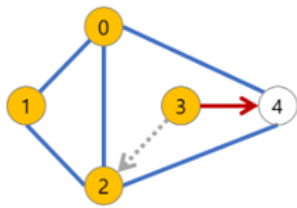
(2) 정점 2 방문



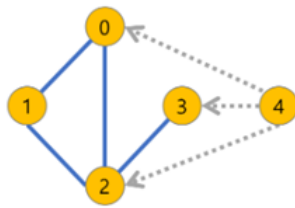
(3) 정점 3 방문



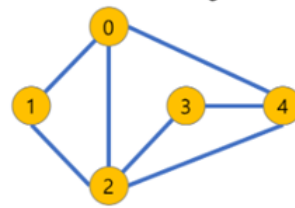
(4) 정점 4 방문



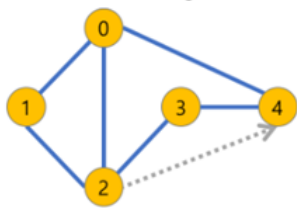
(5) 정점 3으로 backtracking
(다시 돌아와서 탐색하지 않은 정점이 있는지 확인)



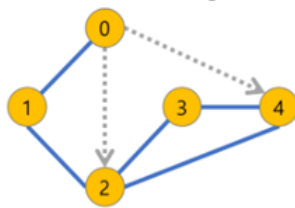
(6) 정점 2로 backtracking



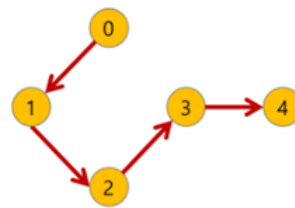
(7) 정점 1로 backtracking



(8) 정점 0으로 backtracking(탐색종료)



(9) 탐색 결과(방문순서: 0,1,2,3,4)



```
int const VNUM = 5;
vector<int> G[VNUM];

G[0].push_back(1);
G[0].push_back(2);
G[0].push_back(4);

...

G[4].push_back(0);
G[4].push_back(3);

int visit[VNUM];
```

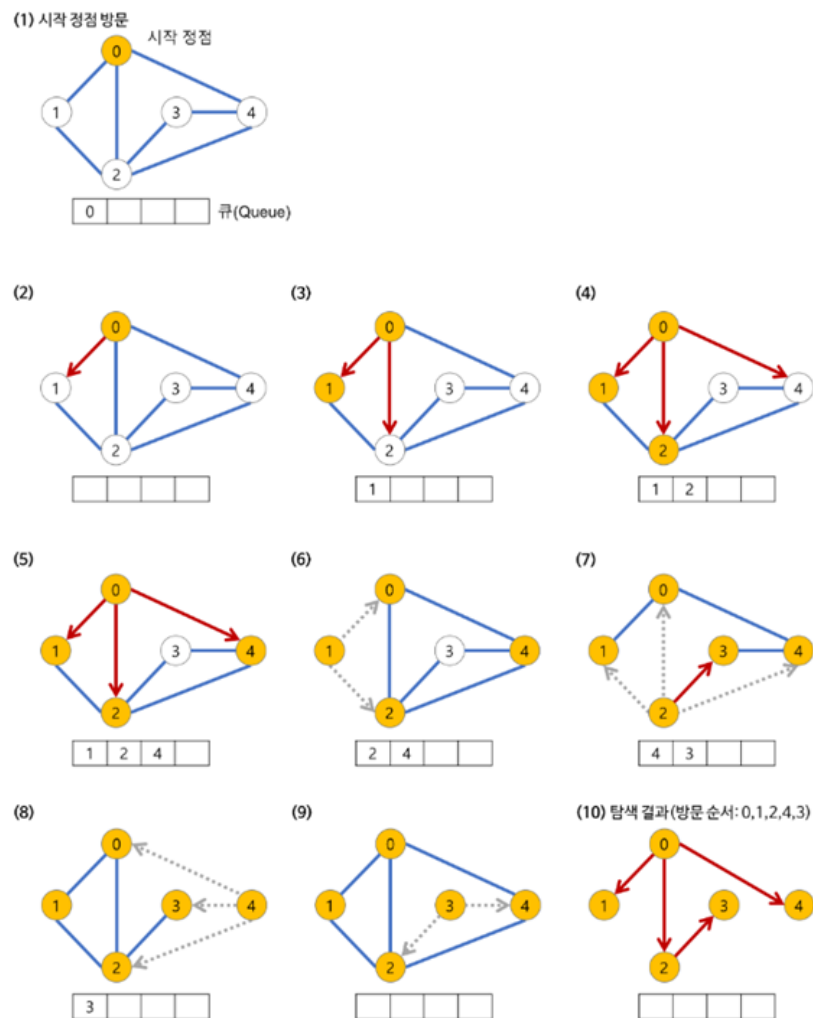
```

void dfs(int v){
    visit[v] = 1;
    int vsize = G[v].size();
    for(int i = 0 ; i < vsize ; i++){
        int nextv= G[v][i];
        if( visit[nextv]) continue;
        dfs(nextv);
    }
}

int main(){
    int start = 3;
    dfs(start);
}

```

BFS



```

int const VNUM = 5;
vector<int> G[VNUM];

G[0].push_back(1);
G[0].push_back(2);
G[0].push_back(4);

```

```

...

G[4].push_back(0);
G[4].push_back(3);

int visit[NUM];

void bfs(int start){
    queue<int> q;
    q.push(start);
    visit[start] = 1;

    while( !q.empty() ){
        int cur = q.front();
        q.pop();

        int vsize = G[cur].size();
        for(int i = 0 ; i < vsize ; i++) {
            int nextv = G[cur][i];
            if( visit[nextv] ) continue;
            q.push(nextv);
            visit[nextv] = 1;
        }
    }
}

int main(){
    int start = 1;
    bfs(start);
}

```

BFS 촌수(Distance) 세는 방법

1. element 에 촌수 추가

queue < pair<int, int > > q 이용
 { vertex, distance } 가 element !

2. queue size 이용하기

```

int distance = 0;
queue<int> q;

q.push(start);
visit[start] = 1;

while( !q.empty() ){
    int qsize = q.size();
    while(qsize--){

        int cur = q.front();
        q.pop();

        // cur의 자식들 push
        int vsize = G[cur].size();
        for(int i = 0 ; i < vsize ; i++) {
            int nextv = G[cur][i];
            if( visit[nextv] ) continue;
            q.push(nextv);
            visit[nextv] = 1;
        }
    }
    distance ++;
}

```

