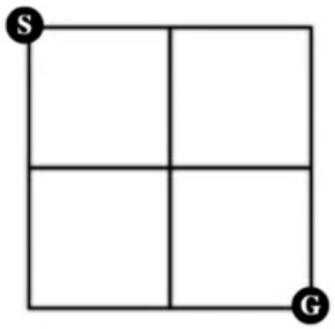
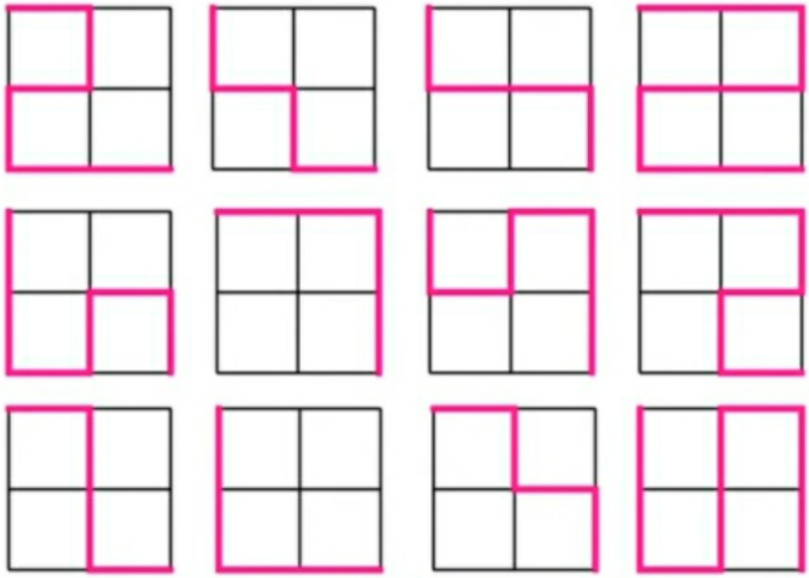
2 × 2





```
Static int count = 0;
FindPath(vertice current) {
    If(current == destination) { ++count; return; }
    forEach neighbor of Current {
        If(!visited[neighbor]){
            visited[neighbor] = 1;
            FindPath(neighbor);
            visited[neighbor] = 0;
        }
    }
    return;
}
```

Cant = {0, 0, 1, 0, 1, 0 ···}

0: can go

1: Can't go

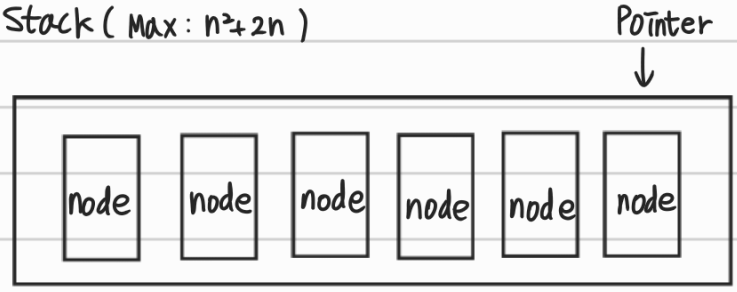
k×k

o o o $(k+1)^2$개

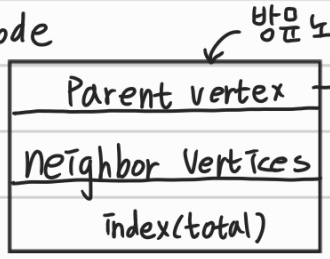
o o o V_i의 neighbors

o o o : $V_{i\pm1}, V_{i\pm(k+1)}$

Stack (Max : n^2+2n)



Node



→ 목표일 경우 count+1, Pop

index == total Pop 방문기록 삭제

index != total Push

```
Push(first node)

while(stack != ∅){
    Node ← Point Node
    if( Node.Parent == destination ){
        pop; count++;
        visited[Node.Parent] = 0;
    }
    if(Node.index == Node.total-1){
        Pop;
        visited[Node.Parent] = 0; break;
    }
    for( Node.index ~ Node.total-1 ){
        if(Node.index not visited){
            Push;
            visited[Node.index] = 1;
            break
        }
    }
}
```

```
Node {
    int current;
    int next[];// 최대 개수
    int index;
}

    Push(Node);
    While(stack != ∅ ) {
        Now ← Stack[Point]
        if( Now.current == end){
            visited[Now.current] =0;
            Pop(); count+1;
            Continue;
        }
        if ( Now.index == Now.total){
            visited[Now.current] =0;
            Pop();
            continue;
        }
        for(Now.index ~ 3 ){
            if( Not visited && Not -1){
                Now.index= i+1;
                Push(Node);
                visited[i] = 1;
                break;
            }
            Now.index = i+1;
        }
    }
```