(훈련반2) Level30

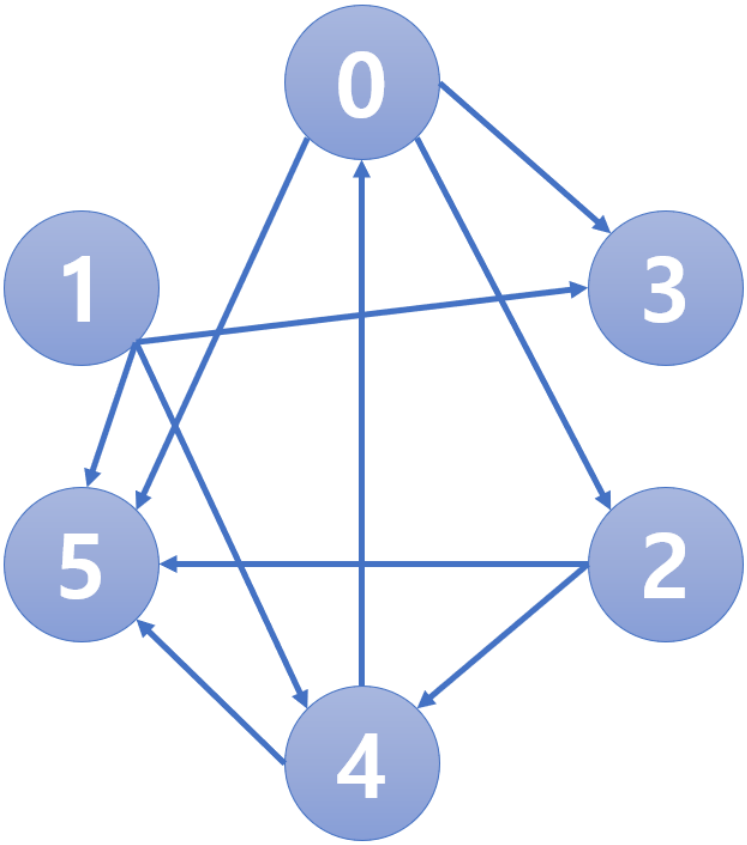
모든 노드를 탐색하는 것과, **모든 경로를 탐색**하는 것은 다릅니다.

모든 경로를 탐색하는 방법을 배우고,

가중그래프에서 최소비용을 구하는 DFS와 BFS를 연습합니다.

Level30 인접행렬 그래프 DFS

문제 1번 [숙제 목록보기]
인접 행렬 그래프를 **깊이 우선 탐색법**으로 탐색 해봅시다.
[그래프]



그래프를 나타내는 인접 행렬을 하드코딩 하세요.
출발지점의 노드 값을 입력 받으세요.
입력 받는 노드 부터 탐색을 시작할때,
DFS로 노드들을 방문할 때마다 노드의 값을 출력해 주세요.
한 노드에 여러 노드로 갈 수 있다면,
숫자가 작은 노드부터 탐색 해주세요.

Ex) 0에서 세개의 노드로 갈 수 있는데, 작은 숫자부터 차례로 2, 3, 5 순으로 탐색이 진행되어야 합니다.

[하드코딩 용 테이블]

	0	1	2	3	4	5
0			1	1		1
1				1	1	1
2					1	1
3						
4	1					1
5						

입력 예제

```
0
```

출력 결과

```
0 2 4 5 3
```

```cpp
#include <iostream>
using namespace std;

const int nodeCnt = 6;
int isVisit[nodeCnt] = {};
int matrix[nodeCnt][nodeCnt] =
{
//      0 1 2 3 4 5
        0,0,1,1,0,1, // 0
        0,0,0,1,1,1, // 1
        0,0,0,0,1,1, // 2
        0,0,0,0,0,0, // 3
        1,0,0,0,0,1, // 4
        0,0,0,0,0,0, // 5
};

void dfs(int _idx)
{
        cout << _idx << " ";
        for (int i = 0; i < nodeCnt; ++i)
        {
                if (matrix[_idx][i] == 1 && isVisit[i] == 0)
                {
                        isVisit[i] = 1;
                        dfs(i);
                }
        }

}

int main(void)
{
        int startNode = 0;
        cin >> startNode;
        isVisit[startNode] = 1;
        dfs(startNode);
        return 0;
}
```

Level30 가중치 인접행렬 DFS

문제 2번 [숙제 목록보기]

각 노드들을 연결하는 선에는 **가중치**가 저장되어 있습니다.

가중치가 저장되어 있는 그래프를 인접행렬로 하드 코딩하고 출발 지점을 입력 받으세요.

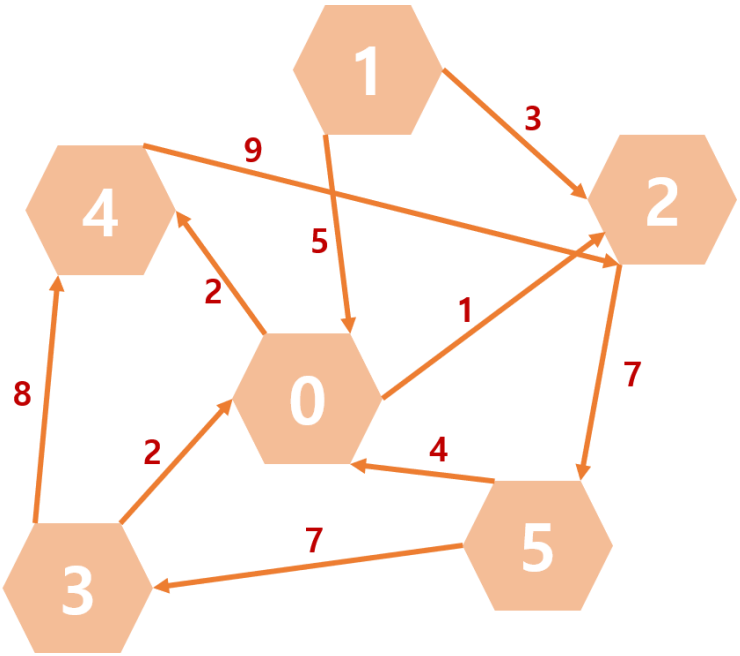
출발 지점부터 깊이 우선 탐색법으로 탐색했을때 **노드 번호와 가중치 경로의 합**을 출력해 주세요.

한번 방문한 노드는 다시 방문할 수 없습니다. 출발지점의 가중치 시작 값은 0 입니다.

한 노드에 여러 노드가 붙어있다면, **숫자가 작은 순서대로 탐색을 시도** 합니다.

Ex) 아래 예제는 0은 2와 4로 이동할 수 있습니다. 2와 4중에 작은 숫자인 2부터 탐색을 시작합니다.

[가중치 그래프]



[가중치 그래프 테이블]

	0	1	2	3	4	5
0			1		2	
1	5		3			
2						7

3	2				8	
4			9			
5	4			7		

입력 예제

```
0
```

출력 결과

```
0 0
2 1
5 8
3 15
4 23
```

```cpp
#include <iostream>
using namespace std;

const int nodeCnt = 6;
int matrix[nodeCnt][nodeCnt] =
{
//      0 1 2 3 4 5
        0,0,1,0,2,0, // 0
        5,0,3,0,0,0, // 1
        0,0,0,0,0,7, // 2
        2,0,0,0,8,0, // 3
        0,0,9,0,0,0, // 4
        4,0,0,7,0,0, // 5
};

int sum = 0;
int isVisit[nodeCnt] = {};
void dfs(int _idx)
{
        for (int i = 0; i < nodeCnt; ++i)
        {
                if (matrix[_idx][i] != 0 && isVisit[i] == 0)
                {
                        isVisit[i] = 1;
                        sum += matrix[_idx][i];
                        cout << i << " " << sum << endl;
                        dfs(i);
                }
        }
}

int main(void)
{
        int startNode = 0;
        cin >> startNode;
        isVisit[startNode] = 1;
        cout << startNode << " " << sum << endl;
        dfs(startNode);
        return 0;
}
```

Level30 트리 인접행렬 BFS

문제 3번 [숙제 목록보기]

트리 자료구조에서 **너비 우선 탐색법**으로 각 노드를 탐색해주세요.

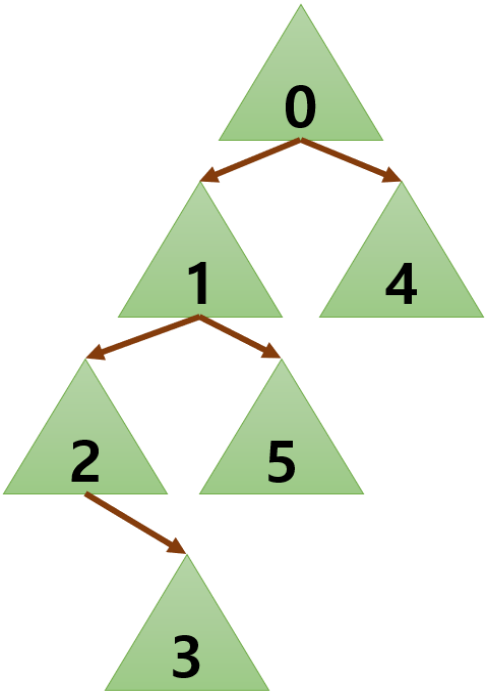
시작 지점 부터, **노드에 방문할 때마다 값을 출력** 해주세요.

출발지점은 입력으로 주어 집니다.

한번 **방문 했던 노드는 방문할 수 없습니다**.

트리 자료구조는 **인접행렬로 하드코딩** 해주세요.

[트리]



[하드코딩용 테이블]

	0	1	2	3	4	5
0		1			1	
1			1			1
2				1		
3						
4						
5						

입력 예제

```
0
```

출력 결과

```
0 1 4 2 5 3
```

```cpp
#include <iostream>
#include <queue>
using namespace std;

const int nodeCnt = 6;
int matrix[nodeCnt][nodeCnt] =
{
//      0 1 2 3 4 5
        0,1,0,0,1,0, // 0
        0,0,1,0,0,1, // 1
        0,0,0,1,0,0, // 2
        0,0,0,0,0,0, // 3
        0,0,0,0,0,0, // 4
        0,0,0,0,0,0, // 5
};

queue<int> que;
int isVisit[nodeCnt] = {};
void bfs()
{

        while (que.empty() == false)
        {
                int target = que.front();
                for (int i = 0; i < nodeCnt; ++i)
                {
                        if (matrix[target][i] == 1 && isVisit[i] == 0)
                        {
                                isVisit[i] = 1;
                                que.push(i);
                        }
                }
                cout << target << " ";
                que.pop();
        }
}

int main(void)
{
        int startNode = 0;
        cin >> startNode;
        que.push(startNode);
        isVisit[startNode] = 1;
        bfs();

        return 0;
}
```

Level30 그래프 BFS

문제 4번 [숙제 목록보기]

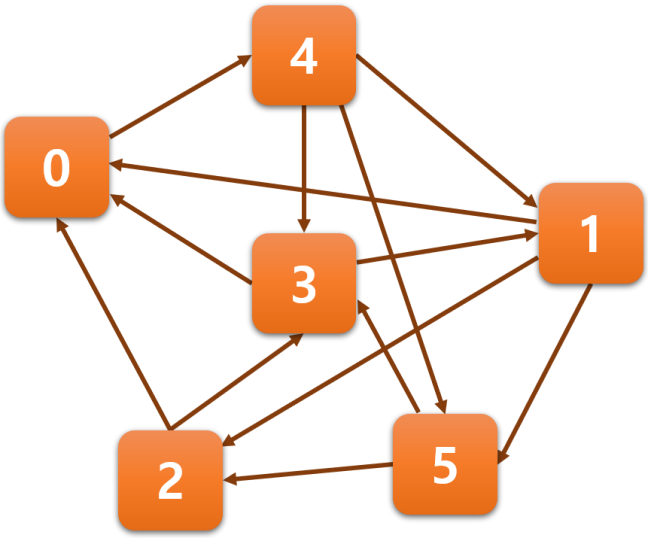
아래에 그래프가 놓여져 있습니다.

입력 받은 출발 지점에서 **너비 우선 탐색법**으로 그래프를 탐색해주세요.

한번 방문했던 노드는 다시 방문할 수 없습니다.

시작 지점부터 BFS가 끝날때까지 방문한 노드를 출력해주세요.

[그래프]



[하드코딩용 테이블]

	0	1	2	3	4	5
0					1	
1	1		1			1
2	1			1		
3	1	1				
4		1		1		1
5			1	1		

입력 예제

```
0
```

출력 결과

```
0
4
1
3
5
2
```

```cpp
#include <iostream>
#include <queue>
using namespace std;

const int nodeCnt = 6;
int matrix[nodeCnt][nodeCnt] =
{
//      0 1 2 3 4 5
        0,0,0,0,1,0, // 0
        1,0,1,0,0,1, // 1
        1,0,0,1,0,0, // 2
        1,1,0,0,0,0, // 3
        0,1,0,1,0,1, // 4
        0,0,1,1,0,0, // 5
};

queue<int> que;
int isVisit[nodeCnt] = {};

void bfs()
{
        while (que.empty() == false)
        {
                int target = que.front();
                for (int i = 0; i < nodeCnt; ++i)
                {
                        if (matrix[target][i] == 1 && isVisit[i] == 0)
                        {
                                que.push(i);
                                isVisit[i] = 1;
                        }
                }
                cout << target << endl;
                que.pop();
        }
}

int main(void)
{
        int startNode = 0;
        cin >> startNode;
        isVisit[startNode] = 1;
        que.push(startNode);
        bfs();
        return 0;
}
```