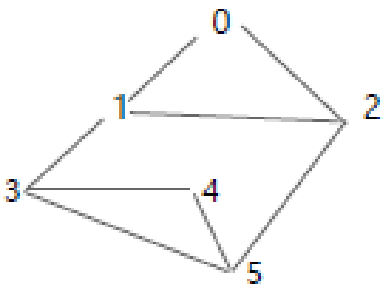
Lab 9 Graph Operations (DFS)

- 입력 그래프 **(Adjacency list 로 표현할 것)**
  - 참고: class Node {private: int data; Node *link;}




```
void main()    {
    initialize VISITED[i]= false;
    get INPUT data;          // graph[i] => head nodes
    print_input data;        // print stored adjacency list
    dfs(v0);                 // starting vertex 0
} //end of main

void dfs(V) { // 강의노트 알고리즘 참조
   visited[V]= true;         print V;
   for (next= graph[v]; next!=NULL; next=next->link)
       if (!visited[next->data]) DFS(next->data);
}
```

- **Output (Data 출력 및 DFS 결과)**

1) Graph Data:

V0: 1->2
V1: 0-> 2->3
V2: 0->1->5
V3: 1->4->5
V4: 3->5
V5: 2->3->4

2) Depth First Search Output (starting from v0)

V0 -> V1 -> V2 -> V5 -> V3 ->V4

Extra Point: Graph Operations - (BFS)

- 그래프 데이터는 다음 Adjacency Matrix 를 사용할 것
- **Output (Data 출력 및 BFS 탐색의 결과)**

******* Adjacent Matrix**

	v0	v1	v2	v3	v4	v5
v0	**0**	**0**	**0**	**0**	**1**	**1**
v1	**0**	**0**	**1**	**1**	**0**	**1**
v2	**0**	**1**	**0**	**1**	**1**	**0**
v3	**0**	**1**	**1**	**0**	**0**	**0**
v4	**1**	**0**	**1**	**0**	**0**	**1**
v5	**1**	**1**	**0**	**0**	**1**	**0**

******* Breadth First Search (BFS) : v0 v4 v5 v2 v1 v3**

- 참고

```
int graph[max][max]={{0,0,0,0,1,1}{0,0,1,1,0,1}.....................}
int front=0,rear=0,queue[5];        char visited[max];

main()
{    initializeQ( );                  // Queue 초기화   front=rear = 0;
     print Adjacent Matrix;     // 그래프 데이터 출력
     for (i=0; i<max;i++) visited[i]='f';    // 방문검사 위한 배열을 초기화
     bfs(v);   // 첫번 노드부터 시작.
}

bfs() {
 addq(v);        // v-> start node;
 v= deque()
 while (!Qempty) {
       for (인접된 모든 노드 w 에 대해서)
            if (not visited)&& (g[v][w] !=0) {
                  addq(w);     visited[w] = 't';     cout << "    " << w;
            }
      v = deletequeue();}
```