图的遍历

和树的遍历类似，我们希望从图中某一顶点出发访遍图中其余顶点，且使每一个顶点仅被访问一次；这一过程叫做图的遍历（Traersing Graph）

深度优先搜索（Depth First Search）

深度优先搜索遍历类似于树的先序遍历，是树的先序遍历的推广

遍历过程

- 假设初始状态是图中所有顶点未曾被访问，则深度优先搜索可以从图中某个顶点v出发
- 访问此顶点v
- 然后依次从v的未曾被访问的邻接点出发深度优先遍历图
- 直至图中所有和v有路径相同的顶点都被访问到；
- 若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为起点，重复上述过程，直至途中所有顶点均被访问到为止

显然，这是一个递归的过程；

为了在遍历过程中区分顶点是否已被访问，需要附设标志数组visited[0,n-1]，其初始值为“false”，一旦某个顶点被访问，则其相应的分量置为“true”

```
void DFS(Graph graph, int vertex){
    visited[v] = TRUE;

    visit(v);

    int w = firstAdjacencyVertex();
    for(;w >= 0;w = nextAdjacencyVertex(graph, v, w)){
        if(!visited[w]){
            DFS(graph, w);
        }
    }
}
```

广度优先搜索（Broadth First Search）

广度优先搜索遍历类似于树的按层序遍历的过程

遍历过程

- 假设从图中某顶点v出发
- 从访问了v之后依次访问v的各个未曾被访问过的邻接点

- 然后分别从这些邻接点出发依次访问他们的邻接点，并使“先被访问的顶点的邻接点”先于“后被访问的的顶点的邻接点”被访问
- 直至图中所有已被访问的顶点的邻接点都被访问到
- 若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为起始点，重复上述过程

```
//广度优先搜索
void BFS(Graph graph){
    //初始化队列
    Queue queue;
    InitQueue(queue);

    for(int v = 0;v < graph.vertexNumber;v++){
        //此顶点已被访问，跳过此顶点
        if(visited[v]){
            continue;
        }

        //将此顶点状态设置为已被访问
        visited[v] = TRUE;
        //访问此顶点
        visit(v);

        //将此顶点入队尾
        EnQueue(queue, v);


        while(!isEmptyQueue(queue)){
            //队头u出列
            DeQueue(queue, u);

            //获取顶点u的第一个邻接顶点w
            int w = FirstAdjacencyVertex(graph, u);
            //依次访问顶点u的邻接顶点，并在访问完之后按照次序访问他们的邻接顶点
            for(;w >= 0;w = NextAdjacencyVertex(graph, u, w)){
                if(!visited[w]){
                    //将此顶点状态设置为已被访问
                    visited[w] = TRUE;
                    //访问顶点w
                    visit(w);
                    //顶点w入队头
                    EnQueue(queue, w);
                }
            }
        }
    }
}
```