20190408 左士海-作业八

1 假设以邻接表表示法作为图的存储结构，设计图的深度优先遍历递归算法。

```
template<typename T>
void ALGraph<T>::DFS(int v, bool *visited) {
    std::cout << adjlist[v].data;    // 访问 v 节点
    visited[v] = true;  // 因为用数组记录状态 该函数传参用顶点索引
    EdgeNode * p = adjlist[v].firstEdge;
    while (p) {
        if (!visited[p->adjvex]) {
            DFS(p->adjvex, visited);    // 没访问过 访问
        }
        p = p->nextEdge;
    }
}
```

```
template<typename T>
void ALGraph<T>::DFSTraverse() {
    bool * visited = new bool[vexnum];
    for (int i = 0; i < vexnum; ++i)
        visited[i] = false;
    for (int v = 0 ; v < vexnum; ++v) {
        if (!visited[v]) {
            DFS(v, visited);
        }
    }
    delete []visited;
}
```

测试：以有向图为例：

```
0 1
0 2
1 3
2 4
3 2
3 4
4 1
DFS: acebd
BFS: acbed
```

2. 试基于图的广度优先搜索策略编写一种算法，判别以邻接表方式存储的有向图中是否存在由顶点 v_i 到顶点 v_j 的路径（$i \neq j$）。

```
template<typename T>
bool ALGraph<T>::isExistPath(int i, int j) {
    int vis[vexnum];
    for (int k = 0; k < vexnum; ++k)
        vis[k] = 0;
    bool ans = false;
    std::queue<int> q;
    q.push(i);
    vis[i] = 1;
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        // 匹配到 就退出
        if (j == v) {
            ans = true;
            break;
        }
        // 将当前节点 v 的所有邻接顶点入队
        EdgeNode * p = adjlist[v].firstEdge;
        while (p) {
            if (!vis[p->adjvex]) {
                vis[p->adjvex] = 1;
                q.push(p->adjvex);
            }
            p = p->nextEdge;
        }
    }
    return ans;
}
```