### **一、无向图**

* ****定义****：边没有方向的图称为无向图。无向图G可以表示为G=<V,E>，其中V是非空集合，称为顶点集；E是V中元素构成的无序二元组的集合，称为边集。直观来说，若一个图中每条边都是无方向的，则称为无向图。无向图中的边均是顶点的无序对，无序对通常用圆括号表示。例如，无序对(vi,vj)和(vj,vi)表示同一条边。
* ****表示****：例如，无向图G2的顶点集V(G2)={v1,v2,v3,v4}，边集E(G2)={(v1,v2),(v1,v3),(v1,v4),(v2,v3),(v2,v4),(v3,v4)}。
* ****特点****：若G是无向图，则0≤e≤n(n-1)/2，恰有n(n-1)/2条边的无向图称无向完全图（Undirected Complete Graph）。注意，完全图具有最多的边数，任意一对顶点间均有边相连。

### **二、连通分量**

****定义****：无向图G的极大连通子图称为G的连通分量（Connected Component）。

****特点****：任何**连通图**的连通分量只有一个，即是其自身。

非连通的无向图有多个连通分量。

### **三、深度优先搜索（DFS）**

* ****定义****：DFS是深度优先搜索（Depth First Search）的缩写，是一种用于遍历或搜索图或树的算法。
* ****过程****：DFS通过从起始节点开始，沿着一条路径一直访问到最深的节点，然后返回到上一个节点，继续探索其他路径，直到所有节点都被访问过为止。
* ****实现****：选择一个起始节点作为当前节点，并标记为已访问。检查当前节点是否满足终止条件，如果满足，则返回结果；如果不满足，则遍历当前节点的所有未访问的邻居节点，标记该邻居节点为已访问，并将该邻居节点作为当前节点，递归调用DFS函数（即进行下一层的深度搜索）。如果当前节点的所有邻居节点都被访问过，则退回到上一层节点，继续搜索其他未被访问的邻居节点。
* ****特点****：DFS沿着一个路径尽可能深入地探索，直到达到最深处或无法继续探索时返回上一级继续探索其他路径。
* ****应用****：DFS算法简单而易于实现，适用于连通性问题和找到可行解的情况。然而，它不适用于求解最优解和最短路径问题，并且在处理大规模和有环图时可能存在一些潜在的问题。因此，在实际应用中需要根据具体情况选择合适的算法。

### **四、广度优先搜索（BFS）**

****定义****：BFS是Breadth-First Search的缩写，即广度优先搜索（也称宽度优先搜索）。

****过程****：BFS从起始节点开始，先访问其所有邻居节点，然后再从这些邻居节点出发，访问它们的未被访问的邻居节点，以此类推，直到所有节点都被访问过为止。

****特点****：BFS强调“宽度”特性，逐层访问节点。

****应用****：BFS广泛应用于图论和搜索问题中，如图的遍历、最短路径寻找（前提是所有边的权重相等）、连通性检测等。此外，在迷宫求解、网络爬虫的数据抓取、游戏AI的路径规划等场景中也常见其身影。

在C++中，**->**操作符用于通过**指针**访问结构体或类的成员，而**.**操作符用于直接**通过对象**访问其成员。当你有一个指向结构体或类实例的指针时，你需要使用 -> 来访问其成员。

struct ArcNode { //弧（arc）结点

int adjvex; //邻结点域 nextarc下一条弧 info数据域存权值

ArcNode\* nextarc;

};

typedef struct VNode {//vertex结点--数组下标就是结点名称

bool vis = false; //标记是否被访问

ArcNode\* firstarc = NULL;//当前顶点出发的第一条弧

}\*AdjList;

struct ALGraph {//采用邻接表形式构建图

AdjList vertices;//vertices变量被声明为一个adjlist指针类型的数组

int vexnum, arcnum; //顶点总数vexnum；弧总数arcnum

};

ArcNode\* p = graph.vertices[src].firstarc;

ArcNode\* q = new(nothrow) ArcNode;

if (!q)

exit(-1);

q->adjvex = dst;