基本概念

* 图是由顶点的有穷非空集合和顶点之间边的集合组成，V是图G中顶点的集合，E是图G中顶点之间边的集合。
* 若代表一条边的顶点序偶是无序的(即该边无方向)，则称此图为无向图。
* 边数相对较少的图称为稀疏图(sparse graph)，边数相对较多的图称为稠密图(dense graph) 任何两顶点间都有边相关联的图称为完全图(complete graph)
* 带权图叫做网络(network)
* 非连通图的极大连通子图称为连通分量
* 极小连通子图
* 连通分量的生成树 就组成了一个非连通图的生成森林
* 邻接矩阵 (Adjacency Matrix) 𝐺. Edge[n][n]

邻接矩阵存储法实现图（代码）

* 定义顶点类、边类
* 定义图类（包括顶点数、边数、顶点数组、邻接矩阵（边类的二维数组）等）
* 图的各方法的实现

邻接表存储法实现图（代码）

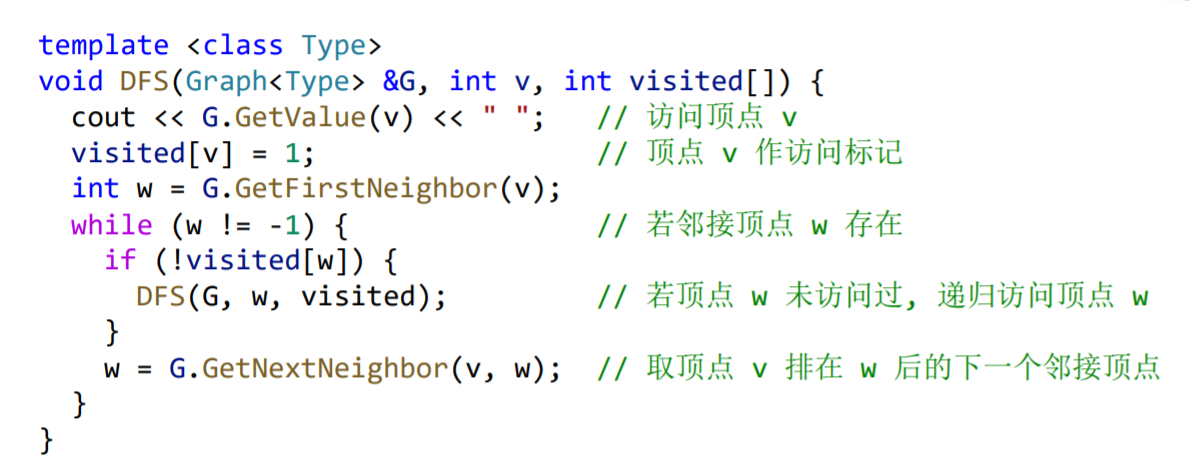
* 顺序存储的顶点表，结点储存了该顶点的值、该顶点的度和指向此顶点边表的指针
* 链接存储的边表（把依附于同一个顶点的边组织成一个单链表），结点储存的值是该边邻接顶点序号（若带权还包括该边的权）
* 图包括了顶点数和顶点表
* 对有向图可能需建立正向逆向两个表

避免同一条边被多次储存的改进：邻接多重表存储法实现图

* 顶点表不变（对有向图顶点结点储存了出度入度两个边表指针）、图不变
* 链接存储的边表（连接了依附于相同顶点的边），结点储存了记录是否处理过的标记、该边两顶点的序号、指向下一条依附顶点1的边的指针、指向下一条依附顶点2的边的指针（对有向图上上述两顶点分为始顶点和终顶点）（若带权还包括该边的权）

图的遍历

* 深度优先搜索DFS (Depth First Search) （代码）



* 广度优先搜索BFS ( Breadth First Search ) 是一种层次遍历

并查集 (Union-Find Data Structure )

* 将一个共同属性（通常为其中一个顶点）赋给子连通图中的每一个顶点，作为标识
* 快速合并（代码）：使标识顶点具有传递性，每次合并仅需改动连通图中作为标识的顶点的标识
* 加权快速合并（代码）：储存连通图的顶点数，避免大树作为小树的子节点

生成树：包含G中全部顶点的一个极小连通子图

最小生成树：无向带权网络中，各边权重总和最小的生成树

* 贪心算法：随意切割多次，每次仅保留权重最小的交叉边，直到剩下n-1条边为止
* 克鲁斯卡尔 (Kruskal) 算法：按权重从小到大的遍历所有边，如果边两端顶点分属不同的连通子图，则将该边加入树中并两连通子图
* 普里姆(Prim)算法：从任一顶点开始遍历所有顶点，将依附于该顶点的不在树中的最小权的边加入树中

最短路径（以一般情况有权有向图为例）

单源最短路径，生成最短路径树（顶点的最短路径数组）：不可避免需边松弛

边松弛算法

* 贝尔曼-福特算法(Bellman–Ford algorithm) 及其代码实现：按顶点的顺序遍历所有边，同时更新各终点的最短路径，重复该遍历直到最短路径不再改变为止（时间复杂度𝑂（𝑉⋅𝐸））
* 贝尔曼-福特算法的改进：如果在某一次遍历时某终点最短路径不变，在下一次遍历则不需要松驰以该顶点出发的边（可利用一个队列保存最短路径改变的终顶点）
* 狄克斯特拉算法 (Dijkstra’s algorithm) 及其代码实现：选择未标记的最短路径顶点，标记该顶点，遍历该顶点的边并更新未标记的相关顶点的最短路径，循环n次直到所有顶点都被标记即遍历完所有边（相比贝尔曼-福特算法，该算法所有边只松驰一次，但要求边都是非负的）

流网络：有向图且包括一个源顶点s和目标顶点t，将边的权视为容量

st-分割：是一种分别包含源顶点𝑠∈𝐴和目标顶点𝑡∈𝐵的分割，容量是指从A到B所有边容量之和。

最小割问题：找到一种最小容量的分割

最大流问题：找到 从s到t流的最大值

基于原有流网络改进各边建立残差网络(Residual network)，进行增广扩大路径流量（若残差容量降为0则除去该边）：边的容量降为剩余容量并增加一条以已用容量为权的逆边

福德-富克逊增广路径算法：找任意一个路径进行增广，重复上述过程直到不再刷新

流值引理：一个流值与一个分割A, B一一对应，流值𝑓小于等于分割的容量，若两者相等则则𝑓是最大流且 𝐴, 𝐵 是最小割