图结构分析实验报告

------2020200982 闫世杰

在给出的无向图中进行 bridge 和 cut_vertex 的查找

运行:

g++ graph.cpp -o xxx

在终端传入参数如下(压缩文件中含给出的测试文件)

./xxx test_data_filename eg: ./xxx graph_1.txt

思路:

根据给出的数据建立图的邻接矩阵,对图进行 dfs,并动态更新各点的 d 和 low 值.根据这两项值进行 bridge 和 cut_vertex 的判断

数据结构:int G_M[][];//采用邻接矩阵方式储存图

bridge:

如果边(u,v)是 bridge,有

1)(u,v)是 tree edge

2)v 的后代节点中没有 backedge 指向 v 的祖先

1)的判断较为简单,在 dfs 中记录 v 的父亲节点 u,所有的 tree_edge=(pred[v],v)

2)的判断:

定义 low[u]为 min{ d[u],min{d[w],w 是 u 的后代节点} }

如果 low 值等于该点的 d 值,说明后代节点中有 backedge 指向其祖先

据上:dfs 遍历整个图,动态更新各个点的 d 值和 low 值,结束之后,遍历所有 vertex,找出 low[v]=d[v]的点,(pred[v],v)即为 bridge

cut_vertex:

dfs 之后,根据 dfs_tree 可以把图中所有的点分成 3 类:

1)dfs_tree 中根节点对应的点:pred[u]==0 的点

2)dfs_tree 中内部节点对应的点:pred[u]!=0 && child_num[u]>0

3)dfs_tree 中叶子节点对应的点:child_num[u]==0

如果 v 是 cut_vertex:

1)根节点:child_num > 1

2)内部节点:v 的后代节点中没有 backedge 指向 v 的祖先

3)叶子节点:必不可能

child_num 的更新:dfs 之后,根据 pred 的值即可更新各个节点 child_num 的值,据此即可进行节点类型的判断,即可完成 1)和 3)

对于 2):同上,根据该点 d 值及其后代节点的 low 值即可进行判断,设置 CV_flag 值,在 dfs 的过程中,根据 low[child]及 d[cur]大小关系动态更新 CV_flag 值即可

据上:dfs遍历整个图之后,遍历整个所有vertex,根据得到的各值对vertex进行节点类型的判断,并针对该节点的类型进行特定的 cut_vertex 的判断即可

时间复杂度:易知两者均分为两步:1)dfs 2)再次遍历 vertex,因此为 O(V+E)

空间复杂度:使用邻接矩阵储存:O(V^2)