数据结构与算法作业3

杨庆龙 1500012956

1. （1）D B

（2）9 29

2.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 迭代步数 | S | A | B | C | D | E | F | G |
| 初态 | {A} | 0 | 1 | infty | Infty | infty | 3 | 4 |
| A | A | void | void | void | A | A |
| 1 | {A,B} | 0 | 1 | 6 | Infty | 8 | 3 | 4 |
| A | A | B | void | B | A | A |
| 2 | {A,B,F} | 0 | 1 | 6 | Infty | 7 | 3 | 4 |
| A | A | B | void | F | A | A |
| 3 | {A,B,F,G} | 0 | 1 | 6 | 12 | 7 | 3 | 4 |
| A | A | B | G | F | A | A |
| 4 | {A,B,F,G,C} | 0 | 1 | 6 | 10 | 7 | 3 | 4 |
| A | A | B | C | F | A | A |
| 5 | {A,B,F,G,C,E} | 0 | 1 | 6 | 10 | 7 | 3 | 4 |
| A | A | B | C | F | A | A |
| 6 | {A,B,F,G,C,E,D} | 0 | 1 | 6 | 10 | 7 | 3 | 4 |
| A | A | B | C | F | A | A |

3.Kruskal算法

只有根结点的树就不列出来了

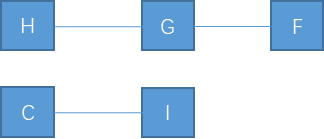
最短的为HG(1)



下一根最短路径为GF(2)



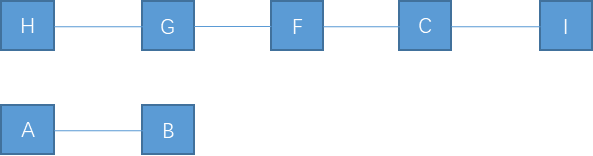
再下一根最短路径为CI(2)



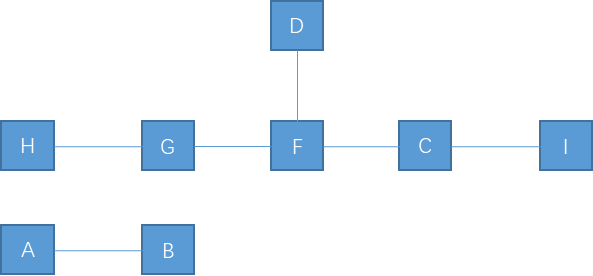
下一根为CF(4)



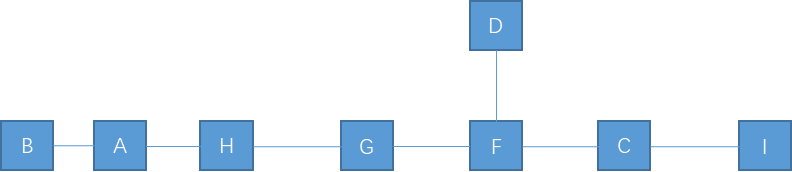
下一根为AB(4)



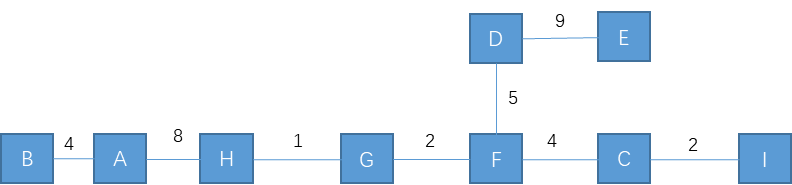
下一根为DF(5)



CD,HI均会成环，不选。选择AH(8)



BC会成环，选择DE(9)



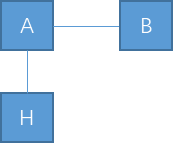
即为总的树

Prim算法

取最短的AB(4)



取最短的AH(8)



取最短的HG(2)



取最短的GF(2)



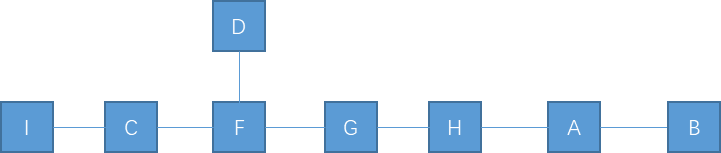
取最短的CF(4)



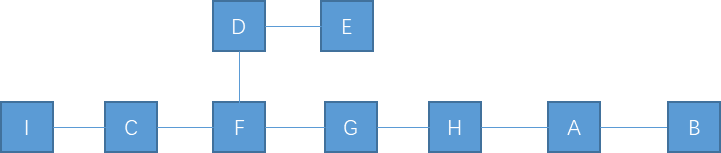
取最短的CI(2)



取DF(5)



取DE(9)



两题结果一致。

4. 证明:对于有向无环图，一定有入度为0的结点，先将这些结点放入矩阵。对于放入的第M行，其子结点一定在M行往下，也就保证了该行的第1到第M-1的元素为0。之后再将已经放入矩阵的结点从图中删除，可以得到一样的结论，即第N行的子结点一定在第N行以下，保证了该行的第1到第N-1的元素为0。重复以上操作，就能使得每一个结点的子结点在矩阵中均位于表示该节点的行的下方，也就实现了主对角线以下的元素全部为0的操作。即，存在操作使得有向无环图的领接矩阵的主对角线以下的元素为0.

5.

Graph getCircle(Graph inGraph) {

Graph outGraph;

Stack stack4zero;

Node tempNode;

for(node in inGraph) {

if(node.inDegree == 0) {

stack4zero.push(node);//将入度为0的节点作为起始结点入栈

}

}

while(!stack4zero.empty()) {

tempNode = stack4zero.top();

stack4zero.pop();

for(node in tempNode.outNodes) {

node.inDegree --;

if(node.inDegree == 0)

stack4zero.push(node);

}

}//将入度为0的结点的节点的每个子结点的入度均减一，并将入度为0的结点入栈，实现深度优先搜索

for(node in inGraph) {

if(node.inDegree)

outGraph.add(node);//将入度还没有变为0的结点加入新图，即可得到环

}

return outGraph;

}