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

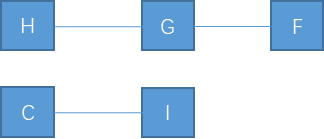
最短的为HG(1)



下一根最短路径为GF(2)



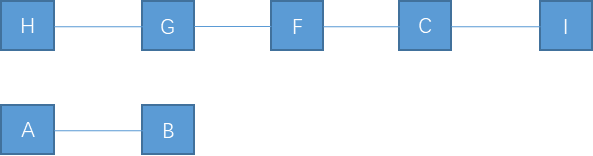
再下一根最短路径为CI(2)



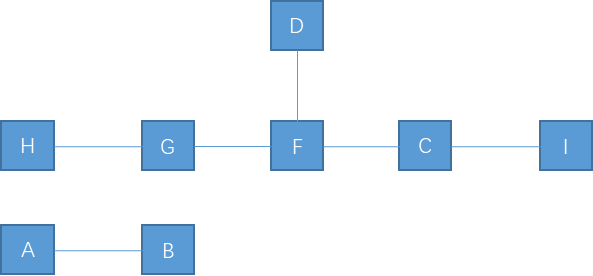
下一根为CF(4)



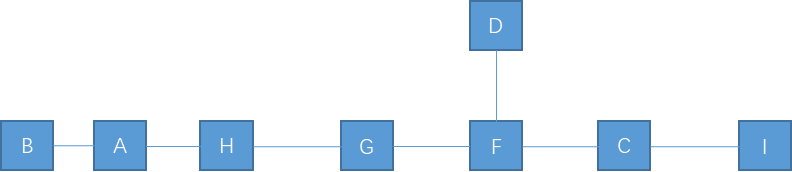
下一根为AB(4)



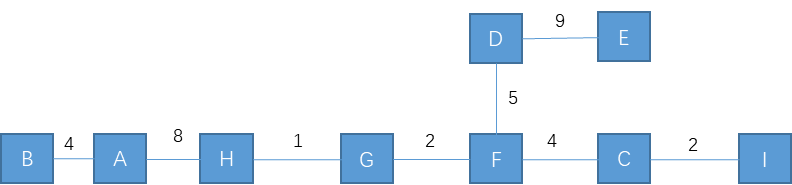
下一根为DF(5)



CD,HI均会成环，不选。选择AH(8)



BC会成环，选择DE(9)



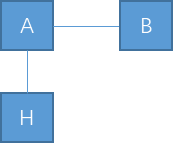
即为总的树

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)



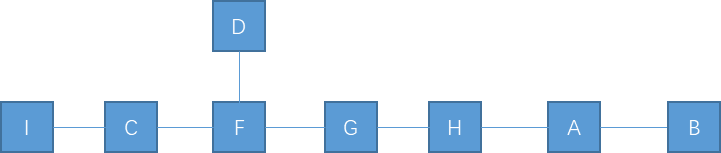
取最短的CF(4)



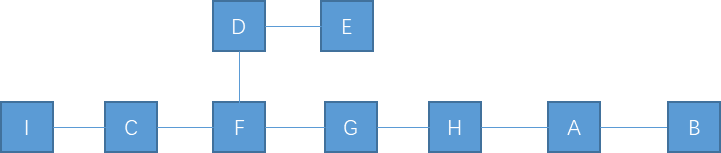
取最短的CI(2)



取DF(5)



取DE(9)



两题结果一致。

4. 证明:对于只有两个点的无环有向图，显然可以表示为主对角线以下元素全为0.

设N个点的无环有向图的邻接矩阵可以表示为主对角线以下元素全为0.

插入第N+1个点，使得这N+1个点依然构成无环有向图。则，对于指向第N+1个点的结点，只是在其最后一列有非零数字m，并设最后一个不为0的是第X行，其坐标为[X,N+1]。对于新加入的第N+1个点，设它的指向向量的第Y个元素n为最左边的不为零的元素，其坐标为[N+1,Y]。显然，此时不满足主对角线以下元素为0。

将第N+1行移动到第Y行，则n移动到[Y,Y+1]，m移动到[X,Y]或[X+1,Y]。当Y<X时

综上，

5.

Graph getCircle(Graph inGraph) {

Graph outGraph;

Stack stack4zero;

Node tempNode;

for(node in inGraph) {

if(node.inDegree == 0) {

stack4zero.push(node);

}

}

while(!stack4zero.empty()) {

tempNode = stack4zero.top();

stack4zero.pop();

for(node in tempNode.outNodes) {

node.inDegree --;

if(node.inDegree == 0)

stack4zero.push(node);

}

}

for(node in inGraph) {

if(node.inDegree)

outGraph.add(node);

}

return outGraph;

}