11.13

#include<algorithm>

int maxDAG(Graphl& G, int v, int depth) {// DFS

int currmax = depth;

for (Edge w = G.first(v); G.isEdge(w); w = G.next(w))

currmax = max(maxDAG(G, G.first(v), depth + 1),currmax);

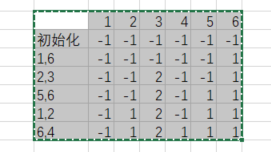
return currmax;

}

11.17

3,2 2,4 4,6 6,1 6,5

11.18



11.19

int maxVertex(Graphl\* G, int \*D)//找到最大费用的节点

{

int i, v = -1;//初始化v为未访问节点

for (i = 0; i<G->n(); i++)

if (G->getMark(i) == UNVISITED) { v = i; break; }

for (i++; i < G->n(); i++)//找到最大D值

if ((G->getMark(i) == UNVISITED) && (D[i] > D[v]))

v = i;

return v;

}

void maxPrim(Graphl\* G, int\* D, int s)

{

int V[5];

int i, w;

for (i = 0; i < G->n(); i++)

{

int v = minVertex(G, D);

G->setMark(v, VISITED);

if (v != s)

AddEdgetoMST(V[v], v);

if (D[v] == INFINITY) return;

for (w = G->first(v); w < G->n();w=G->next(v,w))

if (D[w] > G->weight(v, w))

{

D[w] = G->weight(v, w);

V[w] = v;

}

}

}