①算法分析题6-9 试修改解旅行售货员问题的分支限界法，使得算法保存已产生的排列树。

解：代码如下：

template <class Type>

Type Traveling<Type>::BBTSP(int \*v, Type \*\*G, int tn, Type tNoEdge)

{

priority\_queue<MinHeapNode<Type> > pq;

MinHeapNode<Type> E, N;

Type bestc, cc, rcost, MinSum, \*MinOut, b;

int i, j;

a = G;

n = tn;

NoEdge = tNoEdge;

MinSum = 0; //最小出边费用和

MinOut = new Type[n+1]; //计算MinOut[i]=顶点i的最小出边费用

for(i = 1; i <= n; i++)

{

MinOut[i] = NoEdge;

for(j = 1; j <= n; j++)

if(a[i][j] != NoEdge && (a[i][j] < MinOut[i] || MinOut[i] == NoEdge))

MinOut[i] = a[i][j];

if(MinOut[i] == NoEdge) //无回路

return NoEdge;

MinSum += MinOut[i];

}

//初始化

E.s = 0;

E.cc = 0;

E.rcost = MinSum;

E.x = new int[n];

for(i = 0; i < n; i++)

E.x[i] = i+1;

bestc = NoEdge;

//搜索排列空间树

while(E.s < n-1) //非叶结点

{

if(E.s == n-2) //当前扩展结点是叶结点的父结点 再加2条边构成回路

{ //所构成回路是否优于当前最优解

if(a[E.x[n-2]][E.x[n-1]] != NoEdge && a[E.x[n-1]][1] != NoEdge &&

(E.cc+a[E.x[n-2]][E.x[n-1]]+a[E.x[n-1]][1] < bestc || bestc==NoEdge))

{

//费用更小的路

bestc = E.cc + a[E.x[n-2]][E.x[n-1]] + a[E.x[n-1]][1];

E.cc = bestc;

E.lcost = bestc;

E.s++;

pq.push(E);

}

else

delete []E.x; //舍弃扩展结点

}

else //产生当前扩展结点儿子结点

{

for(i = E.s+1; i < n; i++)

if(a[E.x[E.s]][E.x[i]] != NoEdge)

{

//可行儿子结点

cc = E.cc + a[E.x[E.s]][E.x[i]]; //当前费用

rcost = E.rcost - MinOut[E.x[E.s]]; //更新最小出边费用和

b = cc + rcost; //下界

if(b < bestc || bestc == NoEdge) //子树可能含最优解 结点插入最小堆

{

N.s = E.s + 1;

N.cc = cc;

N.lcost = b;

N.rcost = rcost;

N.x = new int[n];

for(j = 0; j < n; j++)

N.x[j] = E.x[j];

N.x[E.s+1] = E.x[i]; //获得新的路径

N.x[i] = E.x[E.s+1];

pq.push(N); //加入优先队列

}

}

delete []E.x; //完成结点扩展

}

if(pq.empty()) //堆已空

break;

E = pq.top(); //取下一扩展结点

pq.pop();

}

if(bestc == NoEdge) //无回路

return NoEdge;

for(i = 0; i < n; i++) //将最优解复制到v[1:n]

v[i+1] = E.x[i];

while(pq.size()) //释放最小堆中所有结点

{

E = pq.top();

pq.pop();

delete []E.x;

}

return bestc;

}