

**旅行商（TSP）问题：找一条哈密顿回路，使得总代价最小。**

A: 邻接矩阵，A(i,j)=顶点i、j之间的成本。

n：顶点数

minCost：最小成本

optX：最优的哈密顿回路

|  |
| --- |
| 算法1：TSP回溯法（深度优先，递归） |
| input：A[n][n], n |
| output:minCost,optX |
| 1. minCost = MAX |
| 1. optX=[] |
| 1. x = [for i in range(n)]=[0,1,2,…,n-1] |
| 1. **dfs(1,x,0)** |
| 1. output minCost,optX |

|  |
| --- |
| 算法2：dfs(i,x,cost) |
| input: i-处理步骤，x-当前解向量，cost－当前代价（初值为0） |
| output：minCost,optX-全局变量 |
| def dfs(i,x,cost): #第i步处理，得到解向量x和临时代价cost（初值为0） |
| if (i==n-1): #处理到叶子结点 |
| cost += A(x(n-2), x(n-1))+A(x(n-1),x(0)) |
| if cost<minCost: |
| minCost = cost |
| optX = x |
| return |
| for j in range(i, n): #处理中间结点 |
| x(i),x(j) = x(j),x(i) #交换位置（排列树） |
| cost += A(x(i-1),x(i)) |
| if (cost<minCost): #剪枝条件 |
| dfs(i+1,x,cost) #递归处理下一步 |
| cost -= A(x(i-1),x(i)) #回溯处理 |
| x(i),x(j) = x(j),x(i) |

算法时间复杂度：T(n)=(n-1)! = O(n!)

|  |
| --- |
|  |
| #队列结点定义：  struct qNode():  i－处理步骤（位置）  x－当前解向量  cost－当前代价（初值0） |
|  |
| 算法3 TSP分枝限界法（广度优先，队列） |
| input：A[n][n], n |
| output:minCost,optX |
| #初始化 |
| minCost=Max |
| optX=[] |
| #建立初始结点加入队列： |
| e.i=0 |
| e.x=[0,1,2,…,n-1] |
| e.cost=0 |
| qu.push(e) |
| while(!qu.empty()): #处理队列结点直到为空 |
| w = qu.pop()  i = w.i+1 |
| if (i == n): #处理叶子结点 |
| w.cost += A(w.x(n-1),w.x(0)) |
| if w.cost < minCost: #更新最优解 |
| minCost = w.cost |
| optX = w.x |
| else: #处理中间结点 |
| for j in range(i,n):  w1 = createNewNode(i,w.x,w.cost) #创建新结点  w1.x(i), w1.x(j)= w1.x(j), w1.x(i) |
| w1.cost += A(w1.x(i-1), w1.x(i)) |
| if (w1.cost<minCost): |
| qu.push(w1) |
| return minCost,optX |