第五章算法分析题

学号：2209060322　　　姓名：梁桐　　　班级：计算机 2203

第五章 算法分析题

5-3

解：要求最优解，需在算法中记录与当前最优值对应的最优解，在类knap中增加2个成员：

x：从根至当前结点的路径

bestx：当前最优解。

代码：

```
template<class Typew, class Typep>
class knap{
  friend Typep knapsack(Typep*, Typew*, Typew, int, int[]);
private:
  Typep Bound(int i);
  void Backtrack(int i);
  Typew c;
  int n;
      *x;
      *bestx;
  Typew *w;
  Typep *p;
  Typew cw;
  Typep cp;
  Typep bestp;
};
```

```
template<class Typew, class Typep>
void knap<Typew, Typep>::knapsack(int i){
if(i>n){
  for(int j=1; j<=n; j++)
    bestx[j] = x[j];
  bestp = cp;
  return;
}
if(cw+w[i]<=c){
  x[i]=1;
  cw+=w[i];
```

```
  cp += p[i]; knapsack(i+1);
  cw -= w[i];
  cp -= p[i];
  x[i] = 0;
  }
  if(Bound(i+1)>bestp)
    knapsack(i+1);
}
//knapsack()函数进行初始化，回溯法求解.
template <class Typew, class Typep>
Typep knapscak(Typep p[], Typew w[], Typew c, int n, int bestx[])
  Typew w=0;
  Typep P=0;
  Object *Q = new Object[n];
  for(int i =1; i<=n; i++){
    Q[i-1].ID= i;
    Q[i-1].d=1.0*p[i]/w[i];
   P+ =P[i];
   W+ =w[i];
  }
  if(W<=c)
    return P;
  MergesOrt(Q,n);
  Knap<Typew,Typep>k;
  K.p = new Typep[n+1];
  k.w = new Typew[n+1];
  k.x = new int[n+1];
  for (i=1; i<=n; i++){
    k.p[i] =P[Q[i-1].ID];
    k.w[i] = w[Q[i-1].ID];
    k.x[i] = 0;
  }
  k.cp=0;
  k.cw=0;
  k.c =c;
  k.n=n;
  k.bestp=0;
  k.bestx= bestx;
```

```
k.knapsack(1);
for(i=1; i<=n; i++)
   k.x[i]=k.bestx[i];
for(i=1; i<=n; i++)
   k.bestx[Q[i-1].ID]=k.x[i];
delate []Q;
delate []k.w;
delate []k.p;
delate []k.x;
~~delate []k.~~ return k.bestp; }
```

5—5

解：

(1) 任一售货员回路可表示为n个顶点的一个排列 $(\pi(1), \pi(2), \cdots, \pi[n])$ 这个回路的费用为

$h(\pi)=\sum_{i=1}^{n} a(\pi(i)), \pi(i \bmod n+1))$

则 $h(\pi)=\sum_{i=1}^{n} a(\pi(i), \pi(i \bmod n+1)) \leq \sum_{i=1}^{n} \max(\pi(i))$

$=\sum_{i=1}^{n} \max(i) < \sum_{i=1}^{n} \max(i)+1$

(2) 遍历图G即可算出 $\sum_{i=1}^{n} \max(i)+1$ 的值.

```
template< class T>
T Traveling<T>::TSP(int v[]){
  bestc=1;
  for(i=1,MaxCost=0; i<=n; i++){
    for(int j=1; j<=n; j++)
      if(a[i][j] != NoEdge && a[i][j] > MaxCost)
        MaxCost = a[i][j];
```

```
    if(MaxCost == NoEdge)
        return NoEdge;
    bestc += MaxCost;
  }
  x = new int[n+1];
  for(i=1; i<=n; i++)
     x[i]=i;
  bestx = V;
  cc = 0;
  tSP1(2);
  delete []x;
  return bestc;
}
```

//删去TSP回溯法中的语句"bestc == NoEdge;"

```
template <class T>
Void Traveling<T>::tSP1(int i){
  if(i==n){
    if(a[x[n-1]][x[n]] != NoEdge && a[x[n]][1] != NoEdge && (cc + a[x[n-1]]
      [x[n]] + a[x[n]][1] < bestc))
      for(int j=1; j<=n; j++)
         bestx[j] = x[j];
      bestc = cc + a[x[n-1]][x[n]] + a[x[n]][1];
  }
}
else{
```

```
for(int j=i; j<=n; j++){
  if(a[x[i-1]][x[j]] !=NoEdge && (cc+a[x[i-1]][x[j]] <bestc)){
    swap(x[i], x[j]);
    cc+=a[x[i-1]][x[i]];
    tsp1(i+1);
    cc-=a[x[i-1]][x[i]];
    swap(x[i], x[j]);
  }
 }
 }
}
```