【算法分析】

由于商品不超过5种，而每种采购的数量又不超过5，那么可以用一个五维数组来表示第i种商品买ai件的最小费用。

F[a1][a2][a3][ a4][a5] 　　　　　　 　　　　　　　（状态1）

如果我们能够使用第i条商品优惠组合Si即(Si1，Si2，Si3，Si4，Si5)的话，状态就变为了：

F[a1－Si1][a2－Si2][a3－Si3][a4－Si4][a5－Si5] 　　　　　　　 （状态2）

这样的话，状态1的费用为状态2的费用加上Si的费用，而状态2的费用必须最低（很显然，用反证法即可），同时，我们也不管状态2是如何来的（因为每一个优惠商品组合的使用是没有限制的），所以本题既满足无后效性，又符合最优化原理，同时还有大量重叠子问题产生，动态规划解决此题是最好不过了。

所以设F[a1][a2][a3][a4][a5]为买a1件物品1，a2件物品2，a3件物品3，a4件物品4，a5件物品5时所需的最少价格，P[i][j]（j≠0）为第i种优惠方案中j商品的数量，P[i][1001]为第i种优惠方案的优惠价。则有状态转移方程：

F[a1][a2][a3][a4][a5]

＝min{F[a1－P[i][1]][a2－P[i][2]][a3－P[i][3]][a4－P[i][4]][a5－P[i][5]]+P[i][1001]}，（ak－p[i][k]≥0　1 ≤k≤5）

边界条件：F[0][0][0][0][0]＝0

参考程序如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | //商店购物  #include <bits/stdc++.h>  using namespace std;  #define MAXN 10  typedef struct  {  int c,m;  } NODE;  NODE s[MAXN];  long f[MAXN][MAXN][MAXN][MAXN][MAXN];  int m[105][1005];  int Num,t;  int main()  {  int i,j,n,l,A,B,C,D,E;  cin>>Num;  memset(m,0,sizeof(m));  for(i=1; i<=Num; i++)  {  cin>>n;  for(j=1; j<=n; j++)  {  cin>>l;  cin>>m[i][l];  }  cin>>m[i][1001];  }  for(cin>>t,i=1; i<=t; i++)  {  cin>>s[i].c>>s[i].m>>m[++Num][1001];  m[Num][s[i].c]=1;  }  memset(f,1,sizeof(f));  f[0][0][0][0][0]=0;  for(i=1; i<=Num; i++)  {  A=m[i][s[1].c];  B=m[i][s[2].c];  C=m[i][s[3].c];  D=m[i][s[4].c];  E=m[i][s[5].c];  for(int a=A; a<=s[1].m; a++)  for(int b=B; b<=s[2].m; b++)  for(int c=C; c<=s[3].m; c++)  for(int d=D; d<=s[4].m; d++)  for(int e=E; e<=s[5].m; e++)  f[a][b][c][d][e]=  min(f[a][b][c][d][e],f[a-A][b-B][c-C][d-D][e-E]+m[i][1001]);  }  cout<<f[s[1].m][s[2].m][s[3].m][s[4].m][s[5].m]<<endl;  return 0;  } |

|  |  |
| --- | --- |
|  | 再介绍一种最短路算法：把每种状态[a1][a2][a3][a4][a5]（a1件物品1，a2件物品2，a3件物品3，a4件物品4，a5件物品5）看成一个点，则至多7776个点，而每个优惠就是一条边，则至多105条边。接下来就是求[0，0，0，0，0]到目标状态的最短路，用Dijkstra(Heap优化)即可。 |