### 购物问题★

【题目描述】购物问题（shopping）ZJU 1524

老师每周都会交给小光一张购物清单，他需要购买清单上所列的物品并且必须全部买齐。最开始，小光总是穿梭于商店的过道之间，对每样商品选择最便宜的价格购买，但是他逐渐地厌倦了这种方式，并开始了一种新的尝试──对于商店中的每条过道只走一遍，并严格按照清单上物品的顺序购物。现在你要写一个程序来帮助他购物。你所拥有的信息除了清单所列的物品之外，还包括在他选择的整条路线上依次出现的商品和它的价格，你的任务是计算他购齐所有货物的最小花费。

举个例子，如图21.1所示，小光需要购买的货物标号依次是1，1，2，20，他必须从下表中依次选择这四样物品，并使总花费最小（在某些情况下，他也可能根本无法找到一种方案购全所有商品），在这个例子中，他的最小花费是21.30元，即选择0.30元和1元的1号商品，然后选择10元的2号和20号商品。

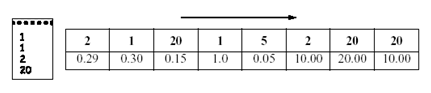


图21.1

【输入格式】

第一行两个数，分别表示清单物品数n（n≤100），路线上物品数m（m≤1 000），接下来一行n个整数描述购物清单的物品，接下来m行每行一个整数和实数，表示物品的编号和价钱。

【输出格式】

如果可以买到所有物品，打印最少花费，精确到小数点后两位。

如果不行，打印“Impossible”。

【输入样例1】

4 8

1 1 2 20

2 0.29

1 0.30

20 0.15

1 1.00

5 0.05

2 10.00

20 20.00

20 10.00

【输出样例1】

21.30

　　【输入样例2】

2 3

1 2

2 0.05

1 10.00

1 3.00

【输出样例2】

Impossible

|  |  |
| --- | --- |
|  | 该题如果用搜索去做会因数据过大而超时，如果用动态规划，只走一遍要么能买齐所有的东西，要么买不齐，所以我们应以这条街的头为起点，经过的每一个商店去买自己所需的东西，最后走到结尾。 |

设Opt[i][j]表示到第i个店买前j件物品所花费的最小价钱。

则动态转移方程为：Opt[i][j]＝min{Opt[i－1][j－1]＋price[j]，Opt[i－1][j]}，条件是该店有这件物品且1≤i≤m，0≤j≤n。

边界条件：Opt[0][0]＝0，Opt[i][0]＝0，Opt[0][j]＝1E100 （科学计数法：1E100＝1×10100）

因为一个店也不去却买了前j件物品是不可能的，所以定为最大值。

输出是Opt[m][n]（m为最后一个店，n为前n件物品）

如果Opt[m][n]＞1E99，则说明前n件物品没有买齐，打印Impossible。

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 | //购物问题  #include <bits/stdc++.h>  #define N 110  #define M 1100  #define INY 1E100  using namespace std;  int sum1[N],sum2[M];  double price[M],opt[M][N];  int main()  {  int n,m;  cin>>n>>m;  for(int i=1; i<=n; i++)  {  cin>>sum1[i];  opt[0][i]=INY; //初始化  }  for(int i=1; i<=m; i++)  cin>>sum2[i]>>price[i];  for(int i=1; i<=m; i++)  for(int j=0; j<=n; j++)  {  opt[i][j]=opt[i-1][j];  if(j>0 && sum1[j]==sum2[i] && opt[i-1][j-1]+price[i]<opt[i][j])  opt[i][j]=opt[i-1][j-1]+price[i];  }  opt[m][n]>1E99?printf("Impossible\n"):printf("%0.2lf\n",opt[m][n]);  return 0;  } |