【算法分析】

本题由矩阵连乘问题改编而来，是一个典型的动态规划问题，但这道题将线性结构改为了环形结构，所以将环形结构展开为线性结构，如果不用求余的方法的话，其数组将扩大一倍。

例如输入样例中的2，3，5，10，数组存为2，3，5，10，2，3，5，10。

设M[i][j]为从第i个矩阵到第j个矩阵所获得的最大值。则：

M[i][j]＝max(M[i][k]＋M[k+1][j]＋r(i－1)r(k)r(j))

例如设珠子的标记为35，40，20，10，15。则可得出：

(1) M[1][2]＝35×40×20＝28 000；M[2][3]＝8 000；M[3][4]＝3 000；

(2) M[1][3]＝max{M[1][2]＋35×20×10，M[2][4]＋35×40×10}＝35 000

　 M[2][4]＝max{M[2][3]＋40×10×15，M[3][4]＋40×20×15}=15 000

(3) M[1][4]＝max{M[2][4]＋35×40×15，M[1][2]＋M[3][4]＋35×20×15，M[1][3]＋35×10×15}＝41 500

则M[1][4]即为所求的结果。

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 | //能量项链  #include <bits/stdc++.h>  using namespace std;  int m[200][200];  int n,temp,p[200];  void matrixch(int p[],int n) //核心算法  {  int i,j,r,k;  for(r=2; r<=n; r++)  for(i=1; i<=n; i++)  {  j=i+r-1;  m[i][j]=0; //初始为0  for(k=i; k<j; k++) //寻找K  {  if(k+1>n) //复制  m[k+1][j]=m[k+1-n][j-n];  temp=m[i][k]+m[k+1][j]+p[i-1]\*p[k]\*p[j];  if(temp>m[i][j])  m[i][j]=temp;  }  }  }  int main()  {  scanf("%d",&n);  for(int i=0; i<=n-1; i++) //从下标0读入  scanf("%d",&p[i]);  for(int i=n; i<2\*n; i++) //复制一遍数组  p[i]=p[i-n];  matrixch(p,n); //调用核心算法  for(int i=1; i<=n; i++)  if(m[i][i+n-1]>temp)  temp=m[i][i+n-1];  printf("%d\n",temp);  return 0;  } |