#### 全排列法★★

|  |  |
| --- | --- |
|  |  |

显然对于给定的牌的序列，可以有多种取法。无论是哪种取法，都要把所有的牌（不包括最两边的两张）都给取走，只是每张牌取的先后不同。这样，一个序列的所有取法其实就是求全排列。

例如对于序列10，1，50，20，5，可以找出1，50，20三张牌的全排列，有：

（1） 1，50，20

（2） 1，20，50

（3） 50，1，20

（4） 50，20，1

（5） 20，1，50

（6） 20，50，1

共6种取法。如取法（1），表示的是先取1，再取50，最后取20。对于每种取法，分别计算出总分并取最小的取法即可。

参考代码如下所示。

|  |  |
| --- | --- |
| 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 | //乘法游戏 — 全排列法  #include <bits/stdc++.h>  using namespace std;  int card[20],permutation[20],n,ans=2147483647;  bool used[20];  void Calculate()  {  int sum=0, remove[20];  for(int i=1; i<=n+2; i++)  remove[i]=1; //remove[i]标记card[i]是否被取走,0:被取走  for(int i=1; i<=n; i++)  {  int j=permutation[i]+1; //取出的牌的位置赋值给j  remove[j]=0;  int mul=card[j]; //取出的牌的值赋给mul  while(remove[j]==0)  j++; //找到右边的牌  mul\*=card[j];  while(remove[j-1]==0)  j--; //找到左边的牌  mul\*=card[j-1];  sum+=mul; //将乘积计入总分  }  if(sum<ans) //选出最小值  ans=sum;  }  void DFS(int dep)  {  if(dep > n)  Calculate();  else  for(int i=1; i<=n; i++)  if(used[i]==0)  {  permutation[dep]=i;  used[i]=1;  DFS(dep+1);  used[i]=0;  }  }  int main()  {  cin>>n;  for(int i=1; i<=n; i++)  cin>>card[i];  n-=2; //要取的牌只有n-2张  DFS(1);  cout<<ans <<endl;  return 0;  } |

#### 区间动态规划法★★

|  |  |
| --- | --- |
| man | 实际上可以使用更为高效的动态规划算法。动态规划算法是算法设计思想中最难，也最有趣的部分。对于该算法，每个刚接触的人都需要一段时间来理解，特别是第一次接触的时候总是想不通为什么这种方法可行。 |

设数组F[i][j]为表示保留第i个位置和第j个位置时的最小值，W[i]表示第i张牌的分值，则可得出：

F[1][3]＝W[1] ×W[2] ×W[3]

F[2][4]＝W[2] ×W[3] ×W[4]

那么求F[1][4]时，取牌方式有两种：

（1）先取第2张牌即F[1][3]＋W[1] ×W[3] ×W[4]

（2）先取第3张牌即F[2][ 4]＋W[1] ×W[2] ×W[4]

则F[1][4]取其两种方式的最小值即可，即：

F[1][4]＝Min{F[1][3]＋W[1] ×W[3] ×W[4]， F[2][ 4]＋W[1] ×W[2] ×W[4]}

则可得一般动态转移方程：

F[i][j]＝Min{F[i][j]，F[i][k]＋W[i] ×W[k] ×W[j]＋F[k][j]} （i＜k＜j）

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 | //乘法游戏 — 动态规划  #include <bits/stdc++.h>  using namespace std;  #define MAXINT 0x7FFFFFFF  #define maxn 110  int f[maxn][maxn],w[maxn];  int main()  {  int n;  scanf("%d",&n);  for(int i=1; i<=n; ++i)  for(int j=1; j<=n; ++j)  f[i][j]=MAXINT;  for(int i=1; i<=n; ++i)  scanf("%d",&w[i]);  for(int i=1; i<=n; ++i) //初始化很关键  f[i][i-1]=f[i][i]=f[i][i+1]=0;  for(int i=2; i<n; ++i) //初始化很关键  f[i-1][i+1]=w[i-1]\*w[i]\*w[i+1];  for(int i=n-2; i>0; --i) //动态规划  for(int j=i+2; j<=n; ++j)  for(int k=i+1; k<j; ++k)  if (f[i][j]>f[i][k]+w[i]\*w[k]\*w[j]+f[k][j])  f[i][j]=f[i][k]+w[i]\*w[k]\*w[j]+f[k][j];  printf("%d\n",f[1][n]);  return 0;  } |