**算法训练 最大的算式**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　题目很简单，给出N个数字，不改变它们的相对位置，在中间加入K个乘号和N-K-1个加号，（括号随便加）使最终结果尽量大。因为乘号和加号一共就是N-1个了，所以恰好每两个相邻数字之间都有一个符号。例如：  
　　N=5，K=2，5个数字分别为1、2、3、4、5，可以加成：  
　　1\*2\*(3+4+5)=24  
　　1\*(2+3)\*(4+5)=45  
　　(1\*2+3)\*(4+5)=45  
　　……

输入格式

　　输入文件共有二行，第一行为两个有空格隔开的整数，表示N和K，其中（2<=N<=15, 0<=K<=N-1）。第二行为 N个用空格隔开的数字（每个数字在0到9之间）。

输出格式

　　输出文件仅一行包含一个整数，表示要求的最大的结果

样例输入

5 2  
1 2 3 4 5

样例输出

120

样例说明

　　(1+2+3)\*4\*5=120

本题的C++参考代码如下：

#include <iostream>

using namespace std;

#define MAXN 15

typedef unsigned long long ULL;

int N, K, All; // 数字总数, 乘号总数, 符号总数

int dp[MAXN][MAXN];

bool Symbol[MAXN-1];

ULL MaxValue = 0;

void Dfs( int deep, int begin )

{

if( deep == K )

{

ULL sum = 1;

for( int i = 0,start=0; i < N; ++i )

{

start = i;

while( Symbol[i] == false && i < All )

++i;

sum \*= dp[start][i];

}

if( sum > MaxValue )

MaxValue = sum;

}

else

{

for( int i = begin; i < All; ++i )

if( !Symbol[i] )

{

Symbol[i] = true;

Dfs(deep+1, i+1);

Symbol[i] = false;

}

}

}

int main(int argc, char\*\* argv) {

cin >> N >> K;

All = N - 1;

for( int i = 0; i < All; ++i) Symbol[i]=false;

for(int i = 0; i < N; ++i ) cin >> dp[i][i];

for( int i = 0; i < N; ++i )

for( int j = i+1; j < N; ++j )

dp[i][j] = dp[i][j-1] + dp[j][j];

Dfs(0,0);

cout << MaxValue;

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<stdlib.h>

long long int jc(int n)

{

long long int sum=1;

while(n)

{

sum\*=n;

n--;

}

return sum;

}

int main()

{

int n,k;

short i,j;

int \*p;

int \*q;

int \*t\_i;

long long Mmax=0;

int times;

int x=0;

scanf("%d%d",&n,&k);

p=(int \*)malloc(n\*sizeof(int));

q=(int \*)malloc((k+1)\*sizeof(int));

t\_i=(int \*)malloc((k+1)\*sizeof(int));

for(i=0;i<n;i++)

scanf("%d",p+i);

times=jc(n-1)/(jc(k)\*jc(n-1-k));

for(i=0;i<k+1;i++)

{

t\_i[i]=1;

}

for(i=1;i<=times;i++) //times次排列组合

{

int sum\_i=0;

int s=0;

for(j=0;j<k;j++) //求最后一个组合数；

{

sum\_i+=t\_i[j];

}

t\_i[j]=n-sum\_i;

for(j=0;j<k+1;j++) //K+1个组合

{

q[j]=0;

int m;

for(m=0;m<t\_i[j];m++) //每个组合的和；

{

q[j]+=p[s++];

}

}

long long max=1;

for(j=0;j<k+1;j++)

{

max\*=q[j];

}

if(max>Mmax)

Mmax=max;

for(j=k-1;j>=0;j--) //每个组合的个数；

{

if((t\_i[j]+1)<=(n-k)&&sum\_i+1<n)

{

t\_i[j]++;

break;

}

else

{

sum\_i-=(t\_i[j]-1);

t\_i[j]=1;

}

}

}

printf("%lld\n",Mmax);

free(q);

free(t\_i);

free(p);

return 0;

}

本题的Java参考代码如下：

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt(), k = sc.nextInt();

long temp;

long[][] arr = new long[n][k + 1];

for(int i = 0; i < n; i++) {

arr[i][0] = sc.nextInt() + (i > 0 ? arr[i - 1][0] : 0);

}

sc.close();

for(int a = 1; a < n; a++) {

for(int b = 1; b <= k; b++) {

for(int c = 0; c < a; c++) {

temp = arr[a][0] - arr[c][0];

arr[a][b] = Math.max(arr[c][b - 1] \* temp, arr[a][b]);

}

}

}

System.out.println(arr[n - 1][k]);

}

}