[编程题] 勇气获得机

时间限制：1秒

空间限制：32768K

妞妞听说Nowcoder Girl女生编程挑战赛要开始了, 但是她没有足够的勇气报名参加, 牛牛为了帮助妞妞,给她准备一台勇气获得机。初始的时候妞妞的勇气值是0, 勇气获得机有两个按钮:

1、N按钮: 如果当期拥有的勇气值为x, 按下之后勇气值将变为2\*x+1，

2、G按钮: 如果当前拥有的勇气值为x, 按下之后勇气值将变为2\*x+2，

勇气值过高也会膨胀,所以妞妞需要将自己的勇气值恰好变为n, 请你帮助她设计一个勇气获得机的按键方案使妞妞的勇气值恰好变为n。

**输入描述:**

输入包括一行, 包括一个正整数n(1 <= n <= 10^9), 表示妞妞最后需要的勇气值。

**输出描述:**

输出一行字符串, 每个字符表示该次妞妞选择按动的按钮,'N'表示该次按动N按钮,'G'表示该次按动G按钮。

**输入例子1:**

20

**输出例子1:**

NGNG

**例子说明1:**

[编程题] 排列

时间限制：1秒

空间限制：32768K

妞妞得到一个(1~n)的排列p1, p2, p3,...,pn, 听村里的老人牛牛说如果让这个排列变为:

对于所有的1 <= i <= n, 都满足pi ≠ i, 就可以获得Google Girl Hackathon的入场券。

妞妞仅允许的操作是: 交换排列中两个相邻的元素, 并且妞妞允许做这个操作任意次。

但是Google Girl Hackathon就快要开始了, 妞妞希望做最少的操作就使排列满足要求, 妞妞希望你能帮助她。

**输入描述:**

输入包括两行, 第一行包括一个正整数n(2 <= n <= 10^5), 表示排列的长度和范围。  
第二行包括n个正整数p1, p2, p3,...,pn, 即妞妞得到的排列, 保证是一个1~n的排列。

**输出描述:**

输出一个整数, 表示妞妞需要的操作次数。

**输入例子1:**

5

1 4 3 5 2

**输出例子1:**

2

**例子说明1:**

[编程题] 打车

时间限制：1秒

空间限制：32768K

妞妞参加完Google Girl Hackathon之后,打车回到了牛家庄。

妞妞需要支付给出租车司机车费s元。妞妞身上一共有n个硬币，第i个硬币价值为p[i]元。

妞妞想选择尽量多的硬币，使其总价值足以支付s元车费(即大于等于s)。

但是如果从妞妞支付的这些硬币中移除一个或者多个硬币，剩下的硬币总价值还是足以支付车费的话，出租车司机是不会接受的。例如: 妞妞使用价值为2，5，7的硬币去支付s=11的车费,出租车司机是不会接受的，因为价值为2这个硬币是可以移除的。

妞妞希望能选取最大数量的硬币，使其总价值足以支付车费并且出租车司机能接受。

妞妞希望你能帮她计算最多可以支付多少个硬币。

**输入描述:**

输入包括两行, 第一行包括两个正整数n和s(1 <= n <= 10, 1 <= s <= 1000), 表示妞妞的硬币个数和需要支付的车费。  
第二行包括n个正整数p[i] (1 <= p[i] <= 100)，表示第i个硬币的价值。  
保证妞妞的n个硬币价值总和是大于等于s。

**输出描述:**

输出一个整数, 表示妞妞最多可以支付的硬币个数。

**输入例子1:**

5 9

4 1 3 5 4

**输出例子1:**

3

**例子说明1:**

[编程题] 勇敢的妞妞

时间限制：1秒

空间限制：32768K

美丽的牛家庄受到了外星人的侵略, 勇敢的妞妞要上战场抵御侵略。

在妞妞上战场前, 村长牛牛给了妞妞N件装备, 妞妞需要选择其中的K件,装备在身上提升自己的战斗力。每件装备有5种属性增幅值,对于第i件装备它的属性增幅值为(ri1, ri2, ri3, ri4, ri5), 分别代表该装备对不同的属性值增幅。

当妞妞装备多件装备的时候,由于装备之前会互相影响, 对于每种属性值的增幅并不是所有装备该属性值之和, 而是该种属性值下所有装备中最大的属性值。而妞妞最终增加的战斗力为这5种属性值增幅之和。

妞妞一定要保卫牛家庄, 所以她希望她能提升尽可能多的战斗力, 请你帮帮她计算她最多能增加多少战斗力。

**输入描述:**

输入包括N+1行,

第一行包括两个正整数N和K(1 <= N <= 10000, 1 <= K <= N), 分别表示一共有的装备数量和妞妞需要选择的装备数量。

接下来的N行,每行5个整数ri1, ri2, ri3, ri4, ri5(0 <= ri1, ri2, ri3, ri4, ri5 <= 10000)表示第i件装备的5种属性值增幅。

**输出描述:**

输出一个整数,表示妞妞最多能增加的战斗力。

**输入例子1:**

4 2

30 30 30 30 0

50 0 0 0 0

0 50 0 50 10

0 0 50 0 20

**输出例子1:**

170

**例子说明1:**

妞妞要从4件装备中选取2件, 如果妞妞选择第1件和第3件装备,那么增加的战斗力为30 + 50 + 30 + 50 + 10 = 170, 这是最大的方案。

[编程题] 平方数

时间限制：1秒

空间限制：32768K

如果一个整数x是某个整数的平方, 我们就把整数x称为平方数。

妞妞最喜欢的数字就是平方数, 妞妞现在给你一个N, 妞妞希望你能帮助她找出不大于N的最大的平方数。

**输入描述:**

输入包括一行, 包括一个正整数N(1 <= N <= 10^9), 表示妞妞给的数字N。

**输出描述:**

输出一个整数, 即不大于N的最大的平方数。

**输入例子1:**

10

**输出例子1:**

9

**例子说明1:**

[编程题] 美丽的项链

时间限制：1秒

空间限制：32768K

妞妞参加了Nowcoder Girl女生编程挑战赛, 但是很遗憾, 她没能得到她最喜欢的黑天鹅水晶项链。

于是妞妞决定自己来制作一条美丽的项链。一条美丽的项链需要满足以下条件:

1、需要使用n种特定的水晶宝珠

2、第i种水晶宝珠的数量不能少于li颗, 也不能多于ri颗

3、一条美丽的项链由m颗宝珠组成

妞妞意识到满足条件的项链种数可能会很多, 所以希望你来帮助她计算一共有多少种制作美丽的项链的方案。

**输入描述:**

输入包括n+1行, 第一行包括两个正整数(1 <= n <= 20, 1 <= m <= 100), 表示水晶宝珠的种数和一条美丽的项链需要的水晶宝珠的数量。

接下来的n行, 每行两个整数li, ri(0 <= li <= ri <= 10), 表示第i种宝珠的数量限制区间。

**输出描述:**

输出一个整数, 表示满足限定条件的方案数。保证答案在64位整数范围内。

**输入例子1:**

3 5

0 3

0 3

0 3

**输出例子1:**

12

**例子说明1：**

**1.**

#include<stdio.h>

#include<string>

using namespace std;

int main(){

    int n;

    while(~scanf("%d",&n)){

        string res="";

        while(n)

            if(n&1) n=(n-1)>>1,res="N"+res;

            else n=(n-2)>>1,res="G"+res;

        printf("%s\n",res.c\_str());

    }

}

2. #include

using namespace std;

int main()

{

    int n; cin >> n;

    int equal = 0, count = 0, data;

    while (count < n) {

        cin >> data;

        if (data == ++count) {

            if (count != n) cin >> data;

            count++; equal++;

        }

    }

    cout << equal;

    return 0;

}

3. #include

#include

using namespace std;

int main()

{

    int n, s; cin >> n >> s;

    int data[10] = { 0 };

    for (int i = 0; i < n; i++) {

        cin >> data[i];

    }

    int result = 0;

    for (int i = 0; i < pow(2, n); i++) {

        int mincoin = 10000, sum = 0, sumcoin = 0;

        int temp = i;

        for (int j = 0; j < n; j++) {

            if (temp % 2) {

                sum += data[j];

                mincoin = mincoin < data[j] ? mincoin : data[j];

                sumcoin++;

            }

            temp >>= 1;

        }

        result = sum >= s && sum - mincoin sumcoin ? result : sumcoin : result;

    }

    cout << result;

    return 0;

}

4. #include <iostream>

#include <algorithm>

using namespace std;

int main()

{

    int n, m; cin >> n >> m;

    int \*\*r = new int\*[n], result = 0;

    int maxr[5] = { 0 };

    for (int i = 0; i < n; i++) {

        r[i] = new int[5];

        for (int j = 0; j < 5; j++) {

            cin >> r[i][j];

            maxr[j] = max(maxr[j], r[i][j]);

        }

    }

    if (m == 1) {

        for (int i = 0; i < n; i++) {

            int temp = 0;

            for (int k = 0; k < 5; k++) {

                temp += r[i][k];

            }

            result = max(result, temp);

        }

    }

    else if (m == 2) {

        for (int i = 0; i < n; i++) {

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

                int temp = 0;

                for (int k = 0; k < 5; k++) {

                    temp += max(r[i][k], r[j][k]);

                }

                result = max(result, temp);

            }

        }

    }

    else if (m == 3) {

        for (int i = 0; i < n; i++) {

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

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

                    int temp = 0;

                    for (int k = 0; k < 5; k++) {

                        temp += max(max(r[i][k], r[j][k]), r[p][k]);

                    }

                    result = max(result, temp);

                }

            }

        }

    }

    else if (m == 4) {

        int maxtemp[5][5] = { 0 };

        for (int p = 0; p < 5; p++) {

            for (int q = p + 1; q < 5; q++) {

                int temp = 0;

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

                    temp = max(temp, r[i][p] + r[i][q]);

                }

                for (int k = 0; k < 5; k++) {

                    if (k != p && k != q) {

                        temp += maxr[k];

                    }

                }

                result = max(result, temp);

            }

        }

    }

    else {

        for (int k = 0; k < 5; k++) {

            result += maxr[k];

        }

    }

    cout << result;

    return 0;

}

5. #include <iostream>

#include <cmath>

using namespace std;

int main()

{

    int n; cin >> n;

    cout << (int)sqrt(n)\*(int)sqrt(n) << endl;

    return 0;

}

6. #include<stdio.h>

#include<string.h>

const int maxn=100;

#define max(a,b) a>b?a:b

int n,m,a[maxn],l[maxn],r[maxn];

long long dp[maxn][maxn+1];

int main(){

    int i,j,k;

    //freopen("input.txt","r",stdin);

    while(~scanf("%d%d",&n,&m)){

        for(i=0;i<n;i++) scanf("%d%d",l+i,r+i);

        memset(dp,0,sizeof(dp));

        for(i=l[0];i<=r[0];i++) dp[0][i]=1;

        for(i=1;i<n;i++)

            for(j=1;j<=m;j++){

                int left=max(0,j-r[i]),right=max(0,j-l[i]);

                for(k=left;k<=right;k++) dp[i][j]+=dp[i-1][k];

            }

        printf("%lld\n",dp[n-1][m]);

    }

}//dp[i][j]表示用前i种宝石组成j颗宝石的项链的方案数

 //但是第i种只能是l[i]到r[i]之间的数量

 //所以dp[i][j]=dp[i-1][j-r[i]]+dp[i-1][j-r[i]+1]+...+dp[i-1][j-l[i]]