hello offer

ALL KILL

- 2018 -

名企校招笔试真题精选

牛客出品

nowcoder.com


燃
京东2018校园招聘
火热启动
更快达到你想要的未来
管理培训生 | 六大方向专业人才 | 国际管培生 | 京锐实习夏令营
网申
07.14 — 09.06
笔试
09.07 — 09.08
面试
09.12 — 10.31
网申入口
campus.jd.com
京东校招 · 燃

目 录

一、京东 2017 秋招笔试真题

1、进制均值

【题目描述】尽管是一个 CS 专业的学生，小 B 的数学基础很好并对数值计算有着特别的兴趣，喜欢用计算机程序来解决数学问题，现在，她正在玩一个数值变换的游戏。她发现计算机中经常用不同的进制表示一个数，如十进制数 123 表达为 16 进制时只包含两位数 7、11（B），用八进制表示为三位数 1、7、3，按不同进制表达时，各个位数的和也不同，如上述例子中十六进制和八进制中各位数的和分别是 18 和 11,。

小 B 感兴趣的是，一个数 A 如果按 2 到 A-1 进制表达时，各个位数之和的均值是多少？她希望你能帮她解决这个问题？

所有的计算均基于十进制进行，结果也用十进制表示为不可约简的分数形式。

输入

输入中有多组测试数据，每组测试数据为一个整数 A(1<=A=<5000).

输出

对每组测试数据，在单独的行中以 X/Y 的形式输出结果。

样例输入

```
5
3
```

样例输出

```
7/3
2/1
```

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//求一个进制数的所有位数之和,n 是数字，a 是进制数

int hexSum(int n,int a) {

    int sum = 0;

    while (n) {

        sum += (n % a);

        n = n / a;

    }

    return sum;

}
```

```
int main() {
    int A;
    while (cin >> A) {
        int sum = 0;
        for (int i = 2; i <= A - 1; i++) {
            sum += hexSum(A, i);
        }
        int r = __gcd(sum, A-2);//约分
        cout << sum / r << '/' << (A - 2) / r << endl;
    }
    return 0;
}
```

2、集合

【题目描述】给你两个集合，要求{A}+{B}。

注：同一个集合中不会有两个相同的元素。

输入：

多组（不超过 5 组）数据。

每组输入数据分为三行，第一行有两个数字 n，m（$0<n,m\leq10000$），分别表示集合 A 和集合 B 的元素个数。后两行分别表述集合 A 和集合 B。每个元素为不超出 int 范围的整数，每个元素之间有一个空格隔开。

输出：

针对每组数据输出一行数据，表示合并后的集合，要求从小到大输出，每个元素之间有一个空格隔开。

样例输入

1 2

1

2 3

1 2

1

1 2

样例输出

1 2 3

1 2

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//用 set 来实现集合的性质即可

set<int> S;

int main() {

    int n, m;

    cin >> n >> m;

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

        int x; cin >> x;

        S.insert(x);

    }

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

        int x; cin >> x;

        S.insert(x);

    }

    for(auto &x : S) {

        if(x != *S.rbegin())

            cout << x << " ";

        else

            cout << x;

    }

    return 0;

}
```

3、通过考试

【题目描述】小明同学要参加一场考试，考试一共有 n 道题目，小明必须做对至少 60%的题目才能通过考试。考试结束后，小明估算出每题做对的概率，p1,p2,...,pn。你能帮他算出他通过考试的概率吗？

输入

输入第一行一个数 n（1<=n<=100），表示题目的个数。第二行 n 个整数，p1,p2,...,pn。表示小明有 pi%的概率做对第 i 题。（0<=pi<=100）

输出

小明通过考试的概率，最后结果四舍五入，保留小数点后五位。

样例输入

4

50 50 50 50

样例输出

0.31250

Hint

第一个样例中，每道题做对的概率都是 0.5，想要通过考试至少要做对三题。所以最后的答案就是

$$(C_4^3+C_4^4)\times 0.5^4=\frac{5}{16}$$

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//dp[i][j]表示状态前 i 个题目对 j 个的概率,挨着 dp 即可。

const int maxn = 100 + 5;

int n,a[maxn];

double dp[maxn][maxn];

int main() {

    scanf("%d",&n);

    for(int i = 1; i <= n; i++) scanf("%d",&a[i]);

    dp[0][0] = 1;

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

        dp[i][0] = dp[i-1][0] * (100.0 - a[i]) / 100;

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

            dp[i][j] = dp[i-1][j] * (100.0 - a[i]) / 100 + dp[i-1][j-1] * 1.0 * a[i] / 100;

        }
```

```
    }

    int begin = (3 * n + 4) / 5;

    double ans = 0;

    for(int i = begin; i <= n; i++) ans += dp[n][i];

    printf("%.5f\n",ans);

    return 0;

}
```

4、异或

【题目描述】异或运算是常见的二进制运算，给出两个 n 位二进制数 a，b。a 异或 b 的运算依次考虑二进制的每一位，若这一位相同，那么这一位的异或结果就是 0，不同就是 1。

例如 a=1100，b=0100。执行 a 异或 b 的运算，a 的最高位是 1，b 的最高位是 0，两个数字不同所以最高位异或结果是 1；a 和 b 次高位都是 1，所以次高位异或为 0；最后两位它们都是 0，所以异或结果也都是 0。那么 a 异或 b 的答案就是 1000。

现在输入两个 n 位二进制数，输出它们异或结果的十进制答案。上述样例中异或的二进制结果为 1000，转化成十进制就是 8。

输入

输入有三行，第一行一个数 n(1<=n<=20)，接下来两行有两个 n 位二进制数。输入的二进制数可能有前导零。

输出

输出一个数，异或结果的十进制数值，不要输出前导零。

样例输入

4

1100

0100

样例输出

8

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//模拟运算然后转为 10 进制输出即可

string s;
```

```
int n;

int solve(int n) {

    int ans = 0;

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

        ans *= 2;

        ans += s[i] - '0';

    }

    return ans;

}

int main() {

    scanf("%d", &n);

    cin >> s;

    int a = solve(n);

    cin >> s;

    int b = solve(n);

    printf("%d\n", a ^ b);

    return 0;

}
```

5、拍卖产品

【题目描述】公司最近新研发了一种产品，共生产了 n 件。有 m 个客户想购买此产品，第 i 个客户出价 Vi 元。为了确保公平，公司决定要以一个固定的价格出售产品。每一个出价不低于要价的客户将会以公司决定的价格购买一件产品，余下的将会被拒绝购买。请你找出能让公司利润最大化的售价。

如果有各种定价方案可以最大化总收入，输出最小的定价。

输入：

输入第一行二个整数 n（1<=n<=1000），m（1<=m<=1000），分别表示产品数和客户数。

接下来第二行 m 个整数 Vi（1<=Vi<=1000000），分别表示第 i 个客户的出价。

输出：

输出一行一个整数，代表能够让公司利润最大化的最小售价。

样例输入：

5 4

2 8 10 7

样例输出：

7

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//排序之后枚举答案计算即可。

const int maxn = 1000 + 5;

int a[maxn];

int main() {

    int n, m, ans = 0, pos;

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

    for(int i = 1; i <= m; i++) scanf("%d", &a[i]);

    sort(a + 1, a + 1 + m);

    for(int i = 1; i <= m; i++) {

        if (ans < a[i] * min(n, m - i + 1)) {

            ans = a[i] * min(n, m - i + 1);

            pos = a[i];

        }

    }

    printf("%d\n", pos);

    return 0;

}
```

二、百度 2017 春招笔试真题

1、 买帽子

【题目描述】度度熊想去商场买一顶帽子，商场里有 N 顶帽子，有些帽子的价格可能相同。度度熊想买一顶价格第三便宜的帽子，问第三便宜的帽子价格是多少？

输入描述:

首先输入一个正整数 N（N <= 50），接下来输入 N 个数表示每顶帽子的价格（价格均是正整数，且小于等于 1000）

输出描述:

如果存在第三便宜的帽子，请输出这个价格是多少，否则输出-1

输入例子:

10

10 10 10 10 20 20 30 30 40 40

输出例子:

30

【答案及解析】

分析:丢进 set 去重,取第三小就行了,没有就-1.

参考 code

```
#include <bits/stdc++.h>

using namespace std;

int n;

set<int> S;

int main() {

        int n;

        cin >> n;

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

                int x; cin >> x;

                S.insert(x);

        }

        int cnt = 0;

        for(auto &x : S) {

                cnt++;

                if(cnt == 3) {
```

```
                    cout << x << endl;
                    break;
                }
            }
        if(cnt < 3) cout << -1 << endl;
        return 0;
}
```

2、度度熊回家

【题目描述】一个数轴上共有 N 个点，第一个点的坐标是度度熊现在位置，第 N-1 个点是度度熊的家。现在他需要依次的从 0 号坐标走到 N-1 号坐标。但是除了 0 号坐标和 N-1 号坐标，他可以在其余的 N-2 个坐标中选出一个点，并直接将这个点忽略掉，问度度熊回家至少走多少距离？

输入描述:

输入一个正整数 N, N <= 50。

接下来 N 个整数表示坐标，正数表示 X 轴的正方向，负数表示 X 轴的负方向。绝对值小于等于 100

输出描述:

输出一个整数表示度度熊最少需要走的距离。

输入例子:

4

1 4 -1 3

输出例子:

4

【答案及解析】

分析：暴力枚举忽略的点,然后维护答案就行了

参考 code

```
#include <bits/stdc++.h>
using namespace std;
int a[55];
int main() {
    int n;
```

```
    cin >> n;
    for(int i = 0; i < n; i++) cin >> a[i];
    int ans = 100000;
    for(int i = 0; i < n; i++) {
        int res = 0;
        int last = a[0];
        for(int j = 1; j < n - 1; j++) {
            if(i == j) continue;
            res += abs(a[j] - last);
            last = a[j];
        }
        res += abs(a[n - 1] - last);
        ans = min(ans, res);
    }
    cout << ans << endl;
    return 0;
}
```

3、寻找三角形

【题目描述】三维空间中有 N 个点，每个点可能是三种颜色的其中之一，三种颜色分别是红绿蓝，分别用'R', 'G', 'B'表示。

现在要找出三个点，并组成一个三角形，使得这个三角形的面积最大。

但是三角形必须满足：三个点的颜色要么全部相同，要么全部不同。

输入描述:

首先输入一个正整数 N 三维坐标系内的点的个数.(N <= 50)

接下来 N 行，每一行输入 c x y z，c 为'R', 'G', 'B' 的其中一个。x，y，z 是该点的坐标。(坐标均是 0 到 999 之间的整数)

输出描述:

输出一个数表示最大的三角形面积，保留 5 位小数。

输入例子:

5

R 0 0 0

R 0 4 0

R 0 0 3

G 92 14 7

G 12 16 8

输出例子:

6.00000

【答案及解析】

分析：暴力枚举,统计就好了。这里需要计算一个三角形面积,用叉积或者海伦公式都可以。

参考 code

```
#include <bits/stdc++.h>

using namespace std;

#define mul(x) ((x)*(x))

char type[55];

double x[55], y[55], z[55];

int main() {

    int n;

    cin >> n;

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

        cin >> type[i] >> x[i] >> y[i] >> z[i];

    }

    double ans = 0.0;

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

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

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

                int ok = 0;

                if(type[i] == type[j]) if(type[i] == type[k]) ok = 1;
```

```
                if(type[i] != type[j]) if(type[i] != type[k]) if(type[j] != type[k]) ok = 1;
                if(!ok) continue;
                double ux = x[j] - x[i], uy = y[j] - y[i], uz = z[j] - z[i];
                double vx = x[k] - x[i], vy = y[k] - y[i], vz = z[k] - z[i];
                double area = sqrt( mul(ux * vy - vx * uy) + mul(uy * vz - vy * uz) + mul(uz * vx - ux * vz) );
                ans = ans > area * 0.5 ? ans : area * 0.5;
            }
        }
    }
    printf("%.5lf\n", ans);
    return 0;
}
```

4、有趣的排序

【题目描述】度度熊有一个 N 个数的数组，他想将数组从小到大 排好序，但是萌萌的度度熊只会下面这个操作：

任取数组中的一个数然后将它放置在数组的最后一个位置。

问最少操作多少次可以使得数组从小到大有序？

输入描述:

首先输入一个正整数 N，接下来的一行输入 N 个整数。(N <= 50，每个数的绝对值小于等于 1000)

输出描述:

输出一个整数表示最少的操作次数。

输入例子:

4

19 7 8 25

输出例子:

2

【答案及解析】

分析:本质是在找原序列可以作为排序后序列前缀序列的长度。然后用 n 减掉这部分长度就好。

看 code,可能更容易明白。

参考 code

```
#include <bits/stdc++.h>

using namespace std;

vector<int> x;

int solve(vector <int> a) {

    vector <int> b;

    b = a;

    sort(b.begin(), b.end());

    int q,n = a.size()-1;

    q = -1;

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

        if (a[i] == b[q + 1]) {

            q++;

            if(q == n) break;

        }

    }

    return n - q;

}

int main() {

    int n;

    cin >> n;

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

        int tmp; cin >> tmp;

        x.push_back(tmp);

    }

    cout << solve(x) << endl;
```

```
    return 0;
}
```

5、不等式数列

【题目描述】度度熊最近对全排列特别感兴趣,对于 1 到 n 的一个排列,度度熊发现可以在中间根据大小关系插入合适的大于和小于符号(即 '>' 和 '<')使其成为一个合法的不等式数列。但是现在度度熊手中只有 k 个小于符号即('<'')和 n-k-1 个大于符号(即'>'),度度熊想知道对于 1 至 n 任意的排列中有多少个排列可以使用这些符号使其为合法的不等式数列。

输入描述:

输入包括一行,包含两个整数 n 和 k(k < n ≤ 1000)

输出描述:

输出满足条件的排列数,答案对 2017 取模。

输入例子:

5 2

输出例子:

66

【答案及解析】

分析:dp[i][j]表示前 i 个数字构成的数列中，恰有 j 个‘<’号的方案数（‘>’号就有 i - j - 1 个）。

dp[i][j] = dp[i - 1][j - 1] * (i - j) + dp[i - 1][j] * (j + 1)

参考 code

```
#include <bits/stdc++.h>
using namespace std;
int n, k, ans;
int dp[1005][1005];
int main() {
    cin >> n >> k;
    for(int i = 1; i <= n; i++)
        dp[i][0] = 1;
    for(int i = 2; i <= n; i++)
```

```
        for(int j = 1; j <= k; j++)
            dp[i][j] = (dp[i - 1][j - 1] * (i - j) + dp[i - 1][j] * (j + 1)) % 2017;
    cout << dp[n][k] % 2017 << endl;
    return 0;
}
```

三、腾讯 2017 秋招笔试真题

1. 编码解码

【题目描述】假定一种编码的编码范围是 a-y 的 25 个字母，从 1 位到 4 位的编码，如果我们把该编码按字典序排序，形成一个数组如下：

a, aa, aaa, aaaa, aaab, aaac, ... , ... , b, ba, baa, baaa, baab, baac, ... , yyyw, yyyx, yyyy

其中 a 的 Index 为 0，aa 的 Index 为 1，aaa 的 Index 为 2，以此类推。编写一个函数，输入是任意一个编码，输出这个编码对应的 index，如：

输入：

baca

输出：

16331

参考用例：

输入：yahy

输出：390833

2、输入：qqq

输出：271250

【答案及解析】

```
int factor[] = {25 * 25 * 25 + 25 * 25 + 25 + 1, 25 * 25 + 25 + 1, 25 + 1, 1};
int encode(char *str) {
    int len = strlen(str);
    int index = len-1;
    for(int i = 0; i < len; ++i) {
        index += factor[i] * (str[i] - 'a');
    }
```

```
    return index;
}
char* decode(int index) {
    char str[4];
    int i = 0;
    while(index >= 0) {
        str[i] = 'a' + index / factor[i];
        index = index % factor[i];
        --index;
        ++i;
    }
    str[i] = '\0';
    return str;
}
```

2、游戏任务

【题目描述】游戏里面有很多各种各样的任务，其中有一种任务玩家只能做一次，这类任务一共有 1024 个，任务 ID 范围[1,1024].请用 32 个 unsigned int 类型来记录着 1024 个任务是否已经完成。初始状态为未完成。
输入两个参数，都是任务 ID，需要设置第一个 ID 的任务为已经完成；并检查第二个 ID 的任务是否已经完成。
输出一个参数，如果第二个 ID 的任务已经完成输出 1，如果未完成输出 0,。如果第一或第二个 ID 不在[1,1024]范围，则输出 1.
如：
输入：
1024 1024
输出：
1
测试用例：
1、
输入：1024 1024
输出：1
2、
输入：1024 1023
输出：0
3、
输入：1022 1025
输出：-1

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;

int main(int argc, char const *argv[])

{

    int open_task(unsigned int task[32],int a,int b);

    unsigned int task[32]= {0};

    int a,b;

    while(scanf("%d,%d",&a,&b))

    {

        printf("the task status of %d is %d\n",b,open_task(task,a,b));

    };

    return 0;

}

int open_task(unsigned int task[32],int a,int b)

{

    if(a>1024||a<1||b>1024||b<1) return -1;

    void set_task(unsigned int task[32],int a);

    int get_task(unsigned int task[32],int b);

    if(!get_task(task,a)) set_task(task,a);

    if(a==b) return 1;

    else return get_task(task,b);

}

void set_task(unsigned int task[32],int a)

{

    int row,column;

    row=(a-1)/32;
```

```
    column=(a-1)%32;


    task[row]+=(unsigned int)pow(2,column);
}
int get_task(unsigned int task[32],int b)
{
    int row,column,status,n;
    unsigned int temp;


    row=(b-1)/32;
    column=(b-1)%32;


    n=0;
    temp=task[row];
    do
    {
        status=temp%2;
        temp=temp/2;
    }
    while((n++)!=column);
    return status;
}
```

3、质数对

【题目描述】给定一个正整数，编写程序计算有多少对质数的和等于输入的这个正整数，并输出结果。输入值小于 1000。如，输入为 10，程序应该输出结果为 2。（共有两对质数的和为 10，分别为（5,5）（3.7））

参考用例：

1、

输入：3

输出：0

2、

输入：50

输出：4

3、

输入：100

输出：6

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;

bool isPrimeNum(int num) {//判断质数

    if(num <= 1)  {

        cout <<"num error!"<<endl;

    }

    if(num == 2) return true;

    if(num % 2 == 0) return false;

    for(int i = 3; i <= sqrt(num); ++i) {

        if(num%i==0) return false;

    }

    return true;

}

int main() {

    int num=0;

    int res=0;

    cin>>num;

    if(isPrimeNum(num-2)) ++res;
```

```
    for(int i = 3; i <= num / 2; i++) {
        if(isPrimeNum(i) && isPrimeNum(num-i)) ++res;
    }
    cout << res << endl;
    return 0;
}
```

4、gohash 解码

【题目描述】geohash 编码：geohash 常用于将二维的经纬度转换为字符串，分为两步：第一步是经纬度的二进制编码，第二步是 base32 转码。

此题考察纬度的二进制编码：算法对纬度[-90, 90]通过二分法进行无限逼近（取决于所需精度，本题精度为 6）。注意，本题进行二分法逼近过程中只采用向下取整来进行二分，针对二分中间值属于右区间。算法举例如下：

针对纬度为 80 进行二进制编码过程：

1) 区间[-90, 90]进行二分为[-90, 0),[0, 90]，成为左右区间，可以确定 80 为右区间，标记为 1；

2) 针对上一步的右区间[0, 90]进行二分为[0, 45),[45, 90]，可以确定 80 是右区间，标记为 1；

3) 针对[45, 90]进行二分为[45, 67),[67,90],可以确定 80 为右区间，标记为 1；

4) 针对[67,90]进行二分为[67, 78),[78,90]，可以确定 80 为右区间，标记为 1；

5) 针对[78, 90]进行二分为[78, 84),[84, 90]，可以确定 80 为左区间，标记为 0；

6) 针对[78, 84)进行二分为[78, 81)，[81, 84)，可以确定 80 为左区间，标记为 0；

已达精度要求，编码为 111100。

样本输入：80

样本输出：111100

参考用例：

1、

输入：80

输出：111100

权重：1

2、

输入：0

输出：100000

3、

输入：-66

输出：001000

【答案及解析】

```
# _*_ coding:utf-8 _*_

right_list = [] # 左区间

left_list = [] # 右区间

geohash_code = []

precision = 6 # 编码精度

def whole_list(right,left):# 生成区间为[-90,90]的列表

        wholelist = []

        for i in range(right,left+1):

                wholelist.append(i)

        return wholelist

whole_list = whole_list(-90,90)# geohash 编码

def geohash(check_number):

        the_whole_list = whole_list

        left_list = the_whole_list[:the_whole_list.index(0)]

        right_list = the_whole_list[the_whole_list.index(0):]

        pre   = precision

        check_number = int(check_number)

        for p in range(1,pre+1):

                if check_number in right_list:

                        geohash_code.append(1)

                        the_whole_list = right_list

                        key = (the_whole_list[0]+the_whole_list[-1])//2

                        left_list = the_whole_list[:the_whole_list.index(key)+1]
```

```
                        right_list = the_whole_list[the_whole_list.index(key):]
                elif check_number in left_list:
                        geohash_code.append(0)
                        the_whole_list = left_list
                        key = (the_whole_list[0]+the_whole_list[-1])//2
                        left_list = the_whole_list[:the_whole_list.index(key)+1]
                        right_list = the_whole_list[the_whole_list.index(key):]
                else:
                        check_number = raw_input(u'您所查询的经纬度不在此区间，请在-90~90 之间选
择并重新输入:')
                        geohash_code(check_number)
def main():
        check_name = raw_input(u'请输入您想要查询的经纬度：')
        geohash(check_name)
        print geohash_code
if __name__ == '__main__':
        main()
```

四、网易 2017 秋招笔试真题

1、计算糖果

【题目描述】A,B,C 三个人是好朋友,每个人手里都有一些糖果,我们不知道他们每个人手上具体有多少个糖果,但是我们知道以下的信息：

A - B, B - C, A + B, B + C. 这四个数值.每个字母代表每个人所拥有的糖果数.

现在需要通过这四个数值计算出每个人手里有多少个糖果,即 A,B,C。这里保证最多只有一组整数 A,B,C 满足所有题设条件。

输入描述:

输入为一行，一共 4 个整数，分别为 A - B，B - C，A + B，B + C，用空格隔开。

范围均在-30 到 30 之间(闭区间)。

输出描述:

输出为一行，如果存在满足的整数 A，B，C 则按顺序输出 A，B，C，用空格隔开，行末无空格。

如果不存在这样的整数 A，B，C，则输出 No

输入例子:

1 -2 3 4

输出例子:

2 1 3

【答案及解析】

分析: 根据三个式子就能求出 A,B,C 了，然后验证一下等式即可。时间复杂度:O(1)

参考代码:

```
#include <iostream>

#include <cstdio>

using namespace std;

int main(){

    int AminusB, BminusC, AplusB, BplusC;

    cin >> AminusB >> BminusC >> AplusB >> BplusC;

    int A = (AplusB + AminusB) / 2;

    int B = (AplusB - AminusB) / 2;

    int C = (BplusC - B);

    if(AminusB == A - B && AplusB == A + B && BminusC == B - C && BplusC == B + C)

        cout << A << " " << B << " " << C << endl;

    else

        cout << "No" << endl;

    return 0;

}
```

2、 买苹果

【题目描述】小易去附近的商店买苹果，奸诈的商贩使用了捆绑交易，只提供 6 个每袋和 8 个每袋的包装(包装不可拆分)。 可是小易现在只想购买恰好 n 个苹果，小易想购买尽量少的袋数方便携带。如果不能购买恰好 n 个苹果，小易将不会购买。

输入描述:

输入一个整数 n，表示小易想购买 n(1 ≤ n ≤ 100)个苹果

输出描述:

输出一个整数表示最少需要购买的袋数，如果不能买恰好 n 个苹果则输出-1

输入例子:

20

输出例子:

3

【答案及解析】

分析: 这题可以考虑使用背包模型去做。但是范围比较小就直接枚举 6 个和 8 个的袋数维护最小值就好了

时间复杂度:O(n^2/48)

参考 code:

```
#include <iostream>
#include <cstdio>
using namespace std;
int main(){
    int n, ans = 1000;
    cin >> n;
    for(int i = 0; i <= 20; i++){
        for(int j = 0; j <= 20; j++){
            if(i * 6 + j * 8 == n){
                ans = min(ans,(i + j));
            }
        }
    }
    if(ans == 1000) ans = -1;
    cout << ans << endl;
}
```

3、数字翻转

【题目描述】对于一个整数 X，定义操作 rev(X)为将 X 按数位翻转过来，并且去除掉前导 0。例如:

如果 X = 123，则 rev(X) = 321;

如果 X = 100，则 rev(X) = 1.

现在给出整数 x 和 y,要求 rev(rev(x) + rev(y))为多少?

输入描述:

输入为一行，x、y(1 ≤ x、y ≤ 1000)，以空格隔开。

输出描述:

输出 rev(rev(x) + rev(y))的值

输入例子:

123 100

输出例子:

223

【答案及解析】

分析： 实现一个 rev()操作，然后输出所求即可时间复杂度：O(length(n)) = O(logn)

参考代码:

```
#include <iostream>

#include <cstdio>

using namespace std;

int rev(int source) {

    int result = 0;

    while (source != 0) {

        int tail = source % 10;

        result = result * 10 + tail;

        source /= 10;

    }

    return result;

}

int main(){

    int x,y;
```

```
    cin >> x >> y;
    cout << rev(rev(x) + rev(y)) << endl;
}
```

4、暗黑的字符串

【题目描述】一个只包含'A'、'B'和'C'的字符串，如果存在某一段长度为 3 的连续子串中恰好'A'、'B'和'C'各有一个，那么这个字符串就是纯净的，否则这个字符串就是暗黑的。例如：

BAACAACCBAAA 连续子串"CBA"中包含了'A','B','C'各一个，所以是纯净的字符串

AABBCCAABB 不存在一个长度为 3 的连续子串包含'A','B','C',所以是暗黑的字符串

你的任务就是计算出长度为 n 的字符串(只包含'A'、'B'和'C')，有多少个是暗黑的字符串。

输入描述:

输入一个整数 n，表示字符串长度(1 ≤ n ≤ 30)

输出描述:

输出一个整数表示有多少个暗黑字符串

输入例子:

2

3

输出例子:

9

21

【答案及解析】

分析： 定义 dp1[i]为结尾两个字符相同长度为 i 的暗黑字符串个数

定义 dp2[i]为结尾两个字符不同长度为 i 的暗黑字符串个数，答案所求就是 dp1[n] + dp2[n]

分别考虑当前这个字符的安放方法，可以得到以下的状态转移方程:

dp1[i + 1] = dp1[i] + dp2[i]

dp2[i + 1] = 2 * dp1[i] + dp2[i]

时间复杂度：O(n)

参考代码:

```
#include <iostream>
#include <cstdio>
using namespace std;
```

```
long long dp1[35];

long long dp2[35];

int main(){

    int n;

    cin >> n;

    dp1[1] = 0, dp1[2] = 3;

    dp2[0] = 1, dp2[1] = 3, dp2[2] = 6;

    for(int i = 3; i <= n; i++){

      dp1[i] = dp1[i - 1] + dp2[i - 1];

      dp2[i] = dp1[i - 1] * 2 + dp2[i - 1];

    }

    cout << dp1[n] + dp2[n] << endl;

}
```

5、 跳石板

【题目描述】小易来到了一条石板路前，每块石板上从 1 挨着编号为：1、2、3.....

这条石板路要根据特殊的规则才能前进：对于小易当前所在的编号为 K 的 石板，小易单次只能往前跳 K 的一个约数(不含 1 和 K)步，即跳到 K+X(X 为 K 的一个非 1 和本身的约数)的位置。 小易当前处在编号为 N 的石板，他想跳到编号恰好为 M 的石板去，小易想知道最少需要跳跃几次可以到达。

例如：

N = 4，M = 24：

4->6->8->12->18->24

于是小易最少需要跳跃 5 次，就可以从 4 号石板跳到 24 号石板

输入描述:

输入为一行，有两个整数 N，M，以空格隔开。

(4 ≤ N ≤ 100000)

(N ≤ M ≤ 100000)

输出描述:

输出小易最少需要跳跃的步数,如果不能到达输出-1

输入例子:

4 24

输出例子:

5

【答案及解析】

分析 题目就是要找个最短路径的长度，这里是个比较显然的 bfs 题。考虑每一步状态拓展，即找某个数的所有约数，这个问题可以在 O(sqrt(num))复杂度完成。所以直接搜出答案即可。

参考代码

```
#include <iostream>
#include <cstdio>
using namespace std;
const int maxn = 130000;
int dst[maxn + 5], Q[maxn + 5], l, r;
int main(){
    int m,n;
    cin >> n >> m;
    for(int i = 0; i < maxn; i++) dst[i] = -1;
    l = r = 0;
    Q[r++] = n; dst[n] = 0;
    for(; l < r; l++){
        int x = Q[l];
    for(int i = 2; i * i <= x; i++){
        if (x % i == 0){
            if (x + i <= m && dst[x + i] == -1)
                dst[x + i] = dst[x] + 1, Q[r++] = x + i;
            if (x + x/i <= m && dst[x + x/i] == -1)
                dst[x + x/i] = dst[x] + 1, Q[r++] = x + x/i;
           }
        }
```

```
    }

    cout << dst[m] << endl;

}
```

6、 回文序列

【题目描述】如果一个数字序列逆置之后跟原序列是一样的就称这样的数字序列为回文序列。例如：

{1, 2, 1}, {15, 78, 78, 15} , {112} 是回文序列,

{1, 2, 2}, {15, 78, 87, 51} ,{112, 2, 11} 不是回文序列。

现在给出一个数字序列，允许使用一种转换操作：

选择任意两个相邻的数，然后从序列移除这两个数，并用这两个数字的和插入到这两个数之前的位置(只插入一个和)。现在对于所给序列要求出最少需要多少次操作可以将其变成回文序列。

输入描述:

输入为两行，第一行为序列长度 n (1 ≤ n ≤ 50)

第二行为序列中的 n 个整数 item[i] (1 ≤ iteam[i] ≤ 1000)，以空格分隔。

输出描述:

输出一个数，表示最少需要的转换次数

输入例子:

4

1 1 1 3

输出例子:

2

【答案及解析】

分析：第一眼以为是个 dp。然后想了想，要变成回文那么两端的值一定要相等，如果已经相等了，把它们移除了继续在剩余的序列上操作；如果不等，我们只能把较小那端(因为没法把较大的值变小)的那个值加到相邻一个位置(等同于题设给的操作)，因此我们一直这样做下去就可以边记录操作次数即可。时间复杂度：O(n)

参考代码:

```
#include <iostream>

#include <cstdio>

using namespace std;

int n;

int a[55];
```

```
int main(){

    cin >> n;

    for(int i = 0; i < n; i++) cin >> a[i];

    int l = 0, r = n - 1;

    int res = 0;

    while(l < r){

        if (a[l] < a[r]){

            a[l+1] += a[l];

            ++l; ++res;

        } else if(a[l] > a[r]){

            a[r-1] += a[r];

            --r; ++res;

        }else{

            ++l, --r;

        }

    }

    cout << res << endl;

}
```

五、华为 2017 秋招笔试真题

1、计算海滩上桃子有多少个

【题目描述】海滩上有一堆桃子，m 只猴子来分。第一只猴子把这堆桃子平均分为 m 份，多了一个，这只猴子把多的一个扔入海中，拿走了一份。第二只猴子把剩下的桃子又平均分成 m 份，又多了一个，它同样把多的一个扔入海中，拿走了一份，第三、第四、第五只、第 m 只猴子都是这样做的，问海滩上原来最少有多少个桃子?

输入：

输入猴子个数 m（3<=m<=9）

输出：

原来最少有多少个桃子

样例输入：

3

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//公式题,推一推就行

int main() {

    int m;

    cin >> m;

    int t = 1;

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

        t = t * m;

    cout << t - m + 1 << endl;

    return 0;

}
```

2、计算重复字符个数

【题目描述】输入一行字符串。如果字符是英文字母，则输出为输入的英文字符+连续出现的次数 ，例如“ABBCCCC”-> “A1B2C4”；否则，丢弃该字符，不输出。

输入：

输入的字符串，长度小于 1024

输出：

输入的英文字符以及其重复的次数

样例输入：

ABBC67%%%CCCAA99

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//按题意模拟即可

const int BUF = 1024;

int main() {
```

```
    char ch = 0;
    char character1[BUF];
    char character[BUF];
    char outputChar[BUF];
    int i = 0, j, t = 1, s = 0, k = 0;
    char *p1, *p2, *p3;
    scanf("%c",&ch);
    while(i < BUF && ch != '\n') {
        character1[i++] = ch;
        scanf("%c",&ch);
    }
    p1 = character1;
    p2 = character;
    for(j = 0; j < i; j++) {
        if ((character1[j]>='A' && character1[j]<='Z') || (character1[j]>='a' &&
character1[j]<='z')) {
            *(p2+k) = *(p1+j);
            k++;
        }
    }
    p1 = character;
    p3 = outputChar;
    for(j = 0; j < k; j++) {
        p1 = character + j;
        p2 = character + j + 1;
        if(*p1 == *p2) {
            t++;
```

```
        }
        else if(t == 1) {
            *(p3+s) = *p1;
            *(p3+s+1) = t;
            s += 2;
            t = 1;
        } else {
            *(p3 + s + 1) = t;
            *(p3 + s) = *p1;
            s += 2;
            t = 1;
        }
    }
    for(i = 0; i < s - 1; i = i + 2) {
        printf("%c%d", outputChar[i], outputChar[i + 1]);
    }
    return 0;
}
```

3、计算字符串中最后一个单词的长度

【题目描述】给定一个字符串 s 由大小写字母和空格组成的字符串，返回字符串的最后一个单词的长度；单词不包含空格；如果最后一个单词不存在，返回 0；

例如"Hello World",返回 5

输入说明：

输入 1 行字符串

输出说明：

输出字符串最后一个单词的长度

样例输入：

I am Lisa

样例输出：

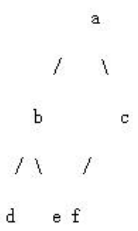
4

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//按着题意读入实现即可

char str[2000];

int main() {

    cin.getline(str, 2000);

    int len = strlen(str);

    int ans = 0;

    while(str[len - 1] != ' ' && len > 0) {

        len--;

        ans++;

    }

    cout << ans << endl;

    return 0;

}
```

4、求二叉树的深度

【题目描述】给出一个字符串形式表达的二叉树，求出指定节点深度。
输入的树形结构字符串格式为：
1、以父节点、左子树、右子树表示的二叉树；每个父节点不会超过两个子节点；
2、树的每一个节点采用单个字母表示；树的层次采用数字表示，树根的层次为 1，下一层为 2，不会超过 9 层；
3、字符串以“节点名称 层次数 节点名称 层次数…”的形式出现，同一个父节点下，先出现的为左子树。



例如，字符串“a1b2c2d3e3f3”生成一棵如下的树：

节点 a 的深度为 3，节点 b 的深度是 2，节点 f 的深度是 1

输入：

一行字符串，表示一个二叉树

一行字符串，一个字符一个节点，输入确保字符不会存在重复节点

输出：

指定节点的深度，如果节点不存在，返回 0；整数之间用空格隔开

样例输入：

```
a1b2c2d3e3f3
ab
```

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//数据结构经典题

int getDepth(vector<vector<char> >& tree,map<char,int>& depth,int i, int j) {

    if (i == tree.size() - 1 || tree[i + 1].size() - 1 < j * 2) {

        depth[tree[i][j]] = 1;

    } else if (tree[i + 1].size() - 1 == j * 2) {

        depth[tree[i][j]] = getDepth(tree, depth, i + 1, j * 2) + 1;

    } else {

        depth[tree[i][j]] = 1 + min(getDepth(tree, depth, i + 1, j * 2), getDepth(tree, depth, i
+ 1, j * 2 + 1));

    }

    return depth[tree[i][j]];

}

int main() {

    string s;

    while (cin >> s) {

        vector<vector<char> > tree;
```

```
        int curLevel = 0;
        vector<char> level;
        for(int i = 0; i < s.size();) {
            if (s[i + 1] - '0' != curLevel) {
                tree.push_back(level);
                level.clear();
                curLevel = s[i + 1] - '0';
            }
            level.push_back(s[i]);
            i = i + 2;
        }
        tree.push_back(level);

        map<char, int> depth;
        getDepth(tree,depth, 1, 0);

        string test;
        cin >> test;
        for(int i = 0; i < test.size(); i++) {
            cout << depth[test[i]] << " ";
        }
        cout << endl;
    }
}
```

六、美团点评 2017 秋招笔试真题

1、大富翁游戏

【题目描述】大富翁游戏，玩家根据骰子的点数决定走的步数，即骰子点数为 1 时可以走一步，点数为 2 时可以走两步，点数为 n 时可以走 n 步。求玩家走到第 n 步（n<=骰子最大点数且是方法的唯一入参）时，总共有多少种投骰子的方法。

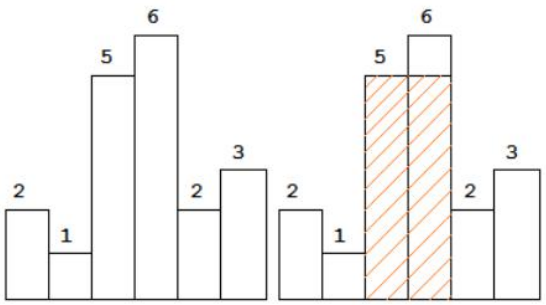
输入：
6
输出：
32

【答案及解析】

```
public class Solution {

      public int JumpFloorII(int target) {

            if (target <= 0) {

                  return -1;

            } else if (target == 1) {

                  return 1;

            } else {

                  return 2 * JumpFloorII(target - 1);

            }

      }

}
```

2、最大矩形面积

【题目描述】给定一组非负整数组成的数组 h，代表一组柱状图的高度，其中每个柱子的宽度都为 1。
在这组柱状图中找到能组成的最大矩形的面积（如图所示）。
入参 h 为一个整型数组，代表每个柱子的高度，返回面积的值。



【答案及解析】

```
# coding: utf-8
def maxArea(histogram):
     histogram.append(-1)
     index = 0
     max = 0
     stack = []
     while(index < len(histogram)):
         if (len(stack) == 0 or histogram[stack[-1]] <= histogram[index]):
            stack.append(index)
            index = index + 1

        else:
           top = stack.pop()
            area = 0
            if (len(stack) == 0):
                area = histogram[top] * index
            else:
                area = histogram[top] * (index-stack[-1]-1)
            if (area > max):
                max = area
    return max
if __name__ == "__main__":
    print(maxArea([2, 1, 5, 6, 2, 3]))
```

3、拼凑钱币

【题目描述】给你六种面额 1、5、10、20、50、100 元的纸币，假设每种币值的数量都足够多，编写程序求组成 N 元（N 为 0~10000 的非负整数）的不同组合的个数。

输入：

N: 0

输出：

1

【答案及解析】//完全背包问题

```
function solve (all) {

    const arr = [1, 5, 10, 20, 50, 100],

        len = arr.length,

        res = [];

    for (let i = 0; i <= len; i++) {

        res[i] = [];

        res[i][0] = 1;
```

```
    }
    for (let j = 1; j <= all; j++) {
        res[0][j] = 0;
    }
    for (let i = 1; i <= len; i++) {
        for (let j = 1; j <= all; j++) {
            res[i][j] = 0;
            for (let k = 0; k <= j / arr[i - 1]; k++) {
                res[i][j] += res[i - 1][j - k * arr[i - 1]];
            }
        }
    }
    return res[len][all];
}
```

4、整数相加

【题目描述】请设计一个算法能够完成两个用字符串存储的整数进行相加操作，对非法的输入则返回"error"， 例如：输入“123”、“123” 则返回 "246", 输入"abc" 、"123"则返回"error"

输入：
"123" ,"123"
"abc", "123"
"-10","123"
"65536000", "3553600000"

输出：
“246”
"error"
"113"
"3619136000"

【答案及解析】

```
function add(m, n){
    var typeM = typeof(m);
    var typeN = typeof(n);
```

```
    if(typeM==typeN){
        console.log(parseInt(m)+parseInt(n));
    }else{
        console.log("error");
    }
}
```

5、最长公共子串

【题目描述】给定两个字符串 s1 和 s2,找出两个字符串的最长公共子串（LCS）的长度。如输入 s1=”abcdef”,s2=”gcdemf”,其最长公共子串为”cde”,输出为 3。

输入：

case1:

s1="abhike" s2="ebhimg

case2:

s1="klmi"

s2="klmiop"

输出：

case1:

bhi

case2:

Klmi

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
int dp[1007][1007]={0};
string a, b;
string jj;
int main() {
    int mmax = -1e9;
```

```
    cin >> a >> b;
    memset(dp, 0, sizeof(dp));
    for(int i = 0; i < a.size(); i++) {
        for(int j = 0; j < b.size(); j++) {
            if(a[i] == b[j])
                dp[i + 1][j + 1] = dp[i][j] + 1;
            if(dp[i + 1][j + 1] > mmax)
                mmax = dp[i + 1][j + 1];
        }
    }
    cout << mmax << endl;
    return 0;
}
```

6、合并有序表

【题目描述】这里有两个有序列表，用自己擅长的语言合并成一个新的有序列表，默认都采用升序。

输入：

[1,3,4,5,8,9]

[2,6,7,10,12,13,15]

输出：

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15]

【答案及解析】

```
def sort(list_a, list_b):
    tmp = []
    i, j = 0, 0
    while len(list_a[i:]) > 0 and len(list_b[j:]):
        if list_a[i] <= list_b[j]:
            tmp.append(list_a[i])
```

```
            i += 1
        else:
            tmp.append(list_b[j])
            j += 1
    tmp.extend(list_a[i:])
    tmp.extend(list_b[j:])
    return tmp


def main():
    list_a = [1,3,4,5,8,9]
    list_b = [2,6,7,10,12,13,15]
    l = sort(list_a, list_b)
    print l


if __name__ == '__main__':
    main()
```

七、今日头条 2017 秋招笔试真题

1、找出函数的最宽尖峰

【题目描述】按数组的形式给出函数 f(x)的取值，即数组 A 的 A[0]元素为 f(0)的取值，数组的取值都为整数，函数在每个点都是严格单调递增或者严格递减（即 A[i-1] != A[i] != A[i+1]），要求找出最宽的先上升后下降的区间（这个区间内函数的值必须先上升到一个点然后下降，区间的上升段和下降段长度必须都大于 0）。

1.如果找到符合条件的最大区间输出数组对应的左右下标（保证只有一个最大区间）

2.找不到那么输出-1 -1

输入格式
n
n 长度的整数数组

输出格式
区间的范围

输入样例
10
1 3 1 2 5 4 3 1 9 10

输出样例
2 7

数据规模
对于 100% 的数据，1 <=n <=10, 000, 000

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//预处理左右各做一遍最长上升子串, 然后维护一个最大和即可

int main() {

    int n;

    cin >> n;

    vector<int> x(n, 0);

    vector<int> l(n, 0);

    vector<int> r(n, 0);

    for(int i = 0; i < n; i++) scanf("%d", &x[i]);

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

        if(x[i] > x[i - 1]) {

            l[i] = l[i - 1] + 1;

        }

    }

    for(int i = n - 2; i >= 0; i--) {

        if(x[i] > x[i + 1]) {

            r[i] = r[i + 1] + 1;

        }

    }

    int mx = 0, ll = -1, rr = -1;

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

```
        if(l[i] > 0 && r[i] > 0 && l[i] + r[i] > mx) {
            mx = l[i] + r[i];
            ll = i - l[i];
            rr = i + r[i];
        }
    }
    cout << ll << " " << rr << endl;
    return 0;
}
```

2、Paragraph

【问题描述】给定一个段落，由 N 个句子组成。第 i 个句子的长度为 L[i]，包含的单词个数为 W[i]。
句子不包含任何除字母和空格( )外的符号。
每个句子内部，含有若干个单词，由空格( )分隔。句子不会包含连续的空格。
随后给定 M 个查询，每个查询包含一个句子，需要在段落中寻找相同单词数量最多的句子。重复的单词只计一次，且不区分大小写。
输入数据将保证结果是存在且唯一的。

输入格式

第一行是两个整数 N 和 M。
接下来的 N+M 行，每行包含一个句子。
前 N 行代表段落中的句子，后 M 行表示查询。

输出格式

输出 M 行，每行代表查询的结果。

输入样例

```
6 3
An algorithm is an effective method that can be expressed within a finite amount of space and time
Starting from an initial state and initial input the instructions describe a computation
That when executed proceeds through a finite number of successive states
Eventually producing output and terminating at a final ending state
The transition from one state to the next is not necessarily deterministic
Some algorithms known as randomized algorithms incorporate random input
Next to the transition
Wormhole, infinite time and space
The transition from one state to the next is not necessarily deterministic
```

输出样例

```
The transition from one state to the next is not necessarily deterministic
An algorithm is an effective method that can be expressed within a finite amount
of space and time
```

The transition from one state to the next is not necessarily deterministic

数据规模

0 < L[i] < 512
0 < W[i] < 32
对于 30% 的数据，0 < N < 30，0 < M < 30。
对于 100% 的数据，0 < N < 500，0 < M < 800。

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;

int main() {

    int n, m;

    char cc;

    cin >> n >> m;

    vector<unordered_set<string>> setence(0);

    vector<string> vec(0);



    string str;

    getline(cin, str);

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

        getline(cin, str);

        vec.push_back(str);

        unordered_set<string> tmp;

        tmp.clear();

        string word = "";

        for(int i = 0; i < str.size(); i++) {

            if(str[i] == ' ') {

                if(word.size() > 0) {

                    transform(word.begin(), word.end(), word.begin(), ::tolower);

                    tmp.insert(word);
```

```
                }
                word = "";
            } else if(str[i] != ' ') {
                word += str[i];
            }
        }
        if(word.size() > 0) {
            transform(word.begin(), word.end(), word.begin(), ::tolower);
            tmp.insert(word);
        }
        setence.push_back(tmp);
    }
    for(int k = 0; k < m; k++) {
        getline(cin, str);
        unordered_set<string> tmp;
        tmp.clear();

        string word = "";
        for(int i = 0; i < str.size(); i++) {
            if(str[i] == ' ') {
                if(word.size() > 0) {
                    transform(word.begin(), word.end(), word.begin(), ::tolower);
                    tmp.insert(word);
                }
                word = "";
            } else if(str[i] != ' ') {
                word += str[i];
```

```
            }
        }
        if(word.size() > 0) {
            transform(word.begin(), word.end(), word.begin(), ::tolower);
            tmp.insert(word);
        }
        word = "";

        int index = 0, mx = 0;
        for(int i = 0; i < n; i++) {
            int count = 0;
            for(auto it = tmp.begin(); it != tmp.end(); it++) {
                if(setence[i].find(*it) != setence[i].end()) {
                    ++count;
                }
            }
            if(count > mx) {
                mx = count;
                index = i;
            }
        }
        cout << vec[index] << endl;
    }
    return 0;
}
```

3、两数组找相同的元素

【题目描述】给两个整数(int)数组，输出相同的元素。

输入格式

m n
a1 a2 … am
b1 b2 … bn

输出格式

相同的的元素，用空白分开

输入样例

5 4
11 15 9 12 3
1 8 3 7

输出样例

3

数据规模

对于 30%的数据， 0 < m < 30, 0 < n < 30
对于 100%的数据，0 < m < 100000, 0 < n < 100000
想考察当 m,n 超过一定规模之后不能直接用暴力求解

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//插入 set,然后对于每个元素在 set 里面查找。。数据量比较大用 scanf

int main() {

    int n, m;

    int x;

    bool first = true;

    set<int> a;

    scanf("%d", &n);

    for (int i = 0; i < n; ++i) scanf("%d", &x), a.insert(x);

    scanf("%d", &m);

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

        scanf("%d", &x);

        if (a.find(x) != a.end()) {

            if (first) first = false; else printf(" ");

            printf("%d", x);
```

```
        }

    }

    return 0;

}
```

4、DAU 统计

【题目描述】日活跃用户数 (DAU) 是衡量一个产品表现的重要指标。
需要编写程序，根据给定的某一天的 N 条访问记录，对当天的用户总数 M 进行统计。每个用户可能访问多次。为了方便，我们用数字 (uid) 唯一标识每个用户。

输入格式

每一行包含一个 uid，遇到 0 时认为输入结束。
输入共包含 N+1 行，可认为是无序的。

输出格式

一个数字：去重后 uid 的数量 M。

输入样例

```
12933
111111
59220
69433
59220
111111
0
```

输出样例

```
4
```

数据范围

0 < uid < 2^63
时间 < 1s, 内存 < 32MB
对于 30% 的数据，0 < N < 100,000, 0 < M < 100,000
对于 100% 的数据，0 < N < 1,000,000, 0 < M < 800,000

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;//统计不重复的数字, 利用 set 就可以了

int main() {

    long long x;

    set<long long> st;

    while (cin >> x) {

        if (x == 0) break;

        st.insert(x);
```

```
    }

    cout << st.size() << endl;;

    return 0;
```

}

5、任务执行策略

【题目描述】我们有一批任务在 mesos 当中。这批任务要么不依赖其它任务，要么一定恰好依赖于两个任务，并且整个依赖关系会构成一个三角模型：

Job(1, 1)

Job(2, 1)　　Job(2, 2)

Job(3, 1)　　Job(3, 2)　　Job(3, 3)

……

Job(n, 1)　　Job(n, 2)　　……　　Job(n, n)

如上图，Job(1, 1) 依赖于 Job(2, 1) 和 Job(2, 2)；Job(2, 2) 依赖于 Job(3, 2) 和 Job(3, 3)；对于任意 1 <= i < n, 1 <= j <= n，Job(i, j) 依赖于 Job(i + 1, j) 和 Job(i + 1, j + 1)。最后一行的任务没有任务依赖。

这批任务有一个特点，每个任务都需要配合它所依赖的任务来执行。也就是说，一个任务某次运行是有效的，当且仅当至少满足下列一个条件：

1.该任务不依赖其它任务；

2.该任务依赖的两个任务都是有效的。

每个任务都预先设定了一个权重 weight(i, j)。现在由于资源上的限制，我们只能挑选其中的 k 个任务来运行。我们希望所有被运行的任务都是有效的，并使得所有运行过的任务的权重之和最大。

输入格式

第一行是两个整数 n 和 k。

接下来 n 行，其中第 i 行 (1 <= i <= n) 包含 i 个整数，给出各个任务的权重。这个三角形也同时描述了任务的依赖关系。

输出格式

输出仅包含一个整数，即所求的最大权重和。

输入样例

3 4

1

2 3

1 1 1

输出样例

6

数据规模

对于 30% 的数据，1 <= n, k <= 50；

对于 100% 的数据，1 <= n <= 100，1 <= m <= C(n + 1, 2)，1 <= weight(i, j) <= 1000。

【答案及解析】

```
#include <bits/stdc++.h>

const int N = 60;

const int M = 500 + 10;//动态规划。把三角形翻转一下，从底部到顶考虑每个元素。

//dp[i][j][k]表示考虑到第(i, j)个，当前选取了 k 个元素的最大值。

//用前缀和维护一下最大值。

int dp[N][N][M], sum[N][N], a[N][N], n, m;

int main() {

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

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

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

            scanf("%d", &a[i][j]);

        }

    }

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

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

            sum[i][j] = sum[i][j - 1] + a[n - j + 1][i - j + 1];

        }

    }

    memset(dp, 200, sizeof(dp));

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

        dp[i][0][0] = 0;

    }

    for(int i = 1; i <= n; ++ i) {
```

```
        for(int j = i; j >= 0; -- j) {
            for(int k = j; k <= m; ++ k) {
                dp[i][j][k] = std::max(dp[i][j + 1][k],
                                       dp[i - 1][std::max(0, j - 1)][k - j] + sum[i][j]);
            }
        }
    }
    printf("%d\n", dp[n][0][m]);
    return 0;
}
```

八、滴滴出行 2017 秋招笔试真题

1、末尾 0 的个数

【题目描述】输入一个正整数 n,求 n!(即阶乘)末尾有多少个 0？ 比如: n = 10; n! = 3628800,所以答案为 2

输入描述:

输入为一行，n(1 ≤ n ≤ 1000)

输出描述:

输出一个整数,即题目所求

输入例子:

10

输出例子:

2

【答案及解析】

分析：考虑 0 是 5 和 2 贡献出来的，但是因子 2 出现的频率高得多，所以统计因子 5 的个数就好了。给了多种写法。

参考代码//第一种

```
#include <iostream>
#include <cstdio>
```

```
using namespace std;

int solve(int n){

    int cnt = 0;

    while(n){

        cnt += n / 5;

        n = n / 5;

    }

    return cnt;

}

int main(){

    int n;

    cin >> n;

    cout << solve(n) << endl;

}

#include <iostream>//第二种

#include <cstdio>

using namespace std;

int solve(int n){

    int cnt = 0;

    for(int i = 5; i <= n; i += 5){

        int res = i;

        while (res % 5 == 0){

            cnt++;

            res /= 5;

        }

    }

    return cnt;
```

```
}

int main(){

    int n;

    cin >> n;

    cout << solve(n) << endl;

}#数据范围比较小,直接用 python 也是可以的

n = int(raw_input())

ans = 1;

for i in range(1,n+1):

    ans *= i

count = 0

for i in range(len(str(ans))-1,-1,-1):

    if(str(ans)[i] == '0'):

        count += 1

    else:

        break

print count
```

2、地下迷宫

【题目描述】小青蛙有一天不小心落入了一个地下迷宫,小青蛙希望用自己仅剩的体力值 P 跳出这个地下迷宫。为了让问题简单,假设这是一个 n*m 的格子迷宫,迷宫每个位置为 0 或者 1,0 代表这个位置有障碍物,小青蛙达到不了这个位置;1 代表小青蛙可以达到的位置。小青蛙初始在(0,0)位置,地下迷宫的出口在(0,m-1)(保证这两个位置都是 1,并且保证一定有起点到终点可达的路径),小青蛙在迷宫中水平移动一个单位距离需要消耗 1 点体力值,向上爬一个单位距离需要消耗 3 个单位的体力值,向下移动不消耗体力值,当小青蛙的体力值等于 0 的时候还没有到达出口,小青蛙将无法逃离迷宫。现在需要你帮助小青蛙计算出能否用仅剩的体力值跳出迷宫(即达到(0,m-1)位置)。

输入描述:

输入包括 n+1 行:

第一行为三个整数 n,m(3 <= m,n <= 10),P(1 <= P <= 100)

接下来的 n 行:

每行 m 个 0 或者 1,以空格分隔

输出描述:

如果能逃离迷宫,则输出一行体力消耗最小的路径,输出格式见样例所示;如果不能逃离迷宫,则输出"Can not escape!"。测试数据保证答案唯一

输入例子:

4 4 10

1 0 0 1

1 1 0 1

0 1 1 1

0 0 1 1

输出例子:

[0,0],[1,0],[1,1],[2,1],[2,2],[2,3],[1,3],[0,3]

【答案及解析】

分析：根据题设，可以简单证明最短路径一定体力消耗最小的路径。因为横向移动 m-1 是必须的，而纵向的路径越短，需要的体力就越少。直接 bfs 然后记录一下路径即可,dfs 应该也是可以的吧。

参考代码

```
#include <iostream>

#include <cstdio>

#include <queue>

using namespace std;

int _map[15][15];

int vis[15][15];

int m,n,P;

struct node{

    int x,y,v;

}ans[15][15];

bool bfs(){

    queue<node> q;

    node tmp;

    tmp.x = 0,tmp.y = 0,tmp.v = P;

    q.push(tmp);
```

```
    while(!q.empty()){
        node tmp = q.front();
        q.pop();
        if(tmp.x == 0 && tmp.y == m - 1 && tmp.v >= 0) return true;
        if(tmp.x + 1 <= n - 1 && _map[tmp.x + 1][tmp.y] && !vis[tmp.x + 1][tmp.y]){
            node temp;
            temp.x = tmp.x + 1;
            temp.y = tmp.y;
            temp.v = tmp.v;
            ans[temp.x][temp.y].x = tmp.x;
            ans[temp.x][temp.y].y = tmp.y;
            vis[temp.x][temp.y] = 1;
            q.push(temp);
        }
        if(tmp.y + 1 <= m - 1 && _map[tmp.x][tmp.y + 1] && !vis[tmp.x][tmp.y + 1]  && tmp.v >
0){
            node temp;
            temp.x = tmp.x;
            temp.y = tmp.y + 1;
            temp.v = tmp.v - 1;
            ans[temp.x][temp.y].x = tmp.x;
            ans[temp.x][temp.y].y = tmp.y;
            vis[temp.x][temp.y] = 1;
            q.push(temp);
        }
        if(tmp.x - 1 >= 0 && _map[tmp.x - 1][tmp.y] && !vis[tmp.x - 1][tmp.y] && tmp.v >= 3){
            node temp;
```

```
            temp.x = tmp.x - 1;
```

3、连续最大和

【题目描述】一个数组有 N 个元素，求连续子数组的最大和。 例如：[-1,2,1]，和最大的连续子数组为[2,1]，其和为 3

输入描述:

输入为两行。

第一行一个整数 n(1 <= n <= 100000)，表示一共有 n 个元素

第二行为 n 个数，即每个元素,每个整数都在 32 位 int 范围内。以空格分隔。

输出描述:

所有连续子数组中和最大的值。

输入例子:

3

-1 2 1

输出例子:

3

【答案及解析】

分析：经典水题。没啥好说的. 时间复杂度：O(n)

参考 code:

```
#include <iostream>

#include <algorithm>

using namespace std;

int main(){

    int n;

    scanf("%d",&n);

    int sum = 0, mx = -99999999;

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

        int temp;

        scanf("%d",&temp);

        if(sum < 0) sum = temp;
```

```
        else sum += temp;

        mx = max(sum, mx);

    }

    printf("%d\n",mx);

}
```

4、餐馆

【题目描述】某餐馆有 n 张桌子，每张桌子有一个参数：a 可容纳的最大人数； 有 m 批客人，每批客人有两个参数:b 人数，c 预计消费金额。 在不允许拼桌的情况下，请实现一个算法选择其中一部分客人，使得总预计消费金额最大

输入描述:

输入包括 m+2 行。第一行两个整数 n(1 <= n <= 50000),m(1 <= m <= 50000)

第二行为 n 个参数 a,即每个桌子可容纳的最大人数,以空格分隔,范围均在 32 位 int 范围内。

接下来 m 行，每行两个参数 b,c。分别表示第 i 批客人的人数和预计消费金额,以空格分隔,范围均在 32 位 int 范围内。

输出描述:

输出一个整数,表示最大的总预计消费金额

输入例子:

3 5

2 4 2

1 3

3 5

3 7

5 9

1 10

输出例子:

20

【答案及解析】

分析：贪心。先把顾客进行消费金额降序,人数升序排序。 然后枚举每波顾客去二分当前最适合的 桌子的容量，维护答案即可,注意答案可能爆 int。 这题裸暴力过不了。 不能拆桌。时间复杂度：O(mlogm + nlogm)

参考 code:

```
#include <iostream>
```

```
#include <map>

#include <vector>

#include <map>

using namespace std;

struct node{

    int b,c;

};

int cmp(node x,node y){

    if (x.c == y.c) {

        return x.b < y.b;

    }

    return x.c > y.c;

}

int n,m;

long long ans;

std::vector<node> v;

std::multimap<int, int> mp;

int main(){

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

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

        int x; scanf("%d",&x);

        mp.insert(std::pair<int, int>(x, 1));

    }

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

        int x, y;

        scanf("%d%d",&x,&y);

        node tmp;
```

```
        tmp.b = x, tmp.c = y;
        v.push_back(tmp);
    }
    sort(v.begin(),v.end(),cmp);
    for(int i = 0; i < m; i++){
        std::map<int,int>::iterator it = mp.lower_bound(v[i].b);
        if (it != mp.end()) {
            mp.erase(it);
            ans += v[i].c;
        }
    }
    printf("%lld\n",ans);
}
```

九、迅雷 2017 秋招笔试真题

1、水仙花数

【题目描述】春天是鲜花的季节，水仙花就是其中最迷人的代表，数学上有个水仙花数，他是这样定义的：“水仙花数”是指一个三位数，它的各位数字的立方和等于其本身，比如：153=1^3+5^3+3^3。现在要求输出所有在 m 和 n 范围内的水仙花数。

输入

输入数据有多组，每组占一行，包括两个整数 m 和 n（100<=m<=n<=999）

输出

对于每个测试实例，要求输出所有在给定范围内的水仙花数，就是说，输出的水仙花数必须大于等于 m,并且小于等于 n，如果有多个，则要求从小到大排列在一行内输出，之间用一个空格隔开 ； 如果给定的范围内不存在水仙花数，则输出 no ； 每个测试实例的输出占一行。

输入样例

100 120

300 380

输出样例

no

370 371

【答案及解析】

分析：本题比较简单，暴力循环判断即可。

```
module.exports = Promise = function(callback){
    this.promise = [];
    this.fail = [];
    var that = this;
    var resolver = function(){
        for(var i=0,len=that.promise.length;i<len;i++){
            var promise = that.promise.shift();
            if(typeof(promise.resolver) == 'function'){
                var ret = promise.resolver.apply(null,Array.prototype.slice(arguments,0));
                if(ret instanceof Promise){
                    ret.promise = that.promise;
                    return ret;
                }
            }
        }
    };

    var rejector = function(){
        for(var i=0,len=that.promise.length;i<len;i++){
            var promise = that.promise.shift();
            if(typeof(promise.rejector) == 'function'){
                var ret = promise.rejector.apply(null,Array.prototype.slice(arguments,0));
                if(ret instanceof Promise){
                    ret.promise = that.promise;
```

```
                    return ret;
                }
            }
        }
    };

    callback.call(null,resolver,rejector);
    return this;
}

Promise.prototype.then = function(resolver,rejector){
    this.promise.push({
        resolver:resolver,
        rejector:rejector
    })
    return this;
}
```

2、点是否在三角形内

【题目描述】在一个二维坐标系中，已知三角形三个点的坐标 ABC(逆时针顺序)，对于坐标系中任意一点 D，判断改点是否在三角形内（点在三角形边上也认为在三角形内）。

输入

输入一组坐标数据（float 类型），分表代表 A、B、C、D 四个点的坐标

输出

如果 D 点位于三角形 ABC 内，则输出“Yes”；

如果 D 点不位于三角形 ABC 内，则输出“No”；

输入样例

0.0 0.0 2.0 0.0 2.0 2.0 1.0 0.5

输出样例

YES

【答案及解析】

分析：本题需要简单的几何知识，重点考察问题分析能力、算法实现能力和逻辑思维能力。

```
#include <iostream>

#include <math.h>

using namespace std;

#define FLOAT_POINT_PRECISION 0.0001


class Point

{

public:

    Point(): x(0), y(0) {}


public:

    float x;

    float y;

};


float get_area(const Point pt0, const Point pt1, const Point pt2)

{

    Point AB,BC;

    AB.x = pt1.x - pt0.x;

    AB.y = pt1.y - pt0.y;

    BC.x = pt2.x - pt1.x;

    BC.y = pt2.y - pt1.y;

    return fabs((AB.x * BC.y - AB.y * BC.x)) / 2.0f;

}
```

```
bool is_in_triangle(const Point A, const Point B, const Point C, const Point D)

{

    float SABC, SADB, SBDC, SADC;

    SABC = get_area(A, B, C);

    SADB = get_area(A, D, B);

    SBDC = get_area(B, D, C);

    SADC = get_area(A, D, C);



    //cout << SABC << " " << SADB << " " << SBDC << " " << SADC << endl;



    float SumSuqar = SADB + SBDC + SADC;



    if ((FLOAT_POINT_PRECISION < (SABC - SumSuqar)) && ((SABC - SumSuqar) <
FLOAT_POINT_PRECISION))

    {

        return true;

    }

    else

    {

        return false;

    }

}



bool is_triangle(const Point pt0, const Point pt1, const Point pt2)

{

    float a = sqrt((pt0.x - pt1.x)*(pt0.x - pt1.x) + (pt0.y - pt1.y)*(pt0.y - pt1.y));

    float b = sqrt((pt0.x - pt2.x)*(pt0.x - pt2.x) + (pt0.y - pt2.y)*(pt0.y - pt2.y));

    float c = sqrt((pt2.x - pt1.x)*(pt2.x - pt1.x) + (pt2.y - pt1.y)*(pt2.y - pt1.y));
```

```
    //cout << a << " " << b << " " << c << endl;

    return ((a + b) > c && (a + c) > b && (b + c) > a);
}

int main(void)
{
    Point A, B, C, P;
    cin >> A.x >> A.y >> B.x >> B.y >> C.x >> C.y >> P.x >> P.y;

    if (!is_triangle(A, B, C))
    {
        cout << "No" << endl;
        return 0;
    }

    if (is_in_triangle(A, B, C, P))
    {
        cout << "Yes" << endl;
    }
    else
    {
        cout << "No" << endl;
    }

    return 0;
```

```
}
```

3、二进制数中 1 的个数

【题目描述】对于一个四字节的整数，求其二进制数中 1 的个数。

输入

输入一个整数 n

输出

该整数对应的二进制数中 1 的个数

输入样例

7

输出样例

3

【答案及解析】

分析：本题考察对存储空间和执行效率的理解、位运算的熟悉度，以及简单的算法实现能力。

```
#include <iostream>

using namespace std;

int count_one(int v)

{

    int num = 0;

    while (v != 0)

    {

        v &= (v - 1);

        num++;

    }


    return num;

}


int main()
```

```
{
    int num = -1;
    cin >> num;
    cout << count_one(num) << endl;
    return 0;
}
```

4、帕斯卡三角问题

【题目描述】通过键盘输入一个高精度的正整数 n(n 的有效位数≤240)，去掉其中任意 s 个数字后，剩下的数字按原左右次序将组成一个新的正整数。编程对给定的 n 和 s，寻找一种方案，使得剩下的数字组成的新数最小

输入

输入 2 个参数，第一个参数为高精度正整数 n， 第二个参数为移除的个数 s。

输出

新数

输入样例

138954， 4

输出样例

13

【答案及解析】

分析：该题需要判断参数是否合法。结果如果高位是 0，则要去掉 0，符合整数书写习惯。

```
#include "stdlib.h"
#include "iostream"
using namespace std;
int digit_to_int(char d)//将 char 数组中的元素转换为整数以便于进行比较
{
    char str[2];

    str[0] = d;
```

```
    str[1] = '/0';
    return (int)strtol(str, NULL, 10);
}

void DeleteOneBit(char * a, int _size){//每次只删除一位的函数
    int i = 0;
    bool _is = false;//用于判断是否需要将部分右边的元素左移一位
    while (i < _size - 1){
        if (digit_to_int(a[i])>digit_to_int(a[i+1])){
            _is = true;
            break;}
        i++;
    }
    if (_is){
        for (; i < _size - 1;i++)
        {
            a[i] = a[i + 1];
        }
    }
    a[_size - 1] = '\0';//写终止符
}

void DeleteZero(char* a, int _size)
{
    int j = 0;
    while (j <  _size){
    if (digit_to_int(a[0])!=0){
```

```
        break;
     }
            for (int i=0; i < _size - 1;i++)
            {
                a[i] = a[i + 1];
            }
           j++;
    }
 a[_size - j] = '\0';//写终止符
}

void GetNewNum(char* b, int _num_d){
    int _length = strlen(b);

    if (_num_d>=_length){
        cout << "删除的位数过多" << endl;
        return;
    }

    for (int i = 0; i < _num_d;i++){
        DeleteOneBit(b, _length);
        _length--;
    }
        DeleteZero(b, _length);//去前面的掉 0
}

int main(int argc, char* argv[])
```

```
{
    cout << "请输入要进行删数的正整数（小于 240 位）"<<endl;
    char a[241];
    cin >> a;
    int _num_d;
    cout << "请输入要删除的位数（小于输入的整数的位数）" << endl;
    cin >> _num_d;
    GetNewNum(a, _num_d);

    if (a[0]=='\0')
    {
       cout<<0;
    }
    else
    {
       cout << a;
    }
    system("pause");
    return 0;
}
```

5、纸牌游戏-斗牛

【题目描述】纸牌游戏斗牛，每人发 5 张牌，从中选三张加起来是十的倍数，即为有牛(J,Q,K 计为十)，然后比较剩下两张加起来个位数的大小，称为牛数，若相加为 10 的整数倍，则为牛 10。

比如：牌面为 A，5,6，9，K。A+9+K=20（即为有牛），5+6=11（个位数为 1，即牛数为 1）。

牌面为 2,3,5,7,3。2+3+5=10（牛），7+3=10（牛 10）。

牌面为 2,4,3,6,7,A。任何 3 个数相加不为 10 的整数倍，则为无牛。

输入

输入 5 个数字，A，J，Q，K 用字母表示。

输出

输出牛数

输入样例

A，5,6，9，K

输出样例

1

【答案及解析】

分析：暴力循环判断即可。关键需要考虑异常情况，如相同是数最多只能 4 个，输入的数合法性检测

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
int main()
{
    cout<<"Please input FIVE poker cards, just like [A 2 3 .. 10 J Q K] :"<<endl;
    int flag[14],num[5];
    int sum=0;
    string str;
    memset(flag,0,sizeof(flag));

    for (int i=1;i<=5;i++)
    {
        cin>>str;
        if (str=="A") num[i]=1;
        else
        {
            if (str=="J") num[i]=11;
```

```
            else
            {
                if (str=="Q") num[i]=12;
                else
                {
                    if (str=="K") num[i]=13;
                    else
                    {
                        if (str!="1") num[i]=atoi(str.c_str());
                        else
                        {
                            cout<<"the input is illegal"<<endl;
                            return 0;
                        }
                    }
                }
            }
        }
        if (num[i]==0 || ++flag[num[i]]>4)
        {
            cout<<"the input is illegal"<<endl;
            return 0;
        }
        sum+=(num[i]>10?10:num[i]);
    }

    int cow=sum%10;
```

```
    for (int i=1;i<=5;i++)
    {
        int need=(cow+10-(num[i]>10?10:num[i]))%10;
        flag[num[i]]--;
        if (need!=0 && flag[need]>0)
        {
            cout<<(cow==0?10:cow)<<endl;
            return 0;
        }
        if (need==0 && (flag[10]>0 || flag[11]>0 || flag[12]>0 || flag[13]>0))
        {
            cout<<(cow==0?10:cow)<<endl;
            return 0;
        }
        flag[num[i]]++;
    }
    cout<<"none"<<endl;
    return 0;
}
```

十、好未来 2017 秋招笔试真题

1、字符串中找出连续最长的数字串

【题目描述】读入一个字符串 str，输出字符串 str 中的连续最长的数字串

输入描述:

个测试输入包含 1 个测试用例，一个字符串 str，长度不超过 255。

输出描述:

在一行内输出 str 中里连续最长的数字串。

输入例子:

abcd12345ed125ss123456789

输出例子:

123456789

【答案及解析】

```
import re

input_val = raw_input()

split_char = re.split(r'\D*', input_val)

print max(split_char, key=len)
```

2、 n 个数里最小的 k 个

【题目描述】找出 n 个数里最小的 k 个

输入描述:

每个测试输入包含空格分割的 n+1 个整数，最后一个整数为 k 值,n 不超过 100。

输出描述:

输出 n 个整数里最小的 k 个数。升序输出

输入例子:

3 9 6 8 -10 7 -11 19 30 12 23 5

输出例子:

-11 -10 3 6 7

【答案及解析】

```
import java.util.Arrays;

import java.util.Scanner;

public class Main {

        public static void main(String[] args) {

                Scanner in=new Scanner(System.in);

                String[] string=in.nextLine().split(" ");

                int[] array=new int[string.length-1];

                for(int i=0;i<string.length-1;i++){
```

```
                array[i]=Integer.parseInt(string[i]);
            }
            Arrays.sort(array);
            for(int i=0;i<Integer.parseInt(string[string.length-1]);i++){
                System.out.print(array[i]);
                if(i!=Integer.parseInt(string[string.length-1])-1){
                    System.out.print(" ");
                }
            }
        }
}
```

3、 n 个数里出现次数大于等于 n/2 的数

【题目描述】输入 n 个整数，输出出现次数大于等于数组长度一半的数。

输入描述:

每个测试输入包含 n 个空格分割的 n 个整数，n 不超过 100，其中有一个整数出现次数大于等于 n/2。

输出描述:

输出出现次数大于等于 n/2 的数。

输入例子:

3 9 3 2 5 6 7 3 2 3 3 3

输出例子:

3

【答案及解析】

```
import java.util.Arrays;
import java.util.Scanner;
public class solution {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        String str=in.nextLine();
```

```
        String s[]=str.split(" ");
        int a[]=new int[s.length];
        for(int i=0;i<s.length;i++)
        {
            a[i]=Integer.parseInt(s[i]);
        }
        Arrays.sort(a);
        int count[]=new int[a[a.length-1]];
        int m=a.length/2;
        for(int i=0;i<a.length;i++)
            for(int j=0;j<a[a.length-1];j++)
        {
            if(j==a[i])
            {
                count[j]++;
            }
        }
        for(int j=0;j<a[a.length-1];j++)
        {
            if (count[j] >= a.length / 2) {
                System.out.println(j);
            }
        }
    }
}
```

4、删除公共字符

【题目描述】输入两个字符串，从第一字符串中删除第二个字符串中所有的字符。例如，输入”They are students.”和”aeiou”，则删除之后的第一个字符串变成”Thy r stdnts.”

输入描述:

每个测试输入包含 2 个字符串

输出描述:

输出删除后的字符串

输入例子:

They are students.

aeiou

输出例子:

Thy r stdnts.

【答案及解析】

```
input_1 = raw_input()

input_2 = raw_input()

input_2 = set(input_2)

for en_char in input_2:

      input_1 = input_1.replace(en_char,"")

print(input_1)
```

5、求和

【题目描述】输入两个整数 n 和 m，从数列 1，2，3.....n 中随意取几个数,使其和等于 m ,要求将其中所有的可能组合列出来

输入描述:

每个测试输入包含 2 个整数,n 和 m

输出描述:

按每个组合的字典序排列输出,每行输出一种组合

输入例子:

5 5

输出例子:

1 4

2 3

5

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;

void help(int n, int m, vector<int>& v, int beg) {

        if (m == 0) {

                for (int i = 0; i<v.size(); i++) {

                        i == 0 ? cout << v[i] : cout << " " << v[i];

                }

                cout << endl;

        }

        for (int i = beg; i <= n&&i <= m; i++) {

                v.push_back(i);

                help(n, m - i, v, i + 1);

                v.pop_back();

        }

}

int main() {

        int n, m;

        while (cin >> n >> m) {

                vector<int>v;

                help(n, m, v, 1);

        }

}
```

6、倒置字符串

【题目描述】将一句话的单词进行倒置，标点不倒置。比如 I like beijing. 经过函数后变为：beijing. like I

输入描述:

每个测试输入包含 1 个测试用例： I like beijing. 输入用例长度不超过 100

输出描述:

依次输出倒置之后的字符串,以空格分割

输入例子:

I like beijing.

输出例子:

beijing. like I

【答案及解析】

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        StringBuffer buf = new StringBuffer();

        while(sc.hasNext()){

            buf.append(sc.next());

            buf.append(" ");

        }

        String str = buf.toString();

        String[] arr = str.split(" ");

        StringBuffer buffer = new StringBuffer();

        for (int i = arr.length-1; i >= 0; i--) {

            if (arr[i] != null && arr[i] != "") {

                buffer.append(arr[i]+" ");

            }

        }

        System.out.println(buffer.toString().trim());

    }

}
```

十一、携程 2017 秋招笔试真题

1、拼图

【题目描述】拼图，是一个老少皆宜的益智游戏，在打乱的 3*3 的范围中，玩家只能每次将空格(0)和相邻的数字格（上、下、左、右）交换，最终调整出一个完整的拼图。

完整拼图为：

1 2 3

4 5 6

7 8 0

输入

测试数据共 3 行，每行 3 个数字 ，包括数字 0、1、2、3.....8（无重复）

样例输入

0 1 3

4 2 5

7 8 6

输出

还原完整拼图的最少移动次数。如不需要调整，则输出 0；如无解，则输出-1。

例如：

0 1 3

4 2 5

7 8 6

依次移动 1,2,5,6，即可还原为正确拼图，移动次数为 4

样例输出

4

【答案及解析】

```
import java.io.*;

import java.util.*;

import java.text.*;

import java.math.*;

import java.util.regex.*;

import java.util.Scanner;

import java.util.Set;

import java.util.HashSet;
```

```
import java.util.ArrayList;

import java.lang.StringBuilder;//bfs

public class Main{

    public static String destNumbers = "123456780";

    public static Set<Integer> set = new HashSet<Integer>();

    public static int[]moveTable = new int[]{12,14,10,13,15,11,5,7,3};


    public static ArrayList<Node> getNextMoveList(Node pNode){

        int position = pNode.numbers.indexOf("0");

        int moveStatus = moveTable[position];

        ArrayList<Node> cNodes = new ArrayList<Node>();


        for(int status=1; status <=8; status=status<<1){

            if((moveStatus & status) > 0){

                char[] charNumbers = pNode.numbers.toCharArray();

                int switchPosition = 0;

                if(status == 1){

                   switchPosition = position - 3;

                } else if(status == 2){

                   switchPosition = position - 1;

                } else if(status == 4){

                   switchPosition = position + 1;

                } else if(status == 8){

                   switchPosition = position + 3;

                }

                charNumbers[position] = charNumbers[switchPosition];

                charNumbers[switchPosition] = '0';
```

```
                String s = String.valueOf(charNumbers);
                if(!set.contains(Integer.valueOf(s))){
                    set.add(Integer.valueOf(s));
                    Node n = new Node(pNode, s, charNumbers[position]);
                    cNodes.add(n);
                }
            }
        }
        return cNodes;
    }

    static int getResult(Node node){
        String result = "";
        while(node.parentNode != null){
            result += node.currentNum;
            node = node.parentNode;
        }
        return new StringBuffer(result).reverse().toString().length();
    }

    static int run(String numbers){
           if(numbers.equals(destNumbers)){
            return 0;
        }
        ArrayList<Node> numsList = new ArrayList<Node>();
        numsList.add(new Node(null, numbers, ' '));
        while(numsList.size() > 0){
```

```
            ArrayList<Node> tmpList = new ArrayList<Node>();


            for(Node pNode : numsList){
                ArrayList<Node> cNodes = getNextMoveList(pNode);


                for(Node cNode : cNodes){
                    if(cNode.numbers.equals(destNumbers)){
                        return getResult(cNode);
                     }
                    tmpList.add(cNode);
                }
            }
            numsList = tmpList;
        }
        return -1;
    }
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);


        String numbers = new String();
        for(int rows=3; rows>0; rows--){
            for(String n: scan.nextLine().split(" ")){
                numbers += n;
            }
        }
        int res = run(numbers);
        System.out.println(String.valueOf(res));
```

```
    }
}
class Node {
    public Node(Node parentNode, String numbers, char currentNum){
        this.numbers = numbers;
        this.currentNum = currentNum;
        this.parentNode = parentNode;
    }
    public char currentNum;
    public String numbers;
    public Node parentNode;
}
```

2、股票交易

【题目描述】一个一维数组，记录 n 天中每天的携程股价。
股市交易规则如下：
a）一天只能有买进或者卖出一种操作，也可以不做任何操作，卖出时价格减买入时价格即为收益
b）每次卖出操作后有冻结期，k 天之后才能进行下一次买进操作（k>=1）
c）买进之后必须卖出才能再次买进
设计一个算法，找到交易收益最大化的买进卖出策略，返回最后的最大收益值

输入
一维正整数数组，表示每天股价
正整数 k，表示冷冻期

样例输入
```
int[] sotckPrice = {1,2,3,5,2,6,3,7};
int k = 2;
```

输入为
8（表示数组长度）
1（开始依次输入数组元素）
2
3
5
2
6
3
7
2（输入 k）

输出
返回最大收益

样例输出
8
(1 买进，5 卖出，3 买进，7 卖出，收益最大为 8)

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1000000 + 5;//根据题意判断冷却时间, 更新答案即可。

int a[maxn],dp[maxn];

int main() {

    int n, k;

    scanf("%d",&n);

    for(int i = 1;i <= n; i++) scanf("%d",&a[i]);

    scanf("%d",&k);

    int cur = -1000000000, ans = 0;

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

        dp[i] = max(a[i] + cur, dp[i - 1]);

        if(i >= k)

            cur = max(cur, dp[i - k] - a[i]);

        else

            cur = max(cur, -a[i]);

        ans = max(ans, dp[i]);

    }

    printf("%d\n",ans);

}
```

3、数组元素查找

【题目描述】 给定一个整形数组，请写一个算法，找到第一个仅出现一次的数组元素，复杂度 O(n) 。

输入

给定一整形数组，数组元素若干

样例输入

1,1,2,2,3,3,4,4,5,6,6,8,9,9

输出

第一个仅出现一次的元素

样例输出

5

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;

int n;

int a[1005];

unordered_map<int, int> mp;

int main() {

    cin >> n;

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

        scanf("%d", &a[i]);

        mp[a[i]]++;

    }

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

        if(mp[a[i]] == 1) {

            cout << a[i] << endl;

            break;

        }

    }

    return 0;

}
```

4、子数组求最大和

【题目描述】输入一个整形数组。

数组中连续的一个或多个整数组成一个子数组，每个子数组都有一个和。

求所有子数组的和的最大值。要求时间复杂度为 O(n)。

输入

给定一个数组，包含若干个整数。

样例输入

1 -3 5 5 -6 -2 -7

输出

测试数组中子数组所能产生的最大和

样例输出

10

【答案及解析】

```
public class Main {

    public static int findGreatestSumOfSubArray(int[] arr) {

        int max = Integer.MIN_VALUE;

        int curMax = 0;

        for (int i : arr) {

            if (curMax <= 0) {

                curMax = i;

            }

            else {

                curMax += i;

            }

            if (max < curMax) {

                max = curMax;

            }

        }

        return max;

    }

    public static void main(String[] args) {
```

```
        int[] case1 = {1, -2, 3, 10, -4, 7, 2, -5, 666};

        int[] case2 = {66666, -2, -8, -1, -5, -9, 233};

        int[] case3 = {2, 8, 1, 5, 9, -3};

        System.out.println(findGreatestSumOfSubArray(case1));

        System.out.println(findGreatestSumOfSubArray(case2));

        System.out.println(findGreatestSumOfSubArray(case3));

    }

}
```

5、字符串去标点

【题目描述】编一个程序，从 string 对象中去掉标点符号。要求输入到程序的字符串必须含有标点符号，输出结果则是去掉标点符号后的 string 对象

输入

带标点的字符串

样例输入

String:a1,b2.c3-d4!

输出

不带标点的字符串

样例输出

Enter a string:
Result:

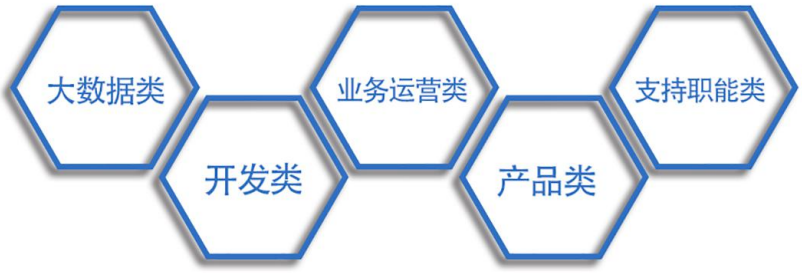
String a1b2c3d4

【答案及解析】

```
#include <bits/stdc++.h>

using namespace std;

#define N 100

char*  change(char putin[],int n) {

    if(putin == NULL)

        return NULL;

    int strcount = 0;

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

```
        if((putin[i]>='a'&&putin[i]<='z')||(putin[i]>='A' &&
putin[i]<='Z')||(putin[i]>='0'&&putin[i]<='9'))
            ++strcount;
    }
    int p1 = n - 1;
    int p2 = strcount - 1;
    char *putout = new char[strcount];
    putout[strcount] = '\0';
    while( p2 >= 0) {
        if((putin[p1]>='a'&&putin[p1]<='z')||(putin[p1]>='A' &&
putin[p1]<='Z')||(putin[p1]>='0'&&putin[p1]<='9')) {
            putout[p2--] = putin[p1--];
        } else {
            --p1;
        }
    }
    return putout;
}
int main() {
    char putin[N];
    char *putout;
    gets(putin);
    int n = strlen(putin);
    putout = change(putin,n);
    puts(putout);
}
```

招聘职位



招聘流程

内推：8月初启动
网申：9月底截止

进校宣讲
直播宣讲

携程简介

携程旅行网创立于1999年，总部设在中国上海，现有员工30000余人遍布海内外。目前携程是中国最大的旅游集团，居在线旅行服务市场领先地位，已成为全球市值前三的在线旅行服务公司。携程旅行网向超过2.5亿注册会员提供包括酒店预订、机票预订、旅游度假、商旅管理及攻略社区在内的全方位旅行服务，线路产品覆盖超过100多个目的地国家和地区。携程旅行网在2012年、2013年进入“胡润品牌榜”，位列100个最有价值的中国品牌。连续三年被《TravelWeeklyChina 旅讯》评为“中国最佳差旅管理公司”，被《财富》评为最受赞赏中国公司。2015年度“中国最佳在线旅游网站”。2016年，携程作为唯一一家在线旅游服务企业入选2016互联网企业百强榜单前十。


GET.OFFER
• Let's NOWCODER.COM •
hello offer
牛客出品
nowcoder.com