下列说法正确的是（ab）

a. display: none；不为被隐藏的对象保留其物理空间；

b. visibility:hidden;所占据的空间位置仍然存在,仅为视觉上的完全透明；

c. visibility:hidden;产生reflow和repaint(回流与重绘)；

d. visibility:hidden;与display: none;两者没有本质上的区别；

下面哪些是HTML5 新增的表单元素？（ac）

a. Datalist

b. Optgroup

c. Output

d. legend

在HTML中，（c ）可以在网页上通过链接直接打开邮件客户端发送邮件。

a.<a href=”telnet:ming.zhou@nowcoder.com”>发送邮件</a>

b.<a href=”mail:ming.zhou@nowcoder.com”>发送邮件</a>

c.<a href=”mailto:ming.zhou@nowcoder.com”>发送邮件</a>

d.<a href=”ftp:ming.zhou@nowcoder.com”>发送邮件</a>

下面哪个属性不会让 div 脱离文档流（normal flow）？（c）

a. position: absolute;

b. position: fixed;

c. position: relative;

d. float: left;

现在有两堆石子,小今与小条玩游戏,2个人都足够聪明,两个人规定:每次每人只能从其中一堆中取走1个或2个或3个石子,最后将石子全部取完的人胜利.现在两堆石子的个数为8和9,请问如何安排才能让小今必胜? （a）

a.让小今先取

b.让小条先取

c.没有策略能够让小今必胜

d.以上说法都不正确

运行以下程序

<script>

var m= 1, j = k = 0;

function add(n) {

return n = n+1;

　 }

y = add(m);

function add(n) {

return n = n + 3;

}

z = add(m);

</script>

y和z的最终结果为（b）:

a.2,4

b.4,4

c.2,2

d.报异常

页面有一个按钮button id为 button1，通过原生的js如何禁用？(IE 考虑IE 8.0以上版本)

下面那个页面调度算法,当进程分配到的页面数增加时,缺页中断的次数可能增加也可能减少

以下描述正确的：（cd）

a. document.getElementById("button1").readolny= true;

b. document.getElementById("button1").setAttribute(“readolny”,”true”);

c. document.getElementById("button1").disabled = true;

d. document.getElementById("button1").setAttribute(“disabled”,”true”);

下面那个页面调度算法,当进程分配到的页面数增加时,缺页中断的次数可能增加也可能减少（a）

a. FIFO算法

b. LRU算法

c. Clock算法

d. LFU算法

以下描述正确的：（bcd）

a. Http协议所使用的运输层协议是UDP

b. Https的端口号是443

c. TCP注重数据可靠性，UDP注重数据传输快

d. 传输层提供端到端的可靠报文传递和错误恢复

假设在今日头条里面,有很多工作人员检查新闻是不是属于虚假新闻,所有新闻真实率到达了98%,工作人员在检验一个真实的新闻把它检验为一个虚假的新闻的概率为2%,而一个虚假的新闻被检验为真实的新闻的概率为5%.那么,一个被检验为真实的新闻确实是真实的新闻的概率是多大? （b）

a.0.9991

b.0.9989

c.0.9855

d.0.96

下面哪条声明能固定背景图片（a）

a. background-attachment:fixed;

b. background-attachment:scroll;

c. background-origin: initial;

d. background-clip: initial;

英文字母全部转为大写正确的是（c）

a. text-transform: capitalize;

b. text-transform: lowercase;

c. text-transform: uppercase;

d. font-weight: bold;

下列关于操作系统进程与线程的区别正确的是：（ad）

a.进程是资源分配的基本单位

b.线程是资源分配的基本单位

c.进程是资源调度的基本单位

d.线程是资源调度的基本单位

请选出所有的置换元素（abcd）

img

input

textarea

select

表toutiao\_tb

title   data       auther  type

abc     2016.2.23   bob     1

bcv     2016.3.3    http    1

cvt     2016.3.3    http    1

bcvvcm  2016.3.5    js      2

nmhh    2016.2.3    html    2

hhj     2016.3.3    java    3

rrr     2016.3.2    cc      1

查询title中包含cv且type是1的记录（b）

a. select \* from toutiao\_tb where title = 'cv' and type='1'

b. select \* from toutiao\_tb where title like '%cv%' and type=1

c. select \* from toutiao\_tb where title like '\*cv' and type=1

d. select \* from toutiao\_tb where title ='\*cv\*' and type='1‘

页面有一个按钮button id为 button1，通过原生的js 设置背景色为红色？（a）

a. document.getElementById('button1').style.backgroundColor="red";

b. document.getElementById('button1').style.backgroundcolor="red";

c. document.getElementById('button1').style.backGroundColor="red";

d. document.getElementById('button1').style.bgcolor="red";

以下关于盒子模型描述正确的是：（a）

a.标准盒子模型中：盒子的总宽度 ＝ 左右margin + 左右border + 左右padding + width

b. IE盒子模型中：盒子总宽度 ＝ 左右margin + 左右border + width

c.标准盒子模型中：盒子的总宽度 ＝ 左右margin + 左右border + width

d. IE盒子模型中：盒子总宽度 ＝ width

(function() {

      var a = b = 5;

  })();

console.log(b);

console.log(a); 上面的输出结果: （d）

a.5，5

b. undefined，undefined

c.5，undefined

d.5，Uncaught ReferenceError: a is not defined

下列排序算法不稳定的有? （bdfg）

a.插入排序

b.希尔排序

c.冒泡排序

d.堆排序

e.归并排序

f.快速排序

g.选择排序

下面标签嵌套正确的是（d）

a.<ul><p>牛客网</p></ul>

b.<a href="#"><a href="#">牛客网</a></a>

c.<dl><li>牛客网</li></dl>

d.<ol><li>牛客网</li></ol>

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。

参考答案:

#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;

}

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。

参考答案:

#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;

}

形式化算式

【题目描述】我们有如下形式化的数字： \*    \*\*\*   \*\*\*   \* \*   \*\*\*   \*\*\*   \*\*\*   \*\*\*   \*\*\*   \*\*\* \*      \*     \*   \* \*   \*     \*       \*   \* \*   \* \*   \* \* \*    \*\*\*   \*\*\*   \*\*\*   \*\*\*   \*\*\*     \*   \*\*\*   \*\*\*   \* \* \*    \*       \*     \*     \*   \* \*     \*   \* \*     \*   \* \* \*    \*\*\*   \*\*\*     \*   \*\*\*   \*\*\*     \*   \*\*\*   \*\*\*   \*\*\* 分别表示 1 2 3 4 5 6 7 8 9 0 有如下形式化的符号： \*        \* \*    \*   \*\*\*\* \*\*\*  \*\*\*   \*    \* \*        \* \*  \*     \*\*\*\*   \*\* \*\* 分别表示 + - \* ／ = 小数点 现在 将输入一个算式 你要将该算式计算之后按照上述形式化的方式输出 各个数字和符号之间空两格 无法整除则保留两位小数 样例输入1： 10 + 31 样例输出1： \*  \*\*\*       \*\*\*  \*        \* \*  \* \*  \* \*   \*     \*  \*  \*\*\*\*  \* \*  \* \*  \* \*  \*\*\*  \*\*\*  \*        \*\*\*  \* \*  \* \*   \*     \*  \*  \*\*\*\*    \*  \* \*  \*\*\*       \*\*\*  \*          \*  \* 样例输入2： 2 ／ 5 样例输出2： \*\*\*       \*\*\*        \*\*\*      \* \* \*    \*  \*    \*\*\*\*  \* \*      \* \* \*\*\*   \*   \*\*\*        \* \*      \*\*\* \*    \*      \*  \*\*\*\*  \* \*  \*\*    \* \*\*\*       \*\*\*        \*\*\*  \*\*    \* 数据范围： 20%的数据 输入的数字和运算结果都是个位数 100%的数据保证 输入数字和答案都小于10000 100%的数据保证 输入数字不会出现小数 80%的数据保证 运算结果不会出现小数

参考答案:

#include <bits/stdc++.h>

using namespace std;

string digit[10][5];

string opr[6][5];

void init() {

    digit[0][0] = ""\*\*\*"";

    digit[0][1] = ""\* \*"";

    digit[0][2] = ""\* \*"";

    digit[0][3] = ""\* \*"";

    digit[0][4] = ""\*\*\*"";

    digit[1][0] = ""\*"";

    digit[1][1] = ""\*"";

    digit[1][2] = ""\*"";

    digit[1][3] = ""\*"";

    digit[1][4] = ""\*"";

    digit[2][0] = ""\*\*\*"";

    digit[2][1] = ""  \*"";

    digit[2][2] = ""\*\*\*"";

    digit[2][3] = ""\*  "";

    digit[2][4] = ""\*\*\*"";

    digit[3][0] = ""\*\*\*"";

    digit[3][1] = ""  \*"";

    digit[3][2] = ""\*\*\*"";

    digit[3][3] = ""  \*"";

digit[3][4] = ""\*\*\*"";

    digit[4][0] = ""\* \*"";

    digit[4][1] = ""\* \*"";

    digit[4][2] = ""\*\*\*"";

    digit[4][3] = ""  \*"";

    digit[4][4] = ""  \*"";

    digit[5][0] = ""\*\*\*"";

    digit[5][1] = ""\*  "";

    digit[5][2] = ""\*\*\*"";

    digit[5][3] = ""  \*"";

    digit[5][4] = ""\*\*\*"";

    digit[6][0] = ""\*\*\*"";

    digit[6][1] = ""\*  "";

    digit[6][2] = ""\*\*\*"";

    digit[6][3] = ""\* \*"";

    digit[6][4] = ""\*\*\*"";

    digit[7][0] = ""\*\*\*"";

    digit[7][1] = ""  \*"";

    digit[7][2] = ""  \*"";

    digit[7][3] = ""  \*"";

    digit[7][4] = ""  \*"";

    digit[8][0] = ""\*\*\*"";

    digit[8][1] = ""\* \*"";

    digit[8][2] = ""\*\*\*"";

    digit[8][3] = ""\* \*"";

    digit[8][4] = ""\*\*\*"";

    digit[9][0] = ""\*\*\*"";

    digit[9][1] = ""\* \*"";

    digit[9][2] = ""\*\*\*"";

    digit[9][3] = ""  \*"";

    digit[9][4] = ""\*\*\*"";

    opr[0][0] = ""   "";

    opr[0][1] = "" \* "";

    opr[0][2] = ""\*\*\*"";

    opr[0][3] = "" \* "";

    opr[0][4] = ""   "";

    opr[1][0] = ""   "";

    opr[1][1] = ""   "";

    opr[1][2] = ""\*\*\*"";

    opr[1][3] = ""   "";

    opr[1][4] = ""   "";

    opr[2][0] = ""   "";

    opr[2][1] = ""\* \*"";

    opr[2][2] = "" \* "";

    opr[2][3] = ""\* \*"";

    opr[2][4] = ""   "";

    opr[3][0] = ""   "";

    opr[3][1] = ""  \*"";

    opr[3][2] = "" \* "";

    opr[3][3] = ""\*  "";

    opr[3][4] = ""   "";

    opr[4][0] = ""    "";

    opr[4][1] = ""\*\*\*\*"";

    opr[4][2] = ""    "";

    opr[4][3] = ""\*\*\*\*"";

    opr[4][4] = ""    "";

    opr[5][0] = ""  "";

    opr[5][1] = ""  "";

    opr[5][2] = ""  "";

    opr[5][3] = ""\*\*"";

    opr[5][4] = ""\*\*"";

}

void print\_num(int x, int p) {

    int dig[10], idx = 0;

    do {

       dig[idx++] = x % 10;

       x /= 10;

    } while (x);

    for (int i = idx-1; i >= 0; --i) {

        cout << digit[ dig[i] ][p];

        if (i) cout << ""  "";

    }

}

void print\_double(double x, int p) {   // 输出数字x的第p行

    int x1 = x;

    int x2 = ceil((x - x1) \* 100);

    int dig[10], idx = 0;

    do {

       dig[idx++] = x1 % 10;

       x1 /= 10;

    } while (x1);

    for (int i = idx-1; i >= 0; --i) {

        cout << digit[ dig[i] ][p];

        cout << ""  "";

    }

    cout << opr[5][p];

    cout << ""  "";

    cout << digit[ x2 / 10 ][p];

    if (x2 % 10 != 0) {

        cout << ""  "";

        cout << digit[ x2 % 10 ][p];

    }

}

int main() {

    int a, b, ans;

    char op;

    init();

    scanf(""%d%\*c%c%\*c%d"", &a, &op, &b);

    if (op == '+') {

        ans = a + b;

        for (int i = 0; i < 5; ++i) {

            print\_num(a, i);

            printf("" "");

            printf("" "");

            cout << opr[0][i];

            printf("" "");

            printf("" "");

            print\_num(b, i);

            printf("" "");

            printf("" "");

            cout << opr[4][i];

            printf("" "");

            printf("" "");

            print\_num(ans, i);

            printf(""\n"");

        }

    } else if (op == '-') {

        ans = a - b; bool mi = false;

        if (ans < 0) mi = true, ans = -ans;

        for (int i = 0; i < 5; ++i) {

            print\_num(a, i);

            printf("" "");

            printf("" "");

            cout << opr[1][i];

            printf("" "");

            printf("" "");

            print\_num(b, i);

            printf("" "");

            printf("" "");

            cout << opr[4][i];

            printf("" "");

            printf("" "");

            if (mi) {

                cout << opr[1][i];

                printf("" "");

                printf("" "");

            }

            print\_num(ans, i);

            printf(""\n"");

        }

    } else if (op == '\*') {

        ans = a \* b;

        for (int i = 0; i < 5; ++i) {

            print\_num(a, i);

            printf("" "");

            printf("" "");

            cout << opr[2][i];

            printf("" "");

            printf("" "");

            print\_num(b, i);

            printf("" "");

            printf("" "");

            cout << opr[4][i];

            printf("" "");

            printf("" "");

            print\_num(ans, i);

            printf(""\n"");

        }

    } else {

        if (a % b == 0) {

            ans = a / b;

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

                print\_num(a, i);

                printf("" "");

                printf("" "");

                cout << opr[3][i];

                printf("" "");

                printf("" "");

                print\_num(b, i);

                printf("" "");

                printf("" "");

                cout << opr[4][i];

                printf("" "");

                printf("" "");

                print\_num(ans, i);

                printf(""\n"");

            }

        } else {

            double fans = (double)a / b;

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

                print\_num(a, i);

                printf("" "");

                printf("" "");

                cout << opr[3][i];

                printf("" "");

                printf("" "");

                print\_num(b, i);

                printf("" "");

                printf("" "");

                cout << opr[4][i];

                printf("" "");

                printf("" "");

                print\_double(fans, i);

                printf(""\n"");

            }

        }

    }

    return 0;

}

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

参考答案:

#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;

}

数列

【题目描述】给定两个长度为 n 的整数数列 A 和 B。再给定 q 组查询，每次查询给出两个整数 x 和 y，求满足 Ai >= x 且 Bi >= y 这样的 i 的数量。 输入格式 第一行给定两个整数 n 和 q。 第二行给定数列 A，包含 n 个整数。 第三行给定数列 B，包含 n 个整数。 接下来 q 行，每行两个整数 x 和 y，意义如上所述。 输出格式 对于每组查询，输出所求的下标数量。 输入样例 3 2 3 2 4 6 5 8 1 1 4 8 输出样例 3 1 数据规模 对于 30% 的数据，1 <= n, q <= 100。 对于 100% 的数据，1 <= n, q, Ai, Bi <= 10^5。

参考答案:

#include <bits/stdc++.h>

using namespace std;//先以A对pair (A, B)排序, 树状数组维护i以后的比y大的个数

const int maxn = 100000 + 10;

struct node {

    int x, y, pos;

    bool operator < (const node &a) const {

        if (x == a.x)

            return y<a.y;

        return x<a.x;

    }

}p[maxn], q[maxn];

int A[maxn];

int B[maxn];

int lowbit(int i) {

    return i & (-i);

}

void add(int a, int i) {

    while (i <= maxn-1) {

        A[i] += a;

        i += lowbit(i);

    }

}

int sum(int i) {

    int res = 0;

    while (i >= 1) {

        res += A[i];

        i -= lowbit(i);

    }

    return res;

}

int main() {

    int n, m;

    cin >> n >> m;

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

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

    sort(p, p+n);

    for(int i = 0; i < m; i++) {

        scanf(""%d%d"", &q[i].x, &q[i].y);

        q[i].pos = i;

    }

    for(int i = 0; i < n; i++)

        add(1, p[i].y);

    sort(q, q+m);

    int k = 0;

    for (int i = 0; i < n; i++) {

        if (p[i].x < q[k].x)

            add(-1, p[i].y);

        else {

            while (1) {

                B[q[k].pos] = sum(maxn - 1) - sum(q[k].y - 1);

                k++;

                if (k >= m)

                    break;                 if (p[i].x < q[k].x) {

                    add(-1, p[i].y);

                    break;

                }

            }

        }

        if (k >= m)

            break;

    }

    for (int i = 0; i < m; i++)

        printf(""%d\n"", B[i]);

    return 0;

}

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

参考答案:

#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;

}

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

参考答案：

#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;

}

绘制括号序列

【题目描述】垂直绘制一个中括号的序列 并用中括号的大小表示层次关系绘制 [[[]]][] 这个括号序列 如图： +-----+ |+---+| |+-+| | | | | |+-+| |+---+| +-----+ +-----+ |     | |     | +-----+ 绘图遵守以下原则： 各个括号之间没有空格 只有在左右括号在最里层配对时 中间才会有一条空行 里层的括号必定小于外层的括号 同一层次的括号大小相同（比如上述的样例 最下面的括号和上面的大括号相同大小） 输入 一个以括号组成的字符串 输出 绘制的图形 保证括号匹配序列合法 样例输入： [][][] 样例输出： +-+ | | | | +-+ +-+ | | | | +-+ +-+ | | | | +-+ 数据范围： 30%的数据 括号层数只有1 100%的数据 括号层数 <= 6 100%的数据 字符串长度 <= 100

参考答案：

#include <bits/stdc++.h>

using namespace std;

int main() {

    string str;

    cin >> str;

    if(str.size() == 0)  return 0;

    vector<int> level(str.size(), 0);

    int l = 0;

    int maxL = 0;

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

        if (str[i] == '[') {

            ++l;

            level[i] = l;

            if (l > maxL)

                maxL = l;

        }

        if (str[i] == ']') {

            level[i] = l;

            --l;

        }

    }

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

        level[i] = maxL - level[i];       --maxL;

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

        if (str[i] == '[') {

            if (level[i] == maxL || level[i] == level[i-1]) {

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << ' ';

                cout << '+';

                for (int k = 0; k < 2 \* level[i] + 1; ++k)

                    cout << '-';

                cout << '+';

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << ' ';

                cout << endl;

            } else {

                for (int j = 0; j < (2 \* maxL - 2 \* level[i] - 2)/2; ++j)

                    cout << ' ';

                cout << ""|+"";

                for (int j = 0; j < (2 \* level[i] + 1); ++j)

                    cout << '-';

                cout << ""+|"";

                for (int j = 0; j < (2 \* maxL - 2 \* level[i] - 2) / 2; ++j)

                    cout << ' ';

                cout << endl;

            }               if (level[i] == level[i + 1]) {

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << "" "";

                cout << ""|"";

                for (int j = 0; j < 2 \* level[i] + 1; ++j)

                    cout << "" "";

                cout << ""|"";

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << "" "";

                cout << endl;

                for (int j = 0; j < 2 \* maxL + 3; ++j)

                    cout << "" "";

                cout << endl;

            }

        } else if (str[i] == ']') {

            if (level[i] == level[i - 1]) {

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << "" "";

                cout << ""|"";

                for (int j = 0; j < 2 \* level[i] + 1; ++j)

                    cout << "" "";

                cout << ""|"";

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << "" "";

                cout << endl;

            }

            if (level[i] == maxL || level[i] == level[i+1]) {

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << ' ';

                cout << '+';

                for (int k = 0; k < 2 \* level[i] + 1; ++k)

                    cout << '-';

                cout << '+';

                for (int j = 0; j < (2 \* maxL - 2 \* level[i]) / 2; ++j)

                    cout << ' ';

                cout << endl;

            } else {

                for (int j = 0; j < (2 \* maxL - 2 \* level[i] - 2) / 2; ++j)

                    cout << ' ';

                cout << ""|+"";

                for (int j = 0; j < (2 \* level[i] + 1); ++j)

                    cout << '-';

                cout << ""+|"";

                cout << endl;

            }

        }

    }

    return 0;

}