算法

二分算法

二分查找算法&LowerBound

code c++

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

int BinarySearch(vector<int> &a, int t){
    int R = a.size()-1;
    int L = 0;
    while (L <= R){
        int M = L + (R - L) / 2;
        if (a[M] == t)return M;
        else if (a[M] > t)R = M - 1;
        else L = M + 1;
    }
    return -1;
}
int LowerBound(vector<int> &a, int t){//比t小的值的最大下标
    int R = a.size() - 1;
    int L = 0;
    int P = -1;
    while (L <= R){
        int M = L + (R - L) / 2;
        if (a[M] >=t)R = M - 1;
        else{ P = M; L = M + 1; }
    }
    return P;
}
void main(){
    int N;
    cin >> N;
    cout << endl;
    int t;
    cin >> t;
    cout << endl;
    vector<int>a(N);
    for (int i = 0; i < N; ++i){
        cin >> a[i];
    }
    //cout<<BinarySearch(a, t);
    cout << LowerBound(a, t);
    system("pause");
}
```

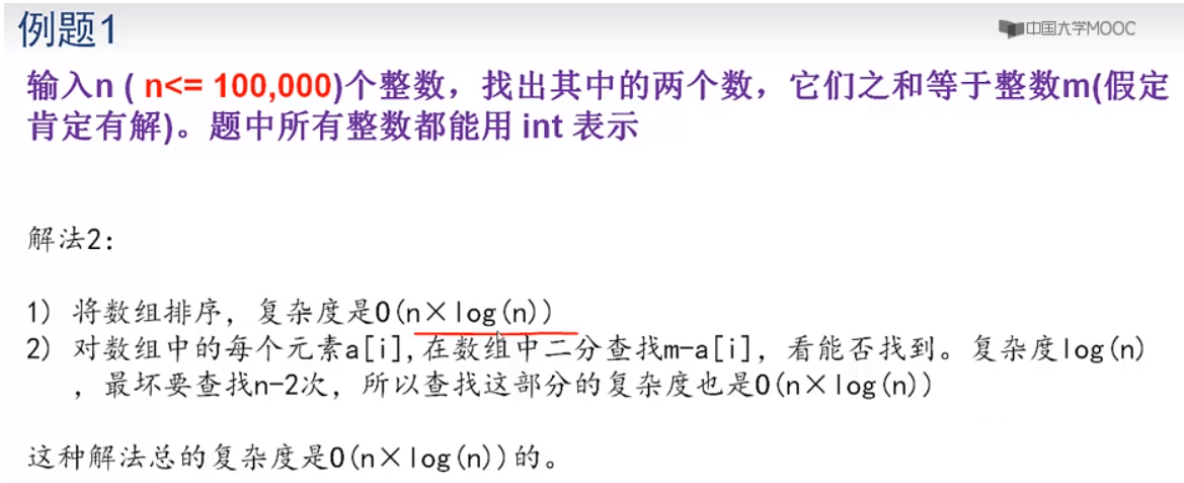
求方程的根

code c++

```
#include<iostream>
#include<cmath>
using namespace std;
double c = 1e-6;
double f(double t){ return (t*t*t - 5*t*t + 10 * t-80); }
void main(){
    double l = 0, r = 100;
    double m = l + (r - l) / 2;
    int t = 1;
    double y = f(m);
    while (y > c||y<-c){
        if (y > 0)r = m;
        else l = m;
        m = l + (r - l) / 2;
        ++t;
        y = f(m);
    }
    cout << t << "ci" << m;
    system("pause");
}
```

找一对数

题解



例题1

中国大学MOOC

输入n (n<= 100,000)个整数，找出其中的两个数，它们之和等于整数m(假定肯定有解)。题中所有整数都能用 int 表示

解法3:

1) 将数组排序，复杂度是O(n×log(n))
2) 查找的时候，设置两个变量i和j，i初值是0，j初值是n-1. 看a[i]+a[j]，如果大于m，就让j减1，如果小于m，就让i加1，直至a[i]+a[j]=m。

这种解法总的复杂度是O(n×log(n))的。

code c++

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
vector<int> yiduishu2(vector<int>&a, int n){
    vector<int>res(2, 0);
    sort(a.begin(), a.end());
    int len = a.size();
    int l = 0;
    int r = len - 1;
    while (l < r){
        if (a[l] + a[r] == n){ res[0] = a[l]; res[1] = a[r]; return res; }
        else if (a[l] + a[r] < n)++l;
        else --r;
    }
    return res;
}
vector<int> yiduishu1(vector<int>&a, int n){
    vector<int>res(2,0);
    sort(a.begin(),a.end());
    int len = a.size();
    for (int i = 0; i < len; ++i){
        int t = n - a[i];
        int l = 0;
        int r = len - 1;
        while (l <= r){
            int m = l + (r - l) / 2;
            if (a[m] == t){ res[0]=a[i]; res[1]=t; return res; }
            else if (a[m] < t)l = m + 1;
            else r = m - 1;
        }
    }
    return res;
}
void main(){
    vector<int> a = { 5, 6, 9, 8, 98,3,50, 100, 477, 5 };
    int N;
    cin >> N;
    vector<int>res;
    res=yiduishu2(a, N);
    cout << res[0] << " " << res[1];
```

```
    system("pause");

}
```

农夫和奶牛

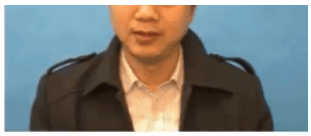
题解

例题2

中国大学MOOC

农夫 John 建造了一座很长的畜栏，它包括N (2≤N≤100,000)个隔间，这些小隔间的位置为$x_0,...,x_{N-1}$ ($0 \le x_i \le 1,000,000,000$,均为整数,各不相同).

John的C (2≤C≤N)头牛每头分到一个隔间。牛都希望互相离得远点省得互相打扰。怎样才能使任意两头牛之间的最小距离尽可能的大，这个最大的最小距离是多少呢？



●解法1：

先得到排序后的隔间坐标 $x_0,...,x_{N-1}$

从1,000,000,000/C到1依次尝试这个“最大的最近距离”D，找到的第一个可行的就是答案。

尝试方法：

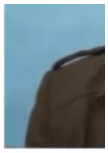
1) 第1头牛放在x_0
2) 若第k头牛放在x_i，则找到x_{i+1}到x_{N-1}中第一个位于[x_i+D, 1,000,000,000]中的X_j
   第k+1头牛放在X_j。找不到这样的X_j，则 D=D-1，转 1)再试

若所有牛都能放下，则D即答案

复杂度 1,000,000,000/C *N，即 1,000,000,000，超时！

24

●解法2：



先得到排序后的隔间坐标 $x_0,...,x_{N-1}$

在[L,R]内用二分法尝试“最大最近距离”D = (L+R)/2 (L,R初值为[1, 1,000,000,000/C]

若D可行，则记住该D，然后在新[L,R]中继续尝试(L= D+1)
若D不可行，则在新[L,R]中继续尝试(R= D-1)

复杂度 log(1,000,000,000/C) * N

code c++

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
```

```
int nainiu(vector<int>&a, int c){
    sort(a.begin(), a.end());
    int len = a.size();
    int l = 1;
    int r = 1000000000 / c;
    int m = l + (r - l) / 2;
    int max = 0;
    while (l <= r){
        int t = c;
        for (int i = 0; i < len;){
            --t;
            if (t == 0){
                max = m;
                l = m + 1;
                m = l + (r - l) / 2;
                break;
            }
            int p = a[i]+m;
            int j;
            for (j = i + 1; j < len; ++j){
                if (a[j]>=p){
                    i = j;
                    break;
                }
            }
            if (j == len){
                r = m - 1;
                m = l + (r - l) / 2;
                break;
            }
        }
        cout << l << m << r;
    }
    return max;
}
void main(){
    vector<int> a = { 1, 200, 500, 800, 440, 1586, 9989, 1985 };
    int n;
    cin >> n;
    cout << " "<<nainiu(a, n);
    system("pause");
}
```