
CS150
初赛专题
集训

CS150 初赛专题集训公布资料的固定网站
请每次课前自行将资料下载到电脑

https://pan.baidu.com/s/1XLQ1mwbfwfDTqFpzmn5HjA

快快编程地址
http://120.132.18.213:9062
请登陆网站提交作业

集训做题步骤

1.在演草纸上填写答案

2.待所有空填写完毕后，打开程序，将草稿纸上的答案填入程序的空白处

3.运行程序，记录所有出现的错误

4.调试程序，尝试改正错误的答案

5.完成整个程序，可以顺利得到期望的结果

6.记录错误和心得经验

二分法

2015 中位数

```
#include<iostream>
using namespace std;
const int MAXN=1000;
int n,i,lbound,rbound,mid,m,count;
int x[MAXN];
int main() {
    cin>>n>>m;
    for(i=0;i<n;i++)
        cin>>x[i];
    lbound=0;rbound=m;
    while(____(1)____) {
        mid=(lbound+rbound)/2;
        ____(2)____;
        for(i=0;i<n;i++)
            if(____(3)____)
                ____(4)____;
        if(count>n/2)
            lbound=mid+1;
        else
            ____(5)____;
    }
    cout<<rbound<<endl;
    return 0;
}
```

给定n（n为奇数且小于1000）个整数，整数的范围在0~m（0<m<2^32）之间，请使用二分法求这n个整数的中位数。所谓中位数，是指将这n个数排序后，排在正中间的数。

边界和中点

```
#include<iostream>
using namespace std;
const int MAXN=1000;
int n,i,lbound,rbound,mid,m,count;
int x[MAXN];
int main() {
    cin>>n>>m;
    for(i=0;i<n;i++)
        cin>>x[i];
    lbound=0;rbound=m;
    while(____(1)____) {
        mid=(lbound+rbound)/2;
        ____(2)____;
        for(i=0;i<n;i++)
            if(____(3)____)
                ____(4)____;
        if(count>n/2)
            lbound=mid+1;
        else
            ____(5)____;
    }
    cout<<rbound<<endl;
    return 0;
}
```

lbound表示下界
rbound表示上界
mid表示中位数

```cpp
#include<iostream>
using namespace std;
const int MAXN=1000;
int n,i,lbound,rbound,mid,m,count;
int x[MAXN];
int main() {
    cin>>n>>m;
    for(i=0;i<n;i++)
        cin>>x[i];
    lbound=0;rbound=m;
    while(lbound<rbound) {  //(1)
        mid=(lbound+rbound)/2;
        count=0;  //(2)
        for(i=0;i<n;i++)
            if(x[i]>mid)  //(3)
                count++;  //(4)
        if(count>n/2)
            lbound=mid+1;
        else
            rbound=mid;  //(5)
    }
    cout<<rbound<<endl;
    return 0;
}
```

2013二叉查找树

二叉查找树具有如下性质：每个节点的值都大于其左子树上所有节点的值、小于其右子树上所有节点的值。试判断一棵树是否为二叉查找树。

输入的第一行包含一个整数n，表示这棵树有n个顶点，编号分别为1, 2, …, n，其中编号为1的为根结点。之后的第i行有三个数value, left_child, right_child，分别表示该节点关键字的值、左子节点的编号、右子节点的编号；如果不存在左子节点或右子节点，则用0代替。输出1表示这棵树是二叉查找树，输出0则表示不是。

```
#include<iostream>
using namespace std;
const int SIZE=100;
const int INFINITE=1000000;
struct node {
    int left_child, right_child, value;
};
node a[SIZE];
int is_bst(int root, int lower_bound, int upper_bound) {
    int cur;
    if(root==0) return 1;
    cur=a[root].value;
    if((cur>lower_bound) && (____(1)____) &&
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
        (is_bst(____(2)____, ____(3)____, ____(4)____)==1))
        return 1;
    return 0;
}
int main() {
    int i,n;
    cin>>n;
    for(i=1;i<=n;i++)
        cin>>a[i].value>>a[i].left_child>>a[i].right_child;
    cout<<is_bst(___(5)____, -INFINITE, INFINITE)<<endl;
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
const int SIZE=100;
const int INFINITE=1000000;
struct node {
    int left_child, right_child, value;
};
node a[SIZE];
int is_bst(int root, int lower_bound, int upper_bound) {
    int cur;
    if(root==0) return 1;
    cur=a[root].value;
    if((cur>lower_bound) && (cur<upper_bound) &&  //(1)
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
        (is_bst(a[root].right_child, cur, upper_bound)==1)) //(2)(3)(4)
        return 1;
    return 0;
}
int main() {
    int i,n;
    cin>>n;
    for(i=1;i<=n;i++)
        cin>>a[i].value>>a[i].left_child>>a[i].right_child;
    cout<<is_bst(1, -INFINITE, INFINITE)<<endl;  //(5)
    return 0;
}
```

2016郊游活动

有n 名同学参加学校组织的郊游活动，已知学校给这n 名同学的郊游总经费为A 元，与此同时第i 位同学自己携带了Mi 元。为了方便郊游，活动地点提供B(≥n)辆自行车供人租用，租用第j 辆自行车的价格为Cj元，每位同学可以使用自己携带的钱或者学校的郊游经费，为了方便账务管理，每位同学只能为自己租用自行车，且不会借钱给他人，他们想知道最多有多少位同学能够租用到自行车。

本题采用二分法。对于区间[l, r]，我们取中间点mid 并判断租用到自行车的人数能否达到mid。判断的过程是利用贪心算法实现的。

```
#include<iostream>
using namespace std;
#define MAXN 1000000
int n,B,A,M[MAXN],C[MAXN],l,r,ans,mid;
bool check(int nn) {
    int count=0, i,j;
    i = ____(1)____;
    j = 1;
    while(i<=n) {
        if(____(2)____)
            count+=C[j]-M[i];
        i++;
        j++;
    }
    return ____(3)____;
}
void sort(int a[], int l, int r) {
    int i = l, j = r, x = a[(l + r) / 2], y;
    while (i <= j) {
        while (a[i] < x) i++;
        while (a[j] > x) j--;
        if (i <= j) {
            y = a[i]; a[i] = a[j]; a[j] = y;
            i++; j--;
        }
    }
    if (i < r) sort(a, i, r);
    if (l < j) sort(a, l, j);
}
```

```
int main() {
    int i;
    cin >> n >> B >> A;
    for (i = 1; i <= n; i++)
        cin >> M[i];
    for (i = 1; i <= B; i++)
        cin >> C[i];
    sort(M, 1, n);
    sort(C, 1, B);
    l = 0;
    r = n;
    while (l <= r) {
        mid = (l + r) / 2;
        if ( ____(4)____ ) {
            ans = mid;
            l = mid + 1;
        } else
            r = ____(5)____;
    }
    cout << ans << endl;
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
#define MAXN 1000000
int n,B,A,M[MAXN],C[MAXN],l,r,ans,mid;
bool check(int nn) {
    int count=0, i,j;
    i = n-nn+1;  //1
    j = 1;
    while(i<=n) {
        if(M[i]<C[j])  //2
            count+=C[j]-M[i];
        i++;
        j++;
    }
    return count<=A;  //3
}
void sort(int a[], int l, int r) {
    int i = l, j = r, x = a[(l + r) / 2], y;
    while (i <= j) {
        while (a[i] < x) i++;
        while (a[j] > x) j--;
        if (i <= j) {
            y = a[i]; a[i] = a[j]; a[j] = y;
            i++; j--;
        }
    }
    if (i < r) sort(a, i, r);
    if (l < j) sort(a, l, j);
}
```

```cpp
int main() {
    int i;
    cin >> n >> B >> A;
    for (i = 1; i <= n; i++)
        cin >> M[i];
    for (i = 1; i <= B; i++)
        cin >> C[i];
    sort(M, 1, n);
    sort(C, 1, B);
    l = 0;
    r = n;
    while (l <= r) {
        mid = (l + r) / 2;
        if (check(mid) {  //4
            ans = mid;
            l = mid + 1;
        } else
            r = mid - 1;  //5
    }
    cout << ans << endl;
    return 0;
}
```

2008找第k大的数

给定一个长度为1,000,000的无序正整数序列, 以及另一个数n (1<=n<=1000000), 然后以类似快速排序的方法找到序列中第n大的数（关于第n大的数：例如序列{1，2，3，4，5，6}中第3大的数是4）

```
#include <iostream>
using namespace std;
int a[1000001],n,ans = -1;
void swap(int &a,int &b) {
    int c;
    c = a; a = b;    b = c;
}
int FindKth(int left, int right, int n) {
    int tmp,value,i,j;
    if (left == right) return left;
    tmp = rand()% (right - left) + left;
    swap(a[tmp],a[left]);
    value =____(1)_____
    i = left;
    j = right;
    while (i < j) {
        while (i < j && ____(2)____) j --;
        if (i < j) {a[i] = a[j]; i ++;} else break;
        while (i < j && ____(3)____) i ++;
        if (i < j) {a[j] = a[i]; j - -;} else break;
    }
    ____(4)____
    if (i < n) return FindKth(____(5)____);
    if (i > n) return ____(6)____
    return i;
}
```

```
int main() {
    int i;
    int m = 1000000;
    for (i = 1;i <= m;i ++)
        cin >> a[i];
    cin >> n;
    ans = FindKth(1,m,n);
    cout << a[ans];
    return 0;
}
```

```cpp
#include <iostream>
#include <cstdlib>
using namespace std;
int a[1000001],n,ans = -1;
void swap(int &a,int &b) {
    int c;
    c = a; a = b;   b = c;
}
int FindKth(int left, int right, int n) {
    int tmp,value,i,j;
    if (left == right) return left;
    tmp = rand()% (right - left) + left;
    swap(a[tmp],a[left]);
    value = a[left];  //1
    i = left;
    j = right;
    while (i < j) {  //从大到小排序
        while (i < j && a[j]<value) j--;  //2
        if (i < j) {a[i] = a[j]; i++;} else break;
        while (i < j && a[i]>value) i++;  //3
        if (i < j) {a[j] = a[i]; j--;} else break;
    }
    a[i]=value;  //4
    if (i < n) return FindKth(i+1, right, n);  //5
    if (i > n) return FindKth(left, i-1, n);  //6
    return i;
}
```

```cpp
int main() {
    int i;
    int m = 1000000;
    for (i = 1;i <= m;i ++)
        cin >> a[i];
    cin >> n;
    ans = FindKth(1,m,n);
    cout << a[ans];
    return 0;
}
```

2017快速幂

请完善下面的程序，该程序使用分治法求x^p mod m 的值。

输入：三个不超过10000 的正整数x，p，m。
输出：x^p mod m 的值。

提示：若 p 为偶数，$x^p=(x^2)^{p/2}$；若p 为奇数，$x^p=x*(x^2)^{(p-1)/2}$

2017快速幂

```
#include <iostream>
using namespace std;
int x, p, m, i, result;
int main() {
    cin >> x >> p >> m;
    result = ____(1)____ ;
    while ( ____(2)____ ) {
        if (p % 2 == 1)
        result = ____(3)____;
        p /= 2;
        x = ____(4)____;
    }
    cout << ____(5)____<< endl;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int x, p, m, i, result;
int main() {
    cin >> x >> p >> m;
    result = 1;  //1
    while ( p>0 ) {  //2
        if (p % 2 == 1)
            result = result*x%m;  //3
        p /= 2;
        x = x*x%m;  //4
    }
    cout << result << endl;  //5
    return 0;
}
```

作业要求

1.在演草纸上填写答案

2.待所有空填写完毕后，补全程序，将草稿纸上的答案填入程序的空白处

3.运行程序，记录所有出现的错误

4.调试程序，尝试改正错误的答案

5.完成整个程序，可以顺利得到期望的结果

6.记录错误和心得经验