


程设错题总结

1.Leetcode202 Happynumber

编写一个算法来判断一个数 `n` 是不是快乐数。

「快乐数」 定义为：

- 对于一个  整数，每一次将该数替换为它每个位置上的数字的平方和。
- 然后重复这个过程直到这个数变为 1，也可能是 **无限循环** 但始终变不到 1。
- 如果这个过程 **结果为 1**，那么这个数就是快乐数。

如果 `n` 是 快乐数 就返回 `true`；不是，则返回 `false`。

示例 1：

输入：n = 19

输出：true

解释：

$$1^2 + 9^2 = 82$$

$$8^2 + 2^2 = 68$$

$$6^2 + 8^2 = 100$$

$$1^2 + 0^2 + 0^2 = 1$$

1.使用快慢指针解决循环问题

当快指针追上慢指针时，代表完成了一次循环。

```
class Solution {
public:
    int happynumber(int n){
        int sum=0;
        while(n!=0){
            sum+=(n%10)*(n%10);
            n/=10;
        }
        return sum;
    }
    bool isHappy(int n) {
        int i=n,j=n;
        do{
            i=happynumber(i);
            j=happynumber(j);
            j=happynumber(j);
            if(i==1){
                return 1;
            }
        }
        while(i!=j);
        return 0;
    }
};
```

2.使用哈希表存储

此处使用容器: unordered_set

```
class Solution {
public:
    int happynumber(int n){
        int sum=0;
        while(n!=0){
            sum+=(n%10)*(n%10);
            n/=10;
        }
        return sum;
    }
    bool isHappy(int n) {
        unordered_set <int> list;
        while(n!=1){
            if(list.find(n)!=list.end()){
                return false;
            }else{
                list.insert(n);
            }
            n=happynumber(n);
        }
        return true;
    }
};
```

2.Acwing 817 数组去重

给定一个长度为 n 的数组 a , 请你编写一个函数:

```
int get_unique_count(int a[], int n); // 返回数组前n个数中的不同数的个数
```

输入格式

第一行包含一个整数 n 。

第二行包含 n 个整数, 表示数组 a 。

输出格式

共一行, 包含一个整数表示数组中不同数的个数。

数据范围

$1 \leq n \leq 1000$,

$1 \leq a_i \leq 1000$ 。

输入样例:

```
5
1 1 2 4 5
```

输出样例:

```
4
```

method1: 使用STL

unique函数:

unique是C++语言中的STL函数，包含于头文件中。**功能是将数组中相邻的重复元素去除。**然而其本质是将重复的元素移动到数组的末尾，最后再将迭代器指向第一个重复元素的下标。

```
#include <iostream>
#include <algorithm>

using namespace std;

int main()
{
    int n, s[1010];
    cin >> n;
    for (int i = 0; i < n; i++) cin >> s[i];
    sort(s, s + n);
    cout << unique(s, s + n) - s;

    return 0;
}
```

method2: 基本方法

```
#include <iostream>

using namespace std;

int b[1001];
int sum=0;
/*两个数组的区别:
a[]代表输入的数组（待去重的数组）
b[]数组是一个状态数组，其下标对应的值和a[]对应，初始值均为0，一旦出现a[i],即代表下标为a[i]的
b[]被访问过，状态值变为1。（且只有初次访问是生效的）
*/
int get_unique_count(int a[], int n)
{
    for(int i=1;i<=n;i++)
    {
        if(b[a[i]]==0)
            b[a[i]]=1;
        sum++;
    }
    return sum;
}
//函数作用：统计数组中一共出现了多少不相同的数。
int main()
{
    int n;
    cin>>n;
    int a[n+1];
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    cout<<get_unique_count(a,n);
}
```

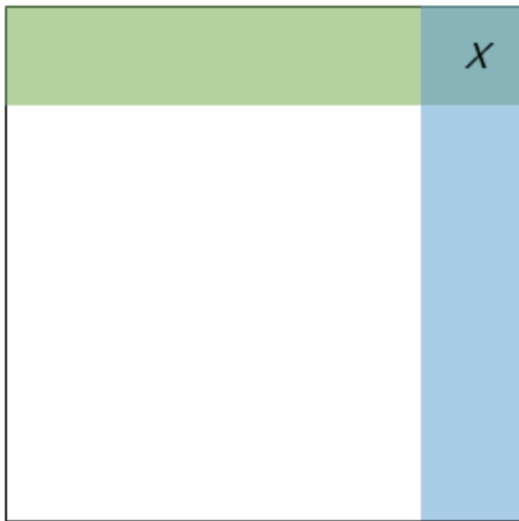
```
return 0;
}
```

3.ACwing 15 二维数组的查找

(单调性扫描) $O(n + m)$

核心在于发现每个子矩阵右上角的数的性质：

- 如下图所示， x 左边的数都小于等于 x ， x 下边的数都大于等于 x 。



因此我们可以从整个矩阵的右上角开始枚举，假设当前枚举的数是 x ：

- 如果 x 等于target，则说明我们找到了目标值，返回true；
- 如果 x 小于target，则 x 左边的数一定都小于target，我们可以直接排除当前一整行的数；
- 如果 x 大于target，则 x 下边的数一定都大于target，我们可以直接排除当前一整列的数；

排除一整行就是让枚举的点的横坐标加一，排除一整列就是让纵坐标减一。

思路：

- 暴力循环（两个for循环嵌套）
- 通过数组大小关系的规律实现**逐步逼近**的策略

```
bool Find(int target, vector<vector<int> > array) {
    int row=array.size();
    int col=array[0].size();
    int i=row-1;
    int j=0;
    //定义扫描的初始位置，(row-1,0),即数组棋盘的左下角
    while(i<=0&&j<col)
    {
        if(target==array[i][j])
            return true;
        //得到return后即可跳出循环，代表查找成功
        else if(target<array[i][j])
            i--;
        //删去最下面一行(判断大小关系)
    }
}
```

```

        else
            j++;
            //删去第一列，移动到第二列(判断大小关系)
        }
        return false;
    }
}

```

4.Acwing3801 最佳连续子数组

给定一个长度为 n 的数组 a_1, a_2, \dots, a_n 。

请你找到其中的最佳**连续**子数组。

最佳连续子数组需满足：

1. 子数组内各元素的算术平均数（即所有元素之和除以元素个数）尽可能大。
2. 满足条件 1 的前提下，子数组的长度尽可能长。

输出最佳连续子数组的长度。

输入格式

第一行包含整数 T ，表示共有 T 组测试数据。

每组数据，第一行包含整数 n 。

第二行包含 n 个整数 a_1, a_2, \dots, a_n 。

输出格式

每组数据输出一行结果，表示最佳连续子数组的长度。

数据范围

$1 \leq T \leq 20$,

$1 \leq n \leq 10^5$,

$0 \leq a_i \leq 10^9$,

同一测试点内所有 n 的和不超过 10^5 。

最暴力解法：分别枚举首项和尾项 $O(n^2)$

```

#include <iostream>
#include <vector>
using namespace std;
int testbestarray(){
    int n; cin>>n;
    int numarray[n];
    int sum=0,targetnumber=0;
    double maxave=0,ave;
    for(int i=0;i<n;i++){
        cin>>numarray[i];
    }
    for(int i=0;i<n;i++){
        for(int j=i;j<n;j++){
            for(int k=i;k<=j;k++){
                sum+=numarray[k];
            }
            //在已知i, j的情况下，遍历对子数组求和
            ave=double(sum)/(j-i+1);
            if(ave>maxave){

```

```

        maxave=ave;
        targetnumber=(j-i+1);
    }
    else if(ave==maxave){
        if((j-i+1)>targetnumber){
            targetnumber=(j-i+1);
        }
    }
    sum=0;
}
}
return targetnumber;
}
int main (){
    int T; cin>>T;
    int count=0;
    while (count<T){
        cout<<testbestarray()<<endl;
        count++;
    }
    return 0;
}

```

解法优化:

分治的思想: 将一个大问题拆分成若干个小问题再分别解决。

子问题1: 最大子数列的值

最大子数列的值一定等于数列中最大项的值

子问题2: 在子数列平均值最大的情况下, 找到最长长度:

找到数列中是否有若干项连续, 且值均为最大值。

优化代码:

```

#include <iostream>
using namespace std;
int testbestarray(){
    int n; cin>>n;
    int numarray[n];
    int sum=0,max=0;
    int count=0;int countmax=0;
    for(int i=0;i<n;i++){
        cin>>numarray[i];
        max=(numarray[i]>max?numarray[i]:max);
    }
    for(int i=0;i<n;){
        if(i<n&&numarray[i]==max){
            while(i<n&&numarray[i]==max){
                count++;
                i++;
            }
            i++;
            countmax=(countmax<count?count:countmax);
            count=0;
        }
    }
}

```

```

    }
    else if(i<n){
        i++;
    }
}

//这一段for循环是筛查的核心，如果某元素是列表中最大值，则进入while循环直到第一个非最大值元素的
//出现终止while循环，同时结束count++，并最大化countmax。在遍历完一整个数组后，即可得到最大子
//区间的长度。
return countmax;
}

int main (){
    int T; cin>>T;
    int count=0;
    while (count<T){
        cout<<testbestarray()<<endl;
        count++;
    }
    return 0;
}

```

进一步简化：

```

#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 10e5 + 10;
int a[N];

int main()
{
    int T;
    cin >> T;
    while(T--)
    {
        int n;
        cin >> n;

        int m = 0;
        for(int i = 0; i < n; i++)
        {
            cin >> a[i];
            m = max(m, a[i]); //m保存最大值
        }

        int res = 1;
        for(int i = 0; i < n; i++) //求长度
        {
            int t = 0;
            while(i < n && a[i] == m)
            {

```

```

        t++;
        i++;
    }
    res = max(res, t);
}
/*核心while循环：
此处不用再添加if语句，直接使用外部的for循环即可。
*/
cout << res << endl;
}
return 0;
}

```

5.ACwing 862 三元组排序

给定 N 个三元组 (x, y, z) ，其中 x 是整数， y 是浮点数， z 是字符串。

请你按照 x 从小到大的顺序将这些三元组打印出来。

数据保证不同三元组的 x 值互不相同。

输入格式

第一行包含整数 N 。

接下来 N 行，每行包含一个整数 x ，一个浮点数 y ，一个字符串 z ，表示一个三元组，三者之间用空格隔开。

输出格式

共 N 行，按照 x 从小到大的顺序，每行输出一个三元组。

注意，所有输入和输出的浮点数 y 均保留两位小数。

数据范围

$1 \leq N \leq 10000$,

$1 \leq x, y \leq 10^5$,

字符串总长度不超过 10^5 。

思路：

- 最基本的排序使用排序算法（此处用algorithm库中的sort（）函数）
- 如何构建一一对应的关系？
 - map容器
 - pair：将两组数据整合成一个数据对
 - （int, (double,string)）
 - 对pair类型的int数排序，一一对应的（double, string）满足映射

代码实现：

①最基本的数组实现

排序算法（Bubblesort）+交换函数swap（）（内置在C++库中）

```

#include <iostream>
#include <string>
using namespace std;
int main () {

```



```

int n; cin>>n;
int intlist[n];
double doublelist[n];
string stringlist[n];
//定义三个数组，分别储存三种不同的数据结构。
for(int i=0;i<n;i++){
    cin>>intlist[i]>>doublelist[i]>>stringlist[i];
}

for(int j=n;j>1;j--){
    for(int i=0;i<j-1;i++){
        if(intlist[i]>intlist[i+1]){
            swap(intlist[i],intlist[i+1]);
            swap(doublelist[i],doublelist[i+1]);
            swap(stringlist[i],stringlist[i+1]);
        }
    }
}
//核心算法：冒泡排序
//从第一位开始逐项与后一位冒泡比较确定是否交换，第一轮下来就确定末尾项为最大值。
//之后通过外层的for循环逐步确定直至首项
for(int i=0;i<n;i++){
    printf("%d %.21f",intlist[i],doublelist[i]);
    cout<<" "<<stringlist[i]<<endl;
}
return 0;
}

```

②map映射

```

#include<iostream>
#include<cstring>
#include<cstdio>
#include<map>

#define x first
#define y second

using namespace std;

const int N = 10010;
typedef pair<double, string> PII;
map<int, PII> ans;
//定义了一个从int向PII的映射，就不用使用两次pair了
int n;

int main()
{
    int a;
    double b;
    string c;
    cin >> n;

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

```

```

{
    cin >> a >> b >> c;
    ans.insert({a, {b, c}});
}

for (auto iter = ans.begin(); iter != ans.end(); iter++)
    printf("%d %.2f %s\n", iter->x, iter->y.x, iter->y.y.c_str());
/*这里 iter是一个迭代器
iter->first代表指向map类型的first成员函数（即自变量）
iter->second代表指向因变量
iter->second.first代表指向因变量pair的第一个元素
iter->second.second          第二个元素
c_str()返回一个指向字符串的指针
*/
return 0;
}

```

③使用pair类型

```

#include<iostream>
#include<cstring>
#include<algorithm>
#include<vector>

#define x first
#define y second

using namespace std;
const int N = 10010;
typedef pair<int, pair<double, string >> PII;

vector<PII> ans;
int n, a;
double b;
string s;
int main()
{
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> a >> b >> s;
        ans.push_back({a, {b, s}});
        //vector类型的push_back函数，{a,{b,s}}是一个PII类型的数据
    }
    sort(ans.begin(), ans.end());
    //使用sort()函数实现ans这个vector的自动排序(默认升序)
    for (auto i: ans)
        printf("%d %.21f %s\n", i.x, i.y.x, i.y.y.c_str());
    //若i是一个pair类型，则i.first代表pair的第一个元素，而i.second代表pair的第二个元素。
    return 0;
}

```