天津大学

程序设计综合实践课程报告

基础算法 2 实验

学生姓名 陈秋澄

学院名称 智能与计算学部

专　　业 大类

学　　号 3022244290

1. 最多水容器

1.1 题目分析

可以使用枚举法，取遍所有“两边”，由木桶效应，两边中较短的一个才会影响装水量。列举所有边乘距离的情况，进行比较，并输出最大值。

1.2 题目代码

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n,m=0,y=0;
    int x=0;
    int a[100],b[100][100];
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>a[i];
    }
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
            y=min(a[i],a[j]);   //取两边中较短的一个
            b[i][j]=y*(j-i);    //算体积
            y=0;
        }
    }
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(b[i][j]>x){
                x=b[i][j];     //逐个比较，选出最大值
            }
        }
    }
    cout<<x<<endl;             //输出最大值
}
```

2. 区间和统计

2.1 题目分析

设置两个 for 循环，从第一个 for 循环的起点对应的元素开始加，如果能正好和第二个输入的数相等，区间个数加 1，否则一直加，直到相等或者超出数组长度，跳出内部 for 循环，在外层 for 循环实现起点向后推移，之后多次遍历。

2.2 题目代码

```
#include <iostream>
using namespace std;
int a[]={};
int main()
{
    int T;
    cin >>T;
    for(int i=0;i<T;i++)
    {
        int n,p;
        cin >>n>>p;
        int num=0;
        int a[n];
        for(int j=0;j<n;j++)
        {
        cin >>a[j];       //输入数组的各个元素
        }
        for(int t=0;t<n;t++)
           {
            int q=0;
            for(int m=t;m<n;m++)
            {
            q+=a[m];      //数组内元素相加
              if(q==p)    //若和为 p
              {
              num++;      //区间数+1
              }
            }
        }
```

```
        cout <<num <<endl;  //输出答案
    }
    return 0;
}
```

3. 子矩阵求和

3.1 题目分析

使用二维数组，便于进行坐标矩阵相应模块的加法，以满足题意。

还有将每次输入的两组横纵坐标分别进行比较，以正确求出矩阵（可以理解成矩形）相应元素之和。

3.2 题目代码

```
#include <iostream>
#include<algorithm>
using namespace std;
long m,n,q;
long long a[1001][1001];    //输入矩阵的行数，列数，测试样例数
int main(){
    cin>>m>>n>>q;
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            cin>>a[i][j];    //将输入的矩阵元素编号
        }
    }
    while(q>0){
        int cnt=0;
        int b[4]={0,0,0,0};
        for(int i=0;i<4;i++){
            cin>>b[i];       //将两组坐标编号
        }
        if(b[0]>b[2]){
            swap(b[0],b[2]);    //比较 x 坐标
        }
        if(b[1]>b[3]){
            swap(b[1],b[3]);        //比较 y 坐标
        }
        for(int i=b[0];i<=b[2];i++){
            for(int j=b[1];j<=b[3];j++){
                cnt+=a[i][j];       //进行相应元素的加法
            }
        }
        cout<<cnt<<endl;           //输出结果
```

```
        q--;
    }
    return 0;
}
```

4. 选择排序

4.1 题目分析

应用循环，控制测试样例数，数据个数等。

用排序算法，依次比较，输出答案即可。

4.2 题目代码

```
#include<iostream>
#include<algorithm>
using namespace std;
int m,n,a[1000],temp;
int main(){
    cin>>m;                          //测试样例数
    while(m>0){
        cin>>n;                      //数据个数
        for(int i=0;i<n;i++){
            cin>>a[i];               //输入待比较数据
        }
        for(int i=0;i<n;i++){
            temp=0;
            for(int j=i;j<n;j++){
                if(a[i]>a[j]){
                    temp=a[j];
                    a[j]=a[i];
                    a[i]=temp;       //进行交换以升序排列
                }
            }
        }
        for(int i=0;i<n;i++){
            cout<<a[i]<<" ";      //输出排好序的数据
        }
        cout<<endl;
        m--;
    }
}
```

5. 前 m 大的数

5.1 题目分析

N 个数两两相加，然后输出前 M 个数字。模拟相加 sort 一遍能过。

提交过程中发现数组开的过大会超时，并且使用散列（哈希表）去解此题效果会更好。

5.2 题目代码

```
#include<iostream>
#include<cstring>
using namespace std;
int main(){
    int n,m,a[3001],b[10001];
    while(cin>>n>>m){
        memset(b,0,sizeof(b));     //数组初始化
        for(int i=0;i<n;i++){
            cin>>a[i];
        }
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                b[a[i]+a[j]]++;    //向后移动
            }
        }
        int k=0;
        for(int i=10000;i>0&&m>0;){
            if(b[i]==0){i--;continue;}
                if(k){
                    cout<<" "<<i;   //注意输出格式，要有空格
                }else {
                    cout<<i;
                }
                k=1;
                b[i]--;
                m--;
        }
        cout<<endl;
    }
    return 0;
```

```
}
```

6. Greed

6.1 题目分析

包含体积和容量两种数据,最后只需判断体积的和是否小于容量中最大的两个数的和。

可以在统计总可乐量后，找到容量最大的可乐罐，确认能否装下。

6.2 题目代码

```
#include <iostream>
#include <cstdio>
using namespace std;
long a[100001]={};
long b[100001]={};
int main(){
    int n;
    long sum=0;
    long max1=0, max2=0;   //进行初始化，防止后续计算错误
    cin >> n;
    for (int i=0;i<n;i++){
        cin >> a[i];
                           // 统计总可乐量
        sum += a[i];
    }

    for (int i = 0; i < n; i++)
    {
        cin >> b[i];
                           // 取最大的两个可乐罐
        if (b[i] > max2)
        {
            max1 = max2;
            max2 = b[i];
        }
        else if (b[i] > max1)
            max1 = b[i];
    }
    if (sum > max1 + max2)        //能装下
        cout << "NO" << endl;
```

```
    else                            //不能装下
        cout << "YES" << endl;
    return 0;
}
```

7. 珠心算测验

7.1 题目分析

将输入的数排序，枚举每一个数，然后分别从头和尾找有没有两数之和等于这个数的，若找到，则计数器加 1 即可，依次进行下一个数的判断。

还要注意排除掉重复计算的数据。

7.2 题目代码

```
#include<bits/stdc++.h>
using namespace std;
long long n,a[1001],b[1001],bk=0;
     //首先定义变量，n 是数的数量，a 是数字，b 是为了排除掉重复的，bk 是为了记
    //录（book）
int main(){

    cin>>n;
    for(int i=0;i<n;i++){
        cin>>a[i];
        b[i]=2;                  //去重
    }
    for(int i=0;i<n;i++){
        for(int o=i+1;o<n;o++){
            for(int p=0;p<n;p++){//穷举
                if(a[p]==a[i]+a[o]&&b[p]!=1){
                    bk++;
                    b[p]=1;       //判定是否是所有组合的和。为了唯一性，所以
                        //要测试 a 是不是被动过的，b 就是在这里派上用场的
                        //如果没动过就把 a 对应的 b 设成动过的
                }
            }
        }
    }
    cout<<bk;
    return 0;
}
```

8. Monthly Expense

8.1 题目分析

采用二分思想,初始化 low 为 n 天中最大花费，high 为 n 天花费总和.每次令 mid=(low+high)/2,判断这个解是否符合题意.若不符合,就令 low=mid+1,反之,令 high=mid。

8.2 题目代码

```
#include <bits/stdc++.h>
using namespace std;
int a[1000];
bool find(int mid,int n,int m)
{
int sum=0,num=1;           //num 表示分了几组
for(int i=0;i<n;i++)
{
if(sum+a[i]<=mid)    //如果当前组的和加上当前这一个小于限制,这一组加上
sum+=a[i];
else                 //反之,需要新的一组
num++,sum=a[i];
}
return (num<=m);
}
int main()
{
int n,m;
while(cin >>n>>m)
{
int sum=0,maxx=0;
for(int i=0;i<n;i++)
{
cin >>a[i];
sum+=a[i];
maxx = max(maxx, a[i]);
}
int low=maxx,high=sum;       //二分的上下界
while(low!=high)
{
```

```
int mid=(high+low)/2;
if(find(mid,n,m))
high=mid;                    //若答案可行,上限下移
else
low=mid+1;                   //答案不可行,下限上移
}
cout <<low<<endl;
}
return 0;
}
```