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 陈昊昆

学院名称 智算学部

专　　业 软件工程

学　　号 **3021001196**

1. 最多水容器

1.1 题目分析

两层 for 循环，计算出全部的每两根柱子间的面积

计算方法：min{柱子 1 长度，柱子 2 长度}*两根柱子的间距

比较得到最大值

1.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n; //容器数量
    int *a = new int[n];
    for (int i = 0; i < n; i++) cin >> a[i];  //输入柱子长度
    int mh, s, max = 0;
    //mh 较短柱子长度    s 两柱子间面积    max 目前最大面积
    for (int i = 0; i < n; i++){
        for (int j = i+1; j < n; j++){
            if (a[i] < a[j]) mh = a[i];
            else mh = a[j];
            s = mh * (j - i);
            if (s > max) max = s;
        }
    }
    cout << max;
    return 0;
}
```

2. 区间和统计

2.1 题目分析

计算该数组所有连续的子集和

判断每个子集和是否是所需的数

2.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int m;
    cin >> m;
    for (int u = 0; u  < m; u++){
        int n, sum;
        int count = 0;
        cin >> n >> sum;
        int *a = new int[n];
        for (int i = 0; i < n; i++) cin >> a[i];  // 输入数组
        // 以每一个数组位置为起点，计算所有以其为起点的子集
        for (int i = 0; i < n; i++){
            int tsum = 0;
            int  j = i;
            while (j < n){
                tsum += a[j];
                j++;
                if (tsum == sum) count++;  // 满足条件 数目+1
            }

        }
        delete a;
        cout << count << endl;
    }
    return 0;
}
```

3. 子矩阵求和

3.1 题目分析

建立动态二维数组，存放矩阵

输入两个顶点，保证 x1<x2,y1<y2

然后遍历子矩阵，求和

3.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int m, n, q;
    cin >> m >> n >> q;

    //建立动态二维数组
    long long **arr = new long long*[m];
    for (int i = 0; i < m; ++i) arr[i] = new long long[n];

    //给数组中的元素赋值
    for (int i = 0; i < m; i++){
        for (int j = 0; j < n; j++){
            cin >> arr[i][j];
        }
    }

    for(int i = 0; i < q; i++){
        int x1,y1,x2,y2;
        int sum = 0;
        cin >> x1 >> y1 >> x2 >> y2;
        // 保证 x1<x2,y1<y2
        if (x1 > x2) {
            int m = x1;
            x1 = x2;
            x2 = m;
        }
```

```
        if (y1 > y2) {
            int m = y1;
            y1 = y2;
            y2 = m;
        }
        //对子矩阵求和
        for (int u = x1; u <= x2; u++){
            for (int w = y1; w <= y2; w++){
                sum += arr[u][w];
            }
        }
        cout << sum << endl;
    }

    delete arr;
    return 0;
}
```

4. 选择排序

4.1 题目分析

把数组分为有序区和无序区

初始状态有序区为零

然后遍历数组，在当前的无序区找到最小的值，与当前锁定的位置值做交换，扩大有序区

4.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;
void selectsort (int a[], int n);

int main(){
    int u;
    cin >> u;
    for (int k = 0; k < u; k++){
        int n;
        cin >> n;
        int *arr = new int [n];
        for (int i = 0; i < n; i++) cin >> arr[i];
        selectsort(arr, n);
        for (int i = 0; i < n; i++) cout << arr[i] << " ";
        cout << endl;
    }
    return 0;
}

// 选择排序
void selectsort (int a[], int n){
    for(int i = 0; i < n-1; i++){
        int min = a[i];
        int num = 0;
        int j = i+1;
        // 找出当前无序区的最小值
        for (; j < n; j++){
```

```
            if (a[j] < min){
                min = a[j];
                num = j;
            }
        }
        // 如果找到，则做交换
        if (num != 0){
            int tem = a[i];
            a[i] = a[num];
            a[num] = tem;
        }

    }
}
```

5. 前 m 大的数

5.1 题目分析

计算两两之和，存入新的数组

对新的数组进行从大到小排序

输入前 m 个数

5.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;
//void bubblesort(int *a, int n);
bool compare(int a,int b)
{
    return a>b;
}

int main(){
    int n;
    while(cin >> n){
        int m;
        cin >> m;
        int *a = new int [n];
        int u = n*(n-1)/2;
        int *arr = new int [u];
        for (int i = 0; i < n; i++) cin >> a[i];
        int num = 0;
        // 计算两两的和，存入数组
        for (int i = 0; i < n-1; i++){
            for(int j = i+1; j < n; j++){
                arr[num] = a[i] + a[j];
                num++;
            }
        }
        sort(arr, arr + u, compare); //从大到小的快排
        for (int i = 0; i < m; i++) cout << arr[i] << " ";
        cout << endl;
```

```
    }
    return 0;
}

//冒泡排序复杂度高
/*void bubblesort(int *a, int n){
    for(int i = 0; i < n-1; i++){
        for (int j = 0; j < n - 1 -i; j++){
            if (a[j] < a[j+1]){
            int t = a[j];
            a[j] = a[j+1];
            a[j+1] = t;

        }
        }

    }
}*/
```

6. Greed

6.1 题目分析

计算出每个 can 的剩余容量 并找到前两个最大的剩余容量

同时记录这两个 can 中的 cola volume，不需要计入

对所有 remaining cola 求和（不计入上述两个 volume）

判断是否小于等于两个最大剩余容量之和

6.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    long long *a = new long long [n];
    long long *b = new long long [n];
    long long *c = new long long [n];



    for (int i = 0; i < n; i++) cin >> a[i];
    for (int i = 0; i < n; i++) cin >> b[i];
    for (int i = 0; i < n; i++) c[i] = b[i] - a[i];  //每个 can 的剩余容
量

    // 找出两个剩余容量最大的 can
    int max = c[0];
    int num = 0;
    for (int i = 0; i < n; i++){
        if (c[i] > max) {
            max = c[i];
            num = i;
        }
    }
    int c1 = c[num];
    int a1 = a[num];
```

```
    c[num] = 0;

    max = c[0];
    num = 0;
    for (int i = 0; i < n; i++){
        if (c[i] > max) {
            max = c[i];
            num = i;
        }
    }

    int c2 = c[num];
    int a2 = a[num];
    c[num] = 0;

    int totalc = c1 + c2;  //剩余容量之和
    int totala = -a1 - a2;; //这两个 can 中的 cola，不需要再占用 totalc
    for (int i = 0; i < n; i++) totala += a[i];

    if (totalc >= totala ) cout << "YES";
    else cout << "NO";
    return 0;
}
```

7. 珠心算测验

7.1 题目分析

先对数组排序

然后三重 for 循环，检查后者是否为前两者之和

为了防止重复统计，建立两个数组

一旦该数被统计过，就置 0

7.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    //为了去重，多建一个数组
    long long *arr = new long long [n];
    long long *brr = new long long [n];
    int cnt = 0;
    for (int i = 0; i < n; i++) cin >> arr[i];
    for (int i = 0; i < n; i++) brr[i] = arr[i];
    //先排序，则只需要检查两个数后面的数
    sort (arr, arr+n);
    sort (brr, brr+n);
    for (int i = 0; i < n-1; i++){
        for(int j = i+1; j < n; j++){
            long long m = arr[i] + arr[j];
            for(int k = j+1; k < n; k++){
                if (m == brr[k]) {
                    brr[k] = 0;  //防止重复统计该数
                    cnt++;
                }
            }
        }
    }
    cout << cnt;
```

```
    return 0;
}
```

8. Monthly Expense

8.1 题目分析

使用二分查找
上界为所有数之和，下界为最大的数
当组数小于目标组数，则提高下限，当组数大于目标组数，则降低上限
直到上界等于下界，就是目标值

8.2 题目代码（带注释）

```
#include <bits/stdc++.h>
using namespace std;
bool find(int top, int *a);

int n, m;
int main(){
    cin >> n >> m;
    int *arr = new int [n];
    int high = 0;
    int low = 0;
    for (int i = 0; i < n; i++){
        cin >> arr[i];
        high += arr[i];  //上界为所有数之和
        if (arr[i] > low) low = arr[i];  //下界为最大的数
    }

    while(high > low){
        int mid = (high + low)/2;
        if(find(mid, arr)) low = mid +1;  //组数小于目标组数，则提高下限
        else high = mid;  //组数大于目标组数，则降低上限
    }
    delete arr;
    cout << high;

    return 0;
}
```

```
// 二分查找
bool find(int top, int *a){
    int cnt = 1;
    int c = 0;
    for (int i = 0; i < n; i++){
        if(c < top) c += a[i];  //当目前的和小于最大，则继续加下一个
        if(c > top) {  //当目前的和大于最大，就新开一组
            cnt++;
            c = a[i];
        }
    }
    if(cnt <= m) return false;
    else return true;
}
```