A Mini-Max Sum

时间限制: 1000 ms 内存限制: 65536 kb
总通过人数: 1318 总提交人数: 1429

题目描述

给你五个正整数。从五个整数中选四个整数相加，找出四个整数和的最大值和最小值。

输入

第一行五个正整数。不超过 10^9

输出

输出一行以空格分隔的最小值和最大值。可能超过 32 位整数。

输入样例

```
1 2 3 4 5
```

输出样例

```
10 14
```

考察知识点：基础运算
解题思路：最大的四整数和的对偶是找到最小的整数， 最小的四整数和同理。

代码：

```
#include <stdio.h>
#include <stdlib.h>
```

```
void miniMaxSum(int* arr) {
    int i;
    long long sum = 0;
    for (i = 0; i < 5; i++) {
        sum += *(arr + i);
    }
    long long min = sum, max = 0;
    for (i = 0; i < 5; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    printf("%lld %lld", sum - max, sum - min);

}

int main() {
    int i;
    int* arr = malloc(5 * sizeof(int));

    for (i = 0; i < 5; i++) {
    scanf("%d", arr + i);
    }

    miniMaxSum(arr);

    return 0;
}
```

B 杨辉三角.改

时间限制：1000ms　内存限制：65536kb

通过率：1216/1278 (95.15%)　正确率：1216/2143 (56.74%)

题目描述

橙橙最近突然迷上了杨辉三角形，它是这样的一个三角形数表： 1，每行端点与结尾的数为 1； 2，其余每个数等于它“肩上”两数之和。 亲爱的同学们，你能不能帮橙橙编程计算一下，打印出 N(1≤N≤30)行的杨辉三角形呢？快动手试试吧！

输入

输入一个正整数 $N(1≤N≤30)$。

输出

NN 行的杨辉三角形，各个数字元素之间以空格隔开。

输入样例

```
4
```

输出样例

```
1

1 1

1 2 1

1 3 3 1
```

考察知识点

基础知识
难度系数：2

解题思路

本题考查数组的基础知识，只需要定义一个 30*30 的二维数组，然后根据杨辉三角形的组成规则对数组相应的元素一一赋值即可，参考代码如下：

```
#include <stdio.h>
#define N 36
int main()
{
    int i, j, k, n=0, a[N][N];
    /*定义二维数组 a[][]*/
    scanf("%d",&n);
    for(i=1;i<=n;i++) a[i][1] = a[i][i] = 1;
 /*两边的数令它为 1，因为现在循环从 1 开始，就认为 a[i][1]为第一个数*/
    for(i=3;i<=n;i++)
        for(j=2;j<=i-1;j++)
            a[i][j]=a[i-1][j-1]+a[i-1][j];
 /*除两边的数外都等于上两顶数之和*/
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=i;j++)
/*j<=i 的原因是不输出其它的数，只输出我们想要的数*/
            printf("%d ",a[i][j]);
        printf("\n");
/*当一行输出完以后换行继续下一行的输出*/
    }
    return 0;
}
```

C Terry 与五角星

时间限制：1000ms　内存限制：65536kb

通过率：1171/1206 (97.10%)　正确率：1171/1667 (70.25%)

题目描述

五角星的五只尖角的角平分线相交于一点，我们称这一点为五角星的中心。

现在，我们将五角星关于它的中心旋转 alpha 角度得到一个新五角星，再与原五角星重叠，得到一个新的平面图形，请问在新的平面图形中二维平面被分成了多少个区

域？

输入

一行，包含一个整数 alpha，代表旋转角度

0<=alpha<360

输出

输出一个整数，表示平面被分割成多少部分。

输入样例

```
180
```

输出样例

```
32
```

HINT

区域是由线段构成的最小封闭图形+最外面的无穷大区域也算在内

考察知识点

基础知识
难度：2

解题思路：

经过观察，我们就会发现答案只有 7 和 32 两种，当旋转角度是 72 的整数倍答案是 7，否则答案是 32

标程

```
#include <stdio.h>

int main() {
            int n;
            scanf("%d",&n);
            if(n%72==0) printf("7\n");
            else printf("32\n");
}
```

D 田忌赛马

时间限制：1000ms　内存限制：65536kb

通过率：981/1093 (89.75%)　正确率：981/2737 (35.84%)

题目描述：

你和对手赛马,双方都有 n 匹马,每匹马的能力数值 ai(0<=ai<=1000),由你来规定双方马匹出场的顺序,请问你最多能赢多少场
(要求对局时候你的马能力值大于对方的才算赢,等于算平手)

输入：

第一行,一个整数 n(1<=n<=100)
接下来的 2 行,每行 n 个整数 a1,a2....an,用空格分割,代表每匹马的能力. 第二行的指数是你的马的能力指数,第三行则是对手的.

输出：

一个整数,表示你最多胜利的场次

输入样例：

```
5
5 4 3 2 1
```

```
1 2 3 4 5
```

输出样例:

```
4
```

样例解释和提示:

```
5 对 4,4 对 3,3 对 2,2 对 1,1 对 5,
提示:需要排序,然后从对手最弱的马开始,分配你能赢过它的能力值最小的马匹.
```

考察知识点

```
排序
难度系数：3
```

思路解析:

题目难度较难,担心同学们没有思路,所以给出了充分的提示,具体的操作则是先对双方马匹排序,然后从对方最弱的开始,一匹一匹的分配你的马,分配掉的是你当前能力值大于这匹马中最弱的那匹,举例
就是对方最弱的马匹为 3,你现在有能力值为 2,4,5,6,7 的 5 匹马,就分配能力为 4 的马出去.

题目难度:

4,数组排序和条件判断,设计基本的贪心思想.

题目代码:

```
#include<stdio.h>
#include<stdlib.h>
#define MAXN 100
int a[MAXN +10];
```

```
int b[MAXN +10];
int compare(const void *a,const void *b)
{
    return *(int*)a - *(int*)b;
}
int main()
{
    int n,i;
    scanf("%d",&n);
    for(i = 0;i < n;i++)
            scanf("%d",&a[i]);
    for(i = 0;i < n;i++)
            scanf("%d",&b[i]);
    qsort(a,n,sizeof(a[0]),compare);
    qsort(b,n,sizeof(b[0]),compare);
    int cura = 0,curb = 0,ans = 0;
    while(cura < n && curb < n){
            if(a[cura] > b[curb])
            {
                    ans++;
                    cura++;
                    curb++;
            }
            else
                    cura++;
    }
    printf("%d",ans);
    return 0;
}
```

E 最大收益

时间限制：1000ms　内存限制：65536kb

通过率：529/883 (59.91%)　正确率：529/5129 (10.31%)

题目描述

W 国的国王手下拥有 n 个猎人，这些猎人能力不一，每个人都有一个自己的能力范围[low,high]。有一天国王要大宴宾客，他想让发布一项任务，让这些猎人带回尽可能多的猎物。这个任务有一个难度值 k，每一个猎人都会对这个任务进行判断，若难度值在自己的

能力范围内（k∈[low,high]），这个猎人将为国王带回价值为 k 的猎物，否则这个猎人将放弃这个任务。现在国王想要确定这个难度值 k，使得他能得到最高的猎物价值总和 value。

输入

第一行为一个整数 n(0<n<=1000)，为猎人的数量。
接下来 n 行每行两个整数 lowi,highi，为第 i 个猎人的能力范围，两个数都在 int 范围内，且 lowi<=highi。

输出

一个整数 value，为能取得的最高猎物价值总和。

输入样例

```
4
1 3
2 4
3 5
4 7
```

输出样例

```
12
```

样例解释

当 k=4 时，后三个猎人都能带来价值为 4 的猎物，此时总和最高为 12

考察知识点

```
数组
```

难度系数：4

解题思路

这道题最直观的想法是求出 k 可能的取值范围（下界为各端点最小值，上界为各端点最大值），然后依次枚举计算收益，求出最大值。但是各端点的取值在 int 范围内，这样直接枚举极有可能超时，因此需要改进算法。
不难想到收益最大时对应的 k 值必在端点处取到（若 k 不是任何一个端点，则取各端点中最小的那个比 k 大的值，此时接受任务的人数不可能缩小，但是 k 变大了，因此答案更优），所以只需要枚举每个端点（甚至只是上界）计算对应的收益取最大值即可，时间复杂度 O(N^2)。
本题还有 O(NlogN)的算法，将所有端点排序后，按顺序扫一遍同时维护当前可接受任务的任务即可，有兴趣的同学可以自行实现。
Ps：本题大数据答案会爆 int，很多同学没有注意到这个。

参考代码：

```
#include<stdio.h>
int n,low[1005],high[1005],i;
typedef long long ll;
ll value = 0;
ll get(int k){
    int i;
    ll sum = 0;
    for (i = 0;i < n;i++)
            if (k >= low[i] && k <= high[i])
                    sum += k;
    return sum;
}
ll max(ll a,ll b){
    return a > b? a : b;
}
int main(){
    scanf("%d",&n);
    for (i = 0;i < n;i++){
            scanf("%d%d",&low[i],&high[i]);
    }
    for (i = 0;i < n;i++)
            value = max(value,max(get(low[i]),get(high[i])));
    printf("%lld",value);
```

```
        return 0;
    }
```

F 均分纸牌

时间限制：1000ms　内存限制：65536kb

通过率：804/887 (90.64%)　正确率：804/1543 (52.11%)

题目描述

有 **N** 堆纸牌，编号分别为 1,2,…,N。
每堆上有若干张，但纸牌总数必为 **N 的倍数**。可以在任一堆上取若干张纸牌，然后移动。

移牌规则为：在编号为 1 堆上取的纸牌，只能移到编号为 2 的堆上；在编号为 N 的堆上取的纸牌，只能移到编号为 N−1 的堆上；其他堆上取的纸牌，可以移到相邻左边或右边的堆上。即只能在相邻牌堆上移动

现在要求找出一种移动方法，用最少的移动次数使每堆上纸牌数都一样多。

输入

两行
第一行为整数 N（N 堆纸牌，1<=N<=100）
第二行为 N 个正整数 Ai（第 i 堆的初始牌数，1<=Ai<=10000）

输出

一行：即所有堆均达到相等时的最少移动次数。

输入样例

```
4
9 8 17 6
```

输出样例

```
3
```

HINT

1.从 A3 取 4 张牌放到 A4
2.从 A3 取 3 张牌放到 A2
3.从 A2 取 1 张牌放到 A1

考察知识点

四则运算
难度系数：4

题解：

模拟+贪心的方法来做，因为题目已经保证了总数是 N 的倍数，那么有一个正好平均的分配结果。只能在相邻的牌堆进行移动，那么我们从第一个牌堆开始和平均数比较，记录下它与平均数的差（也就是需要下一个需要向它移动的数量）；在处理下一个的时候要首先加上这个记录的差值。

```
#include<stdio.h>
int a[150],n,t,count;
int main()
{
    scanf("%d",&n);count=0;t=0;
    int tag = 0;
    int i;
    for(i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
        t+=a[i];
    }
    t/=n;
    for(i=1;i<=n;i++)
    {
     a[i] += tag;
            tag = 0;
        if(a[i]==t) continue;
        else
```

```
                {
                        tag = a[i]-t;
                        count++;
                }
        }
        printf("%d",count);
    }
```