11月10日测试赛题解

本次题目难度可能比平时实验作业难度大，主要考察了一些包括排序、枚举等基础的算法以及复杂问题简单化的思想，也存在较复杂需要使用一些简单数据结构来完成的题目，部分课堂内容不涉及的知识点供有兴趣的同学参考学习。以下是助教整理出来的题解，题目不止一种解法，欢迎同学们拿出自己的解法和大家一起讨论。

A - 奖学金

有各种特殊要求的排序。考察对排序算法的理解。

```
#include <stdio.h>

#define N 305
int c[N], m[N], e[N], pos[N];// c[i], m[i], e[i] 表示学号为i的同学各科成绩，pos[i]表示学号为i的同学排序后位置

void swap(int x, int y) { //交换所在位置为x和y的两个同学所有的信息
    int tmp = c[x];
    c[x] = c[y];
    c[y] = tmp;
    tmp = m[x];
    m[x] = m[y];
    m[y] = tmp;
    tmp = e[x];
    e[x] = e[y];
    e[y] = tmp;
    tmp = pos[x];
    pos[x] = pos[y];
    pos[y] = tmp;
}

int compare(int x, int y) { //比较所在位置为x和y的两位同学谁应该排在前面，若返回1则表示所在位置为y的同学应该排在前面，返回0表示x排在前面
    if (c[x] + m[x] + e[x] == c[y] + m[y] + e[y]) {
        return c[x] < c[y];
    } else return c[x] + m[x] + e[x] < c[y] + m[y] + e[y];
}

int main() {
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)scanf("%d %d %d", c + i, m + i, e + i);
    for (int i = 1; i <= n; i++)pos[i] = i;
    // 选择排序
    for (int i = 1; i < n; i++) {
        for (int j = i + 1; j <= n; j++) {
            if (compare(i, j))swap(i, j);
        }
    }
    for (int i = 1; i <= 5; i++) {
        printf("%d %d\n", pos[i], c[i] + e[i] + m[i]);
    }
    return 0;
```

```
}
```

如果使用 c 语言的结构体来做这道题，会更加简单，容易理解。

B - 烤鸡

「正解是搜索，歪解是枚举。」

歪解

```
#include<stdio.h>

int main() {
    int a, b, c, d, e, f, g, h, i, j, in, x = 0;
    scanf("%d", &in);
    for (a = 1; a <= 3; a++) {
        for (b = 1; b <= 3; b++) {
            for (c = 1; c <= 3; c++) {
                for (d = 1; d <= 3; d++) {
                    for (e = 1; e <= 3; e++) {
                        for (f = 1; f <= 3; f++) {
                            for (g = 1; g <= 3; g++) {
                                for (h = 1; h <= 3; h++) {
                                    for (i = 1; i <= 3; i++) {
                                        for (j = 1; j <= 3; j++) {
                                            if (a + b + c + d + e + f + g + h + i + j == in) {
                                                x++;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    printf("%d\n", x);
    for (a = 1; a <= 3; a++) {
        for (b = 1; b <= 3; b++) {
            for (c = 1; c <= 3; c++) {
                for (d = 1; d <= 3; d++) {
                    for (e = 1; e <= 3; e++) {
                        for (f = 1; f <= 3; f++) {
                            for (g = 1; g <= 3; g++) {
                                for (h = 1; h <= 3; h++) {
                                    for (i = 1; i <= 3; i++) {
                                        for (j = 1; j <= 3; j++) {
                                            if (a + b + c + d + e + f + g + h + i + j == in) {
                                                printf("%d ", a);
                                                printf("%d ", b);
                                                printf("%d ", c);
                                                printf("%d ", d);
                                                printf("%d ", e);
                                                printf("%d ", f);
                                                printf("%d ", g);
```

```
                                        printf("%d ", h);
                                        printf("%d ", i);
                                        printf("%d\n", j);
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
// so beautiful!
// 来自 https://www.luogu.com.cn/problem/solution/P2089 yyy2015c01 的题解
```

正解

这题只有10种配料，可以写10此循环，但是如果有100次循环呢？要写100层循环？这时候我们可以使用递归的枚举方法来实现。这里难度可能较大，同学们可以自行选择了解。

```
#include<stdio.h>

int n, kind = 0, m1[10000][10], m2[10];

void peiliao(int total, int a) {
    // 计算正在枚举第a种材料且已经使用total克材料时的方案数量
    if (a == 10) {
        if (total == n) {
            for (int j = 0; j < 10; j++) m1[kind][j] = m2[j];//符合要求存起来~~
            kind++;
        }
    } else if (total >= n);//材料大于等于n克则这个方案一定不成立，不再继续递归调用
    else
        for (int i = 1; i <= 3; i++) {
            m2[a] = i;
            peiliao(total + i, a + 1);//其实这和十连for没什么区别。。。
        }
}

int main() {
    scanf("%d", &n);
    peiliao(0, 0);
    printf("%d\n", kind);
    for (int j = 0; j < kind; j++) {
        for (int i = 0; i < 10; i++) printf("%d ", m1[j][i]);
        printf("\n");
    }
    return 0;
}
// 整理自 https://www.luogu.com.cn/problem/solution/P2089 万枪先生 的题解
```

更多有关搜索算法的资料可以查阅 https://oi-wiki.org/search/

C - 计数问题

这道题只需要判断一下1到n的每一个数的每一位是否为x，是则总数加一。注意这里用到了 c%10 可以获得个位数的值，c/10 将十进制数右移了一位的技巧。

例如：

不断执行 c%10,c/10 可以有以下结果

```
123456789 9
 12345678 8
  1234567 7
   123456 6
    12345 5
     1234 4
      123 3
       12 2
        1 1
        0
```

题解

```
#include<stdio.h>

int main() {
    long long n, i, x, b, c, t = 0;
    scanf("%d %d", &n, &x);//输入范围与要查的数字；
    for (i = 1; i <= n; i++)//一到n进行循环；
    {
        b = i;//为了不改变i的值，就把i赋值给一个数；
        while (b != 0)//如果b不等于0，继续循环；
        {
            c = b % 10;//求是否是x，是的话计数器加一；
            b = b / 10;//求下一个数字是否为x；
            if (c == x) t++;//计数器加一；
        }
    }
    printf("%d\n", t);//输出计数器的数字；
    return 0;//结束
}
//整理自 https://www.luogu.com.cn/problem/solution/P1980 王超wangchao 的题解
```

D - 火柴棒等式

这道题看似无从下手，实际上可以转换成简单枚举的问题。由于一共最多只有24根火柴，其中必须有4根拿来组成加号和等号，剩下的20根组成等式 A+B=C 中的A, B, C 。显然其中 A, B, C 的值一定不会超过2000（实际上AB并不可能超过1000，C不可能超过2000，虽然结论很明显，但是证明稍复杂，这里留给同学们自己思考），这时候我们可以把2000内的每一个数所需要的火柴数量算出来，再枚举 A 和 B令其满足A+B=C且火柴总数为n即可。

题解

```
#include<stdio.h>

int count[2001];// 表示摆出i需要count[i]根火柴
int num[10] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6}; //摆出一个i需要num[i]根火柴，i为1位数
```

```
int main() {
    int n, sum = 0;
    scanf("%d", &n);
    count[0] = 6;
    for (int i = 1; i <= 2000; i++) {
        int j = i;
        while (j >= 1)//求每个数所用的火柴棒
        {
            count[i] += num[j % 10];
            j = j / 10;
        }
    }
    for (int i = 0; i <= 1000; i++) {
        for (int j = 0; j <= 1000; j++)
            //枚举 A 和 B 并计算出 C
            if (count[i] + count[j] + count[i + j] + 4 == n)sum++;
    }
    printf("%d\n", sum);
    return 0;
}
```

E - 求m区间内的最小值

这道题最直接最朴素的算法就是对于每个数，遍历其前 m 个数并找最小值，虽然这个算法是正确的，时间复杂度为$O(n*m)$。但是题目所给的 m 和 n 实在是太大了，m 和 n 可能会达到 10^6 量级，因此这样的算法可能会有高达 10^{12} 量级的运算量，这显然是不可接受且不可能在题目所给的1秒内运算出结果（一般评测机可以在1秒进行 10^8 量级左右的运算）。

上述算法进行了大量重复的工作，除了开头m-1个和结尾m-1个数外，每个数都进行了m次比较。这时候我们先引入一个新的概念「单调队列」。

顾名思义，单调队列有以下性质

- "单调" 指的是元素的的 "规律"——递增（或递减）
- "队列" 指的是元素只能从队头和队尾进行操作

题目要求的是每连续的m个数中的最小值，很明显，当一个数进入所要 "寻找" 最小值的范围中时，若这个数比其前面（先进队）的数要小，显然，前面的数会比这个数先出队且不再可能是最小值。也就是说——当满足以上条件时，可将前面的数 "弹出"，再将该数真正加入队尾。这就相当于维护了一个递增的队列，符合单调队列的定义，减少了重复的比较次数，不仅如此，由于维护出的队伍是查询范围内的且是递增的，队头必定是该查询区域内的最小值，因此输出时只需输出队头即可。显而易见的是，在这样的算法中，每个数只要进队与出队各一次，因此时间复杂度被降到了 $O(n)$。——参考 https://oi-wiki.org/ds/monotonous-queue/

例如：对于题目中的样例 7 8 1 4 3 2

操作	队列状态
7 入队	{7}
8 比 7 大，8 入队	{7, 8}
由于 7 已经超出范围，先将 7 从队头弹出，1 比 8 小，将 8 从队尾弹出，1 入队	{1}
4 比 1 大，4 入队	{1,4}
由于 1 已经超出范围，先将 1 从队头弹出，3 比 4 小，将 4 从队尾弹出，3 入队	{3}

题解代码

```
#include<stdio.h>

#define maxN 2000000
int q[maxN + 1][2] = {0}, n, m, head = 0, tail = 0;

////q[i][0]表示值， q[i][1]表示在原来顺序的位置
int main() {
    scanf("%d%d", &n, &m);
    printf("0\n"); ////第1个数前没有数，输出0
    scanf("%d", &q[tail][0]); //第1个数进队列
    tail++;
    for (int i = 1; i < n; i++) {
        if (i - q[head][1] > m) head++;
        printf("%d\n", q[head][0]); //输出队首元素（最小）
        int x;
        scanf("%d", &x);
        while (tail > head && x < q[tail - 1][0])
            tail--;
        q[tail][0] = x;
        q[tail][1] = i; //当前数进队列
        tail++;
    }
    return 0;
}
// 整理自 https://www.luogu.com.cn/problem/solution/P1440 Aehnuwx 的题解
```