

第3章 NOIP 2015—2020 普及组复试题解

从本章开始,一起看一下近几年的复试真题。考虑到书稿篇幅的问题,本书并不主张对每一个题目开展深入的探讨,以免限制读者的思维。比赛的题目往往会比较灵活,有较多的解题方法和途径,而在此仅提供一种可行的方法而已,希望读者尽展各自的思维,力求多角度、多方法求解,至于哪个方法是更好的,自然要依据算法评价准则及算法实现的难易度来综合评价。总之,本书的题解只求能起到一个抛砖引玉的作用。

3.1 NOIP 2015 普及组复试题解

3.1.1 T1: 金币

【题目描述】

国王将金币作为工资,发放给忠诚的骑士。第一天,骑士收到一枚金币;之后两天(第二天和第三天),每天收到两枚金币;之后三天(第四、五、六天),每天收到三枚金币;之后四天(第七、八、九、十天),每天收到四枚金币……这种工资发放模式会一直延续下去:当连续 N 天每天收到 N 枚金币后,骑士会在之后的连续 $N+1$ 天里,每天收到 $N+1$ 枚金币。

请计算在前 K 天里,骑士一共获得了多少金币。

【输入】

输入文件只有 1 行,包含一个正整数 K ,表示发放金币的天数。

【输出】

输出文件只有 1 行,包含一个正整数,即骑士收到的金币数。

【样例输入】

6

【样例输出】

14

【解析】

利用循环语句模拟。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int k, mon, ans;
int main(){
```

```
scanf("%d",&k);
mon = 1;
while (k){
    if (k >= mon){
        ans += mon * mon; k -= mon;
    }else{
        ans += mon * k; k = 0;
    }
    mon++;
}
printf("%d\n",ans);return 0;
}
```

3.1.2 T2: 扫雷游戏

【题目描述】

扫雷游戏是一款十分经典的单机小游戏。在 n 行 m 列的雷区中有一些格子含有地雷（称为地雷格），其他格子不含地雷（称为非地雷格）。玩家翻开一个非地雷格时，该格将会出现一个数字——提示周围格子中有多少个是地雷格。游戏的目标是在不翻出任何地雷格的条件下，找出所有的非地雷格。

现在给出 n 行 m 列的雷区中的地雷分布，要求计算出每个非地雷格周围的地雷格数。

注：一个格子的周围格子包括其上、下、左、右、左上、右上、左下、右下 8 个方向上与之直接相邻的格子。

【输入】

输入文件第一行是用一个空格隔开的两个整数 n 和 m ，分别表示雷区的行数和列数。

接下来 n 行，每行 m 个字符，描述了雷区中的地雷分布情况。字符 '*' 表示相应格子是地雷格，字符 '?' 表示相应格子是非地雷格。相邻字符之间无分隔符。

【输出】

输出文件包含 n 行，每行 m 个字符，描述整个雷区。用 '*' 表示地雷格，用周围的地雷个数表示非地雷格。相邻字符之间无分隔符。

【样例输入】

```
3 3
* ??
???
?* ?
```

【样例输出】

```
* 10
221
1 * 1
```

【解析】

循环遍历每个 '?', 对于每个 '?', 统计其四周 8 个点 '*' 的数量。8 个方向可以用数组预处理（参见代码），同时需要注意边界。

【参考代码】

```

#include <bits/stdc++.h>
using namespace std;
char s[105][105];
int n,m,a[105][105];
int dx[] = {0,0,1,-1,1,1,-1,-1};           //8个方向
int dy[] = {1,-1,0,0,1,-1,1,-1};
int main(){
    scanf("%d%d",&n,&m);
    for (int i = 1;i <= n;i++){              //整行读入字符串
        scanf("%s",s[i] + 1);
    }
    for (int i = 1;i <= n;i++){
        for (int j = 1;j <= m;j++){
            if (s[i][j] == '*'){
                printf("*");
            }
            else{
                int cnt = 0;
                for (int k = 0;k < 8;k++){
                    int x = i + dx[k], y = j + dy[k];
                    if (x >= 1 && x <= n && y >= 1 && y <= m && s[x][y] == '*'){
                        cnt++;
                    }
                }
                printf("%d",cnt);
            }
        }
        printf("\n");
    }
    return 0;
}

```

3.1.3 T3: 求和

【题目描述】

一条狭长的纸带被均匀划分出了 n 个格子(图 3-1), 格子的编号为 $1 \sim n$ 。每个格子上都染了一种颜色 $color_i$ (i 用 $[1, m]$ 区间中的一个整数表示), 并且写了一个数字 $number_i$ 。

	5	5	3	2	2	2
编号	1	2	3	4	5	6

图 3-1 样例输入数据的纸带

定义一种特殊的三元组 (x, y, z) , 其中, x, y, z 代表纸带上格子的编号, 这里的三元组要求满足以下两个条件。

- (1) x, y, z 是整数, $x < y < z$, $y - x = z - y$;
- (2) $color_x = color_z$ 。

满足上述条件的三元组的分数规定为 $(x + z) \times (number_x + number_z)$ 。整个纸带的分数规定为所有满足条件的三元组的分数的和。这个分数可能会很大, 只要输出整个纸带的分数除以 10007 所得的余数即可。

【输入】

第一行是用一个空格隔开的两个正整数 n 和 m , n 表示纸带上格子的个数, m 表示纸带上颜色的种类数。

第二行有 n 个用空格隔开的正整数, 第 i 个数字 $number$ 表示纸带上编号为 i 的格子上面写的数字。

第三行有 n 个用空格隔开的正整数, 第 i 个数字 $color$ 表示纸带上编号为 i 的格子染的颜色。

【输出】

共一行, 一个整数, 表示所求的纸带分数除以 10007 所得的余数。

【样例输入】

```
6 2
5 5 3 2 2 2
2 2 1 1 2 1
```

【样例输出】

```
82
```

【解析】

同种颜色, 下标都为奇数的可以两两产生分数, 下标都为偶数的可以两两产生分数, 所以在做题时, 可以根据下标奇偶分为两组分别计算即可。

接下来就要列计算式了。

例如, 对于奇数下标(共有 k 个)的某颜色 $(x_1, num_1), (x_2, num_2), \dots, (x_k, num_k)$, 得分为:

$$\begin{aligned} & (x_1 + x_2) \times (num_1 + num_2) + (x_1 + x_3) \times (num_1 + num_3) + \dots \\ & + (x_1 + x_k) \times (num_1 + num_k) + (x_2 + x_3) \times (num_2 + num_3) + \dots \\ & + (x_2 + x_k) \times (num_2 + num_k) + \dots + (x_{(k-1)} + x_k) \times (num_{(k-1)} + num_k) \end{aligned}$$

调整一下式子得到:

$$\begin{aligned} & (k-1) \times x_1 \times num_1 + x_1 \times (num_2 + num_3 + \dots + num_k) + \\ & (k-1) \times x_2 \times num_2 + x_2 \times (num_1 + num_3 + \dots + num_k) + \dots \\ & (k-1) \times x_k \times num_k + x_k \times (num_1 + num_2 + \dots + num_{(k-1)}) \end{aligned}$$

将每行第一部分取一个 $x_i \times num_i$ 调整到后面, 就为:

$$\begin{aligned} & (k-2) \times x_1 \times num_1 + x_1 \times (num_1 + num_2 + \dots + num_k) + \dots \\ & (k-2) \times x_2 \times num_2 + x_2 \times (num_1 + num_2 + \dots + num_k) + \dots \\ & (k-2) \times x_k \times num_k + x_k \times (num_1 + num_2 + \dots + num_k) \end{aligned}$$

提取公因式后:

$$\begin{aligned} & (k-2) \times (x_1 \times num_1 + x_2 \times num_2 + \dots + x_k \times num_k) + \\ & (x_1 + x_2 + \dots + x_k) \times (num_1 + num_2 + \dots + num_k) \end{aligned}$$

化简到这里, 基本上可以写程序了。

对于 100% 的数据, 其实 m 也不是很大, 可以用桶 $k[c]$ 记录第 $color$ 颜色的数量, $sum_{xn}[c]$ 记录第 $color$ 颜色下标 i 乘以数字 $num[i]$ 的累加和, $sum_x[c]$ 记录第 $color$ 颜色下标 i 的累加, $sum_n[c]$ 记录第 $color$ 颜色数字 $num[i]$ 的累加, 统计出来之后, 就可以直接运算出结果了。

【参考代码】

```

#include <bits/stdc++.h>
using namespace std;
const int M = 100005;
const int P = 10007;
int n, m, num[M], cor[M], ans;
int k1[M], sumxn1[M], sumx1[M], sumn1[M];
int k2[M], sumxn2[M], sumx2[M], sumn2[M];
int main() {
    scanf("%d %d", &n, &m);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &num[i]);
    }
    for (int i = 1; i <= n; i++) {
        scanf("%d", &cor[i]);
    }
    for (int i = 1; i <= n; i++) {
        if (i & 1) { // 奇数的情况
            int c = cor[i];
            k1[c]++;
            sumxn1[c] = (sumxn1[c] + i % P * num[i] % P) % P;
            sumx1[c] = (sumx1[c] + i) % P;
            sumn1[c] = (sumn1[c] + num[i]) % P;
        } else { // 偶数的情况
            int c = cor[i];
            k2[c]++;
            sumxn2[c] = (sumxn2[c] + i % P * num[i] % P) % P;
            sumx2[c] = (sumx2[c] + i) % P;
            sumn2[c] = (sumn2[c] + num[i]) % P;
        }
    }
    for (int i = 1; i <= m; i++) {
        if (k1[i] >= 2) ans = (ans + (k1[i] - 2) * sumxn1[i] + sumx1[i] * sumn1[i]) % P;
        if (k2[i] >= 2) ans = (ans + (k2[i] - 2) * sumxn2[i] + sumx2[i] * sumn2[i]) % P;
    }
    printf("%d\n", ans);
    return 0;
}

```

3.1.4 T4: 推销员

【题目描述】

阿明是一名推销员,他奉命到螺丝街推销他们公司的产品。螺丝街是一条死胡同,出口与入口是同一个,街道的一侧是围墙,另一侧是住户。螺丝街一共有 N 家住户,第 i 家住户到入口的距离为 S_i 米。由于同一栋房子里可以有多家住户,所以可能有多家住户与入口的距离相等。阿明会从入口进入,依次向螺丝街的 X 家住户推销产品,然后再原路走出去。

阿明每走 1 米就会积累 1 点疲劳值,向第 i 家住户推销产品会积累 A_i 点疲劳值。阿明是工作狂,他想知道,对于不同的 X ,在不走多余路的前提下,他最多可以积累多少点疲

劳值。

【输入】

第一行有一个正整数 N , 表示螺丝街住户的数量。

接下来的一行有 N 个正整数, 其中, 第 i 个整数 S_i 表示第 i 家住户到入口的距离。数据保证 $S_1 \leq S_2 \leq \dots \leq S_n < 10^8$ 。

接下来的一行有 N 个正整数, 其中, 第 i 个整数 A_i 表示向第 i 户住户推销产品会积累的疲劳值。数据保证 $A_i < 10^3$ 。

【输出】

输出 N 行, 每行一个正整数, 第 i 行整数表示当 $X=i$ 时, 阿明最多积累的疲劳值。

【样例输入】

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

【样例输出】

```
15
19
22
24
25
```

【解析】

本题可以用贪心算法解决。题目希望疲劳值最大, 可以按疲劳值从大到小排序, 随着 X 的不断增大, 希望可以有更多较大疲劳值的点加入, 而距离只要在这些点中取最远距离。

这里可以得出一个结论: 选取的 X 个点中, 至少有 $X-1$ 个点的疲劳值是取最大的前 $X-1$ 个值, 剩下一个选取距离 $\times 2 +$ 疲劳值最大的点。

可以这么理解: 如果取疲劳值前 X 大点中的 $X-2$ 个, 剩下两个点选非前 X 大的值, 那么剩下两个点中至少有一个点的距离是不用计算的, 这个点换成疲劳值更大的点, 会让答案更大一些。所以 X 个点中, 至少有 $X-1$ 个点的疲劳值是前 $X-1$ 大的, 剩下一个选择距离 $\times 2 +$ 疲劳值最大的点。答案要么选前 i 大疲劳值的点, 要么选前 $i-1$ 大疲劳值的点和距离 $\times 2 +$ 疲劳值最大的点。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int n, sum[100005], pre[100005], last[100005];
//sum[i]表示前 i 大疲劳值的前缀和
//pre[i]表示前 i 个点中距离 * 2 的最大值
//last[i]表示点 i~n 中距离 * 2 + 疲劳值的最大值
struct node{
    int d, v; //距离和疲劳值
}p[100010];
bool cmp(node x, node y)
{
```

```

    return x.v > y.v;
}
int main(){
    scanf("%d",&n);
    for (int i = 1; i <= n; i++){
        scanf("%d",&p[i].d);
    }
    for (int i = 1; i <= n; i++){
        scanf("%d",&p[i].v);
    }

    sort(p+1,p+1+n,cmp);                //按疲劳值从大到小排序
    for (int i = 1; i <= n; i++){
        sum[i] = sum[i-1] + p[i].v;
    }
    for (int i = 1; i <= n; i++){
        pre[i] = max(pre[i-1], 2 * p[i].d);    //前 i 个中最大距离
    }
    for (int i = n; i >= 1; i--){
        last[i] = max(last[i+1], 2 * p[i].d + p[i].v);    //后 i 个中最大值
    }
    for (int i = 1; i <= n; i++){
        printf("%d\n", max(sum[i] + pre[i], sum[i-1] + last[i]));
    }
    return 0;
}

```

3.2 NOIP 2016 普及组复试题解

3.2.1 T1: 买铅笔

【题目描述】

P 老师需要去商店买 n 支铅笔作为小朋友们参加 NOIP 的礼物。她发现商店一共有 3 种包装的铅笔,不同包装内的铅笔数量有可能不同,价格也有可能不同。公平起见,P 老师决定只买同一种包装的铅笔。

商店不允许将铅笔的包装拆开,因此 P 老师可能需要购买超过 n 支铅笔才够给小朋友们发礼物。

现在 P 老师想知道,在商店每种包装的数量都足够的情况下,要买够至少 n 支铅笔最少需要花费多少钱。

【输入】

第一行包含一个正整数 n ,表示需要的铅笔数量。

接下来 3 行中,每行用两个正整数描述一种包装的铅笔:其中,第 1 个整数表示这种包装内铅笔的数量,第 2 个整数表示这种包装的价格。

保证所有的 7 个数都是不超过 10000 的正整数。

【输出】

1 个整数,表示 P 老师最少需要花费的钱。

【样例输入】

```
57
2   2
50  30
30  27
```

【样例输出】

```
54
```

【解析】

枚举每个包装,判断一下 n 是不是当前这个包装数量的整倍数,如果是就直接用 $n/\text{包装数量} \times \text{每个的价钱}$,如果不是,就用 $(n/\text{包装数量} + 1) \times \text{单价}$,最后再判断最小值。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n,ans = 100000000,t;
    scanf("%d",&n);
    for (int i = 1;i <= 3;i++){
        int a,b;
        scanf("%d %d",&a,&b);
        t = ((n - 1)/a + 1) * b;
        ans = min(ans,t);
    }
    printf("%d\n",ans);
    return 0;
}
```

3.2.2 T2: 回文日期

【题目描述】

在日常生活中,通过年、月、日这三个要素可以表示出一个唯一确定的日期。

牛牛习惯用 8 位数字表示一个日期,其中,前 4 位代表年份,接下来 2 位代表月份,最后 2 位代表日期。显然,一个日期只有一种表示方法,而两个不同的日期的表示方法不会相同。

牛牛认为,一个日期是回文的,当且仅当表示这个日期的 8 位数字是回文的。现在牛牛想知道:在他指定的两个日期之间(包含这两个日期本身),有多少个真实存在的日期是回文的。

一个 8 位数字是回文的,当且仅当对于所有的 $i(1 \leq i \leq 8)$ 从左向右数的第 i 个数字和第 $9-i$ 个数字(即从右向左数的第 i 个数字)是相同的。

例如:

- 对于 2016 年 11 月 19 日,用 8 位数字 20161119 表示,它不是回文的。

- 对于 2010 年 1 月 2 日,用 8 位数字 20100102 表示,它是回文的。
- 对于 2010 年 10 月 2 日,用 8 位数字 20101002 表示,它不是回文的。

每一年中都有 12 个月份,其中,1、3、5、7、8、10、12 月每个月有 31 天;4、6、9、11 月每个月有 30 天;而对于 2 月,闰年时有 29 天,平年时有 28 天。

一个年份是闰年当且仅当它满足下列两种情况其中的一种:

- (1) 这个年份是 4 的整数倍,但不是 100 的整数倍。
- (2) 这个年份是 400 的整数倍。

例如,

- 以下几个年份都是闰年: 2000、2012、2016。
- 以下几个年份是平年: 1900、2011、2014。

【输入】

两行,每行包括一个 8 位数字。

第一行表示牛牛指定的起始日期。

第二行表示牛牛指定的终止日期。

保证 $date_i$ 都是真实存在的日期,且年份部分一定为 4 位数字,且首位数字不为 0。

保证 $date_1$ 一定不晚于 $date_2$ 。

【输出】

一个整数,表示在 $date_1$ 和 $date_2$ 之间有多少个日期是回文的。

【样例输入】

20110101

20111231

【样例输出】

1

【解析】

因为是回文,所以符合条件的日期满足后 4 位是前 4 位的逆序,可以枚举后 4 位,即枚举月份和日期,从而确定前 4 位,这样保证枚举的日期是回文日期,然后判断是否在题目给定的日期范围内即可。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int d1,d2,d,ans;
int md[13] = {0,31,29,31,30,31,30,31,31,30,31,30,31};
//预先设置每个月的天数,回文日期肯定是闰年
int main(){
    scanf("%d%d",&d1,&d2);
    for (int i = 1; i <= 12; i++){
        for (int j = 1; j <= md[i]; j++){
            int y = (j % 10) * 1000 + (j / 10) * 100 + (i % 10) * 10 + (i / 10);
            d = y * 10000 + i * 100 + j;
            if (d1 <= d && d <= d2){
                ans++;
            }
        }
    }
}
```

```

    }
}
printf("%d",ans);
return 0;
}

```

3.2.3 T3: 海港

【题目描述】

小 K 是一个海港的海关工作人员,每天都有许多船只到达海港,船上通常有很多来自不同国家的乘客。

小 K 对这些到达海港的船只非常感兴趣,他按照时间记录下了到达海港的每一艘船只情况;对于第 i 艘到达的船,他记录了这艘船到达的时间 t_i (单位: s),船上的乘客数 k_i ,以及每名乘客的国籍 $x_{i,1}, x_{i,2}, \dots, x_{i,k}$ 。

小 K 统计了 n 艘船的信息,希望你帮忙计算出以每一艘船到达时间为止的 $24h$ ($24h = 86400s$) 内所有乘船到达的乘客来自多少个不同的国家。

形式化地讲,你需要计算 n 条信息。对于输出的第 i 条信息,你需要统计满足 $t_i - 86400 < t_p \leq t_i$ 的船只 p ,在所有的 $x_{p,j}$ 中,总共有多少个不同的数。

【输入】

第一行输入一个正整数 n ,表示小 K 统计了 n 艘船的信息。

接下来 n 行中,每行描述一艘船的信息: 前两个整数 t_i 和 k_i 分别表示这艘船到达海港的时间和船上的乘客数量,接下来 k_i 个整数 $x_{i,j}$ 表示船上乘客的国籍。

保证输入的 t_i 是递增的,单位是 s; 表示从小 K 第一次上班开始计时,这艘船在第 t_i 秒到达海港。

【输出】

输出 n 行,第 i 行输出一个整数表示第 i 艘船到达后的统计信息。

【样例输入】

```

3
1 4 4 1 2 2
2 2 2 3
10 1 3

```

【样例输出】

```

3
4
4

```

【解析】

本题可以用队列+桶解决。每次输入直接把船上游客拆开用队列记录时间和国家,存入时如果发现这个国家编号都不与其他在队列中的国家编号相同, $ans+1$ 并记录国家编号出现次数(桶计数加 1),然后每次从当前队首枚举过来,队首元素时间在一天外(超过 86400s)就出队(桶计数减 1),如果出队的国家编号都不与其他在队列中的国家编号相同,