

蓝桥杯省赛知识点 & 精练（上）

蓝桥杯省赛知识点 & 精练（上）

4月9日蓝桥杯省赛正式开赛！！

☑ 蓝桥杯近 3 年常考知识点

📁 蓝桥杯近 5 年真题——知识点对应

📁 省赛一等奖学霸的备赛小贴士

1. 大神备考心得
2. 最后一周时间规划

一. 模拟、思维

1. 《小明的彩灯》

[题目链接](#)

[题目描述](#)

[输入描述](#)

[输出描述](#)

[题解代码](#)

2. 《绝世武功》

[题目链接](#)

[题目描述](#)

[输入描述](#)

[输出描述](#)

[题解代码](#)

二. 基本数据结构

1. 《蓝桥侦探》

[题目链接](#)

[题目描述](#)

[输出描述](#)

[题解代码](#)

三. 基础算法

1. 《哈曼夫树》（贪心法）

[题目链接](#)

[题目描述](#)

[输入描述](#)

[输出描述](#)

[题目分析](#)

四. 搜索

1. 《求立方根》

[题目链接](#)

[题目描述](#)

[输入描述](#)

[输出描述](#)

[题解代码](#)

2. 《走迷宫》

[题目链接](#)

[题目描述](#)

[输入描述](#)

[输出描述](#)

[题解代码](#)

五. 高级数据结构

1. 《百亿富翁》

[题目链接](#)



题目描述
输入描述
输出描述
题解代码

六. 动态规划

1. 《蓝桥骑士》（线性DP）

题目链接
题目描述
输入描述
输出描述
题解代码

2. 《骰子》（概率DP）

题目链接
题解代码

<下册> 知识点7、8、9、10内容正在制作中，敬请期待~

四个快速解题小技巧

技巧一：巧用编辑器

1. 蓝桥杯真题：门牌制作（2020年省赛）

题目描述
解题思路

2. 蓝桥杯真题：卡片（2021年省赛）

题目描述
解题思路

技巧二：眼看手数

1. 蓝桥杯真题：迷宫（2011年省赛）

题目描述
解题思路

2. 蓝桥杯真题：七段码（2020年省赛）

题目描述
解题思路

技巧三：巧用 Excel

1. 蓝桥杯真题：分数（2018年省赛）

题目描述
解题思路

2. 蓝桥杯真题：星期一（2018年省赛）

题目描述
解题思路

技巧四：巧用 Python

蓝桥杯真题：乘积尾零（2018年省赛）

题目描述
解题思路

用Python处理字符

蓝桥杯真题：平方和（2019年省赛）

题目描述
解题思路

试一试1：数列求值（2019年省赛）

题目描述

试一试2：数列求值

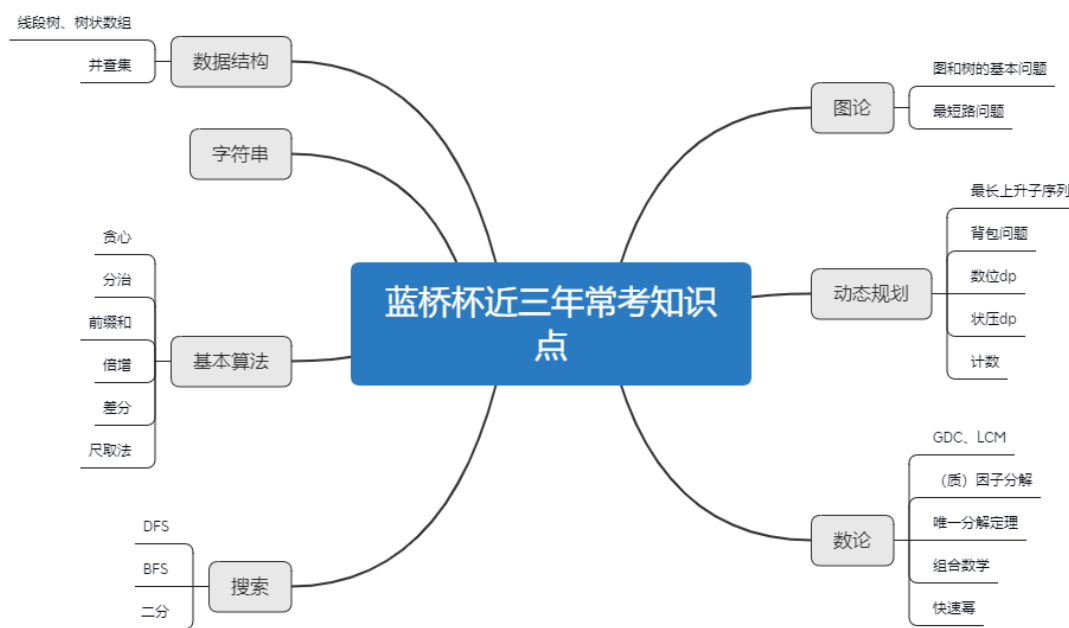
题目描述

实验总结



4月9日蓝桥杯省赛正式开赛！！！！

✓ 蓝桥杯近 3 年常考知识点



众所周知，算法竞赛以知识点的丰富和编码的高难度闻名。其实也不用太担心，算法竞赛的难度是分级、分阶段的，难题、罕见题并不多见。把学习重点放在基础算法、常见算法上，并用大量的实战练习提高编码能力，足以

在蓝桥杯省赛上获得一等奖。

算法竞赛的题目分为这五个大类：杂题、数据结构、基础算法、搜索、动态规划、数学、字符串、图论、几何等。

其中的「杂题」不需要什么算法和数据结构，只考察思维和编码能力，不过，杂题也可能很难。



蓝桥杯近 5 年真题——知识点对应

下面为大家盘点 2017~2021 年度蓝桥杯省赛的知识点。以 C++ A 组为例，其他组别的题目类似，重合的题目很多。

知识点	难度	题目
模拟、思维		2017-10, 2018-10、2019-3、2019-7、2020-3、2020-6、2020-7
基本数据结构	*	二叉树 (2019-6)
基础算法	*	枚举 (2018-5)、差分 (2018-7)、倍增
	**	二分法 (2017-9)
搜索	*	DFS (2017-1、2017-4、2017-7)、BFS (2017-2、2018-8、2019-4)
高级数据结构	*	并查集 (2019-8、2020-4)、线段树
动态规划	*	线性 DP (2017-5, 2017-6、2017-8、2020-10)、记忆化搜索、搜索 (2021)
	**	状态压缩 DP (2019-9)
简单数学		2018-1, 2018-2, 2018-3、2018-4、2019-1、2019-2、2020-1、2020-5
数论	*	余数 (2018-9)、GCD (2017-8, 2020-2) 枚举 (2021)
组合数学	**	卢卡斯定理 (2019-10)
	***	burnside 引理 (2017-3)
其他	*	快速幂 (2019-5)
字符串	*	简单字符串处理 (2018-6、2019-1、2020-8)
图论	*	最短路 BFS (2019-4)
计算几何	*	叉积、面积 (2020-9) (2021)



另外我也统计过前些年蓝桥杯题目，除了上述知识点外，还有一些一星或二星的，例如倍增、线段树等。从上面的统计表格可以看到，省赛涉及的知识点比较基础，考核的是基本的算法思维、基本算法、编码能力。要成功参赛，最重要的是通过大量做基础题目，培养计算思维，提高快速和准确的编码能力。

有一些**几乎是必考的**，因为它们也是整个算法竞赛知识库的基础：

1. 模拟题。不需要算法和数据结构，只需要逻辑、推理的题目，难度可难可易。考察思维能力和编码能力，只能通过大量做题来提高。
2. BFS 搜索和 DFS 搜索，也就是暴搜。这是基本的算法，是基础中的基础。
3. 动态规划，特别是比较简单的线性 DP。
4. 简单数学和简单数论。
5. 简单的字符串处理、输入输出。
6. 基本算法，例如二分法、倍增、差分。
7. 基本数据结构。队列、栈、链表等。

省赛一等奖学霸的备赛小贴士

1. 大神备考心得

通过对蓝桥杯近三年真题的研究，我总结了以下几点极为关键的信息：

1. 蓝桥杯比赛是可以提前入场的。在这段时间内你可以将所需要的模板敲出来并保存。
2. 近年填空题的第一题常考一些不涉及算法的计算机基础问题（例如单位换算），赛前可抽空看看计算机相关基础问题。
3. 填空题不一定需要编程才能得到答案。有时候利用如 Excel、计算器、日历表等工具甚至是更优的选择。
4. 蓝桥杯的命题很有规律，总爱围绕一些知识标签、题型命题（如动态规划）。赛前花点时间整理这
5. 题目的难度不一定是逐级递增的，后面的题目也许比前面的题更好拿分。
6. 蓝桥杯类似 oi 制度，你在提交了程序之后是无法得知答案的正确与否。所以在提交之前，可以先手造几组测试样例对程序进行验证。
7. 压轴题的难度越来越高，如果不是为了冲击省排名，可以减少压轴题花费的时间来检查其它问题。

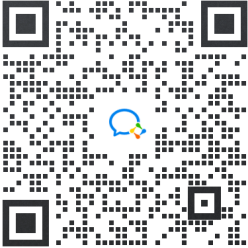
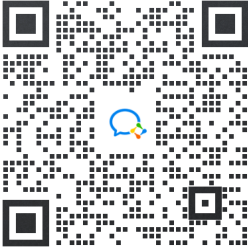
2. 最后一周时间规划

- 最后一周往往是最为关键的一周。如果此前你没有过任何参赛经验，那么我建议你：

1. 花一天时间去整理蓝桥杯的题型、考点、比赛时间、比赛规则等信息。



扫码添加 ▼ 蓝桥杯赛程答疑群

Java组	Python组	C/C++组
		

2. 花一天时间翻读蓝桥杯真题。

近3年蓝桥杯真题解析（官方出品）

组别	真题解析链接
Java-A组:	https://www.lanqiao.cn/courses/5031/?from=djbd321
Java-B组:	https://www.lanqiao.cn/courses/5032/?from=djbd321
Java-C组:	https://www.lanqiao.cn/courses/5033/?from=djbd321
Python:	https://www.lanqiao.cn/courses/5034/?from=djbd321
C/C++-A组:	https://www.lanqiao.cn/courses/5028/?from=djbd321
C/C++-B组:	https://www.lanqiao.cn/courses/5029/?from=djbd321
C/C++-C组:	https://www.lanqiao.cn/courses/5030/?from=djbd321

3. 克隆一场省赛，并花费四个小时模拟参赛。[蓝桥杯省赛14天夺奖冲刺营](#)

4. 针对第一天整理出的题型、考点进行专题训练，具体如下：

- 对于每个考点，每天挑选两道经典问题刷。
- 在刷题结束后，对会做、不会做的题目都进行一波总结。

5. 调整好心态！

- 如果此前你有过参赛经验，那么我建议你：

1. 翻读历年真题。
2. 进行一些算法思想的训练，如动态规划、构造、贪心等。
3. 调整好心态！



下面列出了除「杂题」外几乎所有的算法竞赛知识点，并按知识点本身的难度分成一星、二星、三星。读者应努力掌握一星、二星知识点。

注意知识点的难度和题目的难度并不一定完全对应，简单的知识点也可能出难题。

并且根据每一知识点配套了1-2道精选习题，下面我们就来根据知识点开启精练（上）吧。

一. 模拟、思维

1. 《小明的彩灯》

题目链接

<https://www.lanqiao.cn/problems/1276/learning>

题目描述

小明拥有 N 个彩灯，第 i 个彩灯的初始亮度为 a_i 。

小明将进行 Q 次操作，每次操作可选择一段区间，并使区间内彩灯的亮度 $+x$ (x 可能为负数)。

求 Q 次操作后每个彩灯的亮度 (若彩灯亮度为负数则输出 0)。

输入描述

第一行包含两个正整数 N, Q ，分别表示彩灯的数量和操作的次数。

第二行包含 N 个整数，表示彩灯的初始亮度。

接下来 Q 行每行包含一个操作，格式如下：

$l\ r\ x$ ，表示将区间 $l \sim r$ 的彩灯的亮度 $+x$ 。

$1 \leq N, Q \leq 5 \times 10^5, 0 \leq a_i \leq 10^9, 1 \leq l \leq r \leq N, -10^9 \leq x \leq 10^9$

输出描述

输出共 1 行，包含 N 个整数，表示每个彩灯的亮度。

题解代码

定义 $b_1 = a_1; b_i = a_i - a_{i-1}, i \geq 2$; 那么 b 就是 a 的差分数组，假设现在 $a_l - a_r$ 同时加上 x , 那么 b 数组的变化是什么样的呢？

简单的模拟一下发现 b_{r+1} 加上了一 x , b_l 加上了 x , b 数组就只修改了两个数，那么我在 b 数组上修改怎么回到 a 数组呢？根据 b 数组的定义，我们发现其实 a 就是 b 的前缀和数组，也就是在 b 上修改完对 b 进行前缀和就可以得到修改完的 a 数组

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N = 5e5 + 10;
int a[N] , ans[N];
signed main()
{
    int n , q ;
    cin >> n >> q;
    for(int i = 1 ; i <= n ; i ++ ) cin >> ans[i];
    while(q --)
    {
        int l , r , x;
        cin >> l >> r >> x;
        a[l] += x , a[r + 1] -= x;
    }
    for(int i = 1 ; i <= n ; i ++ ) a[i] += a[i - 1];
    for(int i = 1 ; i <= n ; i ++ ) cout << max(a[i] + ans[i] , 0ll) << " \n"[i == n];
    return 0;
}
```

2. 《绝世武功》

题目链接

<https://www.lanqiao.cn/problems/1368/learning/>

题目描述

小明在练习绝世武功， n 个练功桩排成一排，一开始每个桩的损伤为 0。

接下来小明会练习 m 种绝世武功，每种武功都会对 $[l, r]$ 区间分别造成 $[s, e]$ 的伤害。

这个伤害是一个等差序列。例如 $l = 1, r = 4, s = 2, e = 8$ ，则会对 1 - 4 号练功桩造成 2, 4, 6, 8 点损伤。

小明想让你统计一下所有练功桩的损伤的和。

输入描述

第一行输入 n, m ，代表练功桩的数量和绝世武功的种类数。

接下来 m 行输入 4 个整数 l, r, s, e 。

$1 \leq n \leq 10^7, 1 \leq m \leq 3 \times 10^5, 1 \leq l, r \leq n$

输出描述

输出一个整数代表所有练功桩的损伤和，题目保证所有输入输出都在 $[0, 9 \times 10^{18}]$



题解代码

设 a 是原数组， b 是一阶差分数组， c 是二阶差分数组。

考虑一次在 $[L, R]$ 内加上一个首项是 s ，公差是 d ，末项 $t = s + (R - L) * d$ 的等差数列对 a 数组的影响

$$a_i = a_i + s + (i - L) \times d, i \in [L, R]$$

研究 a 数组的变化对 b 数组的影响，由 $b_i = a_i - a_{i-1}$

$$b_L = (a_L + s) - a_{L-1} = b_L + s$$

$$b_i = a_i + (i - L) \times d - (a_{i-1} + (i - 1 - L) \times d) = b_i + d, i \in [L + 1, R]$$

$$b_{R+1} = a_{R+1} - (a_R + t) = b_{R+1} - t$$

再考虑 b 数组对 c 数组的影响

$$c_L = b_L + s - b_L = c_L + s$$

$$c_{L+1} = (b_{L+1} + d) - (b_L + s) = c_L + d - s$$

$$c_i = (b_i + d) - (b_{i-1} + d) = c_i, i \in [L + 2, R]$$

$$c_{R+1} = (b_{R+1} - t) - (b_R + d) = c_{R+1} - d - t$$

$$c_{R+2} = b_{R+2} - (b_{R+1} - t) = c_{R+2} + t$$

从上面可以看出每组 (L, R, s, d) 在 c 数组的影响只需要修改四个数字，然后将 c 数组做两次前缀和就可以得到 a 数组。


```
#include<bits/stdc++.h>
#define N 10000010
using namespace std;
int n,m,l,r,s,e,d;
long long a[N],b[N],c[N],maxx,yh;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
        scanf("%d%d%d%d",&l,&r,&s,&e); //不用读入优化，本蒟蒻卡着456ms过的
        d=(e-s)/(r-l); //公差
        c[l]=c[l]+s; //K大佬上面写了
        c[l+1]=c[l+1]+d-s;
        c[r+1]=c[r+1]-d-e;
        c[r+2]=c[r+2]+e;
    }
    for(int i=1;i<=n;i++){
        b[i]=b[i-1]+c[i]; //c[i]=b[i]-b[i-1]
        a[i]=a[i-1]+b[i]; //b[i]=a[i]-a[i-1]
        yh += a[i];
    }
    printf("%lld",yh);
    return 0;
}
```

二. 基本数据结构



难度	知识点
*	链表、队列、栈、堆、哈希、二叉树
**	单调队列、单调栈、排序（冒泡排序、交换排序、快速排序、归并排序、基尔排序）

1. 《蓝桥侦探》

题目链接

<https://www.lanqiao.cn/problems/1136/learning/>

题目描述

小明是蓝桥王国的侦探。

这天，他接收到一个任务，任务的名字叫分辨是非，具体如下：

蓝桥皇宫的国宝被人偷了，犯罪嫌疑人锁定在 N 个大臣之中，他们的编号分别为 $1 \sim N$ 。

在案发时这 N 个大臣要么在大厅1，要么在大厅2，但具体在哪个大厅他们也不记得了。

审讯完他们之后，小明把他们的提供的信息按顺序记了下来，一共 M 条，形式如下：

- $x \sim y$, 表示大臣 x 提供的信息, 信息内容为: 案发时他和大臣 y 不在一个大厅。

小明喜欢按顺序读信息, 他会根据信息内容尽可能对案发时大臣的位置进行编排。

他推理得出第一个与先前信息产生矛盾的信息提出者就是偷窃者, 但推理的过程已经耗费了他全部的脑力, 他筋疲力尽的睡了过去。作为他的侦探助手, 请你帮助他找出偷窃者!

###输入描述

第 1 行包含两个正整数 N, M , 分别表示大臣的数量和口供的数量。

之后的第 $2 \sim M + 1$ 行每行输入两个整数 x, y , 表示口供的信息。

$1 \leq N, M \leq 5 \times 10^5, 1 \leq x, y \leq N$ 。

输出描述

输出仅一行, 包含一个正整数, 表示偷窃者的编号。

题解代码

普通的并查集维护的关系是: 朋友的朋友是朋友, 即如果 $A \sim B$ 是一对朋友, $B \sim C$ 是一对朋友, 那么 $A \sim C$ 是一对朋友。但如果我们需要维护这样一个关系“朋友的朋友是朋友, 朋友的敌人是敌人, 敌人的敌人是朋友”, 普通的并查集就无能为力了。这题也是如此, 普通的并查集只能维护多少人是 在一个大厅, 但是题目给的是不在一个大厅, 并且只有俩个大厅, 那么就要维护 和同一个人不在一个大厅的俩人在一个大厅, 就不太行了, 因此, 需要引入种类并查集。

种类并查集又叫做扩展域并查集, 也就是我们扩展出一个域 $i + n$, 作为 i 号的点的敌人, 那么同时和 $i + n$ 相连的点就是 i 号点的敌人, 那么他们之间也就是朋友, 这样就可以维护 “敌人的敌人是朋友” 这类关系了。



```
#include<bits/stdc++.h>
using namespace std;
const int N = 5e5 + 10;
int n, m, far[N << 1];
int find(int x)
{
    if(x == far[x]) return x;
    return far[x] = find(far[x]);
}
void Union(int x, int y)
{
    int tx = find(x), ty = find(y);
    if(tx != ty) far[tx] = ty;
}
signed main()
{
    int res = 0, pos = 0;
    cin >> n >> m;
    for(int i = 1; i <= 2 * n; i++) far[i] = i;
    for(int i = 1; i <= m; i++)
    {
        int x, y;
        cin >> x >> y;
        if(res) continue;
        {
            if(find(x) == find(y) || find(x + n) == find(y + n)) res = x, pos =
i;
```

```

        Union(x , y + n) , Union(x + n , y);
    }
}
cout << res << '\n';
return 0;
}

```

三. 基础算法

难度	知识点
*	打表、枚举、倍增、离散化、差分
**	分治法、贪心法（Huffman 编码）、尺取法、二分法、三分法、整体二分、ST 算法

1. 《哈曼夫树》（贪心法）

题目链接

<https://www.lanqiao.cn/problems/1228/learning/>



题目描述

小明买了 n 件白色的衣服，他觉得所有衣服都是一种颜色太单调，希望对这些衣服进行染色，每次染色时，他会将某种颜色的**所有**衣服寄去染色厂，第 i 件衣服的邮费为 a_i 元，染色厂会按照小明的要求将其一部分衣服染成同一种任意的颜色，之后将衣服寄给小明，请问小明要将 n 件衣服染成不同颜色的最小代价是多少？

输入描述

第一行为一个整数 n ，表示衣服的数量。

第二行包括 n 个整数 a_1, a_2, \dots, a_n 表示第 i 件衣服的邮费为 a_i 元。

$(1 \leq n \leq 10^5, 1 \leq a_i \leq 10^9)$

输出描述

输出一个整数表示小明所要花费的最小代价。

题目分析

将染色过程倒置可以看成两种不同颜色衣服染成同一颜色，代价为两种颜色全部衣服的邮费和，这个合并过程最优解每次选取权值最小的两种颜色衣服合并，这个过程类似哈夫曼树建树过程，合并过程可以用优先队列来模拟。

```
#include<bits/stdc++.h>
```

```

using namespace std;
typedef long long ll;
const int maxn = 1e5 + 50;
int main()
{
    std::ios::sync_with_stdio(false);
    int n;
    cin >> n;
    priority_queue<ll, vector<ll>, greater<ll> > q;
    for(int i = 0; i < n; i++){
        ll x;
        cin >> x;
        q.push(x);
    }
    ll ans = 0;
    while(q.size() > 1){
        ll a = q.top(); q.pop();
        ll b = q.top(); q.pop();
        ll c = a + b;
        ans += c;
        q.push(c);
    }
    cout << ans << endl;
    return 0;
}

```



四. 搜索

难度	知识点
*	基本 DFS、基本 BFS
**	DFS 记忆化搜索、IDA*、BFS 扩展（双向广搜、优先队列、双端队列）、剪枝、爬山算法、随机增量法、模拟退火
***	A*

1. 《求立方根》

show: step
version: 1.0
enable_checker: true

题目链接

<https://www.lanqiao.cn/problems/1217/learning/>

题目描述

给定一个正整数 N ，请你求 N 的立方根是多少。

输入描述

第 1 行为一个整数 T ，表示测试数据数量。

接下来的 T 行每行包含一个正整数 N 。

$1 \leq T \leq 10^5$, $0 \leq N \leq 10^5$ 。

输出描述

输出仅一行，包含一个整数表示答案（答案保留 3 位有效数字）。

题解代码

常见的二分搜索答案，二分的前提是要二分的东西必须具有单调性，很明显立方根是单调递增的。

考虑二分立方根 x ，当 $x^3 < N$ 的时候把二分的区间向左缩小，否则向右缩小区间。

由于题目只要三位有效数字，所以我们实数二分的时候，当当前区间小于某个范围(eps)，就退出搜索。

```
#include<bits/stdc++.h>
using namespace std;
#define double long double
const double eps = 1e-12;
signed main()
{
    int T = 1;
    cin >> T;
    while(T --)
    {
        double n;
        bool flag = 0;
        cin >> n;
        double l = 0 , r = 100000 , res = -1;
        while(l <= r)
        {
            double mid = (l + r) / 2;
            if(fabs(mid * mid * mid - n) <= eps)
            {
                res = mid;
                break ;
            }
            if(mid * mid * mid > n) r = mid - 0.0001;
            else if(mid * mid * mid < n) l = mid + 0.0001 , res = mid;
        }
        cout << setprecision(3) << fixed << res << '\n';
    }
    return 0;
}
```



```
}
```

2. 《走迷宫》

show: step
version: 1.0
enable_checker: true

题目链接

<https://www.lanqiao.cn/problems/1216/learning/>

题目描述

给定一个 $N \times M$ 的网格迷宫 G 。 G 的每个格子要么是道路，要么是障碍物（道路用 1 表示，障碍物用 0 表示）。

已知迷宫的入口位置为 (x_1, y_1) ，出口位置为 (x_2, y_2) 。问从入口走到出口，最少要走多少个格子。

输入描述

输入第 1 行包含两个正整数 N, M ，分别表示迷宫的大小。

接下来输入一个 $N \times M$ 的矩阵。若 $G_{i,j} = 1$ 表示其为道路，否则表示其为障碍物。

最后一行输入四个整数 x_1, y_1, x_2, y_2 ，表示入口的位置和出口的位置。

$1 \leq N, M \leq 10^2$, $0 \leq G_{i,j} \leq 1$, $1 \leq x_1, x_2 \leq N$, $1 \leq y_1, y_2 \leq M$ 。

输出描述

输出仅一行，包含一个整数表示答案。

若无法从入口到出口，则输出 -1 。

题解代码

本题是最常见的广度优先搜索。

搜索算法过程如下：

- 将起始位置加入队列
- 从队列开头取出一个元素，将这个元素四周可到达并且未被访问的点加入队列

因为队列先进先出的特性，只有当所有距离为 1 的点被取出队列后，才会遍历距离为 2 的点，而距离为 1 的点能到达的只有距离为 2 的点，这也就保证这搜索答案的正确性。

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e2 + 10;
int n , m , g[N][N] , vis[N][N];
int xx , yy , x2 , y2;
int nex[4][2] = {{1 , 0} , {-1 , 0} , {0 , 1} , {0 , -1}};
```

```

struct node{
    int a , b , c;
};
bool check(int x , int y)
{
    if(x < 1 || x > n || y < 1 || y > m || !g[x][y] || vis[x][y]) return false;
    return true;
}
int bfs(int x , int y)
{
    queue<node>que;
    que.push(node{x , y , 0});
    vis[x][y] = 1;
    while(!que.empty())
    {
        node u = que.front();
        que.pop();
        if(u.a == x2 && u.b == y2) return u.c;
        for(int i = 0 ; i <= 3 ; i ++){
            int tx = nex[i][0] + u.a;
            int ty = nex[i][1] + u.b;
            if(!check(tx , ty)) continue ;
            vis[tx][ty] = 1;
            que.push(node{tx , ty , u.c + 1});
        }
    }
    return -1;
}
signed main()
{
    cin >> n >> m;
    for(int i = 1 ; i <= n ; i ++){
        for(int j = 1 ; j <= m ; j ++){
            cin >> g[i][j];
        }
    }
    cin >> xx >> yy >> x2 >> y2;
    cout << bfs(xx , yy) << '\n';
    return 0;
}

```



五. 高级数据结构

难度	知识点
*	并查集（带权）、分块
**	莫队算法（树上莫队）、树状数组、线段树、可持久化线段树、二叉搜索树、treap 树、splay 树、块状链表
***	替罪羊树、LCT、树套树、猫树、CDQ 分治、舞蹈链、左偏树、后缀平衡树、KDtree

1. 《百亿富翁》

show: step
version: 1.0
enable_checker: true

题目链接

<https://www.lanqiao.cn/problems/1142/learning/>

题目描述

这天小明买彩票中了百亿奖金，兴奋的他决定买下蓝桥公司旁的一排连续的楼房。

已知这排楼房一共有 N 栋，编号分别为 $1 \sim N$ ，第 i 栋的高度为 h_i 。

好奇的小明想知道对于每栋楼，左边第一个比它高的楼房是哪个，右边第一个比它高的楼房是哪个（若不存在则输出 -1 ）。但由于楼房数量太多，小明无法用肉眼直接得到答案，于是他花了 1 个亿来请你帮他解决问题，你不会拒绝的对吧？

输入描述

第 1 行输入一个整数 N ，表示楼房的数量。

第 2 行输入 N 个整数（相邻整数用空格隔开），分别为 h_1, h_2, \dots, h_N ，表示楼房的高度。

$1 \leq N \leq 7 \times 10^5$, $1 \leq h_i \leq 10^9$ 。



输出描述

输出共两行。

第一行输出 N 个整数，表示每栋楼左边第一栋比自己高的楼的编号。

第二行输出 N 个整数，表示每栋楼右边第一栋比自己高的楼的编号。

题解代码

单调栈，就是一个栈，不过栈内元素保证单调性。即，栈内元素要么从小到大，要么从大到小。

如何求右边第一个比它高的楼房？

通过观察，我们会发现，在当前点后面的，比它矮的，一定会被它遮住。那么，这个点就没用了。而根据现实生活和刚才的推断，我们看到的楼房肯定是一个比一个高的，而没看到的，留着也没用，直接扔了QwQ。那么，这就是符合单调性的。再看，那些没用的楼房什么时候扔掉？当然是遇到比它高的楼房了。那么就可以一个一个地弹出，而且肯定是在已经判断过的人的前面（中间和后面的在之前就弹出了），所以就直接从前面弹出。咦？这不就像一个栈吗？没错，这就是单调栈的实现方法。

再归纳一下：

- 从后往前扫
- 对于每个点：
 - 弹出栈顶比它小的元素
 - 此时栈顶就是答案
 - 加入这个元素

由于是从前往后输出，还要把答案放到一个数组里。

左边第一栋比自己高同理。

```
#include<bits/stdc++.h>
using namespace std;
const int N = 3e6 + 10;
stack<int>st;
int n , a[N] , l[N] , r[N];
signed main()
{
    cin >> n;

    for(int i = 1 ; i <= n ; i ++ ) cin >> a[i];
    for(int i = 1 ; i <= n ; i ++ )
    {
        while(st.size() && a[st.top()] <= a[i]) st.pop();
        if(st.size()) l[i] = st.top();
        else l[i] = -1;
        st.push(i);
    }
    while(!st.empty()) st.pop();
    for(int i = n ; i >= 1 ; i -- )
    {
        while(st.size() && a[st.top()] <= a[i]) st.pop();
        if(st.size()) r[i] = st.top();
        else r[i] = -1;
        st.push(i);
    }

    for(int i = 1 ; i <= n ; i ++ ) cout << l[i] << " \n"[i == n];
    for(int i = 1 ; i <= n ; i ++ ) cout << r[i] << " \n"[i == n];
    return 0;
}
```



六. 动态规划

难度	知识点
*	DP 问题的性质（重叠子问题、最优子结构、无后效性）、编码方法（记忆化递归、递推）、滚动数组；
	常见线性 DP 问题：0/1 背包问题、分组背包、多重背包、最长公共子序列（LCS）、最长递增子序列（LIS）、编辑距离、最小划分、行走问题、矩阵最长递增路径、子集和问题、矩阵链乘法、布尔括号
**	问题区间 DP、状态压缩 DP、树形 DP、数位 DP、计数类 DP、概率 DP
***	插头 DP、基环树 DP；
	DP 优化：数据结构优化、单调队列优化、斜率优化、分治优化、四边形不等式优化

1. 《蓝桥骑士》（线性DP）

show: step
version: 1.0
enable_checker: true

题目链接

<https://www.lanqiao.cn/problems/1188/learning>

题目描述

小明是蓝桥王国的骑士，他喜欢不断突破自我。

这天蓝桥国王给他安排了 N 个对手，他们的战力值分别为 a_1, a_2, \dots, a_n ，且按顺序阻挡在小明的前方。对于这些对手小明可以选择挑战，也可以选择避战。

身为高傲的骑士，小明从不走回头路，且只愿意挑战战力值越来越高的对手。

请你算算小明最多会挑战多少名对手。

输入描述

输入第一行包含一个整数 N ，表示对手的个数。

第二行包含 N 个整数 a_1, a_2, \dots, a_n ，分别表示对手的战力值。

$1 \leq N \leq 3 \times 10^5$, $1 \leq a_i \leq 10^9$ 。

输出描述

输出一行整数表示答案。

题解代码

这是一个经典问题，由于只要求长度，可以考虑维护一个单调递增的序列来优化。

我们可以维护一个单调递增的序列，每当当前元素比栈顶元素大，就直接压入栈，否则我们就在这序列中二分找到第一个比它大的元素将其替换，最后序列长度就是最长上升子序列长度，但是不保证这个递增序列就是最长上升子序列。

为什么这样贪心是可行的呢，不难证明，由于当前值比二分值小，并且当前值的位置在二分的值的后面，那么二者只能取其一，我比你小，就意味着后面可以容纳更多可能的元素，也就是为后面最优解更新创造条件。

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 3e5 + 50;
int a[maxn];
int L[maxn];
int LIS(int n)
{
```



```

L[0] = a[0];
int length = 1;
for(int i = 1; i < n; i++){
    if(L[length - 1] < a[i]) L[length++] = a[i];
    else *lower_bound(L, L + length, a[i]) = a[i];
}
return length;
}
int main()
{
    int n;
    cin >> n;
    for(int i = 0; i < n; i++) cin >> a[i];
    cout << LIS(n) << endl;
}

```

2. 《骰子》（概率DP）

show: step
version: 1.0
enable_checker: true

题目链接

<https://www.lanqiao.cn/problems/1320/learning/>



题解代码

概率 dp 和其他的 dp 有些不同，区别是其他的 dp 通常是顺着推，而概率 dp 有的时候需要逆推，具体要看给的条件，

例如这个题，很明显用 $dp[i]$ 表示已经扔到了 i 个面，还需要扔的期望次数，而可以得出 $dp[n] = 0$ ，所以这个题就需要逆推

求解时考虑 $dp[i]$ 是怎么来的，有可能是扔到了已经扔过的面或者是扔到了没有扔到过的面，就这两种情况：

- 扔到已经扔过的面求出的期望就是 $\frac{i}{n} \times dp[i]$
- 扔到没扔过的面求出的期望就是 $\frac{n-i}{n} \times dp[i+1]$

而无论是哪种情况，期望次数都要比原来多1，所以总的递推式就是：

$$dp[i] = \frac{i}{n} \times dp[i] + \frac{n-i}{n} \times dp[i+1] + 1.$$

化简时把 $dp[i]$ 挪到一边后得到：

$$dp[i] = dp[i+1] + \frac{n}{n-i}$$

```

#include<cstdio>
#include<cstdlib>
#include<iostream>
#include<cstring>

```

```
#include<memory>
using namespace std;
double dp[2000];
int main()
{
    int T,i,j,n;
    scanf("%d",&T);
    while(T--){
        scanf("%d",&n); dp[n]=0;
        for(i=n-1;i>=0;i--) dp[i]=dp[i+1]+1.0*n/(n-i);
        printf("%.2f\n",dp[0]);
    }
    return 0;
}
```

<下册> 知识点7、8、9、10内容正在制作中， 敬请期待~

四个快速解题小技巧

喜闻乐见的「手算题」



手算题的介绍

接下来介绍比赛时让人喜闻乐见的「手算题」，不用编码或编码极为简单的题！

每次比赛都有「送分题」，只需要几分钟就能做出来。特别是部分填空题，只需要填答案，不用提交代码，那么可

以用包括编码在内的多种方法。编码一般比较慢，所以能不编码就不要编码，而是用推理和手算找到答案。

这种不用编码的填空题称为「手算题」。

竞赛的时间极为紧张，应选用最快的实现方式。能节省几分钟就节省几分钟，更重要的是，如果能「投机取巧」尽

快搞出答案，会让人感觉特别爽！然后带着愉快的心情开始做后面的难题。

下面介绍 4 种小技巧：巧用编辑器、眼看手数、巧用 Excel、巧用 Python。

有同学担心比赛机器上没有 Excel、Python 软件。不用担心，这些软件都是标配，比赛机器上肯定有。

知识点

- 巧用编辑器
- 眼看手数
- 巧用 Excel
- 巧用 Python

技巧一：巧用编辑器

首先我们来看 2 道能通过巧用编辑器解决的省赛真题。

1. 蓝桥杯真题：门牌制作（2020年省赛）

[在线刷题](#)

题目描述

小蓝要为一条街的住户制作门牌号。

这条街一共有 2020 位住户，门牌号从 1 到 2020 编号。

小蓝制作门牌的方法是先制作 0 到 9 这几个数字字符，最后根据需要将字符粘贴到门牌上，例如门牌 1017 需要依

次粘贴字符 1、0、1、7，即需要 1 个字符 0，2 个字符 1，1 个字符 7。

请问要制作所有的 1 到 2020 号门牌，总共需要多少个字符 2？

解题思路

这确实是个送分题，编码也很简单：判断每个数字中有几个 2，然后把所有数字中 2 的个数加起来。编码大概 5 分钟。

但是有更简单的做法：先编码打印出 1~2020 这 2020 个数字。大家可以在线上环境中新建 door.cpp，贴入以下

代码：

```
#include<bits/stdc++.h>

using namespace std;

int main(){
    int k=0;
    for(int i=1;i<=2020;i++){
        cout<<i;
    }
```



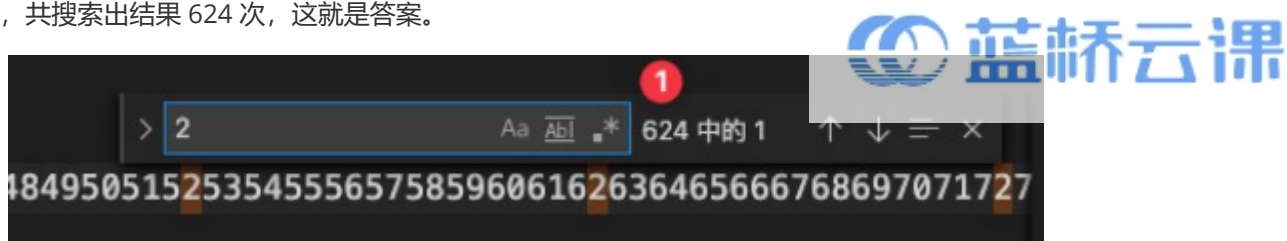
```
资源管理器
> 打开的编辑器
PROJECT
a.out
door.cpp

door.cpp
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main(){
4     int k=0;
5     for(int i=1;i<=2020;i++)
6         cout<<i;
7 }
```

```
shiyancelou:project/ $
shiyancelou:project/ $ g++ door.cpp
shiyancelou:project/ $ ./a.out
12345678910111213141516171819202122232425262728293
0710810911011112131415161718192021222324
71781791801811821831841851861871881891901911921931
24824925025125225325425525625725825926026126226326
1831932032132232324325326327328329330331332333334
83893903913923933943953963973983994004014024034044
45946046146246346446546646746846947047147247347447
```

然后，将输出结果粘贴到编辑器中（比赛时可以选择：Word、Codeblocks 都行）中，选搜索功能，搜索字符

「2」，共搜索出结果 624 次，这就是答案。



2. 蓝桥杯真题：卡片（2021年省赛）

[在线刷题](#)

题目描述

小蓝有很多数字卡片，每张卡片上都是数字 0 到 9。小蓝准备用这些卡片来拼一些数，他想从 1 开始拼出正整数，每拼一个，就保存起来，卡片就不能用来拼其它数了。小蓝想知道自己能从 1 拼到多少。例如，当小蓝有 30 张卡片，其中 0 到 9 各 3 张，则小蓝可以拼出 1 到 10，但是拼 11 时卡片 1 已经只有一张了，不够拼出 11。现在小蓝手里有 0 到 9 的卡片各 2021 张，共 20210 张，请问小蓝可以从 1 拼到多少？提示：建议使用计算机编程解决问题。

解题思路

和上面的例子差不多的做法。

先估计可能拼出 3000 多个数。

编个小程序打印出 1~3500，然后全部贴到 Word 里面，看 1 用了多少次，2 用了多少次。

最后发现，1~3181，用了 2021 个 1。所以答案是 3181。

请在线上环境亲自动手试一试。新建文件 card.cpp 并打印出结果，系统将自动检测。

- name: 检测文件是否存在

script: |

```
ls /home/project/card.cpp
```

error: /home/project 路径下不存在 card.cpp

timeout: 2

- name: 编译

script: |

```
g++ /home/project/card.cpp
```

error: card.cpp 无法正常编译。

timeout: 5

技巧二：眼看手数

1. 蓝桥杯真题：迷宫（2011年省赛）

有的填空题本身比较复杂，但是因为数据简单，此时用不着编码，直接用眼睛看，动手数。

题目描述



X 星球的一处迷宫游乐场建在某个小山坡上。它是由 10×10 相互连通的小房间组成的。

房间的地板上写着一个很大的字母。我们假设玩家是面朝上坡的方向站立，则：

L 表示走到左边的房间，

R 表示走到右边的房间，

U 表示走到上坡方向的房间，

D 表示走到下坡方向的房间。

X 星球的居民有点懒，不愿意费力思考。他们更喜欢玩运气类的游戏。这个游戏也是如此！

开始的时候，直升机把 100 名玩家放入一个个小房间内。玩家一定要按照地上的字母移动。

迷宫地图如下：

UDDLUULRUL

UURLLRRRU

RRUURLDLRD

RUDDDDUUUU

URUDLLRRUU

DURLRLDLRL

ULLURLLRDU

RDLULLRDDD

UUDDUDUDLL

ULRDLUURRR

请你计算一下，最后，有多少玩家会走出迷宫，而不是在里边兜圈子？

如果你还没明白游戏规则，可以参看下面一个简化的 4x4 迷宫的解说图：



解题思路

这道题是典型的 DFS，编码至少 10 分钟。不过因为是个填空题，而且迷宫很简单，只有 100 个字符，可以直接

数，从左往右数，从上往下数，约 2 分钟就能数完。数出来的结果见下面，加粗字符上的人能走出来。



UDDL**U**ULRUL

UURLLLRRRU

RRUURLDLRD

RUDDDDUUUUU

URUDLLRRUU

DURLRLDLRL

ULLURLL**R**DU

RDLULL**R**DDD

UU**D**DUDUDLL

ULRDLUURRR

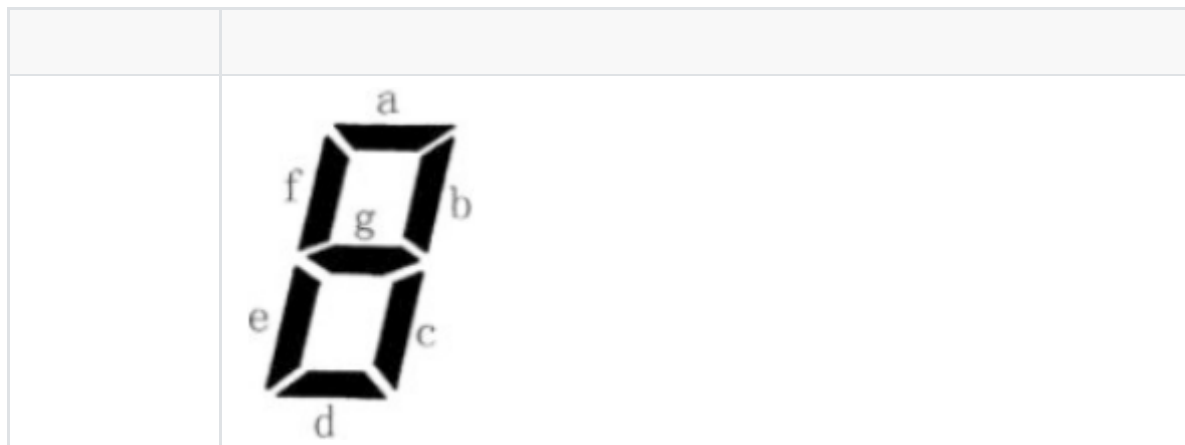
2. 蓝桥杯真题：七段码（2020年省赛）

[在线刷题](#)

题目描述

本题为填空题，只需要算出结果后，在代码中使用输出语句将所填结果输出即可。

小蓝要用七段码数码管来表示一种特殊的文字。



上图给出了七段码数码管的一个图示，数码管中一共有7段可以发光的二极管，分别标记为 a, b, c, d, e, f, g 。

小蓝要选择一部分二极管（至少要有一个）发光来表达字符。在设计字符的表达时，要求所有发光的二极管是连成

一片的。

例如：b 发光，其他二极管不发光可以用来表达一种字符。

例如：c 发光，其他二极管不发光可以用来表达一种字符。这种方案与上一行的方案可以用来表示不同的字符，尽管看上去比较相似。

例如： a, b, c, d, e 发光，f, g 不发光可以用来表达一种字符。

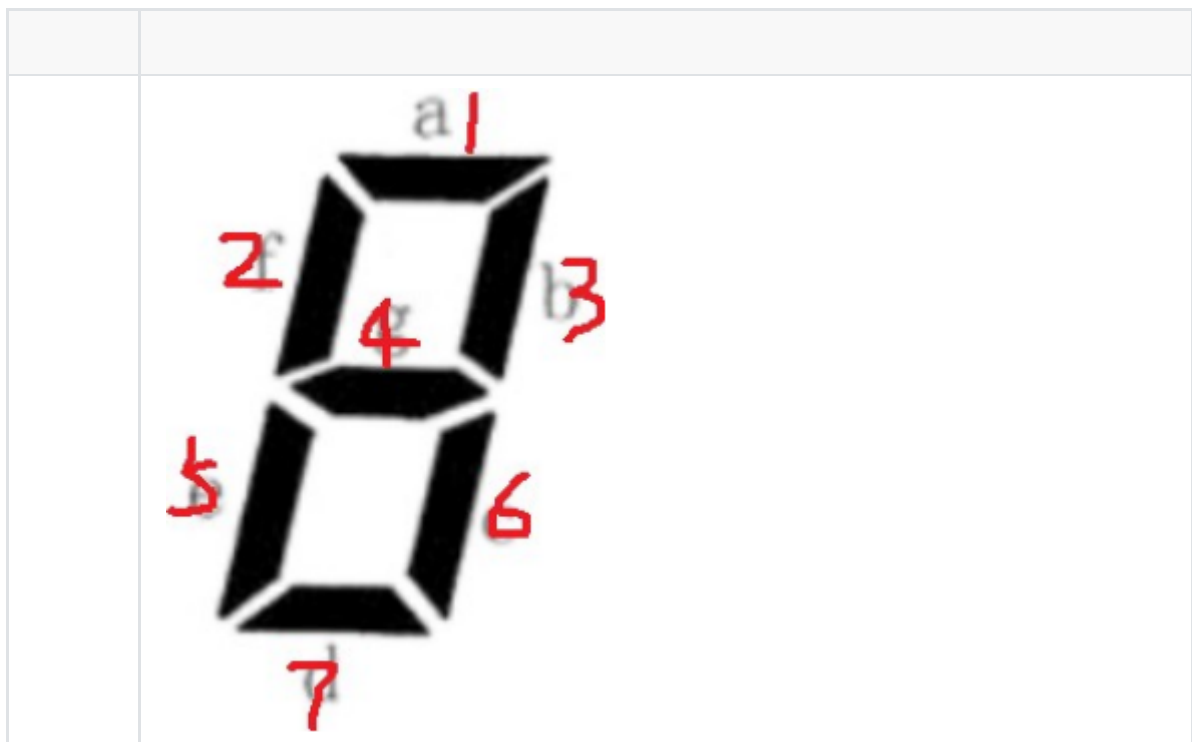
例如： b, f 发光，其他二极管不发光则不能用来表达一种字符，因为发光的二极管没有连成一片。请问，小蓝可以用七段码数码管表达多少种不同的字符？



解题思路

题目要求发光的二极管是相连的，可以用 DFS 或并查集查找连通块，编码时间 15 分钟以上。不过，因为图形简单，直接手算也行，约 3-5 分钟。

用字符表示数码管不太方便，改用数字：



分 7 种情况：

- 亮一个灯：有 7 种情况，1、2、3、4、5、6、7；
- 亮两个灯：有 12、13、24、25、..... 等等；
- 亮三个灯：有 123、124、125、134、136、234、257..... 等等；
- 亮四个灯，这时不要直接数四个灯，情况与灭三个灯是等价的：灭 123、灭 124. 等等；
- 亮五个灯，与灭两个灯等价：灭 12、灭 13、灭 14、等等；
- 亮六个灯，与灭一个灯等价，有 7 种情况；
- 亮七个灯，有 1 种情况。

对以上所有情况求和。



技巧三：巧用 Excel

1. 蓝桥杯真题：分数（2018年省赛）

[在线刷题](#)

题目描述

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots$$

每项是前一项的一半，如果一共有 20 项,求这个和是多少，结果用分数表示出来。

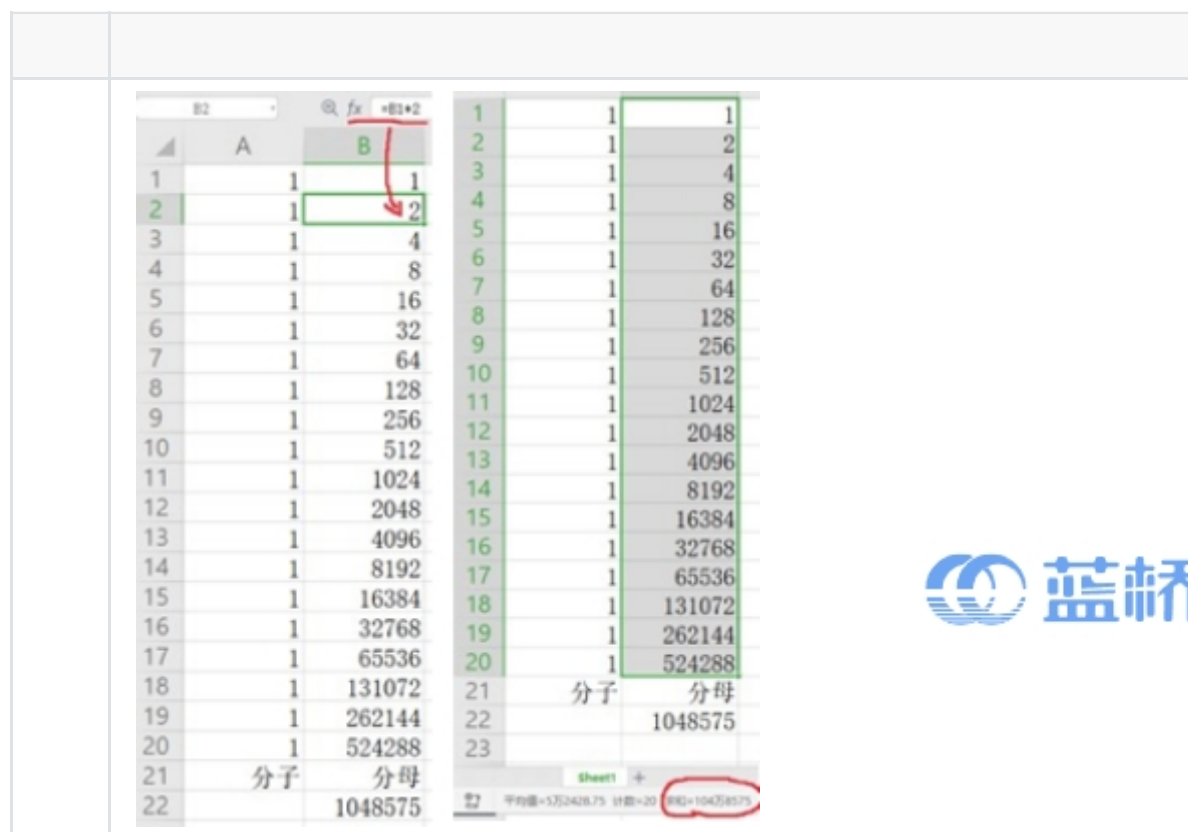
类似： $\frac{3}{2}$ ，当然，这只是加了前 2 项而已。分子分母要求互质。

解题思路

编码很简单，几分钟就好。也可以用 Excel 手算，时间差不多，而且不用思考。

A 列填分子，都是 1；B 列填分母，每行递增 2 倍，做法是，在 B1 填 1，在 B2 填写，然后按住 B2 往下拉到第 20 行，就填好了所有的分母。

然后通分求分子分母。分母就是 B20 的 524288，分子实际上就是「SUM(B1:B20)」，用鼠标选中这个区域，Excel 自动算出 1048575。



Row	A (分子)	B (分母)
1	1	1
2	1	2
3	1	4
4	1	8
5	1	16
6	1	32
7	1	64
8	1	128
9	1	256
10	1	512
11	1	1024
12	1	2048
13	1	4096
14	1	8192
15	1	16384
16	1	32768
17	1	65536
18	1	131072
19	1	262144
20	1	524288
21	分子	分母
22		1048575
23		



2. 蓝桥杯真题：星期一（2018年省赛）

[在线刷题](#)


题目描述

整个 20 世纪（1901 年 1 月 1 日至 2000 年 12 月 31 日之间），一共有多少个星期一？（不要告诉我你不知道今天是星期几）

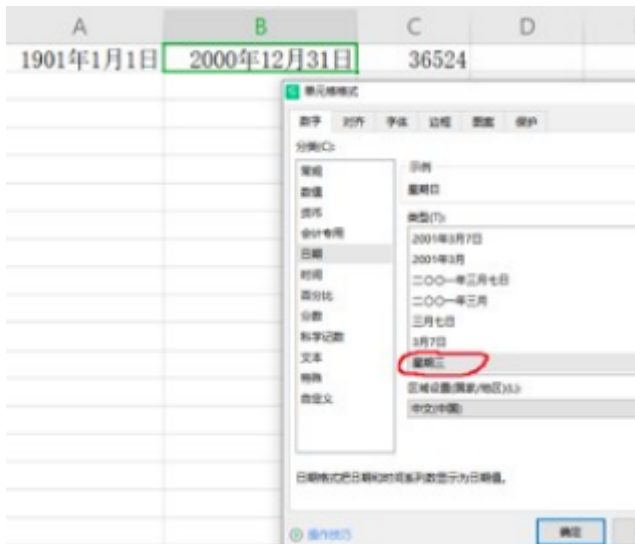
解题思路

首先，用 Excel，一个格子输入日期 1901 年 1 月 1 日，另一个格子输入 2000 年 12 月 31 日，然后两个格子相减

得 36524 天，除以 7 得 5217.7 周。

		
	A	B
	1901年1月1日	2000年12月31日
		C
		36524

再用 Excel 点 2000 年 12 月 31 日的属性，选星期，得「星期日」，说明答案就是 5217。



技巧四：巧用 Python

Python 处理数字非常简单，遇到这样的填空题，可以用 Python。即使是参加 C/C++、Java 组比赛，也要学一些 Python，以方便手算。

Python 的代码长度一般比 C/C++、Java 短很多，例如 30 行的 C++ 代码，用 Python 写只需要 20 行。

用 Python 算大数

蓝桥杯真题：乘积尾零（2018年省赛）

[在线刷题](#)

题目描述

如下的10 行数据，每行有10个整数，请你求出它们的乘积的末尾有多少个零？

5650	4542	3554	473	946	4114	3871	9073	90	4329
2758	7949	6113	5659	5245	7432	3051	4434	6704	3594
9937	1173	6866	3397	4759	7557	3070	2287	1453	9899
1486	5722	3135	1170	4014	5510	5120	729	2880	9019
2049	698	4582	4346	4427	646	9742	7340	1230	7683
5693	7015	6887	7381	4172	4341	2909	2027	7355	5649
6701	6645	1671	5978	2704	9926	295	3125	3878	6785
2066	4247	4800	1578	6652	4616	1113	6205	3264	2915
3966	5291	2904	1285	2193	1428	2265	8730	9436	7074

解题思路

遇到大数的问题，用 Python 处理是最简单的，可以直接硬算。

不过，其实 Python 的大数也不是无限大的，下面的代码，如果一股脑先算出所有的 100 个数的乘积 s ， s 实在太

大了，也是会溢出的。所以乘一个数，就看乘积 s 后面有没有 0，如果有 0 就除以 10，这样 s 就比较小了。

`cnt.py` 请在线上环境中新建 文件并贴入以下代码：



2758	7949	6113	5659	5245	7432	3051	4434	6704	3594
9937	1173	6866	3397	4759	7557	3070	2287	1453	9899
1486	5722	3135	1170	4014	5510	5120	729	2880	9019
2049	698	4582	4346	4427	646	9742	7340	1230	7683
5693	7015	6887	7381	4172	4341	2909	2027	7355	5649
6701	6645	1671	5978	2704	9926	295	3125	3878	6785
2066	4247	4800	1578	6652	4616	1113	6205	3264	2915
3966	5291	2904	1285	2193	1428	2265	8730	9436	7074

```

# 输入放在一行中，不要分10行
num = [int(i) for i in inputs.split()]
# input().split() 读一行以空格分开的元素，然后用int()转为整数
s = 1
cnt = 0
for i in range(len(num)): # 连续乘，一边乘一边统计0的个数
    s *= num[i] # 乘一个数
    while s % 10 == 0: # 末尾是零
        s /= 10 # 除以10，把末尾零去掉
        cnt += 1
print(cnt)

```

执行代码得到题目答案：

```
> 打开的编辑器
PROJECT
  cnt.py 1
cnt.py > ...
1 inputs = """
2 5650 4542 3554 473 946 4114 3871 9073 90 4329
3 2758 7949 6113 5659 5245 7432 3051 4434 6704 3594
4 9937 1173 6866 3397 4759 7557 3070 2287 1453 9899
5 1486 5722 3135 1170 4014 5510 5120 729 2880 9019
6 2049 698 4582 4346 4427 646 9742 7340 1230 7683
7 5693 7015 6887 7381 4172 4341 2909 2027 7355 5649
8 6701 6645 1671 5978 2704 9926 295 3125 3878 6785
9 2066 4247 4800 1578 6652 4616 1113 6205 3264 2915
10 3966 5291 2904 1285 2193 1428 2265 8730 9436 7074
11 689 5510 8243 6114 337 4096 8199 7313 3685 211
12 """
13
14
15 # 输入放在一行中，不要分10行
16 num = [int(i) for i in inputs.split()]
17 # input().split()读一行以空格分开的元素，然后用int()转为整数
18 s = 1
19 cnt = 0
20 for i in range(len(num)): # 连续乘，一边乘一边统计0的个数
21     s *= num[i] # 乘一个数
22     while s % 10 == 0: # 末尾是零
23         s /= 10 # 除以10，把末尾零去掉
24         cnt += 1
25 print(cnt)
26

问题 输出 DEBUG CONSOLE 终端
shiyancelou:project/ $ python cnt.py
31
shiyancelou:project/ $
```



```
- name: 检测文件是否存在
  script: |
    ls /home/project/cnt.py
  error: /home/project 路径下不存在 cnt.py
  timeout: 2
- name: 执行
  script: |
    python /home/project/cnt.py | grep "31"
  error: cnt.py 执行结果不等于 31。
  timeout: 5
```

用Python处理字符

蓝桥杯真题：平方和（2019年省赛）

[在线刷题](#)

题目描述

小明对数位中含有2、0、1、9的数字很感兴趣，在1 到 40 中这样的数包括1、2、9、10至32、39和40，共28个，他们的和是574，平方和是14362。

注意，平方和是指将每个数分别平方后求和。

请问，在1到2019 中，所有这样的数的平方和是多少？

解题思路

用 Python，不用任何算法，直接把数字看成字符来统计。

💡 请在线上环境亲自动手试一试。新建文件sum.py 并打印出结果，系统将自动检测。

```
- name: 检测文件是否存在
  script: |
    ls /home/project/sum.py
  error: /home/project 路径下不存在 sum.py
  timeout: 2
- name: 执行
  script: |
    python /home/project/sum.py | grep "2658417853"
  error: sum.py 执行结果不等于 2658417853。
  timeout: 5
```

▶ 参考答案

```
sum = 0
for i in range(1,2020):
    s = str(i)
    if '2' in s or '0' in s or '1' in s or '9' in s:
        sum += i*i
print(sum)
```



以下是 2 道可以巧用 Python 快速求解的真题，大家试一试看看能否做出来。

试一试1：数列求值（2019年省赛）

[在线刷题](#)

题目描述

434给定数列1, 1, 1, 3, 5, 9, 17, ⋯，从第 项开始，每项都是前 项的和。求第 20190324 项的最后 位数字。

💡 请在线上环境亲自动手试一试。新建文件 try1.py 并打印出结果，系统将自动检测。

```
- name: 检测文件是否存在
  script: |
    ls /home/project/try1.py
  error: /home/project 路径下不存在 try1.py
  timeout: 2
- name: 执行
  script: |
    python /home/project/try1.py | grep "4659"
  error: try1.py 执行结果不等于 4659。
  timeout: 10
```

► 参考答案

```
a,b,c = 1,1,1
for i in range(4,20190325):
    y=(a+b+c)%10000
    a=b
    b=c
    c=y
print(y)
```

试一试2：数列求值

[在线刷题](#)

题目描述

小蓝有很多数字卡片，每张卡片上都是数字 0 到 9。小蓝准备用这些卡片来拼一些数。他想从 1 开始拼出正整数，每拼一个，就保存起来，卡片就不能用来拼其它数了。小蓝想知道自己能从 1 拼到多少。例如，当小蓝有 30 张卡片，其中 0 到 9 各 3 张，则小蓝可以拼出 1 到 10，但是拼 11 时卡片 1 已经只有一张了，不够拼出 11。现在小蓝手里有 0 到 9 的卡片各 2021 张，共 20210 张，请问小蓝可以从 1 拼到多少？提示：建议使用计算机编程解决问题。

💡 请在线上环境亲自动手试一试。新建文件 try2.py 并打印出结果，系统将自动检测。

```
- name: 检测文件是否存在
  script: |
    ls /home/project/try2.py
  error: /home/project 路径下不存在 try2.py
  timeout: 2
- name: 执行
  script: |
    python /home/project/try2.py | grep "3181"
  error: try2.py 执行结果不等于 3181。
  timeout: 10|
```


► 解法 1

```
# 先打印出20210个字符，然后每拼一个数字，就把对应的卡片去掉
s = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"] * 2021
for i in range(1, 10000):
    a = list(str(i))
    try:
        for j in a:
            s.remove(j)
    except:
        print(i - 1)
        break
```

► 解法 2

```
# 数字1用得最多，统计到哪个数字的时候用了2021次就好了
s = ""
for i in range(1, 100000):
    s += str(i)
    if s.count("1") == 2021:
        print(i)
        break
```

实验总结

上面介绍了一些「投机取巧」的手算方法，是不是觉得信心爆棚，能参加蓝桥杯并得奖了？

给大家泼一瓢冷水，蓝桥杯可不是这么简单。

每次比赛手算题最多 1、2 题，而且都是分值不高的简单题，大家一般都能做出来，对能不能得奖并不是决定性的。真正能让你超过他人并得奖的，还是编码比较难的、需要算法的题目。

计算机的算法博大精深，是一片星辰大海！如果所有知识点有中国那么大，蓝桥杯软件赛只能考一小部分，相当于 北京那么大。



----- 未完 见下册 -----