

Lesson 1——基础算法复习

NOIP 非基础新课

Garen Wang

2019 年 3 月 2 日

目录

需要讲的知识点一共有下面这些：

- 1 模拟
- 2 暴力
- 3 二分
- 4 打表
- 5 随机

目录

需要讲的知识点一共有下面这些：

- 1 模拟
- 2 暴力
- 3 二分
- 4 打表
- 5 随机

模拟

以那暴力模拟向正解吊唁——《膜你抄》

- 模拟就是题目叫你做什么你就做什么。

模拟

以那暴力模拟向正解吊唁——《膜你抄》

- 模拟就是题目叫你做什么你就做什么。
- 一般的道理就是把整个算法过程都说给你听，然后请你实现。

模拟

以那暴力模拟向正解吊唁——《膜你抄》

- 模拟就是题目叫你做什么你就做什么。
- 一般的道理就是把整个算法过程都说给你听，然后请你实现。
- 模拟题在实现方面一般没有多大难度，但是需要对题意有充分的了解。

模拟

以那暴力模拟向正解吊唁——《膜你抄》

- 模拟就是题目叫你做什么你就做什么。
- 一般的道理就是把整个算法过程都说给你听，然后请你实现。
- 模拟题在实现方面一般没有多大难度，但是需要对题意有充分的了解。
- 总之一句话：注意细节！

模拟

以那暴力模拟向正解吊唁——《膜你抄》

- 模拟就是题目叫你做什么你就做什么。
- 一般的道理就是把整个算法过程都说给你听，然后请你实现。
- 模拟题在实现方面一般没有多大难度，但是需要对题意有充分的了解。
- 总之一句话：注意细节！
- 细节做的不好的话可能让你去掉一大半分数。

例题 1: luoguP1015

题目描述

若一个数（首位不为零）从左向右读与从右向左读都一样，我们就将其称之为回文数。

例如：给定一个十进制数56，将56加65（即把56从右向左读），得到121是一个回文数。

又如：对于十进制数87：

STEP1: $87+78 = 165$

STEP2: $165+561 = 726$

STEP3: $726+627 = 1353$

STEP4: $1353+3531 = 4884$

在这里的一步是指进行了一次 N 进制的加法，上例最少用了4步得到回文数4884。

写一个程序，给定一个 N ($2 \leq N \leq 10, N = 16$) 进制数 M (100位之内)，求最少经过几步可以得到回文数。如果在30步以内（包含30步）不可能得到回文数，则输出 **Impossible!**

输入输出格式

输入格式：

两行，分别是 N ， M 。

例题 1——分析

例题 1——分析

- 无 f**k 说。直接模拟就好了。

例题 1——分析

- 无 f**k 说。直接模拟就好了。
- 求一个数的回文数都懂吧。。。

例题 2：某题

题目描述

法法电影院有 n 排 m 列座位 ($m \leq 26$)。排从前往后编号为 1 至 n 的正整数，列从左往右编号为 A 至第 m 个大写字母。

大片《法法传说：mcfx 阿克传》即将首映，电影院中已经有若干座位被预订。接下来会有 q 群人来买票，每群人买票时会输入一个想坐的排数和这群人的人数。

当收到一个买票请求时，法法电影院的售票机会依次进行如下判断：

- 假设输入的排数是 r ，人数是 k ；
- 如果输入的排数 $r > n$ ，机器会取消这次售票并输出 `Theater only contains X rows`，其中 X 应替换为电影院座位的排数 n ；
- 如果剩余的空座位一共少于 k 个，机器会取消这次售票并输出 `Sold out`；
- 如果第 r 排有至少 k 个空座位，机器会找到最靠左的连续 k 个空座位，将它们售出；如果第 r 排没有这样的连续 k 个空座位，机器会从左至右依次售出 k 个空座位。
- 如果第 r 排剩余少于 k 个空座位，机器会找到最靠前的有连续 k 个空座位的排，将这一排最靠左的连续 k 个空座位售出；
- 如果以上条件均不符合，机器会取消这次售票并输出 `Please contact our staff to buy tickets`。

例题 2：某题

输入输出样例

输入样例#1: [复制](#)

```
3 3
---
#--
-#-
10
1 7
1 9
5 1
2 1
2 2
2 2
3 2
3 1
3 1
```

输出样例#1: [复制](#)

```
Please contact our staff to buy tickets
Sold out
Theater only contains 3 rows
2B
1A 1B
Please contact our staff to buy tickets
3A 3C
1C
2C
Sold out
```

例题 2——分析

例题 2——分析

- 这道题就是真实的模拟。

例题 2——分析

- 这道题就是真实的模拟。
- 照着题意做，认真读题后再做。

例题 2——分析

- 这道题就是真实的模拟。
- 照着题意做，认真读题后再做。
- 最后，注意数据范围，开好数组。

例题 2——分析

- 这道题就是真实的模拟。
- 照着题意做，认真读题后再做。
- 最后，注意数据范围，开好数组。
- 然后就没事了。

例题 2——分析

- 这道题就是真实的模拟。
- 照着题意做，认真读题后再做。
- 最后，注意数据范围，开好数组。
- 然后就没事了。
- 因为这道题不是公开题，所以想提交的 qq 发给我，我帮你交上去。
- 我就不提供代码了。注意一下就能写。

作业

作业

- luoguP1003 铺地毯
- luoguP1009 阶乘之和
- luoguP1014 Cantor 表
- luoguP1076 寻宝
- luoguP1328 生活大爆炸版剪刀石头布

暴力

骗分过样例，暴力出奇迹

暴力

骗分过样例，暴力出奇迹

- 外国真的有这门算法，名字叫 brute force。

暴力

骗分过样例，暴力出奇迹

- 外国真的有这门算法，名字叫 `brute force`。
- 能用暴力做的题的前提是数据范围极其极其小。

暴力

骗分过样例，暴力出奇迹

- 外国真的有这门算法，名字叫 `brute force`。
- 能用暴力做的题的前提是数据范围极其极其小。
- 一般暴力可以用来找规律。有些结论题可以用暴力猜规律。

暴力

骗分过样例，暴力出奇迹

- 外国真的有这门算法，名字叫 **brute force**。
- 能用暴力做的题的前提是数据范围极其极其小。
- 一般暴力可以用来找规律。有些结论题可以用暴力猜规律。
- 优秀的码暴力能力是学习其他算法的坚实基础。

例题 1: luoguP1024

题目描述

有形如: $ax^3 + bx^2 + cx^1 + dx^0 = 0$ 这样的一个一元三次方程。给出该方程中各项的系数(a, b, c, d 均为实数), 并约定该方程存在三个不同实根(根的范围在 -100 至 100 之间), 且根与根之差的绝对值 ≥ 1 。要求由小到大依次在同一行输出这三个实根(根与根之间留有空格), 并精确到小数点后2位。

提示: 记方程 $f(x) = 0$, 若存在2个数 x_1 和 x_2 , 且 $x_1 < x_2$, $f(x_1) \times f(x_2) < 0$, 则在 (x_1, x_2) 之间一定有一个根。

输入输出格式

输入格式:

一行, 4个实数 A, B, C, D 。

输出格式:

例题 1——分析

例题 1——分析

- 这道题为什么可以暴力而不 TLE 呢？原因就在于：

例题 1——分析

- 这道题为什么可以暴力而不 TLE 呢？原因就在于：
 - 根的范围在 -100 至 100 之间

例题 1——分析

- 这道题为什么可以暴力而不 TLE 呢？原因就在于：
 - ① 根的范围在 -100 至 100 之间
 - ② 精确到小数点后两位

例题 1——分析

- 这道题为什么可以暴力而不 TLE 呢？原因就在于：
 - ① 根的范围在 -100 至 100 之间
 - ② 精确到小数点后两位
- 这意味着我们只需要判断 10000 个 x 是否合法即可。

例题 1——分析

- 这道题为什么可以暴力而不 TLE 呢？原因就在于：
 - ① 根的范围在 -100 至 100 之间
 - ② 精确到小数点后两位
- 这意味着我们只需要判断 10000 个 x 是否合法即可。
- 所以解决的方法就是找到三个 $f(x) = 0$ 的 x 即可。

例题 1——分析

- 这道题为什么可以暴力而不 TLE 呢？原因就在于：
 - ① 根的范围在 -100 至 100 之间
 - ② 精确到小数点后两位
- 这意味着我们只需要判断 10000 个 x 是否合法即可。
- 所以解决的方法就是找到三个 $f(x) = 0$ 的 x 即可。
- 唯一要注意的就是浮点数的精度问题，判断零点还不能看是否等于 0。

例题 1——分析

- 这道题为什么可以暴力而不 TLE 呢？原因就在于：
 - ① 根的范围在 -100 至 100 之间
 - ② 精确到小数点后两位
- 这意味着我们只需要判断 10000 个 x 是否合法即可。
- 所以解决的方法就是找到三个 $f(x) = 0$ 的 x 即可。
- 唯一要注意的就是浮点数的精度问题，判断零点还不能看是否等于 0。
- 代码不贴。

例题 2: luoguP1149

题目描述

给你 n 根火柴棍，你可以拼出多少个形如“ $A + B = C$ ”的等式？等式中的 A 、 B 、 C 是用火柴棍拼出的整数（若该数非零，则最高位不能是0）。用火柴棍拼数字0—9的拼法如图所示：



注意：

1. 加号与等号各自需要两根火柴棍
2. 如果 $A \neq B$ ，则 $A + B = C$ 与 $B + A = C$ 视为不同的等式($A, B, C \geq 0$)
3. n 根火柴棍必须全部用上

输入输出格式

输入格式：

一个整数 n ($n \leq 24$)。

例题 2——分析

例题 2——分析

- 思路比较简单：枚举两个加数，那么和就唯一确定，判断能否成立即可。

例题 2——分析

- 思路比较简单：枚举两个加数，那么和就唯一确定，判断能否成立即可。
- 这道题的难点就是你不知道枚举到多少才能枚举完。

例题 2——分析

- 思路比较简单：枚举两个加数，那么和就唯一确定，判断能否成立即可。
- 这道题的难点就是你不知道枚举到多少才能枚举完。
- （参照题解）发现可以把所有符合的等式都打出来，看看我们有必要枚举到哪个地方。

例题 2——分析

- 思路比较简单：枚举两个加数，那么和就唯一确定，判断能否成立即可。
- 这道题的难点就是你不知道枚举到多少才能枚举完。
- （参照题解）发现可以把所有符合的等式都打出来，看看我们有必要枚举到哪个地方。
- 听说答案是 711。当然大于 711 不 TLE 的程序都是能过的。

例题 2——分析

- 思路比较简单：枚举两个加数，那么和就唯一确定，判断能否成立即可。
- 这道题的难点就是你不知道枚举到多少才能枚举完。
- （参照题解）发现可以把所有符合的等式都打出来，看看我们有必要枚举到哪个地方。
- 听说答案是 711。当然大于 711 不 TLE 的程序都是能过的。
- 判断过程可以写个函数方便操作。

例题 2——分析

- 思路比较简单：枚举两个加数，那么和就唯一确定，判断能否成立即可。
- 这道题的难点就是你不知道枚举到多少才能枚举完。
- （参照题解）发现可以把所有符合的等式都打出来，看看我们有必要枚举到哪个地方。
- 听说答案是 711。当然大于 711 不 TLE 的程序都是能过的。
- 判断过程可以写个函数方便操作。
- 代码一大堆，不贴。

作业

作业

找不到什么作业啊!!! 所以就把一些显然的题目给大家。

作业

找不到什么作业啊!!! 所以就把一些显然的题目给大家。

- luoguP1097 统计数字

作业

找不到什么作业啊!!! 所以就把一些显然的题目给大家。

- luoguP1097 统计数字
- luoguP1088 火星人

作业

找不到什么作业啊!!! 所以就把一些显然的题目给大家。

- luoguP1097 统计数字
- luoguP1088 火星人
- CF1100A Roman and Browser

作业

找不到什么作业啊!!! 所以就把一些显然的题目给大家。

- luoguP1097 统计数字
- luoguP1088 火星人
- CF1100A Roman and Browser

二分

二分

- Binary search 是比较简单的带 \log 的算法，所以比暴力要快得多。数据越大越明显。

二分

- Binary search 是比较简单的带 \log 的算法，所以比暴力要快得多。数据越大越明显。
- 生活例子：猜数字炸弹、翻字典

二分

- Binary search 是比较简单的带 \log 的算法，所以比暴力要快得多。数据越大越明显。
- 生活例子：猜数字炸弹、翻字典
- 二分的流程是这样的：（对区间 $[l, r]$ 进行操作）

二分

- Binary search 是比较简单的带 \log 的算法，所以比暴力要快得多。数据越大越明显。
- 生活例子：猜数字炸弹、翻字典
- 二分的流程是这样的：（对区间 $[l, r]$ 进行操作）
 - ① 计算 l 和 r 的中点为 mid 。

二分

- Binary search 是比较简单的带 \log 的算法，所以比暴力要快得多。数据越大越明显。
- 生活例子：猜数字炸弹、翻字典
- 二分的流程是这样的：（对区间 $[l, r]$ 进行操作）
 - ① 计算 l 和 r 的中点为 mid 。
 - ② 判断 mid 是否满足条件，如果符合，将区间往不符合方向移动；否则，向符合方向移动。移动方式是修改端点值。

二分

- Binary search 是比较简单的带 \log 的算法，所以比暴力要快得多。数据越大越明显。
- 生活例子：猜数字炸弹、翻字典
- 二分的流程是这样的：（对区间 $[l, r]$ 进行操作）
 - ① 计算 l 和 r 的中点为 mid 。
 - ② 判断 mid 是否满足条件，如果符合，将区间往不符合方向移动；否则，向符合方向移动。移动方式是修改端点值。
 - ③ 一直做 2 过程直到 l 和 r 不表示区间（可理解为 $l > r$ ）为止。

二分的前提

二分的前提

- 使用二分是有前提的：条件的符合情况一定要具有单调性。

二分的前提

- 使用二分是有前提的：条件的符合情况一定要具有单调性。
- 在数组里找到小于等于 *val* 的最后一个元素，即 `std::lower_bound` 操作。前提是要把数组排序。

二分的前提

- 使用二分是有前提的：条件的符合情况一定要具有单调性。
- 在数组里找到小于等于 *val* 的最后一个元素，即 `std::lower_bound` 操作。前提是要把数组排序。
- 如果有一个问题，*val* 越小越难满足，*val* 越大越容易满足，那么 *jiushuo* 这个问题有单调性，可以二分。反之亦然。

二分的前提

- 使用二分是有前提的：条件的符合情况一定要具有单调性。
- 在数组里找到小于等于 *val* 的最后一个元素，即 `std::lower_bound` 操作。前提是要把数组排序。
- 如果有一个问题，*val* 越小越难满足，*val* 越大越容易满足，那么 *jiushuo* 这个问题有单调性，可以二分。反之亦然。
- 那我们二分能找到什么呢？

二分的前提

- 使用二分是有前提的：条件的符合情况一定要具有单调性。
- 在数组里找到小于等于 *val* 的最后一个元素，即 `std::lower_bound` 操作。前提是要把数组排序。
- 如果有一个问题，*val* 越小越难满足，*val* 越大越容易满足，那么 *jiushuo* 这个问题有单调性，可以二分。反之亦然。
- 那我们二分能找到什么呢？
- 找到这个所谓的临界点。即“最小的最大”“最大的最小”。

例题 1

例题 1

- 在数组里找到小于等于 *val* 的最后一个元素。

例题 1

- 在数组里找到小于等于 *val* 的最后一个元素。
- 首先先排序，满足大小的单调性，然后就可以二分了。

例题 1

- 在数组里找到小于等于 *val* 的最后一个元素。
- 首先先排序，满足大小的单调性，然后就可以二分了。
- 这里给大家提供两个二分的模板，自行选用。

例题 1

- 在数组里找到小于等于 *val* 的最后一个元素。
- 首先先排序，满足大小的单调性，然后就可以二分了。
- 这里给大家提供两个二分的模板，自行选用。
- 镜像问题：在数组里找到大于 *val* 的第一个元素，即 `std::upper_bound` 操作。

模板 1

模板 1

模板 1 适用于浮点数。*left* 和 *right* 就是前面所说的左右端点。

模板 1

模板 1 适用于浮点数。*left* 和 *right* 就是前面所说的左右端点。

```
ll left = 1, right = n;
while(left < right) {
    ll mid = (left + right) / 2;
    if(a[mid] <= val) left = mid;
    else right = mid - 1;
}
```

模板 2

模板 2

模板 2 只适用于整数类型。浮点数显然用不了这个，如果要用要写 `eps`。

模板 2

模板 2 只适用于整数类型。浮点数显然用不了这个，如果要用要写 `eps`。

```
ll left = 1, right = n, ans = -1;
while(left <= right) {
    ll mid = (left + right) / 2;
    if(check(mid)) ans = mid, left = mid + 1; // 看
    else right = mid - 1; // 同上
}
```

例题 2: luoguP2678

题目背景

一年一度的“跳石头”比赛又要开始了!

题目描述

这项比赛将在一条笔直的河道中进行,河道中分布着一些巨大岩石。组委会已经选择好了两块岩石作为比赛起点和终点。在起点和终点之间,有 N 块岩石(不含起点和终点的岩石)。在比赛过程中,选手们将从起点出发,每一步跳向相邻的岩石,直至到达终点。

为了提高比赛难度,组委会计划移走一些岩石,使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制,组委会至多从起点和终点之间移走 M 块岩石(不能移走起点和终点的岩石)。

输入输出格式

输入格式:

第一行包含三个整数 L, N, M , 分别表示起点到终点的距离,起点和终点之间的岩石数,以及组委会至多移走的岩石数。保证 $L \geq 1$ 且 $N \geq M \geq 0$ 。

接下来 N 行,每行一个整数,第 i 行的整数 $D_i (0 < D_i < L)$, 表示第 i 块岩石与起点的距离。这些岩石按与起点距离从小到大的顺序给出,且不会有两个岩石出现在同一个位置。

例题 2——分析

例题 2——分析

- 这是二分在题目中的应用，叫做“二分答案”。顾名思义，就是通过发现答案具有单调性，从而对答案来二分。

例题 2——分析

- 这是二分在题目中的应用，叫做“二分答案”。顾名思义，就是通过发现答案具有单调性，从而对答案来二分。
- 一眼看出题目考点：使得在比赛过程中的最短跳跃距离尽可能长。

例题 2——分析

- 这是二分在题目中的应用，叫做“二分答案”。顾名思义，就是通过发现答案具有单调性，从而对答案来二分。
- 一眼看出题目考点：使得在比赛过程中的最短跳跃距离尽可能长。
- 答案单调性：最短跳跃距离越短难度越小，越长则难度越大。

例题 2——分析

- 这是二分在题目中的应用，叫做“二分答案”。顾名思义，就是通过发现答案具有单调性，从而对答案来二分。
- 一眼看出题目考点：使得在比赛过程中的最短跳跃距离尽可能长。
- 答案单调性：最短跳跃距离越短难度越小，越长则难度越大。
- 所以套上二分的模板，具体化端点移动模式，然后就发现一个最大的问题：

例题 2——分析

- 这是二分在题目中的应用，叫做“二分答案”。顾名思义，就是通过发现答案具有单调性，从而对答案来二分。
- 一眼看出题目考点：使得在比赛过程中的最短跳跃距离尽可能长。
- 答案单调性：最短跳跃距离越短难度越小，越长则难度越大。
- 所以套上二分的模板，具体化端点移动模式，然后就发现一个最大的问题：
- 你说的 `check` 函数怎么写啊？

例题 2——分析

例题 2——分析

- 二分答案的本质是把临界问题转变为判定问题。

例题 2——分析

- 二分答案的本质是把临界问题转变为判定问题。
- `check` 函数应为：当最短跳跃距离为 mid 时，选手能否完成比赛。

例题 2——分析

- 二分答案的本质是把临界问题转变为判定问题。
- `check` 函数应为：当最短跳跃距离为 *mid* 时，选手能否完成比赛。
- 显然 `check` 函数返回 `bool`，能完成返回 `true`，不能就返回 `false`。

例题 2——分析

- 二分答案的本质是把临界问题转变为判定问题。
- `check` 函数应为：当最短跳跃距离为 *mid* 时，选手能否完成比赛。
- 显然 `check` 函数返回 `bool`，能完成返回 `true`，不能就返回 `false`。
- 假设现在拿到最短跳跃距离，该如何安排石头？

例题 2——分析

- 二分答案的本质是把临界问题转变为判定问题。
- `check` 函数应为：当最短跳跃距离为 *mid* 时，选手能否完成比赛。
- 显然 `check` 函数返回 `bool`，能完成返回 `true`，不能就返回 `false`。
- 假设现在拿到最短跳跃距离，该如何安排石头？
- 显然，要判断与前边已放石头的距离是否大于等于答案。如果可以，石头可以保留；否则，这块要拿走，看下一个石头能否放置。

例题 2——分析

- 二分答案的本质是把临界问题转变为判定问题。
- `check` 函数应为：当最短跳跃距离为 *mid* 时，选手能否完成比赛。
- 显然 `check` 函数返回 `bool`，能完成返回 `true`，不能就返回 `false`。
- 假设现在拿到最短跳跃距离，该如何安排石头？
- 显然，要判断与前边已放石头的距离是否大于等于答案。如果可以，石头可以保留；否则，这块要拿走，看下一个石头能否放置。
- 其实这里用到贪心的思想，我们建立满足答案的拿走最少的局面。

例题 2——分析

- 二分答案的本质是把临界问题转变为判定问题。
- `check` 函数应为：当最短跳跃距离为 *mid* 时，选手能否完成比赛。
- 显然 `check` 函数返回 `bool`，能完成返回 `true`，不能就返回 `false`。
- 假设现在拿到最短跳跃距离，该如何安排石头？
- 显然，要判断与前边已放石头的距离是否大于等于答案。如果可以，石头可以保留；否则，这块要拿走，看下一个石头能否放置。
- 其实这里用到贪心的思想，我们建立满足答案的拿走最少的局面。
- 以上就是 `check` 函数的写法。`check` 函数的写法常常是一种贪心。

作业

作业

- luoguP1316 丢瓶盖
- luoguP1024 一元三次方程求解（提示：极点之间的区域来二分）
- luoguP1083 借教室
- luoguP1182 数列分段 'Section II'
- luoguP3382 【模板】三分法（提示：对导函数二分）
- 看 luogu 日报第 12 期：浅谈二分的边界问题

打表

打表

暴搜挂着机，打表出省一

打表

暴搜挂着机，打表出省一

- 打表虽然很赖皮，而且基本都是非正解，但是这种方法能让我们拿到一些会超时或者会超空间的一些数据点（来自 [luogu 试炼场](#)）

打表

暴搜挂着机，打表出省一

- 打表虽然很赖皮，而且基本都是非正解，但是这种方法能让我们拿到一些会超时或者会超空间的一些数据点（来自 [luogu 试炼场](#)）
- 打表的三种手段：（by: 百度百科）

打表

暴搜挂着机，打表出省一

- 打表虽然很赖皮，而且基本都是非正解，但是这种方法能让我们拿到一些会超时或者会超空间的一些数据点（来自 [luogu 试炼场](#)）
- 打表的三种手段：（by: 百度百科）
 - ❶ 暴力出奇迹，算出答案。适用于数据范围较大的题目。

打表

暴搜挂着机，打表出省一

- 打表虽然很赖皮，而且基本都是非正解，但是这种办法能让我们拿到一些会超时或者会超空间的一些数据点（来自 [luogu 试炼场](#)）
- 打表的三种手段：（by: 百度百科）
 - ① 暴力出奇迹，算出答案。适用于数据范围较大的题目。
 - ② 手算也出奇迹，算出答案。适用于数据范围较小但难的题目。

打表

暴搜挂着机，打表出省一

- 打表虽然很赖皮，而且基本都是非正解，但是这种方法能让我们拿到一些会超时或者会超空间的一些数据点（来自 `luogu` 试炼场）
- 打表的三种手段：（by: 百度百科）
 - ① 暴力出奇迹，算出答案。适用于数据范围较大的题目。
 - ② 手算也出奇迹，算出答案。适用于数据范围较小但难的题目。
 - ③ 给数据处理一下，作为算法的新条件。其实就是预处理。

适用范围

适用范围

- 什么题目可以用到打表？

适用范围

- 什么题目可以用到打表？
 - ① 题目需要的输入条件很少，一般是一个或两个。

适用范围

- 什么题目可以用到打表？
 - ① 题目需要的输入条件很少，一般是一个或两个。
 - ② 数据范围极大，但是可以用暴力用很久算出来，并且答案不多。这些题目以数学题为主。

适用范围

- 什么题目可以用到打表？

- ① 题目需要的输入条件很少，一般是一个或两个。
- ② 数据范围极大，但是可以用暴力用很久算出来，并且答案不多。这些题目以数学题为主。
- ③ 疑似有规律的题目，通过打表推出正解。

适用范围

- 什么题目可以用到打表？
 - ① 题目需要的输入条件很少，一般是一个或两个。
 - ② 数据范围极大，但是可以用暴力用很久算出来，并且答案不多。这些题目以数学题为主。
 - ③ 疑似有规律的题目，通过打表推出正解。
- 注意：答案非常多的题目打不了表，因为 C++ 不允许初始化一个特别大的表。

例题 1: luoguP1463

题目描述

对于任何正整数 x ，其约数的个数记作 $g(x)$ 。例如 $g(1)=1$ 、 $g(6)=4$ 。

如果某个正整数 x 满足： $g(x)>g(i)$ $0<i<x$ ，则称 x 为反质数。例如，整数1, 2, 4, 6等都是反质数。

现在给定一个数 N ，你能求出不超过 N 的最大的反质数么？

输入输出格式

输入格式：

一个数 N ($1 \leq N \leq 2,000,000,000$) 。

输出格式：

例题 1——分析

例题 1——分析

- n 的范围是 2×10^9 ，这辈子都不可能想出数学方法的解法了。让我们用打表的思维解决这道题。

例题 1——分析

- n 的范围是 2×10^9 ，这辈子都不可能想出数学方法的解法了。让我们用打表的思维解决这道题。
- 假设 n 很小，那直接读 n 然后直接做。

例题 1——分析

- n 的范围是 2×10^9 ，这辈子都不可能想出数学方法的解法了。让我们用打表的思维解决这道题。
- 假设 n 很小，那直接读 n 然后直接做。
- 我们要做的是枚举 $[1, n]$ 内所有整数，算出它们的约数个数，看看是否满足条件。

例题 1——分析

- n 的范围是 2×10^9 ，这辈子都不可能想出数学方法的解法了。让我们用打表的思维解决这道题。
- 假设 n 很小，那直接读 n 然后直接做。
- 我们要做的是枚举 $[1, n]$ 内所有整数，算出它们的约数个数，看看是否满足条件。
- 有一个睿智的优化：开一个变量，维护 $[1, i-1]$ 内整数约数个数最大值，每次维护一下即可。

例题 1——分析

- n 的范围是 2×10^9 ，这辈子都不可能想出数学方法的解法了。让我们用打表的思维解决这道题。
- 假设 n 很小，那直接读 n 然后直接做。
- 我们要做的是枚举 $[1, n]$ 内所有整数，算出它们的约数个数，看看是否满足条件。
- 有一个睿智的优化：开一个变量，维护 $[1, i-1]$ 内整数约数个数最大值，每次维护一下即可。
- 求一个数的约数个数，要用到约数个数定理：

例题 1——分析

- n 的范围是 2×10^9 ，这辈子都不可能想出数学方法的解法了。让我们用打表的思维解决这道题。
- 假设 n 很小，那直接读 n 然后直接做。
- 我们要做的是枚举 $[1, n]$ 内所有整数，算出它们的约数个数，看看是否满足条件。
- 有一个睿智的优化：开一个变量，维护 $[1, i-1]$ 内整数约数个数最大值，每次维护一下即可。
- 求一个数的约数个数，要用到约数个数定理：
- 若一个数 x 唯一分解后等于 $\prod_{i=1}^n p_i^{a_i}$ ，则 x 的约数个数为 $\prod_{i=1}^n (a_i + 1)$ 。

例题 1——分析

例题 1——分析

- 前面的算法复杂度是 $O(n\sqrt{n})$ 的，只能做到十多万的数据。

例题 1——分析

- 前面的算法复杂度是 $O(n\sqrt{n})$ 的，只能做到十多万的数据。
- 那我们如果把这个程序交上去，妥妥地 T 掉。

例题 1——分析

- 前面的算法复杂度是 $O(n\sqrt{n})$ 的，只能做到十多万的数据。
- 那我们如果把这个程序交上去，妥妥地 T 掉。
- 但是我们可以把所有的答案在本地都打出来，存入常量数组，然后就可以二分出答案来了。

例题 1——分析

- 前面的算法复杂度是 $O(n\sqrt{n})$ 的，只能做到十多万的数据。
- 那我们如果把这个程序交上去，妥妥地 T 掉。
- 但是我们可以把所有的答案在本地都打出来，存入常量数组，然后就可以二分出答案来了。
- 写一个快速打表的程序十分重要，不然你可能一天都打不出表来。

例题 1——分析

- 前面的算法复杂度是 $O(n\sqrt{n})$ 的，只能做到十多万的数据。
- 那我们如果把这个程序交上去，妥妥地 T 掉。
- 但是我们可以把所有的答案在本地都打出来，存入常量数组，然后就可以二分出答案来了。
- 写一个快速打表的程序十分重要，不然你可能一天都打不出表来。
- 这种程序只需要跑 3min 左右。我本来的暴力程序 10min 还跑没好。

例题 1——分析

- 前面的算法复杂度是 $O(n\sqrt{n})$ 的，只能做到十多万的数据。
- 那我们如果把这个程序交上去，妥妥地 T 掉。
- 但是我们可以把所有的答案在本地都打出来，存入常量数组，然后就可以二分出答案来了。
- 写一个快速打表的程序十分重要，不然你可能一天都打不出表来。
- 这种程序只需要跑 3min 左右。我本来的暴力程序 10min 还跑没好。
- 输出到文件，用 C++ 的语法格式直接输出，然后粘贴到你交的程序中。

例题 1——分析

- 前面的算法复杂度是 $O(n\sqrt{n})$ 的，只能做到十多万的数据。
- 那我们如果把这个程序交上去，妥妥地 T 掉。
- 但是我们可以把所有的答案在本地都打出来，存入常量数组，然后就可以二分出答案来了。
- 写一个快速打表的程序十分重要，不然你可能一天都打不出表来。
- 这种程序只需要跑 3min 左右。我本来的暴力程序 10min 还跑没好。
- 输出到文件，用 C++ 的语法格式直接输出，然后粘贴到你交的程序中。
- 最后的问题就变成了一个 sb 二分问题了。甚至你都不用二分。

例题 2: HDU1061

给你一个正整数 n , 求 n^n 的个位数。 $n \leq 1 \times 10^9$ 。

例题 2——分析

例题 2——分析

- 这道题看起来就很不可做。让我们用暴力打表看看有没有规律。

例题 2——分析

- 这道题看起来就很不可做。让我们用暴力打表看看有没有规律。
- 随便打了前 1000 的答案，然后就发现了规律：

例题 2——分析

- 这道题看起来就很不可做。让我们用暴力打表看看有没有规律。
- 随便打了前 1000 的答案，然后就发现了规律：
- 除了第一个数是 1，剩下的答案都是有循环节的。

例题 2——分析

- 这道题看起来就很不可做。让我们用暴力打表看看有没有规律。
- 随便打了前 1000 的答案，然后就发现了规律：
- 除了第一个数是 1，剩下的答案都是有循环节的。
- 仔细观察下，一种循环节是 901 后开始，直到第二个 901 结束，长度 20。

例题 2——分析

- 这道题看起来就很不可做。让我们用暴力打表看看有没有规律。
- 随便打了前 1000 的答案，然后就发现了规律：
- 除了第一个数是 1，剩下的答案都是有循环节的。
- 仔细观察下，一种循环节是 901 后开始，直到第二个 901 结束，长度 20。
- 最后就 n 的大小分类讨论，输出哪个答案即可。

作业

作业

- luoguP1820 寻找 AP 数
- luoguP3951 小凯的疑惑
- luoguP4942 小凯的数字
- 看 luogu 日报第 59 期：我有独特的骗分技巧

随机

随机

- 随机算法是一种神奇的算法，也是十分玄学的算法。

随机

- 随机算法是一种神奇的算法，也是十分玄学的算法。
- 当你需要取出其中的元素进行检验，但又不能规定死了取，这时候我们随机抽取即可。

随机

- 随机算法是一种神奇的算法，也是十分玄学的算法。
- 当你需要取出其中的元素进行检验，但又不能规定死了取，这时候我们随机抽取即可。
- 这里只介绍低级随机算法。像模拟退火等随机算法就不介绍了。

随机

- 随机算法是一种神奇的算法，也是十分玄学的算法。
- 当你需要取出其中的元素进行检验，但又不能规定死了取，这时候我们随机抽取即可。
- 这里只介绍低级随机算法。像模拟退火等随机算法就不介绍了。
- 随机部分的知识一般不会直接列入考点，但是如何用随机数是很重要的。

随机

- 随机算法是一种神奇的算法，也是十分玄学的算法。
- 当你需要取出其中的元素进行检验，但又不能规定死了取，这时候我们随机抽取即可。
- 这里只介绍低级随机算法。像模拟退火等随机算法就不介绍了。
- 随机部分的知识一般不会直接列入考点，但是如何用随机数是很重要的。
- 随机的一个重要运用：大名鼎鼎的数据生成器。

随机的实现

随机的实现

- 这里首先要介绍 `rand` 函数，这个函数的返回值就是随机数。

随机的实现

- 这里首先要介绍 `rand` 函数，这个函数的返回值就是随机数。
- 用 `rand` 之前要 `srand`。`srand` 需要一个玄学种子作为参数。

随机的实现

- 这里首先要介绍 rand 函数，这个函数的返回值就是随机数。
- 用 rand 之前要 srand。srand 需要一个玄学种子作为参数。
- rand 函数能生成 $[0, RAND_MAX]$ 之间的随机整数，这个 $RAND_MAX$ 多大要看具体的编译器。

随机的实现

- 这里首先要介绍 `rand` 函数，这个函数的返回值就是随机数。
- 用 `rand` 之前要 `srand`。`srand` 需要一个玄学种子作为参数。
- `rand` 函数能生成 $[0, RAND_MAX]$ 之间的随机整数，这个 $RAND_MAX$ 多大要看具体的编译器。
- 古老的编译器的 $RAND_MAX$ 是 32768，现在已经很大了。

例题 1: luoguP4703

题目描述

Alice 和 Bob 生活在一个 $l \times l$ 的正方形房子里, 由于 Bob 最近沉迷隔膜, Alice 决定要限制 Bob 上网的频率。

Alice 建造了 n 个无线信号屏蔽器, 第 i 个位于 (x_i, y_i) , 屏蔽范围为 $\frac{l}{n}$ 。

Bob 网瘾发作按捺不住上网的冲动, 找到了你, 帮他找到一个位置 (x, y) , 使得没有被 Alice 的无线信号屏蔽器覆盖。

输入输出格式

输入格式:

第一行两个整数 n, l ($1 \leq n \leq 10, 1 \leq l \leq 10^5$), 分别表示无线信号屏蔽器的个数和房子的大小。

接下来 n 行, 每行 2 个数, 分别是 x_i, y_i ($0 \leq x_i, y_i \leq l$), 意义如上所述。

输出格式:

如果可以找到, 输出两个数 x, y ($0 \leq x, y \leq l$), 意义如上所述, 如果有多组解, 输出任意一组即可。如果你输出的解满足到任意一个屏蔽器的距离都不小于 $\frac{l}{n} + 10^{-6}$ 则视为正确

例题 2: luoguP2503

题目描述

已知N个正整数：A1、A2、.....、An。今要将它们从小到大排列。均方差公式如下：

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$$

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

例题 1、2——分析

例题 1、2——分析

- 第一道题没办法用数组存下来，因为有圆的存在。

例题 1、2——分析

- 第一道题没办法用数组存下来，因为有圆的存在。
- 其实随机出里面的点，其实不用随多少次就能搞定。

例题 1、2——分析

- 第一道题没办法用数组存下来，因为有圆的存在。
- 其实随机出里面的点，其实不用随多少次就能搞定。
-
- 第二道题其实只要用一个函数把数据随机打乱，再算均方差即可。

例题 1、2——分析

- 第一道题没办法用数组存下来，因为有圆的存在。
- 其实随机出里面的点，其实不用随多少次就能搞定。
-
- 第二道题其实只要用一个函数把数据随机打乱，再算均方差即可。
- 显然想要答案越正确，你要随更多次，但是要确保不 T 掉。

例题 1、2——分析

- 第一道题没办法用数组存下来，因为有圆的存在。
- 其实随机出里面的点，其实不用随多少次就能搞定。
-
- 第二道题其实只要用一个函数把数据随机打乱，再算均方差即可。
- 显然想要答案越正确，你要随更多次，但是要确保不 T 掉。
- 把数据随机打乱有一个 STL 函数可以做：`std::random_shuffle`。

例题 1、2——分析

- 第一道题没办法用数组存下来，因为有圆的存在。
- 其实随机出里面的点，其实不用随多少次就能搞定。
-
- 第二道题其实只要用一个函数把数据随机打乱，再算均方差即可。
- 显然想要答案越正确，你要随更多次，但是要确保不 T 掉。
- 把数据随机打乱有一个 STL 函数可以做：`std::random_shuffle`。
- 数据是水的，不用用到模拟退火降温等玄学操作。

作业

作业

- CF1108A Two distinct points
- 自己学会针对某一道题写数据生成器，数据要保证在数据范围内。