# 贪心/二分/倍增

SRwudi

October 5, 2019

# 开篇

二分，倍增，贪心思想在 NOIP 提高组中是非常基础和非常重要的一环，基本上每年都会涉及至少 1 道题及以上（如果今年没有则当我没说）。

二分和倍增之间的思想有共通之处也有区别，不过只是一种将问题简化的方法。都可以优化一类问题达到 $O(\log n)$ 的复杂度。其深层次的本质是分治，将一个问题划归成若干小规模问题来加快求解复杂度。

贪心是一种一类优化解决问题的思想，和动归在 OI 范围中是属于优化和计划问题中的一环。

其实不然发现，很多题目二分之后需要用到贪心，亦或是在倍增的维护和求解中需要用到二分。

所以很多题目都是这些算法和 idea 拼一拼的。

# 概述

贪心只是一种思想，不是一种算法。

# 概述

贪心只是一种思想，不是一种算法。

贪心法是一种解决最优问题的策略。它是从问题的初始解出发，按照当前最佳的选择，把问题归纳为更小的相似的子问题，并使子问题最优，再由子问题来推导出全局最优解。

# 概述

贪心只是一种思想，不是一种算法。

贪心法是一种解决最优问题的策略。它是从问题的初始解出发，按照当前最佳的选择，把问题归纳为更小的相似的子问题，并使子问题最优，再由子问题来推导出全局最优解。

如何观察并知道是否有这个性质：猜想 + 证明！(数学化的角度)

# 概述

贪心只是一种思想，不是一种算法。

贪心法是一种解决最优问题的策略。它是从问题的初始解出发，按照当前最佳的选择，把问题归纳为更小的相似的子问题，并使子问题最优，再由子问题来推导出全局最优解。

如何观察并知道是否有这个性质：猜想 + 证明！(数学化的角度)

一般没有贪心性质的题只能往 dp 方向考虑。dp 相对贪心来说更加全面，但是往往会拥有更大的复杂度。

# 帮大家整理好的贪心证明方法关键

常用的证明法有反证（比较）法，划归法，归纳法。

# 帮大家整理好的贪心证明方法关键

常用的证明法有反证（比较）法，划归法，归纳法。

1. 反证法：

对于当前的贪心策略，假设存在比它不同而且更优的解，证明该解不存在。证得满足条件。

或者我们证明我们在贪心策略的基础上做任何改动都不能使答案变优，则贪心策略适用。

# 帮大家整理好的贪心证明方法关键

常用的证明法有反证（比较）法，划归法，归纳法。

1. 反证法：

对于当前的贪心策略，假设存在比它不同而且更优的解，证明该解不存在。证得满足条件。

或者我们证明我们在贪心策略的基础上做任何改动都不能使答案变优，则贪心策略适用。

2. 划归法：贪心策略时，把问题构造成已知的算法或数据结构去证明正确性。

# 帮大家整理好的贪心证明方法关键

常用的证明法有反证（比较）法，划归法，归纳法。

1. 反证法：

对于当前的贪心策略，假设存在比它不同而且更优的解，证明该解不存在。证得满足条件。

或者我们证明我们在贪心策略的基础上做任何改动都不能使答案变优，则贪心策略适用。

2. 划归法：贪心策略时，把问题构造成已知的算法或数据结构去证明正确性。

3. 归纳法：归纳证明证明前 i 步是最优解，从而证到全局解最优。

或者你证明你每一步，对后面一定没有坏处，一定是最优的局面之一。

# 帮大家整理好的贪心证明方法关键

常用的证明法有反证（比较）法，划归法，归纳法。

1. 反证法：

对于当前的贪心策略，假设存在比它不同而且更优的解，证明该解不存在。证得满足条件。

或者我们证明我们在贪心策略的基础上做任何改动都不能使答案变优，则贪心策略适用。

2. 划归法：贪心策略时，把问题构造成已知的算法或数据结构去证明正确性。

3. 归纳法：归纳证明证明前 i 步是最优解，从而证到全局解最优。

或者你证明你每一步，对后面一定没有坏处，一定是最优的局面之一。

其实贪心的证明环节其实也是一种对于你贪心直觉和思维的培养，提升你对于一个贪心算法的正确或错误性的判断能力，能更好地帮助你在赛场上面发挥。

# 常见运用策略的算法

一些常见的经典的运用到贪心的算法：

(1) 单源最短路（dijkstra）
(2) 最小生成树（kruskal+prim）
(3) 哈夫曼编码（合并果子）

# 哈夫曼编码（合并果子）

一篇文章有 $n$ 个单词，其中第 $i$ 个单词的出现次数为 $w_i$。你要用 2 进制串 $s_i$ 替换第 $i$ 种单词。要求：对于任意 $i! = j$，都有 $s_i$ 不是 $s_j$ 的前缀。要使替换后的文章总长度最小。

# 哈夫曼编码（合并果子）

一篇文章有 $n$ 个单词，其中第 $i$ 个单词的出现次数为 $w_i$。你要用 2 进制串 $s_i$ 替换第 $i$ 种单词。要求：对于任意 $i! = j$，都有 $s_i$ 不是 $s_j$ 的前缀。要使替换后的文章总长度最小。

给你 $n$ 个果子，每次可以选择两堆果子，支付这堆果子合计的代价可以将二者合并成一堆。然后最小化全部合成一堆的总代价。

# 哈夫曼编码（合并果子）

一篇文章有 $n$ 个单词，其中第 $i$ 个单词的出现次数为 $w_i$。你要用 2 进制串 $s_i$ 替换第 $i$ 种单词。要求：对于任意 $i! = j$，都有 $s_i$ 不是 $s_j$ 的前缀。要使替换后的文章总长度最小。

给你 $n$ 个果子，每次可以选择两堆果子，支付这堆果子合计的代价可以将二者合并成一堆。然后最小化全部合成一堆的总代价。

证明思路：画出合并树，并且归纳证明合并两个最小的果子是不亏（最优的）。

# 哈夫曼编码（合并果子）

一篇文章有 $n$ 个单词，其中第 $i$ 个单词的出现次数为 $w_i$。你要用 2 进制串 $s_i$ 替换第 $i$ 种单词。要求：对于任意 $i! = j$，都有 $s_i$ 不是 $s_j$ 的前缀。要使替换后的文章总长度最小。

给你 $n$ 个果子，每次可以选择两堆果子，支付这堆果子合计的代价可以将二者合并成一堆。然后最小化全部合成一堆的总代价。

证明思路：画出合并树，并且归纳证明合并两个最小的果子是不亏（最优的）。

《荷马史诗》。

一篇文章有 $n$ 个单词，其中第 $i$ 个单词的出现次数为 $w_i$。你要用 $k$ 进制串 $s_i$ 替换第 $i$ 种单词。要求：对于任意 $i! = j$，都有 $s_i$ 不是 $s_j$ 的前缀。要使替换后的文章总长度最小。

# 哈夫曼编码（合并果子）

一篇文章有 $n$ 个单词，其中第 $i$ 个单词的出现次数为 $w_i$。你要用 2 进制串 $s_i$ 替换第 $i$ 种单词。要求：对于任意 $i! = j$，都有 $s_i$ 不是 $s_j$ 的前缀。要使替换后的文章总长度最小。

给你 $n$ 个果子，每次可以选择两堆果子，支付这堆果子合计的代价可以将二者合并成一堆。然后最小化全部合成一堆的总代价。

证明思路：画出合并树，并且归纳证明合并两个最小的果子是不亏（最优的）。

《荷马史诗》。

一篇文章有 $n$ 个单词，其中第 $i$ 个单词的出现次数为 $w_i$。你要用 $k$ 进制串 $s_i$ 替换第 $i$ 种单词。要求：对于任意 $i! = j$，都有 $s_i$ 不是 $s_j$ 的前缀。要使替换后的文章总长度最小。

给你 $n$ 个果子，每次可以选择 $k$ 堆果子，支付这堆果子合计的代价可以将二者合并成一堆。然后最小化全部合成一堆的总代价。

# 区间调度问题/任务分配

一个学生有 $N$ 个任务，每项工作分别在 $s_i$ 时间开始，$t_i$ 结束。对于每个任务来说至多只能有一个学生去完成，而且每个学生完成任务的时间段不能重叠，最大化能完成的任务数量。

# 区间调度问题/任务分配

一个学生有 $N$ 个任务，每项工作分别在 $s_i$ 时间开始，$t_i$ 结束。对于每个任务来说至多只能有一个学生去完成，而且每个学生完成任务的时间段不能重叠，最大化能完成的任务数量。

$m$ 个学生有 $N$ 个任务，每项工作分别在 $s_i$ 时间开始，$t_i$ 结束。对于每个任务来说至多只能有一个学生去完成，而且每个学生完成任务的时间段不能重叠，最大化能完成的任务数量。

# ddl 问题

一个学生有 $N$ 个任务，每个任务有一个 $d_i, t_i$。$d_i$ 表示截止时间，$t_i$ 代表该任务要花多少时间。若完成时间大于截止时间，则会造成 $d_i - t_i$ 的损失。最小化最大的损失。

# ddl 问题

一个学生有 $N$ 个任务，每个任务有一个 $d_i, t_i$。$d_i$ 表示截止时间，$t_i$ 代表该任务要花多少时间。若完成时间大于截止时间，则会造成 $d_i - t_i$ 的损失。最小化最大的损失。

$m$ 个学生有 $N$ 个任务，每个任务有一个 $d_i, t_i$。$d_i$ 表示截止时间，$t_i$ 代表该任务要花多少时间。若完成时间大于截止时间，则会造成 $d_i - t_i$ 的损失。最小化最大的损失。

# NOIP2012 国王游戏

恰逢 $H$ 国国庆，国王邀请 $n$ 位大臣来玩一个有奖游戏。首先，他让每个大臣在左、右手上面分别写下一个整数，国王自己也在左、右手上各写一个整数。然后，让这 $n$ 位大臣排成一排，国王站在队伍的最前面。排好队后，所有的大臣都会获得国王奖赏的若干金币，每位大臣获得的金币数分别是：排在该大臣前面的所有人的左手上的数的乘积除以他自己右手上的数，然后向下取整得到的结果。

国王不希望某一个大臣获得特别多的奖赏，所以他想请你帮他重新安排一下队伍的顺序，使得获得奖赏最多的大臣，所获奖赏尽可能的少。注意，国王的位置始终在队伍的最前面。

# 玩具车

Jasio 喜欢玩玩具，他有 $n$ 个不同的玩具放在了架子上。但是由于场地限制，任何时刻地板上都不会有超过 $k$ 个玩具。

Jasio 每一个时刻都会想要玩一个玩具，所以需要这个玩具出现在地面上，否则她妈妈就需要从架子上拿一个玩具放到地面上，如果地面已经满 $k$ 个玩具了，还要将一个玩具放回架子上。

他的妈妈很清楚自己的孩子，所以他能够预料到 Jasio 想玩些什么玩具。所以她想尽量的使自己去架子上拿玩具的次数尽量的少。

$n, k, p$ (1<=k<=n <=100000, 1<=p<=500000),

# 打游戏

小强在游戏遇到了 $n$ 个怪物。每个怪物有一个生命值，第 $i$ 个怪物的生命值是 $h_i$。而小强有一个属性为魔法值 $m$。

小强和怪物们依次行动。每一回合，小强先行动，然后怪物们同时行动。小强每次可以选择以下行动之一：

- 普通攻击：令某个怪物的生命值减少 1。
- 重击：消耗 1 魔法值，令某个怪物的生命值减少 2。
- 群体攻击：消耗 1 魔法值，令全体怪物的生命值减少 1。

而每个存活的怪物（生命值严格大于 0）每次会令小强的生命值减少 1。假设小强有足够的生命值来维持存活，小强想知道自己至少需要被消耗多少生命值才能击败所有怪物。

# CF1012D

给定字符串 $s, t$，其中只包含小写字母 $a$ 和 $b$，而且 $a$ 和 $b$ 至少在任意一个字符串中各出现一次。

现在允许你执行一种操作：交换 $a$ 的一段前缀和 $b$ 的一段前缀 (可以为空)。

问至少执行多少次操作，使得一个字符串为全 $a$，另一个字符串为全 $b$，并输出方案。

# CF360E

已知 $N$ 个点，$M+K$ 条有向边及边权，可能有重边和自环。

可以任意修改后面的给定 $K$ 条边中一些边的边权，每条边有一个修改范围 $[l_i, r_i]$。

修改给定 K 条边的长度使从 $s1$ 到 $f$ 比 $s2$ $f$ 的时间短，判断是否可以并输出方案。

$(1\ n, m\ 10^4, 1\ k\ 100)$

# 二分

## 二分查找

在一个长度为 $n$ 的有序数列中找出某个数 $X$ 的位置。或是寻找不大于某个数的最大值，和不小于某个数的最小值。

例：在 1 2 4 5 7 8 10 12 中寻找数 4 的位置。

设置递归函数 $search(l, r, x)$ 表示查询数 $x$ 在区间 $[l, r]$ 内的位置，令 $mid = (l + r)/2$，若 $x \leq a[mid]$ 则递归运行 $search(l, mid, x)$，否则运行 $search(mid + 1, r, x)$。边界：$l = r$。

# 二分

## 二分答案

对于离散区间：若当前的答案可能存在的区间为 $[l, r]$，那试探取中间值 $mid$，判断中间值 $mid$ 是否符合答案条件，然后判断继续递归进入 $[l, mid]$ 区间内还是 $[mid + 1, r]$ 区间内。直到 $l = r$。

对于连续区间：若当前的答案可能存在的区间为 $[l, r)$，那试探取中间值 $mid$，判断 $mid$ 是否在符合答案条件，然后判断递归进入在 $[l, mid)$ 区间内还是 $[mid, r)$ 区间内。直到 $l$ 和 $r$ 之间的差足够小。

本质是问题思路的转换，将求某个符合条件的最大值和最小值转换为判断答案是否小于等于某个数。

经典复杂度分析：

$T(n, k) = T(n, k/2) + O(n) \rightarrow T(n, k) = O(n \log k)$

$T(n, k) = T(n, k/2) + O(1) \rightarrow T(n, k) = O(\log k)$

# NOIP2012 借教室

我们需要处理 $n$ 天的借教室信息，其中第 $i$ 天学校有 $r_i$ 个教室可供租借。共有 $m$ 份订单，每份订单表示从第 $s_j$ 天到第 $t_j$ 天，每天租 $d_j$ 个教室。即对于每份订单，我们只需要每天提供 $d_j$ 个教室。订单先到先得，如果所有订单均满足则输出 $0$，否则不满足输出 $-1$ 第一个无法满足的订单。

$1 \leq n, m \leq 10^6$

# codeforces562 Div2 C

有 $n$ 个数，每次可以选择任意 $k$ 个数，将他们 $+1$ 并对 $m$ 取模。问最少进行多少次操作，使得序列是非递减的。

# Codeforces 985D

你有 $n$ 堆沙子，最左边的沙子的最大高度不能超过 $H$，让你在一个从 1 到正无穷的一维平面内放沙子，且要满足相邻两个坐标的沙子的高度不能超过 1。问沙子能够占用的最少坐标点的个数。

例如：

$n = 5, H = 2, [2, 2, 1, 0, ...], [2, 1, 1, 1, 0, ...], [1, 0, 1, 2, 1, 0, ...]$

$n = 6, H = 8, [3, 2, 1, 0, ...], [2, 2, 1, 1, 0, ...], [0, 1, 0, 1, 2, 1, 1, 0...]$

# 第 K 大区间

定义一个区间的值为其众数出现的次数。

现给出 $n$ 个数，求将所有区间的值排序后，第 $K$ 大的值为多少。

# IOI2013 玩具

小沐把玩具扔在地板上，乱七八糟。庆幸的是，有一种特殊的机器人可以收拾玩具。不过他需要确定哪个机器人去拣哪个玩具。

一共有 $T$ 个玩具，整数 $w[i]$ 表示这个玩具的重量，整数 $s[i]$ 表示这个玩具的体积。机器人有两种，分别是：弱机器人和小机器人。

有 $A$ 个弱机器人。每个弱机器人有一个重量限制 $x[i]$，它只能拿起重量严格小于 $x[i]$ 的玩具，与玩具的体积大小没有关系。

有 $B$ 个小机器人。每个小机器人有一个体积限制 $y[i]$，它只能拿起体积严格小于 $y[i]$ 的玩具，与玩具的重量大小没有关系。

每个机器人用 1 分钟将一个玩具拿走放好。不同的机器人可以同时拿走并放好不同的玩具。

你的任务是确定机器人是否可以将所有玩具都收拾好，

# NOIP2018 赛道修建

给你一棵 $n(n \leq 5\times10^4)$ 个结点的树，从中找出 $m$ 个没有公共边的路径，使得最短路径最长。

# What

倍增思想是什么呢？

# What

倍增思想是什么呢？
倍增，顾名思义，就是每次增加一倍。

# What

倍增思想是什么呢？

倍增，顾名思义，就是每次增加一倍。

展开来说，就是每次根据已经得到的信息，将考虑的范围增加一倍，从而加速操作。

# What

倍增思想是什么呢？

倍增，顾名思义，就是每次增加一倍。

展开来说，就是每次根据已经得到的信息，将考虑的范围增加一倍，从而加速操作。

这是一种非常巧妙的思想，可以用来解决信息学竞赛中的很多问题，一个经典应用是后缀数组的构造。

# How

倍增思想的最基本最通用模型：

# How

倍增思想的最基本最通用模型：

考虑这样一个比较一般的模型，在一个有向图中，每个点最多只有一条出边，每条边有一定的信息，走过一条路径时，就将路径上边的信息依次按一定的规则合并，并且合并的规则满足结合律。

# How

那么，对于点 $x$，我们可以倍增地得到它经过 $2^i$ 条边后到达的点 $next(x,i) = next(next(x,i-1),i-1)$，并维护路径上边的信息合并后的结果
$info(x,i) = merge(info(x,i-1), info(next(x,i-1),i-1))$。

# How

那么，对于点 $x$，我们可以倍增地得到它经过 $2^i$ 条边后到达的点 $next(x,i) = next(next(x,i-1),i-1)$，并维护路径上边的信息合并后的结果
$info(x,i) = merge(info(x,i-1), info(next(x,i-1),i-1))$。

预处理出上面的内容后，我们就要回答有关这个图的询问，这就是我们接下来讨论的主要内容。

# Example 1

求 $x^y \mod p (1 \leq x, y, p \leq 10^9)$。

# Example 1

```
int pow(int x, int y) {
    int t = 1;
    for (; y; y >>= 1) {
        if (y & 1)
            t = (long long)t * x % p;
        x = (long long)x * x % p;
    }
    return t;
}
```

# 传统模型：RMQ

给出长度为 $n(1 \leq n \leq 10^5)$ 的序列 $a$ 和 $q(1 \leq q \leq 10^7)$ 组形如 $(l, r)$ 的询问，每次询问 $\min_{i=l}^{r} a_i$。

# RMQ

联系前面的模型，第 $i$ 项指向第 $i+1$ 项，边权是 $a_i$，合并的规则是取 min。

# RMQ

联系前面的模型，第 $i$ 项指向第 $i+1$ 项，边权是 $a_i$，合并的规则是取 min。

```
for (int i = 1; i <= n; ++i) {
    lg[i] = lg[i - 1] + (1 << lg[i - 1] + 1 == i);
    info[i][0] = a[i];
}
for (int k = 1; k <= lg[n]; ++k)
    for (int i = 1; i + (1 << k) - 1 <= n; ++i)
    info[i][k] = min(info[i][k - 1], info[i + (1 << k - 1)
```

# RMQ

仍然使用二进制拆分回答询问。

# RMQ

仍然使用二进制拆分回答询问。
时间复杂度 $O((n+q)\log n)$？

# RMQ

仍然使用二进制拆分回答询问。
时间复杂度 $O((n+q)\log n)$？
取 min 的特殊性？

# RMQ

仍然使用二进制拆分回答询问。

时间复杂度 $O((n+q)\log n)$？

取 min 的特殊性？

有重叠部分仍然可以合并！对于询问 $(l,r)$，令 $k=\lfloor \log_2(r-l+1) \rfloor$，则答案为 $\min\{info(l,k), info(r-2^k+1,k)\}$。

# RMQ

仍然使用二进制拆分回答询问。

时间复杂度 $O((n+q)\log n)$？

取 min 的特殊性？

有重叠部分仍然可以合并！对于询问 $(l,r)$，令 $k=\lfloor \log_2(r-l+1) \rfloor$，则答案为 $\min\{info(l,k), info(r-2^k+1,k)\}$。

时间复杂度 $O(n\log n+q)$。

# 传统模型：LCA

给出一棵 $n(1 \leq n \leq 10^5)$ 个点的有根树和 $q(1 \leq q \leq 10^5)$ 组形如 $(x, y)$ 的询问，每次询问 $LCA(x, y)$ 即 $x$ 和 $y$ 的最近公共祖先。

# 传统模型：LCA

给出一棵 $n(1 \leq n \leq 10^5)$ 个点的有根树和 $q(1 \leq q \leq 10^5)$ 组形如 $(x, y)$ 的询问，每次询问 $LCA(x, y)$ 即 $x$ 和 $y$ 的最近公共祖先。

同时还可以引申出各种两点之间的最小/最大权值，gcd，以及任意满足结合律的信息。

# LCA

树就是模型中的图？不过在这里我们不关心边权。

# LCA

树就是模型中的图？不过在这里我们不关心边权。

在 DFS 或 BFS 遍历这棵树的过程中，走到点 $x$ 时我们就可以完成对点 $x$ 的预处理。

```
for (int k = 1; k <= lg[dep[x]]; ++k)
    nxt[x][k] = nxt[nxt[x][k - 1]][k - 1];
```

# LCA

怎么回答询问呢？一起往上爬？

# LCA

怎么回答询问呢？一起往上爬？
$x$ 和 $y$ 深度不同？

# LCA

怎么回答询问呢？一起往上爬？
$x$ 和 $y$ 深度不同？
先爬到同一深度！

# LCA

怎么回答询问呢？一起往上爬？

$x$ 和 $y$ 深度不同？

先爬到同一深度！

不妨设 $dep(x) > dep(y)$，我们只要对 $dep(x) - dep(y)$ 进行二进制拆分，就可以使 $x$ 爬到 $y$ 的深度。

# LCA

怎么回答询问呢？一起往上爬？

$x$ 和 $y$ 深度不同？

先爬到同一深度！

不妨设 $dep(x) > dep(y)$，我们只要对 $dep(x) - dep(y)$ 进行二进制拆分，就可以使 $x$ 爬到 $y$ 的深度。

已经到同一个点了？完工！

# LCA

否则就可以一起往上爬了？

# LCA

否则就可以一起往上爬了？

我们要求的是 $x$ 和 $y$ 一起往上爬且不到达同一个点经过的极限边数。似曾相识？

# LCA

否则就可以一起往上爬了？

我们要求的是 $x$ 和 $y$ 一起往上爬且不到达同一个点经过的极限边数。似曾相识？

同样是从二进制高位向低位确定！

# LCA

否则就可以一起往上爬了？

我们要求的是 $x$ 和 $y$ 一起往上爬且不到达同一个点经过的极限边数。似曾相识？

同样是从二进制高位向低位确定！

别忘了经过极限边数到达的点的父亲才是我们要的。

# LCA

```
int lca(int x, int y) {
    if (dep[x] < dep[y])
        swap(x, y);
    while (dep[x] > dep[y])
        x = nxt[x][lg[dep[x] - dep[y]]];
    if (x == y)
        return x;
    for (int k = lg[dep[x]]; k >= 0; --k)
        if (nxt[x][k] != nxt[y][k])
            x = nxt[x][k], y = nxt[y][k];
    return nxt[x][0];
}
```

# LCA

```
int lca(int x, int y) {
    if (dep[x] < dep[y])
        swap(x, y);
    while (dep[x] > dep[y])
        x = nxt[x][lg[dep[x] - dep[y]]];
    if (x == y)
        return x;
    for (int k = lg[dep[x]]; k >= 0; --k)
        if (nxt[x][k] != nxt[y][k])
            x = nxt[x][k], y = nxt[y][k];
    return nxt[x][0];
}
```

时间复杂度 $O((n+q)\log n)$。

# LCA

维护其他信息？预处理一样处理，在倍增跳跃的时候同时统计信息即可。

# Codeforces 587C

给出若干询问，求树上两点路径中的前 $a(<=10)$ 小的点权值。数据范围都在 $10^5$ 范围内。

# 开车旅行

$n(1 \leq n \leq 10^5)$ 个城市在一直线上，自西向东编号为 $1 \sim n$，且有互不相同的海拔高度。两城市间距离为他们海拔高度差的绝对值。

A, B 两人轮流开车，A 先开，之后每天轮换。他们从起点 $s$ 出发一直向东开，B 每次沿前进方向选最近的城市作为目的地，A 则选第二近的（距离相同时离海拔低的更近），当其中一人无法选择目的地或到达目的地将会使总距离超过 $x$ 则停止。

给出 $m(1 \leq m \leq 10000)$ 组形如 $(s, x)$ 的询问，每次询问 A 和 B 分别开的距离。

# 最大公约数

给出长度为 $n(1 \leq n \leq 10^5)$ 的序列 $a(1 \leq a_i \leq 10^{12})$，定义其任意连续子序列 $a_l, a_{l+1}, \ldots, a_r$ 的权值为 $(r-l+1) \times \gcd(a_l, a_{l+1}, \ldots, a_r)$，求权值最大的连续子序列的权值。

# 排列问题

给出一个长度为 $n$ 的 $1-n$ 的排列，再给定一个长度为 $m$ 的每个数字在 $1-n$ 范围内的序列。然后给定 $q$ 个询问，每次询问 $[l,r]$ 这个区间内是否包含一个子序列是给定排列，或给定排列的轮换形式。

$1 \leq n, m \leq 10^5, 1 \leq q \leq 10^7$.

# CodeForces 97 E. Leaders

给你一个有 $n$ 个点 $m$ 条边的无向图，有 $q$ 次询问，每次询问两个点 $u, v$ 之间是否存在长度为奇数的简单路径。
$1 \leq n, m, q \leq 10^5$.

# 区间覆盖问题

给你 $n$ 个区间，形如 $[x, y]$，每次询问给出一组 $[l, r]$，问 $[l, r]$ 中最多可以取出多少个不同的区间。

# 区间覆盖问题

给你 $n$ 个区间，形如 $[x, y]$，每次询问给出一组 $[l, r]$，问 $[l, r]$ 中最多可以取出多少个不同的区间。

给你 $n$ 个区间，形如 $[x, y]$，每次询问给出一组 $[l, r]$，问最少要多少个区间才能完美覆盖 $[l, r]$。