俄罗斯南部和伏尔加地区竞赛

萨拉托夫国立大学

18 2024 .

有两种情况：一辆公共汽车到站时*有* b 个空座位，p 人到站。需要确定每辆公交车上是否有空余座位。

J.等待…

- 我们将维持现有的停靠站人数。

J.等待...

- 我们将维持现有的停靠站人数。
- 如果有 p 人到达，我们将执行 $s \mathrel{+}= p$

- 我们将维持现有的停靠站人数。
- 如果有 p 人到达，我们将执行 $s \mathrel{+}= p$
- 如果一辆巴士到达时有 b 个座位，我们将比较 b 和 s

- 我们将维持现有的停靠站人数。
- 如果有 p 人到达，我们将执行 $s += p$
- 如果一辆巴士到达时有 b 个座位，我们将比较 b 和 s
- 如果 $b > s$，答案为 "是"，否则为 "否"。

- 我们将维持现有的停靠站人数。
- 如果有 p 人到达，我们将执行 s += p
- 如果一辆巴士到达时有 b 个座位，我们将比较 b 和 s
- 如果 $b > s$，则答案为 "是"，否则为 "否"
- 然后我们将执行 s -= min(b, s)

N.固定表达式

有这样一个逻辑表达式，一个数字小于/大于/等于另一个数字，改变其中最小的符号个数即可使其为真。

N.固定表达式

- 您可以更改数字之间的符号，而不是更改数字

N.固定表达式

- 您可以更改数字之间的符号，而不是更改数字
- 如果表达式已经正确，符号将变为相同的符号，即不会有任何变化

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 贪婪的想法

L.桥梁翻新

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 贪婪的想法
- 创建成对的 25 + 25 是亲子表

L.桥梁翻新

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 贪婪的想法
- 创建成对的 25 + 25 是亲子表
- 需要$^{n}_{2}$ 对 25 + 25

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 贪婪的想法
- 创建成对的 25 + 25 是亲子表
- 需要一对 25 + 25
  2
- 创建成对的 21 + 21 + 18 是亲桌

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 贪婪的想法
- 创建成对的 25 + 25 是亲子表
- 需要一对 25 + 25
- 创建成对的 21 + 21 + 18 是亲桌
- 需要$\frac{n}{2}$ 对 21 + 21 + 18

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 贪婪的想法
- 创建成对的 25 + 25 是亲子表
- 需要一对 25 + 25
  2
- 创建成对的 21 + 21 + 18 是亲桌
- 需要一对 21 + 21 + 18
  2
- 此外，需要 (n mod 2) 对 25 + 21

L.桥梁翻新

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 贪婪的想法
- 创建成对的 25 + 25 是亲子表
- 需要一对$_2$ 25 + 25
- 创建成对的 21 + 21 + 18 是亲桌
- 需要一对$_2$ 21 + 21 + 18
- 此外，需要 (n mod 2) 对 25 + 21
- 木板 25 和 21 恰好需要 n *块* 60 的木板

L.桥梁翻新

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 还有^{n} 18 块木板$_{2}$

L.桥梁翻新

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 还有 18 块木板 $\frac{n}{2}$
- 创建成对的 18 + 18 + 18 是亲子桌

L.桥梁翻新

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 还有 18 块木板$_{2}$
- 创建成对的 18 + 18 + 18 是亲子桌
- 需要 $\frac{\lceil \frac{n}{2} \rceil}{3} = \frac{n}{6}$ 一对 18 + 18 + 18

L.桥梁翻新

将长度为 60 的木板分成 18、21 和 25 块，每种长度 n 块。

- 还有 18 块木板 $_2$
- 创建成对的 18 + 18 + 18 是亲子桌
- 需要 $\left\lceil \frac{n}{\frac{2}{3}} \right\rceil = \frac{n}{6}$ 一对 18 + 18 + 18

问题答案：$n + \frac{n}{6}$

有一个数字列表。您需要选择其中的 8 个数字，并用它们作为矩形四角的坐标，该矩形的边平行于

坐标轴。矩形的面积应尽可能大。

- 数字必须成对出现，因为每条水平线/垂直线上必须有 2 个点

- 数字必须成对出现，因为每条水平线/垂直线上必须有 2 个点
- 如果没有 4 对完全相同的元素，则答案为 "否"。

C.DIY

- 数字必须成对出现，因为每条水平线/垂直线上必须有 2 个点
- 如果没有 4 对完全相同的元素，则答案为 "否"。
- 否则，我们来压缩列表，这样，如果原始列表中有 x 次出现，在压缩列表中就会变成 $\lfloor \frac{x}{2} \rfloor$ 次出现。

- 对压缩列表进行排序我们称新列表为 b，其长度为 m

- 对压缩列表进行排序我们称新列表为 b，其长度为 m
- 在每个坐标轴上，左坐标越小，右坐标越大。

  右边，矩形的尺寸就越大

- 对压缩列表进行排序我们称新列表为 b，其长度为 m
- 在每个坐标轴上，左坐标越小，右坐标越大。

  右边，矩形的尺寸就越大
- 您总是可以将元素 b_1, b_2, b_{m-1}, b_m

- 对压缩列表进行排序我们称新列表为 b，其长度为 m
- 在每个坐标轴上，左坐标越小，右坐标越大。

  右边，矩形的尺寸就越大
- 您总是可以将元素 b_1, b_2, b_{m-1}, b_m
- 可以证明，将成对（b_1, b_{m-1}）和 $(b_2, b\ )_m$

A.奖励项目

n 名开发人员正在开发该项目。要完成这个项目，他们必须总共花费 k 个小时。开发人员轮流命名一个整数的

c_i ，每个人将为该项目工作几个小时。如果他们完成了项目，第 i 位将获得 $a_i - c_i \cdot b_i$ 的奖金。如果每个开发者都想最大限度地提高自己的收益，那么每个人将花费多少小时?

A.奖励项目

- 每个开发人员的工作时数都有限制，超过限制就会出现亏损

A.奖励项目

- 每个开发人员的工作时数都有限制，超过限制就会出现亏损
- 如果限值之和小于 k，则所有答案均为 0

A.奖励项目

- 每个开发人员的工作时数都有限制，超过限制就会出现亏损
- 如果极限之和小于 k，则所有答案都是 0
- 让我们从最后一个开发者到第一个开发者解决问题

A.奖励项目

- 每个开发人员的工作时数都有限制，超过限制就会出现亏损
- 如果极限之和小于 k，则所有答案都是 0
- 让我们从最后一个开发者到第一个开发者解决问题
- 最后一名开发人员知道项目还剩多少工作时间

A.奖励项目

- 每个开发人员的工作时数都有限制，超过限制就会出现亏损
- 如果极限之和小于 k，则所有答案都是 0
- 让我们从最后一个开发者到第一个开发者解决问题
- 最后一名开发人员知道项目还剩多少工作时间
- 如果他们的限额少于这个数，他们就不会工作，也就没有人领取补贴。

A.奖励项目

- 为了倒数第二位开发人员的利益，应尽可能少地工作，但要足够多，以便最后一位开发人员不会拒绝工作

A.奖励项目

- 为了倒数第二位开发人员的利益，应尽可能少地工作，但要足够多，以便最后一位开发人员不会拒绝工作
- 最理想的情况是，倒数第二位开发人员留下的工作量正好等于最后一位开发人员留下的工作量。

- 为了倒数第二位开发人员的利益，应尽可能少地工作，但要足够多，以便最后一位开发人员不会拒绝工作
- 最理想的情况是，倒数第二位开发人员留下的工作量正好等于最后一位开发人员留下的工作量。
- 最后一个开发人员将工作 $\min(k,)^{a_i}$

b_i

A.奖励项目

- 为了倒数第二位开发人员的利益，应尽可能少地工作，但要足够多，以便最后一位开发人员不会拒绝工作
- 最理想的情况是，倒数第二位开发人员留下的工作量正好等于最后一位开发人员留下的工作量。
- 最后一个开发人员将工作 $\min(k,)^{a_i}$

b_i

- 删除最后一名开发人员，用 k 减去他们的工时，然后转到

$n - 1$

A.奖励项目

- 为了倒数第二位开发人员的利益，应尽可能少地工作，但要足够多，以便最后一位开发人员不会拒绝工作
- 最理想的情况是，倒数第二位开发人员留下的工作量正好等于最后一位开发人员留下的工作量。
- 最后一个开发人员将工作 $\min(k, \frac{a_i}{b_i})$
- 删除最后一名开发人员，用 k 减去他们的工时，然后转到 $n - 1$

时间复杂性$O(n)$

G. 猜一个角色

陪审团有一个二进制字符串，您需要猜出其中的一个字符。您可以问不超过 3 个这样的问题：某个字符串作为子串在陪审团字符串中出现了多少次?

G. 猜一个角色

- 除最后一个字符外，每个字符都有下一个字符 1 或 0

G. 猜一个角色

- 除最后一个字符外，每个字符都有下一个字符 1 或 0
- 据此，我们可以提出以下查询 1、11 和 10

- 除最后一个字符外，每个字符都有下一个字符 1 或 0
- 据此，我们可以提出以下查询 1、11 和 10
- 除了最后一个 1 之外，每个 1 后面都会有一个字符，因此如果第一个问题的答案等于另外两个问题的答案之和，那么最后一个字符不是 1

B.实现平等

有一个整数数组。让数组由相同的非负数组成，使用最少的运算次数将一个元素减少 2，并将下一个元素（循环）增加 1。

B.实现平等

- 如果每个元素都等于 k，那么再进行 n 次运算，也可以得到 $(k-1)$ 的值

- 如果每个元素都等于 k，那么再进行 n 次运算，也可以得到 $(k - 1)$ 的值
- 您可以对数组的 nal 值使用二进制搜索

- 在二进制搜索中，要检查一个值是否可以达到，我们将循环 x (进行运算，使元素不超过 k）元素，直到数组的总和小于或等于 $k - n$

B.实现平等

- 在二进制搜索中，要检查一个值是否可以达到，我们将循环 x (进行运算，使元素不超过 k）元素，直到数组的总和小于或等于 $k - n$
- 有了这样的 xes，我们就可以执行一些必然需要完成的操作

B.实现平等

- 在二进制搜索中，要检查一个值是否可以到达，我们将循环 x (进行运算，使元素不超过 k）元素，直到数组的总和小于或等于 $k - n$
- 有了这样的 xes，我们就可以执行一些必然需要完成的操作
- 这样的 xes 序列收敛速度为 $O(n + \log A)$，因为在第一轮 xes 之后，只有一个元素大于所需值

K.网格行走

找出从 $(1, 1)$ 到 (n, n) 的最便宜路径，其中 $c(i, j) = \gcd(i, a) + \gcd(j, b)$.

- $P(x, y)$ 是从 $(1, 1)$ 到 (x, y) 的某条路径

K.网格行走

找出从 (1, 1) 到 (n, n) 的最便宜路径，其中 $c(i, j) = \gcd(i, a) + \gcd(j, b)$.

- $P(x, y)$ 是从 (1, 1) 到 (x, y) 的某条路径
- $\sum_{(i,j)\in P} c(i, j) = \sum_{(i,j)\in P} \gcd(i, a) + \gcd(j, b)$

找出从 (1, 1) 到 (n, n) 的最便宜路径，其中 $c(i, j) = \gcd(i, a) + \gcd(j, b)$.

- $P(x, y)$ 是从 (1, 1) 到 (x, y) 的某条路径
- $\sum_{(i,j)\in P} c(i, j) = \sum_{(i,j)\in P} \gcd(i, a) + \gcd(j, b)$
- $\sum_{(i,j)\in P} c(i, j) \geq (x - 1) + (y - 1) + \sum_{i=1}^{x} \gcd(i, a) + \sum_{j=1}^{y} \gcd(j, b)$

找出从 (1, 1) 到 (n, n) 的最便宜路径，其中

$c(i, j) = \gcd(i, a) + \gcd(j, b)$.

- $P(x, y)$ 是从 (1, 1) 到 (x, y) 的某条路径
- $$\sum_{(i,j)\in P} c(i, j) = \sum_{(i,j)\in P} \gcd(i, a) + \gcd(j, b)$$
- $$\sum_{(i,j)\in P} c(i, j) \geq (x - 1) + (y - 1) + \sum_{i=1}^{x} \gcd(i, a) + \sum_{j=1}^{y} \gcd(j, b)$$
- 如果 gcd（x，a） = 1 或 gcd（y，b） = 1，则达到下限

K.网格行走

- 设 x 是最大整数，使得 $x \leq n$ 且 $\gcd(x, a) = 1$

K.网格行走

- 设 x 是最大整数，使得 $x \leq n$ 且 $\gcd(x, a) = 1$
- 设 y 是最大整数，使得 $y \leq n$ 且 $\gcd(y, b) = 1$

K.网格行走

- 设 x 为最大整数，使得 $x \leq n$ 且 $\gcd(x, a) = 1$
- 设 y 为最大整数，使得 $y \leq n$ 且 $\gcd(y, b) = 1$
- 存在一条经过 (x, y) 的最优路径

K.网格行走

- 设 x 为最大整数，使得 $x \leq n$ 且 $\gcd(x, a) = 1$
- 设 y 为最大整数，使得 $y \leq n$ 且 $\gcd(y, b) = 1$
- 存在一条经过 (x, y) 的最优路径
- 路径 $(1, 1)$ - (x, y) 的成本已经已知

K.网格行走

- 设 x 为最大整数，使得 $x \le n$ 且 $\gcd(x, a) = 1$
- 设 y 为最大整数，使得 $y \le n$ 且 $\gcd(y, b) = 1$
- 存在一条经过 (x, y) 的最优路径
- 已知道路径 $(1, 1)$ - (x, y) 的成本
- 需要找到路径 (x, y) - (n, n) 的成本

K.网格行走

- 设 x 为最大整数，使得 $x \leq n$ 且 $\gcd(x, a) = 1$
- 设 y 为最大整数，使得 $y \leq n$ 且 $\gcd(y, b) = 1$
- 存在一条经过 (x, y) 的最优路径
- 路径 $(1, 1)$ - (x, y) 的成本已经已知
- 需要求出路径 (x, y) - (n, n) 的成本
- 对于 $n \leq 10^6$，$n - x \leq 25$ 和 $n - y \leq 25$

K.网格行走

- 设 x 是最大整数，使得 $x \le n$ 且 $\gcd(x, a) = 1$
- 设 y 是最大整数，使得 $y \le n$ 且 $\gcd(y, b) = 1$
- 存在一条经过 (x, y) 的最优路径
- 路径 $(1, 1)$ - (x, y) 的成本已经已知
- 需要求出路径 (x, y) - (n, n) 的成本
- 对于 $n \le 10^6$，$n - x \le 25$ 和 $n - y \le 25$
- $dp[26][26]$

K.网格行走

- 设 x 是最大整数，使得 $x \leq n$ 且 $\gcd(x, a) = 1$
- 设 y 是最大整数，使得 $y \leq n$ 且 $\gcd(y, b) = 1$
- 存在一条经过 (x, y) 的最优路径
- 路径 $(1, 1) - (x, y)$ 的成本已经已知
- 需要求出路径 $(x, y) - (n, n)$ 的成本
- 对于 $n \leq 10^6$，$n - x \leq 25$ 和 $n - y \leq 25$
- $dp[26][26]$

时间复杂性$O(n \log n)$

有 n 名运动员参加的 m 个项目的比赛正在进行。给出一个二进制矩阵，第 i 位参赛选手在第 j 个运动项目中是否很强?

运动 x、$x + 1$、......循环进行，每项运动结束后，弱

如果至少有一个强参与者，则淘汰该参与者。对于每个起始 x，输出获胜者。

I.铁人三项

- 将其视为一个字符串问题

I.铁人三项

- 将其视为一个字符串问题
- 词性最大的运动员获胜

I.铁人三项

- 将其视为一个字符串问题
- 在每个循环移位中，找出最大值的运动员获胜。

- 我们将保持整个行的递减顺序

- 我们将保持整个行的递减顺序
- 有了 $x+1$ 的顺序，我们将重新计算 x

I.铁人三项

- 我们将保持整个行的递减顺序
- 有了 $x+1$ 的顺序，我们将重新计算 x
- 第 x 个位置上有 1 名运动员的运动员按相同顺序移动到前面
- x位置上0的运动员按相同顺序留在后面

I.铁人三项

- 您可以使用弧度排序

I.铁人三项

- 您可以使用弧度排序
- 每次迭代需要 $O(n)$

I.铁人三项

- 您可以使用弧度排序
- 每次迭代需要 $O(n)$
- 我们将计算 $x = m$ 的答案，进行 m 次迭代

I.铁人三项

- 您可以使用弧度排序
- 每次迭代需要 $O(n)$
- 我们将计算 $x = m$ *的*答案，进行 m 次迭代
- 我们将再进行 m 次迭代，每次迭代后都保留最大值。

- 您可以使用弧度排序
- 每次迭代需要 $O(n)$
- 我们将计算 $x = m$ *的*答案，进行 m 次迭代
- 我们将再进行 m 次迭代，每次迭代后都保留最大值 复杂度： O(nm)$O(nm)$

给定一个由 n 个整数组成的数组，你必须计算有多少种方法可以将它切成若干段，从而使这些段上的位相 OR 形成一个

非递减序列。

D.分割或征服

- 考虑具有相同右边界的线段

D.分割或征服

- 考虑具有相同右边界的线段
- 在这些分段中，有对数$_2$ 10^9 不同的 OR

D.分割或征服

- 考虑具有相同右边界的线段
- $_2{}^9$ 使用动态程序设计 dp[i][x]。

D.分割或征服

- 考虑具有相同右边界的线段
- ${}_2{}^9$ 使用动态程序设计 dp[i][x]。
- 将长度为 i 的预 x 切成若干段，使最后一段的 OR 恰好为 x 的方法有几种?

- 对于每个 l, 找出所有对 (r, x) 的最小 r，对其 OR $[l, r)$ 等于 x

D.分割或征服

- 对于每个 l, 找出所有对 (r, x) 的最小 r，对其 OR $[l, r)$ 等于 x
- 从 $l + 1$ 的对子中可以得到 l 的对子

- 对于每个 l, 找出所有对 (r, x) 的最小 r，对其 OR $[l, r)$ 等于 x
- 可从 $l+1$ 的对子中得到 l 的对子
- 按 x 的降序排列对子 (r, x)

D.分割或征服

- 我们将计算动态程序设计的前

- 我们将计算动态程序设计的前向
- 考虑两个相邻对 (r_j, x_j) 和 $(r_{j+1}, x)_{j+1}$

- 我们将计算动态程序设计的前向
- 考虑两个相邻对 (r_j, x_j) 和 (r_{j+1}, x_{j+1})
- 我们需要在 $\sum$ dp$[i][x]$ 对于所有 $x \leq x_j$ 上的 dp 值。
  从 r_j 到 r_{j+1}

D.分割或征服

- 我们将计算动态程序设计的前向
- 考虑两个相邻对 (r_j, x_j) 和 (r_{j+1}, x_{j+1})
- 我们需要在 $\sum$ dp$[i][x]$ 对于所有 $x \le x_j$ 上的 dp 值。
  从 r_j 到 r_{j+1}
- 我们将用扫线法做延迟加法

D.分割或征服

- 我们将计算动态程序设计的前向
- 考虑两个相邻对 (r_j, x_j) 和 (r_{j+1}, x_{j+1})
- 我们需要在 $\sum$ dp$[i][x]$ 对于所有 $x \leq x_j$ 上的 dp 值。从 r_j 到 r_{j+1}
- 我们将用扫线法做延迟加法
- 在位置 r_j，我们将加上，在位置 r_{j+1}，我们将减去相同的值

D.分割或征服

- 在每个位置上，我们都将保留一个形式为（OR 值、dp 值总和）的列表

D.分割或征服

- 在每个位置上，我们都将保留一个形式为（OR 值、dp 值总和）的列表
- 我们将根据扫线活动更新列表

D.分割或征服

- 在每个位置上，我们都将保留一个形式为（OR 值、dp 值总和）的列表
- 我们将根据扫线活动更新列表
- 通过两个指针，我们将计算出位置的所有动力值

D.分割或征服

- 在每个位置上，我们都将保留一个形式为（OR 值、dp 值总和）的列表
- 我们将根据扫线活动更新列表
- 通过两个指针，我们将计算出位置的所有动力值
- 求解复杂度O（$n \log n \log n$）

M.皇家同花顺

有一副从 1 到 4 种花色的牌。我们从中抽取最上面的 5 张牌，每轮可以选择弃牌和留牌。

每次弃牌后，我们都要取下一张牌，直到手中有 5 张牌为止。我们的任务是用最少的预期步数获得王牌。

M.皇家同花顺

- 一共有 n+1 种扑克牌：每种扑克牌都有 n 种花色，其中每种花色的扑克牌都能组成王牌乌希，其他的扑克牌都不能组成王牌乌希。

- 一共有 n+1 种扑克牌：每种扑克牌都有 n 种花色，其中每种花色的扑克牌都能组成王牌乌希，其他的扑克牌都不能组成王牌乌希。
- 皇室牌局中 不需要的牌型可以立即丢弃，但其他牌型的丢弃则比较复杂

- 一共有 n+1 种扑克牌：每种扑克牌都有 *n* 种花色，其中每种花色的扑克牌都能组成王牌乌希，其他的扑克牌都不能组成王牌乌希。
- 皇室牌局中 不需要的牌型可以立即丢弃，但其他牌型的丢弃则比较复杂
- 对于每个状态，我们都需要最优化地选择弃牌，让我们尝试用动态编程来选择吧

- 状态可以用以下参数来描述：牌组中有多少张牌、手头有多少张不同类型的牌

M.皇家同花顺

- 状态可以用以下参数来描述：牌组中有多少张牌、手头有多少张不同类型的牌
- 如果我们丢弃了王牌乌希所需的一张牌，那么就不能再收集该花色的王牌乌希；对于每种花色，我们都需要记录是否还能收集王牌乌希

- 状态可以用以下参数来描述：牌组中有多少张牌、手头有多少张不同类型的牌
- 如果我们丢弃了王牌乌希所需的一张牌，那么就不能再收集该花色的王牌乌希；对于每种花色，我们都需要记录是否还能收集王牌乌希
- 例如，我们可以用-1 来存储无法再收集的卡牌类型的数量，但为此我们需要将手头卡牌的总数作为附加状态来维护

M.皇家同花顺

- 过渡：如果手中牌的数量少于 5 张，我们随机抽取一张牌（dp 值等于我们过渡到的状态的 dp 值的加权和）

M.皇家同花顺

- 过渡：如果手中牌的数量少于 5 张，我们随机抽取一张牌（dp 值等于我们过渡到的状态的 dp 值的加权和）
- 如果手牌数是 5，我们将遍历要弃掉的牌（dp 值等于 1+ 我们过渡到的 dp 值的最小值）

- 过渡：如果手中牌的数量少于 5 张，我们随机抽取一张牌（dp 值等于我们过渡到的状态的 dp 值的加权和）
- 如果手牌数是 5，我们将遍历要弃掉的牌（dp 值等于 1+ 我们过渡到的 dp 值的最小值）
- 如果运行速度太慢，我们可以在本地预先计算答案

一共有 n 个木桶。我们在桶底投掷粘土，以最大限度地增加第一桶中的水量。

- 贪婪的想法

一共有 n 个木桶。我们在桶底投掷粘土，以最大限度地增加第一桶中的水量。

- 贪婪的想法
- 将木桶从右向左排列是亲桌。

E.桶

一共有 n 个木桶。我们在桶底投掷粘土，以最大限度地增加第一桶中的水量。

- 贪婪的想法
- 将枪管从右向左排列是最简单的方法。
- 简化程序：

E.桶

一共有 n 个木桶。我们在桶底投掷粘土，以最大限度地增加第一桶中的水量。

- 贪婪的想法
- 将枪管从右向左排列是最简单的方法。
- 简化程序：
  - 立即投掷 $h[i-1]$ 个单位的粘土

一共有 n 个木桶。我们在桶底投掷粘土，以最大限度地增加第一桶中的水量。

- 贪婪的想法
- 将枪管从右向左排列是最简单的方法。
- 简化程序：
  - 立即投掷 $h[i-1]$ 个单位的粘土
  - 重新计算余量

一共有 n 个木桶。我们在桶底投掷粘土，以最大限度地增加第一桶中的水量。

- 贪婪的想法
- 将枪管从右向左排列是最简单的方法。
- 简化程序：
  - 立即投掷 $h[i-1]$ 个单位的粘土
  - 重新计算余量
  - 用一个额外的粘土单位封桶

E.桶

如何方便地重新计算?

如何方便地重新计算?

- 维护通信船舶分段并与分段合作

如何方便地重新计算?

- 维护通信舱段并处理舱段
- 在一般情况下，我们在 su x 上存储一个堆栈，堆栈中的舱段为

E.桶

如何方便地重新计算?

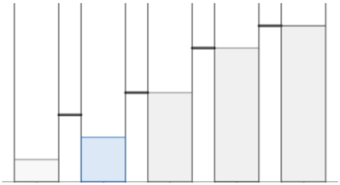
- 维护通信船只的分段，并处理分段
- 在一般情况下，在 su x 上，我们存储一叠已分段
- 水收集在最后一个未分 段 中。

如何方便地重新计算?

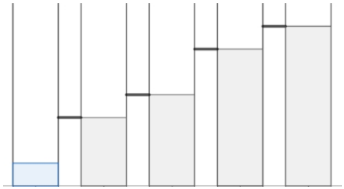
- 维护通信船只的分段，并处理分段
- 在一般情况下，在 su x 上，我们存储一叠已分段
- 水收集在最后一个未分 段 中。
- 2 + 1 个案例:

E.桶

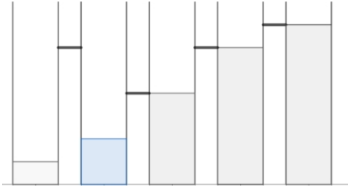
第一个案例



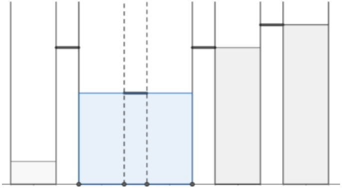
堆栈扩展



第二个案例



分段合并



- 我们将堆栈存储在一个列表中，这样就可以在 $O(1)$ 内从头开始添加/删除，并从尾部删除。

- 我们将堆栈存储在一个列表中，这样就可以在 $O(1)$ 内从头开始添加/删除，并从尾部删除。
- 复杂性总共 $O(n)$。

对于每个 k，计算所有大小为 k 的子集的最小值的最大值。

F.替代平台

对于每个 k，计算所有大小为 k 的子集的最小值的最大值。

第一部分：对于一个 k 值。

对于每个 k，计算所有大小为 k 的子集的最小值的最大值。

第一部分：对于一个 k 值。

- 让 $F(k) = \sum_{S} \mathrm{E}(S)$，其中 $|S| = k$

对于每个 k，计算所有大小为 k 的子集的最小值的最大值。

第一部分：对于一个 k 值。

- 让 $F(k) = \sum_S E(S)$，其中 $|S| = k$
- $$平均_k = \frac{F(k)}{\binom{n}{k}}$$

对于每个 k，计算所有大小为 k 的子集的最小值的最大值。

第一部分：对于一个 k 值。

- 让 $F(k) = \sum_S E(S)$，其中 $|S| = k$
- $$平均_k = \frac{F(k)}{\binom{n}{k}}$$
- $F(k) = \sum_{e=1}^{10^6} e \cdot |\{S \mid E(S) = e\}|$

F.替代平台

对于每个 k，计算所有大小为 k 的子集的最小值的最大值。

第一部分：对于一个 k 值。

- 让 $F(k) = \sum_S E(S)$，其中 $|S| = k$
- $$\text{平均}_k = \frac{F(k)}{\binom{n}{k}}$$
- $F(k) = \sum_{e=1}^{10^6} e \cdot |\{S \mid E(S) = e\}|$
- $F(k) = \sum_{e=1}^{10^6} |\{S \mid E(S) \geq e\}|$

F.替代平台

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$

F.替代平台

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$ add $|\{S \mid \min_{s \in S} v(S) \geq e\}|$

F.替代平台

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$ ● add $|\{S \mid \min_{s \in S} v(S) \geq e\}|$

- $+ \, {}^{p_1}{}_{k}$ 对于某些 p_1

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$

- add $|\{S \mid \min_{s \in S} v(S) \geq e\}|$
- p_1 k + 对于某些 p_1
- 添加 $|\{S \mid \min_{s \in S} r(S) \geq e\}|$

F.替代平台

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$ • add $|\{S \mid \min_{s \in S} v(S) \geq e\}|$

- p_1 k
- 添加 $|\{S \mid \min_{s \in S} r(S) \geq e\}|$ + 对于某些 p_1
- + $\binom{p_2}{k}$ 对于某些 p_2

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$ • add $|\{S \mid \min_{s \in S} v(S) \geq e\}|$

- p_1 k
- 添加 $|\{S \mid \min_{s \in S} r(S) \geq e\}|$ + 对于某些 p_1
- p_2 k + 对于某些 p_2
- 减去 $|\{S \mid \min_{s \in S} v(S)$ 和 $\min_{s \in S} v(S) \geq e\}|$

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$ • add $|\{S \mid \min_{s\in S} v(S) \geq e\}|$

- p_1 k
- + 对于某些 p_1
- 添加 $|\{S \mid \min_{s\in S} r(S) \geq e\}|$
- + 对于某些 p_2 p_2 k
- 减去 $|\{S \mid \min_{s\in S} v(S)$ 和 $\min_{s\in S} v(S) \geq e\}|$ • $m_i = \min(v_i, r_i)$

使用包含/排除公式计算 $|\{S \mid E(S) \geq e\}|$ • add $|\{S \mid \min_{s\in S} v(S) \geq e\}|$

- p_1 k
- 添加 $|\{S \mid \min_{s\in S} r(S) \geq e\}|$ + 对于某些 p_1
- p_2 k + 对于某些 p_2

减去 $|\{S \mid \min_{s\in S} v(S)$ 和 $\min_{s\in S} v(S) \geq e\}|$ • $m_i = \min(v_i, r_i)$

- p_3 $-_k$ for some p_3

$$F(k) = \sum_{i=1}^{n} f_{i}^{i} \quad k$$

F.替代平台

$$F(k) = \sum_{i=1}^{n} f_i k^i$$

- $f_i = (v_{i-1} - v_i) + (r_{i-1} - r_i) - (m_{i-1} - m_i)$

F.替代平台

$$F(k) = \sum_{i=1}^{n} f_i k^i$$

- $f_i = (v_{i-1} - v_i) + (r_{i-1} - r_i) - (m_{i-1} - m_i)$
- f_i 不依赖于 k

F.替代平台

$$F(k) = \sum_{i=1}^{n} f_i \binom{i}{k}$$

- $f_i = (v_{i-1} - v_i) + (r_{i-1} - r_i) - (m_{i-1} - m_i)$
- f_i 不依赖于 k

$$F(k) = \frac{1}{k!} \sum_{i=1}^{n} f_i \cdot \frac{i!}{(i-k)!}$$

F.替代平台

$$F(k) = \frac{1}{k!} \sum_{i=1}^{n} \frac{f_i \cdot i!}{(i-k)!}$$

$$F(k) = \frac{1}{k!} \sum_{i=1}^{n} \frac{f_i \cdot i!}{(i-k)!}$$

- FFT (NTT)

$$F(k) = \frac{1}{k!}\sum_{i=1}^{n} \frac{f_i \cdot i!}{(i-k)!}$$

- FFT (NTT)
- $a = [f_1\, 1!, f_2\, 2!, \ldots, f_n\, n!, 0, \ldots]$。

$$F(k) = \frac{1}{k!} \sum_{i=1}^{n} \frac{f_i \cdot i!}{(i-k)!}$$

- FFT（NTT）
- $a = [f_1\ 1!, f_2\ 2!, \ldots, f_n\ n!, 0, \ldots]$。
- $b = [\frac{1}{n!}$、$\frac{1}{(n-1)!}, \ldots, \frac{1}{0!}, 0, \ldots]$

$$F(k) = \frac{1}{k!}\sum_{i=1}^{n} \frac{f_i \cdot i!}{(i-k)!}$$

- FFT (NTT)
- $a = [f_1\, 1!, f_2\, 2!, \ldots, f_n\, n!, 0, \ldots]$。
- $b = [\frac{1}{n!}, \frac{1}{(n-1)!}, \ldots, \frac{1}{0!}, 0, \ldots]$

$$F(k) = \frac{1}{k!}(a \times b)[n-1+k]。$$

$$F(k) = \frac{1}{k!} \sum_{i=1}^{n} \frac{f_i \cdot i!}{(i-k)!}$$

- FFT (NTT)
- $a = [f_1\, 1!, f_2\, 2!, \ldots, f_n\, n!, 0, \ldots]$。
- $b = [\frac{1}{n!}, \frac{1}{(n-1)!}, \ldots, \frac{1}{0!}, 0, \ldots]$

$$F(k) = \frac{1}{k!}(a \times b)[n-1+k]。$$

时间复杂性$O(n \log n)$

H.银河理事会

正式说明：有一个数组，初始时所有值都等 于 0。我们进行 m 次操作：将不是最左边最大值的元素增加 1。每次操作后，都需要确定最左边最大值的位置。

- 第一个查询将第一个最大值的元素无限增加 1

- 第一个查询将第一个最大值的元素无限增加 1
- 对于每个后续查询，有两种情况

- 第一个查询将第一个最大值的元素无限增加 1
- 对于每个后续查询，有两种情况
- 如果对位置的要求是针对同一元素，则其值保持不变

- 第一个查询将第一个最大值的元素无限增加 1
- 对于每个后续查询，有两种情况
- 如果对位置的要求是针对同一元素，则其值保持不变
- 否则，成为最大值的元素将在此操作过程中成为最大值

- 第一个查询将第一个最大值的元素无限增加 1
- 对于每个后续查询，有两种情况
- 如果对位置的要求是针对同一元素，则其值保持不变
- 否则，成为最大值的元素将在此操作过程中成为最大值
- 无论如何，我们知道最大值

H.银河理事会

- 如果我们知道最大值的位置及其值，那么所有的

  左边的元素必须小于最大值，而右边的所有元素都小于或等于最大值。

H.银河理事会

- 如果我们知道最大值的位置及其值，那么所有的

  左边的元素必须小于最大值，而右边的所有元素都小于或等于最大值。
- 对于每个元素和每个运算，运算后都有一个下 限 和上限

- 如果我们知道最大值的位置及其值，那么所有的
  左边的元素必须小于最大值，而右边的所有元素都小于或等于最大值。
- 对于每个元素和每个运算，运算后都有一个下 限 和上限
- 执行操作是有成本的，约束条件很小，让我们试着建立一个最低成本的 Ow

- 在网络中，每一对元素-天都会有一个顶点。从每个顶点都有一条边通向第二天的相应顶点（如果是最后一天，则通向汇）。

H.银河理事会

- 在网络中，每一对元素-天都会有一个顶点。从每个顶点都有一条边通向第二天的相应顶点（如果是最后一天，则通向汇）。
- 该边的 下限和上限与元素的下限和上限相对应

H.银河理事会

- 在网络中，每一对元素-天都会有一个顶点。从每个顶点都有一条边通向第二天的相应顶点（如果是最后一天，则通向汇）。
- 该边的 下限和上限与元素的下限和上限相对应
- 我们还将为每一天创建一个顶点，从这个顶点开始，将有一条容量为 1 的边从源点出发。

  元素边缘，容量为 1，成本等于这一天申请该元素的负成本

H.银河理事会

- 在网络中，每一对元素-天都会有一个顶点。从每个顶点都有一条边通向第二天的相应顶点（如果是最后一天，则通向汇）。
- 该边的 下限和上限与元素的下限和上限相对应
- 我们还将为每一天创建一个顶点，从这个顶点开始，将有一条容量为 1 的边从源点出发。

  元素边缘，容量为 1，成本等于这一天申请该元素的负成本
- 要对某些边上的欠进行下限建模，可以设置分割

  将它们一分为二，其中一个的容量等于下限，成本等于减数

H.银河理事会

- 在网络中，每一对元素-天都会有一个顶点。从每个顶点都有一条边通向第二天的相应顶点（如果是最后一天，则通向汇）。
- 该边的 下限和上限与元素的下限和上限相对应
- 我们还将为每一天创建一个顶点，从这个顶点开始，将有一条容量为 1 的边从源点出发。

  元素边缘，容量为 1，成本等于这一天申请该元素的负成本
- 要对某些边上的欠进行下限建模，可以设置分割

  将它们一分为二，其中一个的容量等于下限，成本等于减数
- 这样一个网络的欠大小为 m，顶点数为 $O(nm)$，边数为 $O(nm)$，因此即使没有电位，它的运行时间也为 $O(n\,m\,)^{23}$