贪心与搜索

pwq

Cap1taL

2026 年 7 月

CodeForces 496E Distributing Parts

热身题

n 首歌曲，每首的音域为 $[a_i, b_i]$

m 个演员，每个人的音域范围为 $[c_i, d_i]$，他最多可以演唱 k_i 首歌曲

一首歌曲能被一个演员演唱，当且仅当演员 $[c, d]$ 完全包含歌曲 $[a, b]$

判断是否存在一个分配歌曲演唱的方案，并给出构造

$n, m \leq 10^5$

其余不重要

Solution

先忽略 k 的影响，不难转化为经典模型

Solution

先忽略 k 的影响，不难转化为经典模型

将演员与歌曲，一同按区间左端点升序排列，从左向右依次考虑它们

Solution

先忽略 k 的影响，不难转化为经典模型

将演员与歌曲，一同按区间左端点升序排列，从左向右依次考虑它们

1. 若考虑到演员-左端，将其纳入维护
2. 若考虑到演员-右端，将其移出维护
3. 若考虑到歌曲-左端，我们从维护的演员里，找到能演唱当前歌曲，且右端最小的的演员，将其与当前乐曲匹配

Solution

现在考虑 k 的影响

Solution

现在考虑 k 的影响

我们将一个人可唱 k 次，等效为 k 个人，则可直接套用刚才的做法

Solution

现在考虑 k 的影响

我们将一个人可唱 k 次，等效为 k 个人，则可直接套用刚才的做法

k 可以到 10^9 量级，因此我们不能真的拆人（?!）

Solution

现在考虑 k 的影响

我们将一个人可唱 k 次，等效为 k 个人，则可直接套用刚才的做法

k 可以到 10^9 量级，因此我们不能真的拆人（?!）

我们在纳入维护演员时，为每个人额外维护一个“剩余次数”即可

Solution

现在考虑 k 的影响
我们将一个人可唱 k 次，等效为 k 个人，则可直接套用刚才的做法
k 可以到 10^9 量级，因此我们不能真的拆人（?!）
我们在纳入维护演员时，为每个人额外维护一个“剩余次数”即可
具体维护可以使用 set 或者 multiset

CodeForces 525D Arthur and Walls

给定一个 $n \times m$ 的网格，每个格子是空地. 或墙壁 *

房间定义为极大四连通空地连通块

可以拆除一些墙壁（将 * 变成.），目标是让每个房间都是矩形

求最少拆除数量，并输出任意一种方案

$n, m \leq 2000$

Solution

我们考虑：什么样的“房间”不是一个“矩形”?

Solution

我们考虑：什么样的“房间”不是一个“矩形”?

```
..
.#
```

容易发现，既然房间不是矩形，必然存在上图所示的“拐角”（可以旋转）。且这样的拐角必须被修复

Solution

我们考虑：什么样的“房间”不是一个“矩形”?

```
..
.#
```

容易发现，既然房间不是矩形，必然存在上图所示的“拐角”（可以旋转）。且这样的拐角必须被修复
又发现，若我们修复了一个拐角，可能会引入新的拐角
搜索！我们抽象每个 $2x2$ 的单元为一个新元素即可

CodeForces 442C Artem and Array

给定长度为 n 的数组 a，你需要进行 n 次操作：删去某一元素 a_i，并获得 $\min\{a_{i-1}, a_{i+1}\}$ 的分数。若不存在 a_{i-1} 或 a_{i+1}，则此次操作不得分。
请你计算至多能得到多少分。
$n \leq 5 \times 10^5$，$a_i \leq 10^6$

Solution

给出方案：

Solution

给出方案：

- 删除所有极小值点，序列变为单峰
- 答案为除了最大值和次大值的元素之和

我们来证明这个思路

Solution

“删除所有极小值点”

Solution

“删除所有极小值点”

- 感性上容易理解，删除凹下去的部分是“不亏”的

Solution

“删除所有极小值点”

- 感性上容易理解，删除凹下去的部分是“不亏”的
- 设极小值 y 附近是 $[p, x, y, z, q]$，我们尝试证明：删除 x 或 z 之前先删除 y 一定不劣

Solution

“删除所有极小值点”

- 感性上容易理解，删除凹下去的部分是“不亏”的
- 设极小值 y 附近是 $[p, x, y, z, q]$，我们尝试证明：删除 x 或 z 之前先删除 y 一定不劣
- A. 若先删除 x，则收益为 $\min(p, y)$，剩余序列为 $[p, y, z, q]$
- B. 若先删除 y，则收益为 $\min(x, z)$，剩余序列为 $[p, x, z, q]$

Solution

“删除所有极小值点”

- 感性上容易理解，删除凹下去的部分是“不亏”的
- 设极小值 y 附近是 $[p, x, y, z, q]$，我们尝试证明：删除 x 或 z 之前先删除 y 一定不劣
- A. 若先删除 x，则收益为 $\min(p, y)$，剩余序列为 $[p, y, z, q]$
- B. 若先删除 y，则收益为 $\min(x, z)$，剩余序列为 $[p, x, z, q]$
- 注意到 B 方案的收益 $\min(x, z) \geq y \geq \min(p, y)$，且 B 的剩余序列更大，显然其剩余答案也大于 A
- 因此 B 方案更优，证明成立

Solution

“答案为除了最大值和次大值的元素之和”

Solution

“答案为除了最大值和次大值的元素之和”

- 显然最大值不可能进入答案

Solution

“答案为除了最大值和次大值的元素之和”

- 显然最大值不可能进入答案
- 次大值成为答案，当且仅当“存在一次 min(最大 , 次大)”，而这在单峰序列中是不可能的

Solution

“答案为除了最大值和次大值的元素之和”

- 显然最大值不可能进入答案
- 次大值成为答案，当且仅当“存在一次 min(最大 , 次大)”，而这在单峰序列中是不可能的
- 因此答案必定不大于“除了最大值和次大值的元素之和”
- 构造方案：从“峰尖”开始取即可

Solution

- 第一步在理解题目大意后可以感受到
- 第二步可以手玩出来
- 第一步可以带 log，实现方案很多，链表或者一点数据结构都可以。或者采用类似搜索的手法。

洛谷 P6243 The Milk Queue G

经典?!

N 头奶牛排队挤奶，每头牛需先后经过两道工序

第 i 头牛的第一道工序耗时 A_i，第二道工序耗时 B_i

若牛 i 先于牛 j 开始第一道工序，则她也先开始第二道工序

求最优排队顺序下，完成所有挤奶的最短总时间

$N \leq 25000,\ A_i, B_i \leq 2 \times 10^4$

Solution

结论是众所周知的：

- A. 当 $a_i \le b_i$ 时，按照 a_i 升序
- B. 当 $a_i > b_i$ 时，按照 b_i 降序
- 将 A 组放到 B 组的左侧

不乏感性理解方法。如何严格证明？

Solution

1. 用表达式刻画答案

$$\max_{k=1}^{n}\left(\sum_{i=1}^{k} a_i + \sum_{i=k}^{n} b_i\right)$$

证明？数学归纳法

Solution

2. 临项交换
   - 考虑何时交换 (a_i, b_i) 与 (a_j, b_j) 会变优
   - 可得到 $\min(b_i, a_j) > \min(a_i, b_j)$

Solution

2. 临项交换
   - 考虑何时交换 (a_i, b_i) 与 (a_j, b_j) 会变优
   - 可得到 $\min(b_i, a_j) > \min(a_i, b_j)$
   - 但它没有传递性。我们无法直接使用它排序。我们使用刚才构造的全序

Solution

2. 临项交换
   - 考虑何时交换 (a_i, b_i) 与 (a_j, b_j) 会变优
   - 可得到 $\min(b_i, a_j) > \min(a_i, b_j)$
   - 但它没有传递性。我们无法直接使用它排序。我们使用刚才构造的全序
   - 一方面，其满足我们推得的不等式；另一方面，任意满足我们推得不等式的排列，都可以通过交换变为全序排序后唯一的形式
   - 证明？考虑交换一对元素对实际答案的影响

洛谷 P6631 序列

给定长度为 n 的非负整数序列 a
每一步可选择以下三种操作之一：

- 选择区间 $[l, r]$，所有数减 1
- 选择区间 $[l, r]$，其中下标为奇数的数减 1
- 选择区间 $[l, r]$，其中下标为偶数的数减 1

求将全部数归零的最少步数
$n \leq 10^5$，$a_i \leq 10^9$，$T \leq 10$

Solution

存在线性规划的套路做法，不过这不是我们今天的主题

Solution

存在线性规划的套路做法，不过这不是我们今天的主题

- 观察范围和时空限制，结合题面操作的高自由度，我们猜测是某种神秘贪心

Solution

存在线性规划的套路做法，不过这不是我们今天的主题

- 观察范围和时空限制，结合题面操作的高自由度，我们猜测是某种神秘贪心
- 注意到后两种操作其实是同一种，即隔一个操作一个

Solution

存在线性规划的套路做法，不过这不是我们今天的主题

- 观察范围和时空限制，结合题面操作的高自由度，我们猜测是某种神秘贪心
- 注意到后两种操作其实是同一种，即隔一个操作一个
- 一个常见的思考角度是从端点情况入手。我们考虑 a_n 和 a_{n-1}

Solution

存在线性规划的套路做法，不过这不是我们今天的主题

- 观察范围和时空限制，结合题面操作的高自由度，我们猜测是某种神秘贪心
- 注意到后两种操作其实是同一种，即隔一个操作一个
- 一个常见的思考角度是从端点情况入手。我们考虑 a_n 和 a_{n-1}
  - 若 $a_{n-1} = 0$，显然操作二不劣
  - 若 $a_n \neq 0$，不做任何事
  - 若 $a_{n-1}, a_n \neq 0$，虽然操作二可能延伸更远，但直接用操作一覆盖不劣。可以转化为上述情形
- 具体证明？

Solution

从右向左扫描，假设我们已经用刚才的方法把 $[L+1,n]$ 都变成 0，考虑 a_L

- 基本与刚才的情形类似，不同的是右侧“传来”了几个尚未结束的操作一/二或三。不妨假设其数量为 x 和 y。

Solution

从右向左扫描，假设我们已经用刚才的方法把 $[L+1,n]$ 都变成 0，考虑 a_L

- 基本与刚才的情形类似，不同的是右侧“传来”了几个尚未结束的操作一/二或三。不妨假设其数量为 x 和 y。
- 若 $a_L < x+y$，贪心选取，则必定有 $x+y-a_L$ 操作需要在 L 右侧结束
  - 我们暂时无法决策具体保留哪些操作，于是一个技巧是，将 x 与 y 均扣除 $x+y-a_L$，再“保留 $x+y-a_L$ 的自由操作额度”
  - x 和 y 可能会小于 $x+y-a_L$，需要先用鸽巢原理处理。

Solution

从右向左扫描，假设我们已经用刚才的方法把 $[L+1,n]$ 都变成 0，考虑 a_L

- 基本与刚才的情形类似，不同的是右侧“传来”了几个尚未结束的操作一/二或三。不妨假设其数量为 x 和 y。
- 若 $a_L < x+y$，贪心选取，则必定有 $x+y-a_L$ 操作需要在 L 右侧结束
  - 我们暂时无法决策具体保留哪些操作，于是一个技巧是，将 x 与 y 均扣除 $x+y-a_L$，再“保留 $x+y-a_L$ 的自由操作额度”
  - x 和 y 可能会小于 $x+y-a_L$，需要先用鸽巢原理处理。
- 若 $a_L \geq x+y$，继续延伸右侧的操作——将 a_L 减去 $x+y$。此后若 a_L 仍不为 0，我们先尝试使用之前积累的" 自由操作额度"，若不足再使用新操作
- 如何想到？

洛谷 P8314 Parkovi

给定一棵 n 个节点的树，边有长度
选择 k 个位置修建公园（同一节点最多一个）
最小化每个节点到最近公园距离的最大值
$n \leq 2 \times 10^5$

Solution

- 一道小清新的树上贪心

Solution

- 一道小清新的树上贪心
- 最大值最小，首先套一个二分答案，问题转化为如何 check

Solution

- 一道小清新的树上贪心
- 最大值最小，首先套一个二分答案，问题转化为如何 check
- 类似上一道题！我们从端点（叶子）开始考虑

Solution

- 一道小清新的树上贪心
- 最大值最小，首先套一个二分答案，问题转化为如何 check
- 类似上一道题！我们从端点（叶子）开始考虑
- 使用贪心策略，则一个公园会被尽量放的更浅，以期望覆盖更多的节点

Solution

- 一道小清新的树上贪心
- 最大值最小，首先套一个二分答案，问题转化为如何 check
- 类似上一道题！我们从端点（叶子）开始考虑
- 使用贪心策略，则一个公园会被尽量放的更浅，以期望覆盖更多的节点
- 每个节点并不关心子树内的具体决策

Solution

- 一道小清新的树上贪心
- 最大值最小，首先套一个二分答案，问题转化为如何 check
- 类似上一道题！我们从端点（叶子）开始考虑
- 使用贪心策略，则一个公园会被尽量放的更浅，以期望覆盖更多的节点
- 每个节点并不关心子树内的具体决策
- 使用 DFS 遍历树，合并子树时先考虑能不能子树间相互覆盖，再考虑当前根节点是否必须放置公园
- 细节：根节点没有父节点

洛谷 P5665 划分

将数列 a 划分成若干连续段
要求每段之和单调不降
最小化每段之和的平方的和 (?! 绕绕?!)。即最小化：

$$\left(\sum_{i=1}^{k_1} a_i\right)^2 + \left(\sum_{i=k_1+1}^{k_2} a_i\right)^2 + \cdots + \left(\sum_{i=k_p+1}^{n} a_i\right)^2$$

$n \leq 4 \times 10^7$

Solution

- 题外话：感觉属于那种赛场上不严谨猜着做比严格证明简单的多的题

Solution

- 题外话：感觉属于那种赛场上不严谨猜着做比严格证明简单的多的题
- 首先有一个显然的 DP。设 $f_{i,j}$ 表示前 i 个数，最后一段是 $[j+1,i]$ 的最小答案。转移需要枚举倒数第二段的段首，复杂度 $O(n^3)$。

Solution

- 题外话：感觉属于那种赛场上不严谨猜着做比严格证明简单的多的题
- 首先有一个显然的 DP。设 $f_{i,j}$ 表示前 i 个数，最后一段是 $[j+1,i]$ 的最小答案。转移需要枚举倒数第二段的段首，复杂度 $O(n^3)$。
- 半感性地，我们知道 $(a+b)^2=a^2+b^2+2ab>a^2+b^2$。即在保证合法的情况下，拆开一段总是更优的，每段更短更好。

Solution

- 题外话：感觉属于那种赛场上不严谨猜着做比严格证明简单的多的题
- 首先有一个显然的 DP。设 $f_{i,j}$ 表示前 i 个数，最后一段是 $[j+1,i]$ 的最小答案。转移需要枚举倒数第二段的段首，复杂度 $O(n^3)$。
- 半感性地，我们知道 $(a+b)^2=a^2+b^2+2ab>a^2+b^2$。即在保证合法的情况下，拆开一段总是更优的，每段更短更好。
- 先给出结论：在所有合法的划分中，最优划分一定最后一段最短

Solution

“在所有合法的划分中，最优划分一定最后一段最短”

Solution

“在所有合法的划分中，最优划分一定最后一段最短”

- 数学归纳法。当 $n = 1$ 时显然成立，我们尝试证明 n 时也成立。

Solution

“在所有合法的划分中，最优划分一定最后一段最短”

- 数学归纳法。当 $n = 1$ 时显然成立，我们尝试证明 n 时也成立。
- 借用刚才 DP 转移的思想。
- 我们现在不需要再枚举 k，数学归纳法保证了 f_j 的最优解的最后一段最短，则它的和也最小，最易合法。取更小的 k 不优。

Solution

“在所有合法的划分中，最优划分一定最后一段最短”

- 数学归纳法。当 $n = 1$ 时显然成立，我们尝试证明 n 时也成立。
- 借用刚才 DP 转移的思想。
- 我们现在不需要再枚举 k，数学归纳法保证了 f_j 的最优解的最后一段最短，则它的和也最小，最易合法。取更小的 k 不优。
- 设 g_i 表示最优分拆下，最后一段的和。设 s_i 表示前缀和。转移条件可以写成 $g_j + s_j < s_i$，单调性表明可行的 j 是一个前缀。
- 问题转化为证明，取最大的 j 最优

Solution

“转移时取最大的 j 最优”

Solution

“转移时取最大的 j 最优”

- 我们证明任何一个 $k < j$ 都劣于 j

Solution

“转移时取最大的 j 最优”

- 我们证明任何一个 $k < j$ 都劣于 j
- 为了简化问题，单独提出所有分拆点，可以证明 k 分拆的一段不可能被 j 分拆的一段完全包含。这表明排布一定形如先重合，后交替，最后一侧有一侧无的形式
- “最后一侧有一侧无”可以忽略

Solution

“转移时取最大的 j 最优”

- 我们证明任何一个 $k < j$ 都劣于 j
- 为了简化问题，单独提出所有分拆点，可以证明 k 分拆的一段不可能被 j 分拆的一段完全包含。这表明排布一定形如先重合，后交替，最后一侧有一侧无的形式
- “最后一侧有一侧无”可以忽略
- 直接考虑二者各段的和序列为 $[X_1, X_n]$ 和 $[Y_1, Y_m]$。我们下面通过一系列操作，使 X_i 与 Y_i 相同，同时保证 X_i 不更优 Y_i 不更劣，从而说明原本的 X_i 优于 Y_i

Solution

“转移时取最大的 j 最优”

- 我们证明任何一个 $k < j$ 都劣于 j
- 为了简化问题，单独提出所有分拆点，可以证明 k 分拆的一段不可能被 j 分拆的一段完全包含。这表明排布一定形如先重合，后交替，最后一侧有一侧无的形式
- “最后一侧有一侧无”可以忽略
- 直接考虑二者各段的和序列为 $[X_1, X_n]$ 和 $[Y_1, Y_m]$。我们下面通过一系列操作，使 X_i 与 Y_i 相同，同时保证 X_i 不更优 Y_i 不更劣，从而说明原本的 X_i 优于 Y_i
- 找到最大的 p 最小的 q，满足 $Y_1 = Y_p, Y_q = Y_m$，将 $Y_p + 1$ 同时 $Y_q - 1$，这使得 Y_i 变得更优
- 一旦 X_i 和 Y_i 存在了一段前缀相同，我们将前缀砍掉继续操作

Solution

“转移时取最大的 j 最优”

- 最终 $Y_1 = Y_m$ 或 $Y_1 + 1 = Y_m$

Solution

“转移时取最大的 j 最优”

- 最终 $Y_1 = Y_m$ 或 $Y_1 + 1 = Y_m$
- 注意 $k < j$ 使得 $X_n \leq Y_m - 1 = Y_1$，由此得到 X_i 每个元素都小于 Y_1

Solution

“转移时取最大的 j 最优”

- 最终 $Y_1 = Y_m$ 或 $Y_1 + 1 = Y_m$
- 注意 $k < j$ 使得 $X_n \leq Y_m - 1 = Y_1$，由此得到 X_i 每个元素都小于 Y_1
  - 若两序列等长，则二者必定完全恒等，以此保证总和一致

Solution

“转移时取最大的 j 最优”

- 最终 $Y_1 = Y_m$ 或 $Y_1 + 1 = Y_m$
- 注意 $k < j$ 使得 $X_n \le Y_m - 1 = Y_1$，由此得到 X_i 每个元素都小于 Y_1
  - 若两序列等长，则二者必定完全恒等，以此保证总和一致
  - 若 $n = m + 1$，我们把 X_1 分配给 $[X_2, X_n]$，从而让二者相等，同时 X_i 变的更劣

Solution

“转移时取最大的 j 最优”

- 最终 $Y_1 = Y_m$ 或 $Y_1 + 1 = Y_m$
- 注意 $k < j$ 使得 $X_n \le Y_m - 1 = Y_1$，由此得到 X_i 每个元素都小于 Y_1
  - 若两序列等长，则二者必定完全恒等，以此保证总和一致
  - 若 $n = m + 1$，我们把 X_1 分配给 $[X_2, X_n]$，从而让二者相等，同时 X_i 变的更劣
- 综上，我们总能通过一系列操作，使 X_i 与 Y_i 相同，同时保证 X_i 不更优，Y_i 不更劣

Solution

“转移时取最大的 j 最优”

- 最终 $Y_1 = Y_m$ 或 $Y_1 + 1 = Y_m$
- 注意 $k < j$ 使得 $X_n \le Y_m - 1 = Y_1$，由此得到 X_i 每个元素都小于 Y_1
  - 若两序列等长，则二者必定完全恒等，以此保证总和一致
  - 若 $n = m + 1$，我们把 X_1 分配给 $[X_2, X_n]$，从而让二者相等，同时 X_i 变的更劣
- 综上，我们总能通过一系列操作，使 X_i 与 Y_i 相同，同时保证 X_i 不更优，Y_i 不更劣
- 因此原本的 X_i 优于 Y_i，数学归纳法成立，转移时只需要找到最大的合法 j 即可。

Solution

“转移时取最大的 j 最优”

- 最终 $Y_1 = Y_m$ 或 $Y_1 + 1 = Y_m$
- 注意 $k < j$ 使得 $X_n \le Y_m - 1 = Y_1$，由此得到 X_i 每个元素都小于 Y_1
  - 若两序列等长，则二者必定完全恒等，以此保证总和一致
  - 若 $n = m + 1$，我们把 X_1 分配给 $[X_2, X_n]$，从而让二者相等，同时 X_i 变的更劣
- 综上，我们总能通过一系列操作，使 X_i 与 Y_i 相同，同时保证 X_i 不更优，Y_i 不更劣
- 因此原本的 X_i 优于 Y_i，数学归纳法成立，转移时只需要找到最大的合法 j 即可。
- 使用数据结构可能会带 log，可以使用单调队列做到线性。卡常。

QOJ 5173 染色

一条纸条上有 n 个格子，第 i 个格子的颜色为 a_i
每秒钟可以执行以下两种操作之一：

- 将某个格子染成任意颜色
- 移动到相邻格子（要求移动前后颜色相同）

Q 次询问，每次求从格子 u 到 v 的最短时间
询问之间独立（不实际修改纸条）
$n, Q \leq 10^6$

Solution

- 先考虑单次询问怎么做，不妨设 $u < v$

Solution

- 先考虑单次询问怎么做，不妨设 $u < v$
- 最暴力的做法是每格都染，这给出了答案的上界

Solution

- 先考虑单次询问怎么做，不妨设 $u < v$
- 最暴力的做法是每格都染，这给出了答案的上界
- 发现若 u, v 之间存在一对 $i < j$ 满足 $a_i = a_j$，则我们可以少染一次
- 问题转化为怎么在 u, v 之间选出最多这样的区间

Solution

- 先考虑单次询问怎么做，不妨设 $u < v$
- 最暴力的做法是每格都染，这给出了答案的上界
- 发现若 u, v 之间存在一对 $i < j$ 满足 $a_i = a_j$，则我们可以少染一次
- 问题转化为怎么在 u, v 之间选出最多这样的区间
- 立马发现可以用简单的 DP 解决，并优化到线性
- 多次询问怎么办？

Solution

- 多次询问下发现 DP 没什么前途。这个问题这么简单，能不能不用 DP 呢？

Solution

- 多次询问下发现 DP 没什么前途。这个问题这么简单，能不能不用 DP 呢？
- 贪！心！预处理 R_i 表示 i 右侧最靠左的完整区间的右端点下标

Solution

- 多次询问下发现 DP 没什么前途。这个问题这么简单，能不能不用 DP 呢？
- 贪！心！预处理 R_i 表示 i 右侧最靠左的完整区间的右端点下标
- 我们断言：从 u 开始不断跳 R_i，超过 v 之前跳的次数就是最大区间数
- 可以用反证法证明

Solution

- 多次询问下发现 DP 没什么前途。这个问题这么简单，能不能不用 DP 呢？
- 贪！心！预处理 R_i 表示 i 右侧最靠左的完整区间的右端点下标
- 我们断言：从 u 开始不断跳 R_i，超过 v 之前跳的次数就是最大区间数
- 可以用反证法证明
- 具体实现方面，把询问挂到序列上，使用扫描线，同时使用并查集加树上差分，可以做到线性

QOJ 1173 Knowledge Is...

N 个项目，M 名学生
项目 i 在时间 L_i 可被领取，必须在 R_i 或之前完成
一名学生不能同时做两个项目（即若选 i, j，须满足 $R_i < L_j$ 或 $R_j < L_i$）
每名学生最多做两个项目
求最多能完成的项目数，并输出每个项目由哪名学生（1~M）完成，未完成输出 0
$N \leq 3 \times 10^5$

Solution

- 说人话：给一堆区间，选最多的匹配，满足匹配内的区间不交

Solution

- 说人话：给一堆区间，选最多的匹配，满足匹配内的区间不交
- 不太能 DP，尝试反悔贪心
- 套路地，我们将区间按右端点降序排序进行扫描

Solution

- 说人话：给一堆区间，选最多的匹配，满足匹配内的区间不交
- 不太能 DP，尝试反悔贪心
- 套路地，我们将区间按右端点降序排序进行扫描
  - 如果右侧存在区间能与当前区间匹配，我们直接进行匹配。我们尽量使用左端点更大的区间

Solution

- 说人话：给一堆区间，选最多的匹配，满足匹配内的区间不交
- 不太能 DP，尝试反悔贪心
- 套路地，我们将区间按右端点降序排序进行扫描
  - 如果右侧存在区间能与当前区间匹配，我们直接进行匹配。我们尽量使用左端点更大的区间
  - 如果不存在可以直接匹配的区间，我们尝试拆散已有匹配。设当前区间是 $[L, R]$，若存在匹配的最左端点大于 L，我们将当前区间与匹配中的区间交换。我们尽量交换左端点更大的区间

Solution

- 说人话：给一堆区间，选最多的匹配，满足匹配内的区间不交
- 不太能 DP，尝试反悔贪心
- 套路地，我们将区间按右端点降序排序进行扫描
  - 如果右侧存在区间能与当前区间匹配，我们直接进行匹配。我们尽量使用左端点更大的区间
  - 如果不存在可以直接匹配的区间，我们尝试拆散已有匹配。设当前区间是 $[L, R]$，若存在匹配的最左端点大于 L，我们将当前区间与匹配中的区间交换。我们尽量交换左端点更大的区间
  - 这样的交换总能扩大“未匹配区间”的待匹配范围，因此是更优的
- 使用优先队列维护即可

CodeForces 802M3 April Fools' Problem (hard)

有 n 天，需要准备 k 个问题
第 i 天最多准备一个问题（花费 a_i），最多打印一个问题（花费 b_i）
一个问题必须先准备再打印（可在同一天完成）
求准备并打印 k 个问题的最小总花费
$n \leq 5 \times 10^5$

Solution

- 原题要求必须打印 k 个问题

Solution

- 原题要求必须打印 k 个问题
- 容易发现点集 $\{(k, mincost_k)\}$ 是下凸的，使用 wqs 二分拿掉 k 的限制
  - 不会 wqs 二分的可以参考 P2619 [国家集训队] Tree I
  - wqs 二分并非今天的重点，现在可以先 skip 掉

Solution

- 原题要求必须打印 k 个问题
- 容易发现点集 $\{(k, mincost_k)\}$ 是下凸的，使用 wqs 二分拿掉 k 的限制
  - 不会 wqs 二分的可以参考 P2619 [国家集训队] Tree I
  - wqs 二分并非今天的重点，现在可以先 skip 掉
- 将 b_i 减去 wqs 二分的斜率后，变为纯粹的最优化问题

Solution

- 考虑一个简单的贪心：每天都尝试“打印”，如果前缀存在某天的“准备”开销和今天的“打印”开销加起来为负，就直接美美打印得吃

Solution

- 考虑一个简单的贪心：每天都尝试“打印”，如果前缀存在某天的“准备”开销和今天的“打印”开销加起来为负，就直接美美打印得吃
- 这显然是错误的，因为后面可能某天打印更划算

Solution

- 考虑一个简单的贪心：每天都尝试“打印”，如果前缀存在某天的“准备”开销和今天的“打印”开销加起来为负，就直接美美打印得吃
- 这显然是错误的，因为后面可能某天打印更划算
- 因此使用反悔贪心。打印的时候尝试能不能把之前的打印拆掉，比较一下与直接打印谁更优即可

Solution

- 考虑一个简单的贪心：每天都尝试“打印”，如果前缀存在某天的“准备”开销和今天的“打印”开销加起来为负，就直接美美打印得吃
- 这显然是错误的，因为后面可能某天打印更划算
- 因此使用反悔贪心。打印的时候尝试能不能把之前的打印拆掉，比较一下与直接打印谁更优即可
- 具体实现可以用优先队列。方便起见可以直接把当前打印的 $-b_i$ 和正常的 a_i 混起来，之后如果取出 $-b_i$ 就相当于进行反悔操作。

CodeForces 436E Cardboard Box

n 个关卡，每个关卡可以：

- 不玩（得 0 星）
- 花 a_i 时间通关得 1 星
- 花 b_i 时间通关得 2 星（$a_i < b_i$）

每个关卡最多通关一次
至少要获得 w 颗星，求最短总时间
$n \leq 3 \times 10^5$

Solution

- 一个简单但错误的贪心思路：每次选最便宜的星进行选择，同时“解锁”“补差价”到第二颗星的选项继续贪心

Solution

- 一个简单但错误的贪心思路：每次选最便宜的星进行选择，同时“解锁”“补差价”到第二颗星的选项继续贪心
- 它可以被一个简单的 Hack 卡掉，例如第一关为 (100,105)，而第二关为 (101 102)，只需要两颗星。上面的贪心会一头扎进第一关里，这是错误的
- 发现问题在于，我们的贪心无法“看到”相邻第二层的选项

Solution

- 一个简单但错误的贪心思路：每次选最便宜的星进行选择，同时“解锁”“补差价”到第二颗星的选项继续贪心
- 它可以被一个简单的 Hack 卡掉，例如第一关为 (100,105)，而第二关为 (101 102)，只需要两颗星。上面的贪心会一头扎进第一关里，这是错误的
- 发现问题在于，我们的贪心无法“看到”相邻第二层的选项
- 于是使用反悔贪心。我们有四个选项：
  1. 开一个新的 a_i
  2. 补一个 $b_i - a_i$ 的差价
  3. 反悔一个 a_i 的选择，改选另一个补差价 $b_j - a_j$
  4. 反悔一个 $b_i - a_i$ 的补差价，改选另一个 a_j

Solution

- 一个简单但错误的贪心思路：每次选最便宜的星进行选择，同时“解锁”“补差价”到第二颗星的选项继续贪心
- 它可以被一个简单的 Hack 卡掉，例如第一关为 (100,105)，而第二关为 (101 102)，只需要两颗星。上面的贪心会一头扎进第一关里，这是错误的
- 发现问题在于，我们的贪心无法“看到”相邻第二层的选项
- 于是使用反悔贪心。我们有四个选项：
  1. 开一个新的 a_i
  2. 补一个 $b_i - a_i$ 的差价
  3. 反悔一个 a_i 的选择，改选另一个补差价 $b_j - a_j$
  4. 反悔一个 $b_i - a_i$ 的补差价，改选另一个 a_j
- 开五个优先队列进行维护即可。与前几道题的思路基本一致，只是选择更多
- 为什么正确？

Solution

一个较为简单的正确性理解

Solution

一个较为简单的正确性理解

- 我们对每道关卡的选择只有“2 层”

Solution

一个较为简单的正确性理解

- 我们对每道关卡的选择只有“2 层”
- 为了保持正确性，我们需要在每个节点选择时，具有跳出到任意临近关卡的任意层数的能力。

Solution

一个较为简单的正确性理解

- 我们对每道关卡的选择只有“2 层”
- 为了保持正确性，我们需要在每个节点选择时，具有跳出到任意临近关卡的任意层数的能力。
- 由于贡献可拆分，这需要 2^K 个“出路”。因此我们开了 2^2 个额外的优先队列用于维护各个选择。

Solution

一个较为简单的正确性理解

- 我们对每道关卡的选择只有“2 层”
- 为了保持正确性，我们需要在每个节点选择时，具有跳出到任意临近关卡的任意层数的能力。
- 由于贡献可拆分，这需要 2^K 个“出路”。因此我们开了 2^2 个额外的优先队列用于维护各个选择。
- 某种意义上，也类别为执行一个隐性的优化建图 + Prim 的过程，边权是我们的赚星的开销。

洛谷 P4511 日程管理

有 T 天，每天恰好可以完成一个任务
每个任务有截止日期 t_i 和收益 p_i，必须在截止日前完成才获得收益
支持 Q 次操作：

- ADD t p：添加一个任务
- DEL t p：删除一个任务

每次操作后输出当前最大可获收益
$T, Q \leq 3 \times 10^5$

Solution

- 先考虑静态版本怎么处理，即问题集固定
- 什么情况下才会完不成？

Solution

- 先考虑静态版本怎么处理，即问题集固定
- 什么情况下才会完不成？
  - 若存在 k，使得前第 k 天必须做完 $> k$ 个任务，我们就会爆炸
  - 否则我们一定可以构造一个完成任务的方案，每天贪心做最早的题即可

Solution

- 先考虑静态版本怎么处理，即问题集固定
- 什么情况下才会完不成？
  - 若存在 k，使得前第 k 天必须做完 $> k$ 个任务，我们就会爆炸
  - 否则我们一定可以构造一个完成任务的方案，每天贪心做最早的题即可
- 如何确定选哪些任务？

Solution

- 先考虑静态版本怎么处理，即问题集固定
- 什么情况下才会完不成？
  - 若存在 k，使得前第 k 天必须做完 $> k$ 个任务，我们就会爆炸
  - 否则我们一定可以构造一个完成任务的方案，每天贪心做最早的题即可
- 如何确定选哪些任务？
  - 反悔贪心。一个个考虑任务
  - 若还塞得下就塞进任务序列
  - 若塞不下就找替换一个收益不如它的

Solution

- 先考虑静态版本怎么处理，即问题集固定
- 什么情况下才会完不成？
  - 若存在 k，使得前第 k 天必须做完 $> k$ 个任务，我们就会爆炸
  - 否则我们一定可以构造一个完成任务的方案，每天贪心做最早的题即可
- 如何确定选哪些任务？
  - 反悔贪心。一个个考虑任务
  - 若还塞得下就塞进任务序列
  - 若塞不下就找替换一个收益不如它的
- 发现我们实际上已经可以做只有 ADD 的动态版本。删除怎么办？

Solution

方案 #1 修补贪心方案

Solution

方案 #1 修补贪心方案

- 若删除的任务根本没使用，直接删除
- 若删除的任务已经被使用，我们将其删除，并尝试放入一个尚未被使用的任务

Solution

方案 #1 修补贪心方案

- 若删除的任务根本没使用，直接删除
- 若删除的任务已经被使用，我们将其删除，并尝试放入一个尚未被使用的任务
- 如何维护合法性？
  - “若存在 k，使得前第 k 天必须做完 $> k$ 个任务，我们就会爆炸”

Solution

方案 #1 修补贪心方案

- 若删除的任务根本没使用，直接删除
- 若删除的任务已经被使用，我们将其删除，并尝试放入一个尚未被使用的任务
- 如何维护合法性？
  - “若存在 k，使得前第 k 天必须做完 $> k$ 个任务，我们就会爆炸”
  - 定义 $c_i = k - [$ 必须在前 k 天完成的任务数 $]$。这个值必须非负
  - 同时我们可以发现，插入一个任务 (t, p) 只要求 $[c_t, c_T]$ 必须为正，插入后会对这个后缀减一。
  - 使用线段树维护即可

Solution

方案 #2 线段树分治

- 用线段树分治的板子把 $O(q)$ 次增加和删除变成 $O(q \log q)$ 次增加和撤销
- 撤销可以硬维护，用栈记录操作序列
- 比较套路的做法，不是我们今天的重点

SPOJ MTREECOL Color a tree

给定一棵 N 个节点的树，根为 R
每个节点有染色代价因子 C_i
必须先染父节点才能染子节点
每染一个节点耗时 1，若节点 i 在时刻 F_i 完成染色，代价为 $C_i \times F_i$
求最小总染色代价
$N \leq 1000$，$C_i \leq 500$

洛谷 P3620 数据备份

N 栋办公楼在一条直线上，坐标 s_i 已排序
需要铺设 K 条电缆，每条连接两栋楼
每栋楼至多属于一对
最小化电缆总长度
$N \leq 10^5$

CodeForces 2128F Strict Triangle

给定连通无向图 G 和节点 k

每条边 i 的权重 w_i 可在 $[l_i, r_i]$ 中任取实数

判断是否存在赋值方案，使得

$\mathrm{dist}_w(1, n) \neq \mathrm{dist}_w(1, k) + \mathrm{dist}_w(k, n)$

多组数据，$\sum n, \sum m \leq 2 \times 10^5$

CodeForces 2034G2 Simurgh's Watch (Hard Version)

给定 n 个区间 $[l_i, r_i]$，每个区间需分配一种颜色
对任意整数时刻 t，若至少有一个区间覆盖 t，则存在一种颜色在 t 恰好只出现一次
求最小所需颜色数，并给出构造
$\sum n \leq 2 \times 10^5$

Thanks!