Solution
简单 NOIP 模拟赛 Day1

JKLover

October 3, 2020

算法一

dfs 枚举所有分组情况并计算答案，期望得分 20 分。

算法二

若 u, v 被分在同一组中，则对答案有 $w_u + w_v$ 的贡献（$u = v$ 也算）。

那么就有

$ans = (\sum w_i) \cdot \left\{ {n \atop k} \right\} + \sum_{u \neq v} (w_u + w_v) \cdot \left\{ {n-1 \atop k} \right\}$

$ans = (\sum w_i) \cdot (\left\{ {n \atop k} \right\} + (n-1) \left\{ {n-1 \atop k} \right\})$

其中 $\left\{ {n \atop k} \right\}$ 是第二类斯特林数，即将 n 个数分为 k 个非空集合的方案数。

若按照递推式计算，时间复杂度 $O(n^2)$ ，期望得分 40 分。

考虑经典的做法，利用容斥原理计算第二类斯特林数，

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n$$

直接用快速幂计算每个 $(k-i)^n$，时间复杂度 $O(n \log n)$ ，期望得分 70 到 100 分。

若利用线性筛预处理所有 $(k-i)^n$ ，时间复杂度 $O(n)$ ，期望得分 100 分。

算法一

dfs 去哪些商店以及去的顺序，期望得分 20 分。

算法二

考虑如果在时刻 t 在商店 i 购买物品，结束后立即去商店 j 购买物品。
那么 j 会因为在 i 处等候而额外花费 $(a_i \cdot t + b_i + 1) \cdot a_j$ 的时间。
如果我们将二者交换顺序，在时刻 t 在 j 购买，结束后立即去 i 购买，i 会额外花费 $(a_j \cdot t + b_j + 1) \cdot a_i$ 的时间。
若先去 i 比先去 j 更优，就需要满足

$$(a_i \cdot t + b_i + 1) \cdot a_j \leq (a_j \cdot t + b_j + 1) \cdot a_i$$

即

$$(b_i + 1) \cdot a_j \leq (b_j + 1) \cdot a_i$$

可以发现 i 是否比 j 优与当前时刻 t 无关。
于是可以先对所有商店排序，得到序列 p，那么我们实际去的商店按照时间先后形成的序列一定是 p 的一个子序列。

算法二

那么我们可以进行 dp ，设 $dp(i,j)$ 表示考虑了序列 p 中前 i 家商店，已经买了 j 个物品花费的最少时间。
转移时考虑是否去第 i 家商店即可，时间复杂度 $O(n^2)$ ，期望得分 50 分。

算法二

那么我们可以进行 dp，设 $dp(i,j)$ 表示考虑了序列 p 中前 i 家商店，已经买了 j 个物品花费的最少时间。
转移时考虑是否去第 i 家商店即可，时间复杂度 $O(n^2)$，期望得分 50 分。

算法三

当没有 $a_i = 0$ 的商店时，可以发现花费的时间是随着去过的商店数目指数级增长的，即最多去 $O(\log T)$ 个商店。
将第二维只开到 $O(\log T)$，这部分的时间复杂度就优化到了 $O(n\log T)$，结合算法二，期望得分 70 分。

算法三

当没有 $a_i = 0$ 的商店时，可以发现花费的时间是随着去过的商店数目指数级增长的，即最多去 $O(\log T)$ 个商店。
将第二维只开到 $O(\log T)$ ，这部分的时间复杂度就优化到了 $O(n \log T)$ ，结合算法二，期望得分 70 分。

算法四

按照我们的排序方式，$a_i = 0$ 的商店一定是在序列 p 末尾的。
设共有 k 个 $a_i > 0$ 的商店，对这 k 个商店做算法三的 dp，对于每个最终状态 $dp(k, j)$ ，再贪心地按照 b_i 从小到大的顺序检查还能去几个 $a_i = 0$ 的商店。
时间复杂度 $O(n \log n + n \log T)$ ，期望得分 100 分。
为了模拟 NOIP ，没有开子任务，所以一些假的乱搞做法可能也可以得到一些分数。

算法一

dfs 枚举所有情况，期望得分 20 分。

算法二

考虑在限定每堆石子数目互不相同的前提下，用所有方案数减去先手必败的方案数。

设 $p(i) = (2^n - 1)^{\underline{i}}$ ，即 i 堆石子的总方案数。

设 $f(i)$ 表示 i 堆石子时先手必败的方案数。

我们考虑让前 $i-1$ 堆石子任意取，通过调整最后一堆石子的数目使得异或和为 0 ，方案数为 $p(i-1)$ 。

若前 $i-1$ 堆石子异或和为 0 ，因为最后一堆不能取 0 ，这种情况是不合法的，方案数为 $f(i-1)$ 。

若前 $i-1$ 堆石子中，有 $i-2$ 堆石子异或起来是 0 ，那么最后一堆石子就只能和另一堆石子数目相同，也是不合法的，方案数为 $(i-1)\cdot(2^n-i+1)\cdot f(i-2)$ 。

于是得到 $f(i) = p(i-1) - f(i-1) - (i-1)\cdot(2^n-i+1)\cdot f(i-2)$ ，边界为 $f(1) = f(2) = 0$ ，直接 $O(n)$ 递推即可。

可能存在一些其他容斥做法，根据实现优劣可以得到不同的分数。

算法一

暴力（如小范围内枚举分母，对分子 dfs），期望得分 20 分。
随机乱搞，贪心乱搞，根据实现优劣可以获得 20 到 100 分。

算法一

暴力（如小范围内枚举分母，对分子 dfs），期望得分 20 分。
随机乱搞，贪心乱搞，根据实现优劣可以获得 20 到 100 分。

算法二

当图为以 1 为中心的菊花图时，收益是 $t_1 \cdot (x - t_1)$，令 $t_1 = x/2$ 即可。
结合算法一，期望得分 40 分。

算法三

注意到，若 u, v 之间没有边，记 s_u, s_v 分别表示与 u, v 相连的点的 t 之和。
则 u, v 的总贡献应当是 $\frac{1}{2}(s_u \times t_u + s_v \times t_v)$ 。
这意味着在保持 $t_u + t_v$ 不变时，让 t_u 或 t_v 中的一者为 0 答案不会变劣。
那么一定有一种最优解是满足所有 $t > 0$ 的点之间两两有边，即它们的导出子图在原图上形成了一个团。
设这个团有 k 个点，不难发现令每个点 $t = \frac{x}{k}$ 最优，此时答案为 $x^2 \times \frac{k(k-1)}{2k^2}$ 。
可以看出答案是关于 k 递增的，我们需要在图中找出一个最大团。
枚举点集的每个子集，并检验导出子图是否为团，时间复杂度 $O(2^n \cdot n)$ ，期望得分 70 分。

算法四

使用 meet in the middle + 状压 dp 搜索图中的最大团，时间复杂度 $O(2^{n/2} \cdot n)$ ，期望得分 100 分。
具体地，取 $p = n/2$，对前 p 个点，预处理 $f(S)$ 表示点集 S 的导出子图的最大团大小，对后 $n - p$ 个点，枚举一个集合 T，若点集 T 的导出子图是团，并且 T 中所有点邻居的交与前 p 个点交集为 S ，就找到了一个大小为 $|T| + f(S)$ 的团。

算法五

使用 Bron–Kerbosch 算法搜索图中的最大团，时间复杂度 $O(3^{n/3})$ ，期望得分 100 分。
也存在一些随机化算法能够以较大概率找出最大团，此处不详细展开。

本套题综合考察了多种 NOIP 常见的知识点，相信能给拼搏于逐梦之路上的你，提供一个有力的援助。