

The background of the slide features a close-up, slightly blurred image of an anime character's face. The character has long, flowing blonde hair and large, expressive blue eyes. Their mouth is open in a wide, joyful smile, showing their teeth. The overall tone is bright and cheerful, with a soft, pastel-like color palette.

浅谈时间复杂度

复旦大学白学研究会

温昕岳

渐进记号

- **O**: 若 $\exists c > 0, n_0 > 0: \forall n \geq n_0, 0 \leq T(n) \leq c \cdot f(n)$, 则称 $T(n) = O(f(n))$
- **Ω** : 若 $\exists c > 0, n_0 > 0: \forall n \geq n_0, 0 \leq c \cdot f(n) \leq T(n)$, 则称 $T(n) = \Omega(f(n))$
- **Θ** : 若 $T(n) = O(f(n))$ 且 $T(n) = \Omega(f(n))$, 则称 $T(n) = \Theta(f(n))$
- **o**: 若 $\forall c > 0, \exists n_0 > 0: \forall n \geq n_0, 0 \leq T(n) < c \cdot f(n)$, 则称 $T(n) = o(f(n))$
- **ω** : 若 $\forall c > 0, \exists n_0 > 0: \forall n \geq n_0, 0 \leq c \cdot f(n) < T(n)$, 则称 $T(n) = \omega(f(n))$

欧几里得算法

- $\gcd(a, b) = \gcd(b, a \% b)$
- 时间复杂度 $O(\log \min(a, b))$
- 证明：不妨设 $a > b$
- 若 $b \leq \frac{a}{2}$, 则 $a \% b < b \leq \frac{a}{2}$
- 若 $\frac{a}{2} < b < a$, 则 $a \% b = a - b < a - \frac{a}{2} = \frac{a}{2}$
- 综上所述, $a \% b < \frac{a}{2}$

欧几里得算法

- $O(\log \min(a, b))$ 的时间复杂度不可改进，即最坏情况下，其运行时间复杂度为 $\Theta(\log \min(a, b))$
- 构造数据，令 $a = \text{fib}(n), b = \text{fib}(n - 1)$ ，可发现每次递归过程是把两个变量对应的斐波那契数列项数-1。
- $\lim_{n \rightarrow \infty} \frac{\text{fib}(n)}{\text{fib}(n-1)} = \frac{\sqrt{5}+1}{2}$
- $\log_2 b \leq \text{递归层数} \leq \log_{\frac{\sqrt{5}+1}{2}} b$

CF 193 B

- 给定四个长度为 n ，下标从 1 到 n 的数组 a, b, k, p ，保证 $p[1], p[2], \dots, p[n]$ 是 $1, 2, \dots, n$ 的一个排列。你要对数组 a 进行恰好 u 次操作，每次可以在以下两种操作中选择一种：
 - 1. 对所有 $i = 1, 2, \dots, n$ ，将 $a[i]$ 修改为 $a[i] \text{ xor } b[i]$;
 - 2. 对所有 $i = 1, 2, \dots, n$ ，将 $a[i]$ 修改为 $a[p[i]] + r$ 。
- 问 u 次操作之后， $\sum_{i=1}^n a_i k_i$ 的最大值是多少。

暴力搜索的时间复杂度优化

- 原始的暴力搜索时间复杂度为 $O(nu2^u)$ 。
- 在搜索树的每一层计算，而非到了叶子节点之后再计算。时间复杂度为 $O(n2^u)$ 。
- 连续执行两次1操作是没有意义的。
- 时间复杂度的递推式为 $T(u) = T(u - 1) + T(u - 2)$ ，即斐波那契数列。时间复杂度为 $O(n1.62^u)$ 。

bitset

- 维护一个01序列支持：
- 与/并/异或一个给定01序列
- 查询1的个数
- 查询第1个1的位置
- 查询某个位置之后的第1个1的位置
- 时间复杂度 $O(\frac{n}{w})$, w 为机器字长。

CF 506 D

- 给出一个 n 个点 m 条边的无向图，每条边有一个颜色。
- q 次询问，问有多少个颜色满足：只加入该颜色的边后 u_i 与 v_i 连通。
- $2 \leq n \leq 10^5, 1 \leq m, q \leq 10^5$

Solution

- 保存每个已出现的询问的答案，现在只需要考虑所有询问都不同的情况
- 枚举颜色，加入对应的所有边，hash表存所有点所属连通块的编号。
- 对于询问 (u, v) ，设 $\deg(u) \geq \deg(v)$ 不失一般性，暴力枚举 v 临边的颜色，hash表查询 u, v 的连通块编号是否相同。
- 时间的最坏情况是选出来度数前 $O(\sqrt{q})$ 大的点两两询问，时间复杂度为 $O(m\sqrt{q})$ 。

均摊时间复杂度

- 若 $\forall n$, 完成长度为 n 的操作序列的总时间复杂度为 $O(n * T(\dots))$, 则称单次操作的均摊时间复杂度为 $O(T(\dots))$ 。

高精度加法

- 高精度整数：使用int/char等基本类型的数组来表示高位的整数
- 高精度加法：对高精度整数进行加法运算
- 暴力算法：按照加法的定义，从低位到高位依次运算，直到没有进位为止。

BZOJ 2054

- 给出一个长度为 n 的全0序列，支持 m 次区间赋值操作。问最后每个位置的值是什么。
- $1 \leq n \leq 10^6, 1 \leq m \leq 10^7$

启发式合并

- 给出一张 n 个点的无向图，初始时没有边，进行 m 个下列操作：
- 加一条边
- 询问两点是否连通
- 启发式合并：合并两个连通块时，遍历元素比较少的一个连通块，暴力修改其中的元素的信息。一共有 $O(n \log n)$ 次修改操作。

轻重链剖分

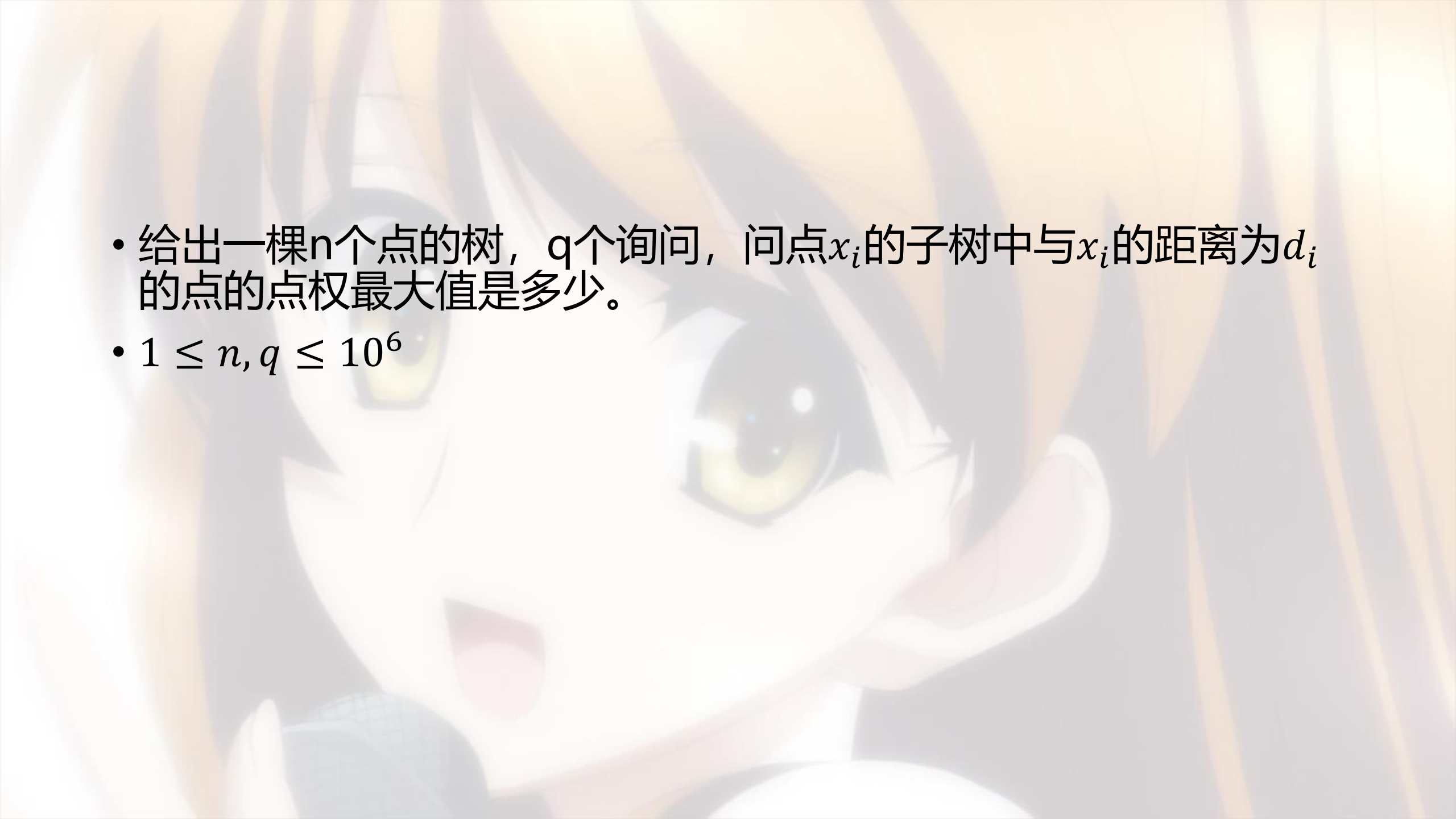
- 如何快速求出树上两点的LCA?
- 设每个点子树大小最大的儿子为重儿子，其他儿子为轻儿子。
- 连接轻儿子的边称为轻边。一段连续的重边称为重链。
- 一个点到根的路径上经过的轻边数量为 $O(\log n)$ 。

CF 741D

- 给出一棵 n 个点的树，每条边上有一个范围为 $a \sim v$ 的字母，问最长满足该条件的简单路径：取出路径上的边的字母，对字母进行重排列可以得到一个回文串。
- $1 \leq n \leq 5 \times 10^5$

轻重链剖分式启发式合并

- 设 a_x 为 x 到根的路径上的边权的异或和
- 路径 (u,v) 的边权异或和= $a_u \text{ xor } a_v$
- 启发式合并?
- map TLE, 数组MLE
- 若想得到点 u 处的数据结构, 先递归解决 u 的所有轻儿子, 然后清空数据结构。然后递归解决 u 的重儿子, 再将轻儿子子树中的所有元素加入数据结构中。
- 任意时刻只需要维护一个数据结构
- 时间复杂度 $O(n|\Sigma| \log n)$

- 
- 给出一棵 n 个点的树， q 个询问，问点 x_i 的子树中与 x_i 的距离为 d_i 的点的点权最大值是多少。
 - $1 \leq n, q \leq 10^6$

长链剖分

- 设 $d[i][j]$ 表示以 i 为根的子树中与 i 距离为 j 的点的权值最大值。
- j 最大为 i 的子树大小
- 按子树大小启发式合并?
- j 最大为以 i 为根的子树的高度
- 将高度小的子树合并到高度大的子树
- 时间复杂度 $O(n)$

长链剖分

- 对树长链剖分，从一个点跳到根节点会经过多少条轻边？
- 考虑在节点数 $\leq n$ 的情况下构造最坏情况：
- 一定是跳到根过程中一直是轻边
- 跳的过程中深度为 $k, k - 1, \dots, 1$
- 为了使其成为轻边，至少需要配上深度为 $k, k - 1, \dots, 1$ 的终边
- k 为 $O(\sqrt{n})$ 级别

BZOJ 4033

- 有一棵点数为 N 的树，树边有边权。给你一个在 $0 \sim N$ 之内的正整数 K ，你要在这棵树中选择 K 个点，将其染成黑色，并将其他的 $N-K$ 个点染成白色。将所有点染色后，你会获得黑点两两之间的距离加上白点两两之间距离的和的收益。问收益最大值是多少。
- $N \leq 2000, 0 \leq K \leq N$

树上背包

- 一条边的贡献 = 子树内黑点个数 \times 子树外黑点个数 + 子树内白点个数 \times 子树外白点个数
- 令 $f[i][j]$ 表示以 i 为根的子树中选了 j 个黑点时，子树内边的贡献和的最大值
- 转移时合并两棵子树，枚举两棵子树中的黑点个数。
- 时间复杂度 $O(n^3)$ 。
- 有意义的 j 最大为对应的点集的大小。
- 只枚举有意义的 j ，时间复杂度 $O(n^2)$ 。

HDOJ 6091

- 给出一棵 n 个点的数，问有多少种删边方式满足剩下的图的最大匹配数是 m 的倍数。
- $1 \leq n \leq 5 \times 10^4, 1 \leq m \leq 200$

树上背包

- 求树上最大匹配——贪心
- $dp[i][j][0/1]$ 表示以 i 为根的子树的最大匹配数 $\%m=j$ 且贪心结果中 i 是否被选的删边方案数
- 时间复杂度 $O(nm^2)$ 。
- 合并只枚举到 $\min(m, size_u)$ ，总时间复杂度为 $O(nm)$ 。
- 若有一方的大小 $\geq m$ ，则可认为是把另一方的每个尚存节点用 $O(m)$ 的时间删除了，时间复杂度为 $O(nm)$
- 若两方的大小都 $< m$ ，则这样的极大集合是不相交的，对于一个大小为 k 的极大集合，每个节点消耗的时间为 $O\left(\frac{k^2}{k}\right) = O(k) = O(m)$

期望时间复杂度

- 对于任意可能的数据，运行的期望时间为 $O(T(n))$ 的，则称期望时间复杂度为 $O(T(n))$ 。
- 与平均时间复杂度的区别：平均时间复杂度是建立在每种可能的数据等概率出现情况下的期望时间复杂度。

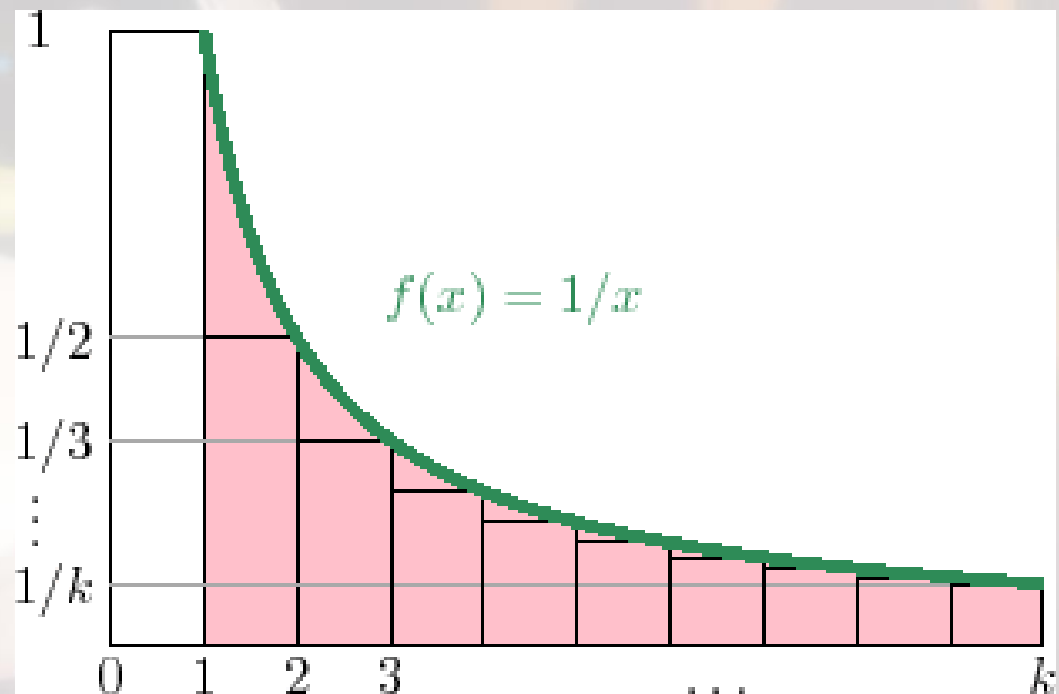
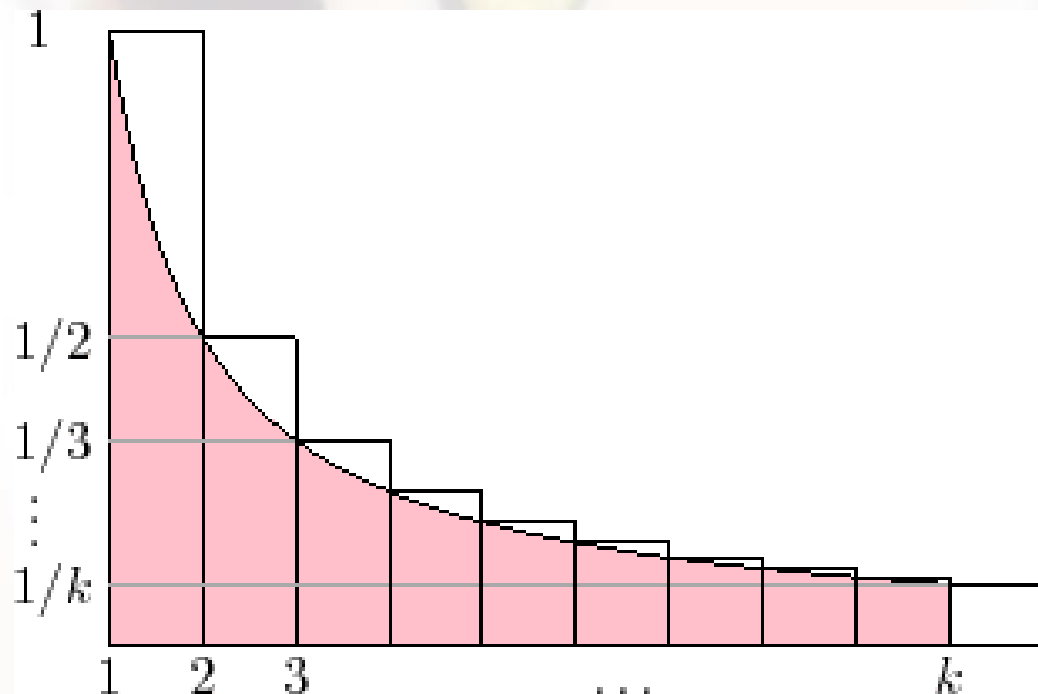
用随机化优化一类最优化问题

2	1	3	7	4	6	10	8	5	9
■	■	■	■	■	■	■	■	■	■

- 假如有一个随机的1~n的排列，那么期望有多少个位置会成为从起始位置到当前位置的最大值？
- $E(\sum_{i=1}^n [\text{第}i\text{个位置为前缀最大值}]) =$
 $\sum_{i=1}^n E([\text{第}i\text{个位置为前缀最大}]) = \sum_{i=1}^n \frac{1}{i}$

积分计算时间复杂度

- 如何估算 $\sum_{i=1}^n \frac{1}{i}$ 的量级?
- $O\left(\sum_{i=1}^n \frac{1}{i}\right) = O\left(\int_1^n \frac{1}{x} dx\right) = O\left(1 + \int_1^n \frac{1}{x} dx\right) = O(\log n)$



用随机化优化一类最优化问题

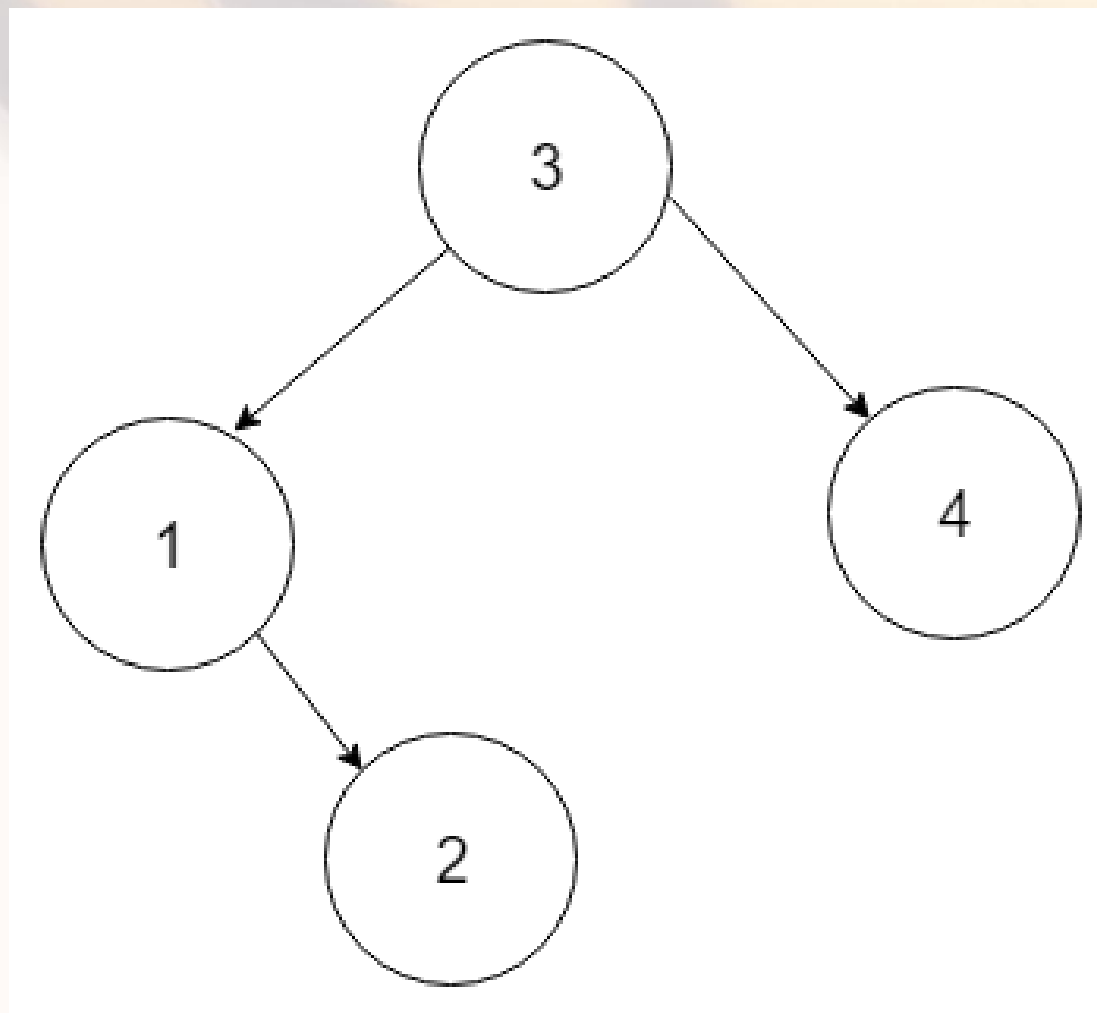
- 有些最优化问题想要直接求出答案是困难，但是想要判断答案是否至少是某个值是容易的。更形式化地说就是，求出最优解的时间复杂度是 $O(T_1)$ ，判断最优解是否至少是 V 的时间复杂度是 $O(T_2)$ ，且 $O(T_1)$ 是一个比 $O(T_2)$ 更高的复杂度。
- 有时会碰到这样的问题：给出 n 个子问题，求出所有子问题的最优解的最大值。

用随机化优化一类最优化问题

- 设第 i 个子问题的最优解是 ans_i ，如果我们对这 n 个问题随机排列，那么 ans 序列中任意两个数的相对大小就是完全随机的。
- 我们以随机序列中的顺序去试图求出每个子问题的解，但加上一个小小的剪枝：先用 $O(T_2)$ 的时间判断一下这个子问题的最优解是否至少是它之前问题的最大值，如果不是，就不必求出这个子问题的最优解。
- 期望时间复杂度是 $O(T_2n + T_1 \log n)$ 。

二叉搜索树

左子树中的元素都小于根节点，右子树中的元素都大于根节点



建立平衡二叉搜索树

- 理想情况下，左右子树的大小应该尽量均等，树高为 $O(\log n)$ 。
- 给出所有数字后，建立平衡的二叉搜索树：
- 找到当前数字的中位数作为根节点，递归建立左右子树
- 时间复杂度 $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$ ，解得 $T(n) = O(n \log n)$

二叉搜索树的平均时间复杂度

- 在此我们讨论插入一个节点的平均时间复杂度，亦即节点的深度。
- 设插入顺序是 $1 \sim n$ 的一个随机排列。
- $\forall i < x$, i 是 x 的祖先当且仅当 i 是“ $i, i + 1, \dots, x$ ”中第1个插入的。
- $\forall i > x$, i 是 x 的祖先当且仅当 i 是“ $x, x + 1, \dots, i$ ”中第1个插入的。
- $$E[h_x] = E(\sum_{i=1}^n [i \text{ 是 } x \text{ 的祖先}]) = \sum_{i=1}^n E([i \text{ 是 } x \text{ 的祖先}]) = 1 + \sum_{i=1}^{x-1} \frac{1}{x-i+1} + \sum_{i=x+1}^n \frac{1}{i-x+1} = 1 + \sum_{i=2}^x \frac{1}{i} + \sum_{i=2}^{n-x+1} \frac{1}{i} = O(\log n)$$

父亲随机的有根树

- 1 号点为根，之后按照点的编号从小到大加入每个点，加入点 i 时，从点 $1, 2, \dots, i - 1$ 中随机选一个点，将其与点 i 连边。
- 期望树高为 $O(\log n)$ 。
- 证明：假设对于 $1 \dots n$ 都满足 $dep_i \leq k \log i$ ， k 为常数。则有
$$dep_{n+1} = \frac{\sum_{i=1}^n dep_i}{n} + 1 \leq \frac{k \sum_{i=1}^n \log i}{n} + 1 \approx k(\log n - \log e) + 1$$
（斯特林公式）， k 取足够大时，上式 $\leq k \log(n + 1)$ 。

CF 896 C

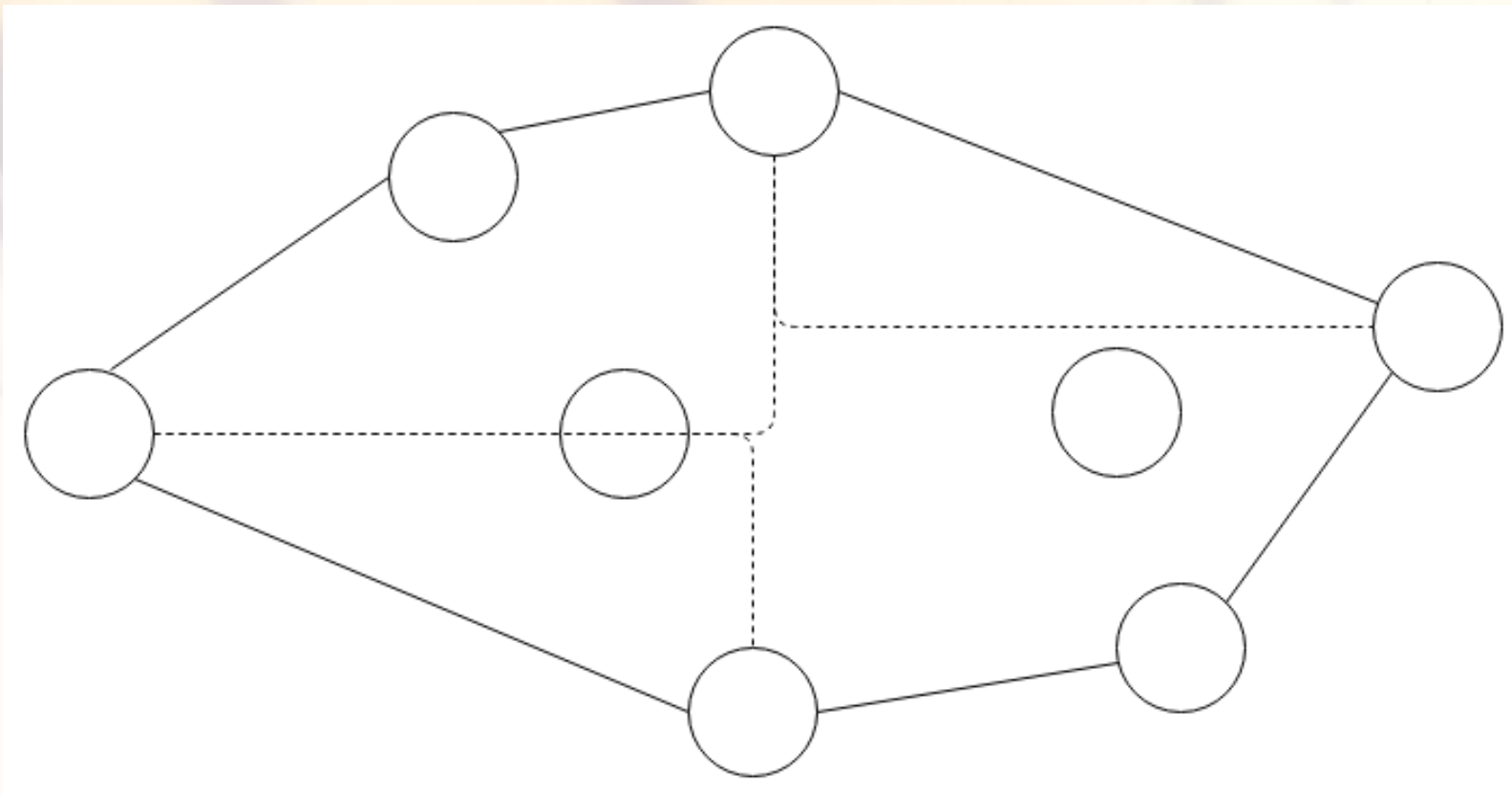
- 给出一个长度为 n 的整数序列，进行 m 次操作。
- 操作类型：
 1. 区间加
 2. 区间赋值
 3. 求区间内第 k_i 大的数
 4. 求区间内元素的 x_i 次幂之和，答案模 y_i
- 保证包括操作类型在内的数据都是随机的
- $1 \leq n, m \leq 10^5, 1 \leq k_i \leq r_i - l_i + 1, 1 \leq x_i, y_i \leq 10^9$

Chtholly Tree

- 将连续的一段相同的数压成一个点，用线段树/平衡树维护
- 每个点期望经过4次会被删除
- 考虑到线段树和平衡树的时间复杂度，总的维护时间复杂度为 $O(n \log n)$

随机点的凸包

- 平面上有 n 个坐标随机生成的点，期望下它们之中有多少个点在凸包边界上？
- 找到最左、最右、最上、最下的4个点，将凸包分成4部分
- 在每部分中，每个点的纵坐标为前缀最值是该点在凸包上的必要条件
- $E([\sum \text{点} i \text{在凸包上}]) = \sum E([\text{点} i \text{在凸包上}]) \leq \sum E([\text{点} i \text{为前缀最值}]) = O(\log n)$



整点的凸包

- 平面坐标系上，横纵坐标范围都为 $[0, R]$ 的点构成的凸包边界上的点最多有多少个（不考虑凸包边界上多点共线的情况）？
- 考虑右下凸包，因为任意相邻两点的斜率为正，所以 $\frac{y_{i+1}-y_i}{x_{i+1}-x_i} = \frac{p_i}{q_i}$ ，其中 p_i, q_i 为互质的正整数。
- 设右下凸包上共有 n 个点。
- $\sum_{i=1}^{n-1} p_i + q_i \leq \sum_{i=1}^{n-1} y_{i+1} - y_i + x_{i+1} - x_i = y_n - y_1 + x_n - x_1 \leq 2R$
- 至多有 $x - 1$ 对 (p, q) 满足 $p + q = x$
- $\sum_{i=1}^{x+1} i(i-1) \geq \sum_{i=1}^x i^2 = \frac{x(x+1)(2x+1)}{6}$ ，故 $x = O(R^{\frac{1}{3}})$ ， $n = \frac{(1+x)x}{2} = O(R^{\frac{2}{3}})$