Scoins

$n \le 2$签到分。

认真读题可以得到:

$$Ans = \left\lfloor \sum_{i=1}^{n} \frac{1}{2\sqrt{i}} \right\rfloor$$

这样可以做$n \le 10^5$。

然后通过将$\sqrt{i}$替换为$\frac{\sqrt{i}+\sqrt{i-1}}{2}$和$\frac{\sqrt{i}+\sqrt{i+1}}{2}$进行放缩。

最后得到$Ans \in (\sqrt{n+1}-1, \sqrt{n})$。

Hanoi

$n, m \le 5$手玩签到分。

$m \le 3$分类讨论:

$m = 1$答案肯定是0。

$m = 2$只有$n = 1$答案是0，否则无解。

$m = 3$经典汉诺塔问题，答案是2^{n-1}。

$m > 3$的时候答案总是在`long long`范围内。通过手玩$m = 4$观察可以猜想/归纳一个DP式子：$f(n, m)$表示n个盘子m个柱子的汉诺塔的答案。

$$f(n, m) = \max_{0 \le k < n} \{f(k, m) + 2f(n-k, m-1)\}$$

这样是$O(n^2 m)$，可以得到70分。

然后可以发现m固定的时候k随n单调变化：当圆盘增多的时候一次操作的圆盘一定更多，否则如果圆盘少的时候操作的圆盘多，减少操作的圆盘总会得到一个不会更差的解。

于是用单调指针优化$O(nm)$。虽然有T组询问，但是预处理以后就可以$O(1)$回答了。

Tree

题目本质就是在问直径在$[L,R]$区间的子树连通块个数。

$n\le 5$爆搜判断即可。

对于链的做法非常简单，一个直径为K的连通块个数自然是$n-K$。

$n\le 100$显然我们需要一个多项式做法。要求直径大于一个值不太好做，但是直径小于一个值会比较好做，先只考虑直径在$[1,R]$区间的连通块个数。

类似于求直径的DP，$f[p][d]$表示以点p为最浅点，最深点距离点p深度为d的连通块个数。进行树形DP，将一个新的儿子v加入时：

$$f'[p][\max(d_1,d_2+1)]=\sum_{d_1+d_2+1\le R} f[p][d_1]\times f[v][d_2]$$

这已经给出了一个明显的$O(n^3)$的做法。

$n\le 5000$注意左侧有一个max无法优化，那么就分类讨论，枚举d_2：

$$f'[p][d_2+1]+=\left(\sum_{d_1+d_2+1\le R,d_1\le d_2} f[p][d_1]\right)\times f[v][d_2]$$
$$f'[p][d_1]+=\sum_{d_1+d_2+1\le R,d_1>d_2} f[p][d_1]\times f[v][d_2]$$

第一个式子显然可以一个前缀和优化(或数据结构优化)，但是第二个式子不好处理，似乎更类似一个区间加。

需要维护单点修改($f'[p][d_2+1]$)以及区间加上一个数组的若干倍($f'[p][d_1]$)并维护区间和。这个用线段树是可以维护的。

注意到最开始$f'[p][d]=f[p][d]$，那么事实上"区间加上一个数组的x倍"就是"区间加上自己的x倍"，那就是区间乘$x+1$倍。

因此可以用一个`map<int,int>`来维护每个点需要加上多少倍的自己，最后用线段树实现区间乘、单点修改、区间求和就做完了。

$O(n^2\log n)$。这个做法有一个log，在5000的数据下可能有些困难，但是依然是进一步优化算法的一个可能方向。

事实上，仔细审视之前$O(n^3)$的做法。如果在枚举d_1, d_2的时候保证只枚举到子树最深深度。那么每次合并的复杂度是两棵子树最深深度的乘积，显然小于等于两棵子树点数的乘积。而如果每次合并的代价是两棵子树点数的乘积，那么代价之和存在一个物理意义：有多少对点(每对点都会在其LCA处被计算一次贡献)。显然有C_n^2对点，那么这个算法的总复杂度就是$O(n^2)$的。

$O(n^2)$。这个做法启发我们要积极理由树形DP是各种均摊性。然而注意到这里使用了一个放缩【每次合并的复杂度是两棵子树最深深度的乘积，显然小于等于两棵子树点数的乘积】。如果放弃这个放缩，利用上子树深度的性质，也许可以得到一个更好的界，于是引出本题最重要的一个均摊性质：长链剖分中短链长度和的均摊性。

$n \leq 2 \times 10^5$：树上和深度有关的DP问题一般都可以用长链剖分来优化。每次先DFS子树深度最大的儿子，DP过程用之前提到的线段树做法实现。这样第一个子树只需要执行$f'[p][d+1] = f[p][d]$，如果可以$O(1)$实现这样一个平行拷贝的操作，那么总复杂度(不考虑线段树和`map<int,int>`操作复杂度)就是$O(\sum d_2)$，即短链长度和，而每个短链最多只会备用一次，因此复杂度是$O(n)$的。

而如何$O(1)$平行拷贝是一个实现上的技巧问题：先用类似于树链剖分的方法按照长链标号，这样一条长链就是一条连续的区间，就可以直接在这条区间上操作了。

$O(n \log n)$。