7 月 3 日模拟赛题解

mcfx

2019 年 7 月 3 日

枚举根，模拟即可。

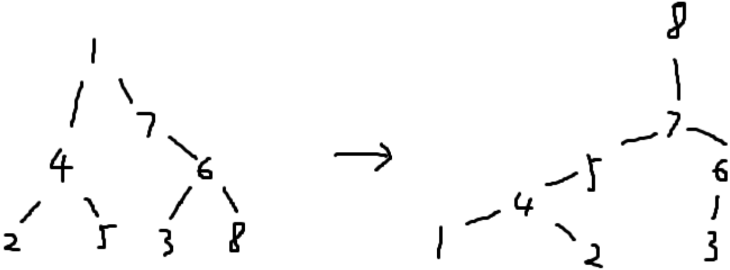
实际上这个条件只是为了保证点权两两不同。

实际上这个条件只是为了保证点权两两不同。
考虑一个点 x，在哪些点作为根时是神仙的，显然是与 x 只通过点权小于 s_x 的点就能连通的点。

实际上这个条件只是为了保证点权两两不同。
考虑一个点 x，在哪些点作为根时是神仙的，显然是与 x 只通过点权小于 s_x 的点就能连通的点。
那么如果按照点权从小到大依次加入，这些点就是在加入 x 时和 x 连通的所有点。

实际上这个条件只是为了保证点权两两不同。
考虑一个点 x，在哪些点作为根时是神仙的，显然是与 x 只通过点权小于 s_x 的点就能连通的点。
那么如果按照点权从小到大依次加入，这些点就是在加入 x 时和 x 连通的所有点。
把并查集的过程建一棵树，那么这棵树上每个点到根的路径就是这个点的答案。

实际上这个条件只是为了保证点权两两不同。
考虑一个点 x，在哪些点作为根时是神仙的，显然是与 x 只通过点权小于 s_x 的点就能连通的点。
那么如果按照点权从小到大依次加入，这些点就是在加入 x 时和 x 连通的所有点。
把并查集的过程建一棵树，那么这棵树上每个点到根的路径就是这个点的答案。



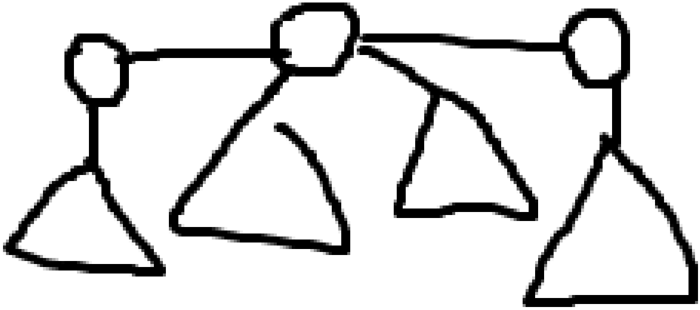
当点权不同时，前面的做法会遇到问题。相同点权的点需要特殊处理。

当点权不同时，前面的做法会遇到问题。相同点权的点需要特殊处理。

对于相同点权的点，只需要先遍历一遍，把每个点相连的子树打上标记，然后再依次加入并查集即可。

当点权不同时，前面的做法会遇到问题。相同点权的点需要特殊处理。

对于相同点权的点，只需要先遍历一遍，把每个点相连的子树打上标记，然后再依次加入并查集即可。



按照题意模拟。

m 的约数个数不超过 1000（似乎最大是 720）。

m 的约数个数不超过 1000（似乎最大是 720）。
可以直接预处理出每个数的约数，然后对每个位置枚举约数求和。

m 的约数个数不超过 1000（似乎最大是 720）。
可以直接预处理出每个数的约数，然后对每个位置枚举约数求和。
对于 m 是质数，需要使用 miller rabin 算法判断。或者如果不打算过后面的点，可以只使用 $1 \sim 10^5$ 的质数试除。

先用 pollard rho 分解 m。

先用 pollard rho 分解 m。
如果实现两个函数 $modify(x,y)$ 和 $query(x)$，$modify(x,y)$ 表示 $f[x]+=y$，$query(x)$ 表示求出 $\sum_{y|x} f[y]$，那么可以很容易解决原问题。

先用 pollard rho 分解 m。
如果实现两个函数 $modify(x, y)$ 和 $query(x)$，$modify(x, y)$ 表示 $f[x]+=y$，$query(x)$ 表示求出 $\sum_{y|x} f[y]$，那么可以很容易解决原问题。
考虑把 m 的质数分成两部分，即找到 A, B，满足 $gcd(A, B) = 1, A \cdot B = m$，然后使 $d(A) + d(B)$ 尽量小。

先用 pollard rho 分解 m。
如果实现两个函数 $modify(x,y)$ 和 $query(x)$，$modify(x,y)$ 表示 $f[x]+=y$，$query(x)$ 表示求出 $\sum_{y|x} f[y]$，那么可以很容易解决原问题。
考虑把 m 的质数分成两部分，即找到 A,B，满足 $gcd(A,B)=1, A\cdot B=m$，然后使 $d(A)+d(B)$ 尽量小。
对于 $modify$，可以对所有满足 $x|z$ 且 $gcd(\frac{z}{x},B)=1$ 的 $g[z]+=y$，这部分复杂度是 $O(d(A))$。

先用 pollard rho 分解 m。
如果实现两个函数 $modify(x, y)$ 和 $query(x)$，$modify(x, y)$ 表示 $f[x] += y$，$query(x)$ 表示求出 $\sum_{y|x} f[y]$，那么可以很容易解决原问题。
考虑把 m 的质数分成两部分，即找到 A, B，满足 $gcd(A, B) = 1, A \cdot B = m$，然后使 $d(A) + d(B)$ 尽量小。
对于 $modify$，可以对所有满足 $x|z$ 且 $gcd(\frac{z}{x}, B) = 1$ 的 $g[z] += y$，这部分复杂度是 $O(d(A))$。
对于 $query$，可以所有满足 $z|x$ 且 $gcd(\frac{x}{z}, A) = 1$ 的 $g[z]$ 的和，这部分复杂度是 $O(d(B))$。

先用 pollard rho 分解 m。
如果实现两个函数 $modify(x, y)$ 和 $query(x)$，$modify(x, y)$ 表示 $f[x] += y$，$query(x)$ 表示求出 $\sum_{y|x} f[y]$，那么可以很容易解决原问题。
考虑把 m 的质数分成两部分，即找到 A, B，满足 $gcd(A, B) = 1, A \cdot B = m$，然后使 $d(A) + d(B)$ 尽量小。
对于 $modify$，可以对所有满足 $x|z$ 且 $gcd(\frac{z}{x}, B) = 1$ 的 $g[z] += y$，这部分复杂度是 $O(d(A))$。
对于 $query$，可以所有满足 $z|x$ 且 $gcd(\frac{x}{z}, A) = 1$ 的 $g[z]$ 的和，这部分复杂度是 $O(d(B))$。
$d(A) + d(B)$ 近似是 $O(\sqrt{d(M)})$ 的，那么总复杂度是 $O(n\sqrt{d(M)} + m^{\frac{1}{4}} \log m)$。

同样先分解 m。

同样先分解 m。
考虑对序列分块，每块内暴力，块间进行一次高维前缀和。

同样先分解 m。
考虑对序列分块，每块内暴力，块间进行一次高维前缀和。
设块大小为 S，m 的质因子次数之和为 $f(m)$，那么复杂度为 $n \cdot S + \frac{n}{S} f(m) d(m)$。

同样先分解 m。
考虑对序列分块，每块内暴力，块间进行一次高维前缀和。
设块大小为 S，m 的质因子次数之和为 $f(m)$，那么复杂度为 $n \cdot S + \frac{n}{S} f(m) d(m)$。
取 $S = \sqrt{f(m)d(m)}$ 时取到最优，复杂度为 $n \cdot \sqrt{f(m)d(m)}$。

枚举每个物品选不选即可。

dp，可以用 bitset 记录每个重量是否合法。

dp，可以用 bitset 记录每个重量是否合法。
输出方案有一种空间和时间都只多 $\log$ 的做法：

dp，可以用 bitset 记录每个重量是否合法。
输出方案有一种空间和时间都只多 $\log$ 的做法：
分治，每次加入左半部分的物品，然后递归右半部分。求出右半部分的答案之后，再递归左半部分。

将物品分成两半，每一半暴力枚举，然后 sort。(也可以依次插入，然后归并排序，会快一些)

将物品分成两半，每一半暴力枚举，然后 sort。(也可以依次插入，然后归并排序，会快一些)
两半之间可以 two pointers。

同测试点 3，不过由于 $\frac{n}{2} = 30$，可能内存开不下，一种方法是枚举前几个是否选择，剩下的用上面的方法。

观察发现，每个物品是两个 1 和一堆 0，而目标对应位置是若干个 1。那么这实际是一个求图的匹配。

观察发现，每个物品是两个 1 和一堆 0，而目标对应位置是若干个 1。那么这实际是一个求图的匹配。
进一步观察发现，每个物品的两个 1，一个在 150 之前，一个在 150 之后，那么这是一个二分图。

观察发现，每个物品是两个 1 和一堆 0，而目标对应位置是若干个 1。那么这实际是一个求图的匹配。
进一步观察发现，每个物品的两个 1，一个在 150 之前，一个在 150 之后，那么这是一个二分图。
求一个二分图匹配即可。

同上，不过是一般图匹配。可以尝试随机化算法以获得一定暴力分~~或者去 uoj 粘个板子过掉~~。

同上，不过是一般图匹配。可以尝试随机化算法以获得一定暴力分~~或者去 uoj 粘个板子过掉~~。
~~uoj 上现在最快的代码也不是带花树，而是个像二分图匹配一样的东西。~~

对给定的二分图需要找一个边的子集使得每个点的度数都是 3。

对给定的二分图需要找一个边的子集使得每个点的度数都是 3。
考虑如下建图：

对给定的二分图需要找一个边的子集使得每个点的度数都是 3。
考虑如下建图：
对原图中的一个点 x，建三个新点 x_0, x_1, x_2。

对给定的二分图需要找一个边的子集使得每个点的度数都是 3。
考虑如下建图：
对原图中的一个点 x，建三个新点 x_0, x_1, x_2。
对原图中的一条边 $e(x, y)$，建两个新点 e_x, e_y，然后连 7 条边：

对给定的二分图需要找一个边的子集使得每个点的度数都是 3。
考虑如下建图：
对原图中的一个点 x，建三个新点 x_0, x_1, x_2。
对原图中的一条边 $e(x, y)$，建两个新点 e_x, e_y，然后连 7 条边：
$(e_x, x_0), (e_x, x_1), (e_x, x_2), (e_y, y_0), (e_y, y_1), (e_y, y_2), (e_x, e_y)$。

对给定的二分图需要找一个边的子集使得每个点的度数都是 3。
考虑如下建图：
对原图中的一个点 x，建三个新点 x_0, x_1, x_2。
对原图中的一条边 $e(x, y)$，建两个新点 e_x, e_y，然后连 7 条边：
$(e_x, x_0), (e_x, x_1), (e_x, x_2), (e_y, y_0), (e_y, y_1), (e_y, y_2), (e_x, e_y)$。
对这个新图求出最大匹配，而不在匹配中的 (e_x, e_y) 就是答案。

对给定的二分图需要找一个边的子集使得每个点的度数都是 3。
考虑如下建图：
对原图中的一个点 x，建三个新点 x_0, x_1, x_2。
对原图中的一条边 $e(x, y)$，建两个新点 e_x, e_y，然后连 7 条边：
$(e_x, x_0), (e_x, x_1), (e_x, x_2), (e_y, y_0), (e_y, y_1), (e_y, y_2), (e_x, e_y)$。
对这个新图求出最大匹配，而不在匹配中的 (e_x, e_y) 就是答案。
当然随机化算法应该也有不错的期望得分。

物品都是随机生成的，可以无限取。

物品都是随机生成的，可以无限取。
标程用的方法是，考虑一个 $solve(x)$ 函数，返回 K 个最接近 x 的值，K 是一个自己选择的值。

物品都是随机生成的，可以无限取。
标程用的方法是，考虑一个 $solve(x)$ 函数，返回 K 个最接近 x 的值，K 是一个自己选择的值。
首先所有物品本身可以加入堆中。

物品都是随机生成的，可以无限取。
标程用的方法是，考虑一个 $solve(x)$ 函数，返回 K 个最接近 x 的值，K 是一个自己选择的值。
首先所有物品本身可以加入堆中。
然后考虑 $solve(\frac{x}{2})$，把结果中两两之和也加入堆。可以 two pointers 优化取到近似较优的解。

物品都是随机生成的，可以无限取。
标程用的方法是，考虑一个 $solve(x)$ 函数，返回 K 个最接近 x 的值，K 是一个自己选择的值。
首先所有物品本身可以加入堆中。
然后考虑 $solve(\frac{x}{2})$，把结果中两两之和也加入堆。可以 two pointers 优化取到近似较优的解。
最后取出堆中最优的 K 个返回。

物品都是随机生成的，可以无限取。
标程用的方法是，考虑一个 $solve(x)$ 函数，返回 K 个最接近 x 的值，K 是一个自己选择的值。
首先所有物品本身可以加入堆中。
然后考虑 $solve(\frac{x}{2})$，把结果中两两之和也加入堆。可以 two pointers 优化取到近似较优的解。
最后取出堆中最优的 K 个返回。
测试点 9 是取 $K = 10^5$ 通过的，大约用时 1 分钟。

物品都是随机生成的，可以无限取。
标程用的方法是，考虑一个 $solve(x)$ 函数，返回 K 个最接近 x 的值，K 是一个自己选择的值。
首先所有物品本身可以加入堆中。
然后考虑 $solve(\frac{x}{2})$，把结果中两两之和也加入堆。可以 two pointers 优化取到近似较优的解。
最后取出堆中最优的 K 个返回。
测试点 9 是取 $K = 10^5$ 通过的，大约用时 1 分钟。
测试点 8 是取 $K = 10^6$ 通过的，大约用时 10 分钟。

物品都是随机生成的，可以无限取。
标程用的方法是，考虑一个 $solve(x)$ 函数，返回 K 个最接近 x 的值，K 是一个自己选择的值。
首先所有物品本身可以加入堆中。
然后考虑 $solve(\frac{x}{2})$，把结果中两两之和也加入堆。可以 two pointers 优化取到近似较优的解。
最后取出堆中最优的 K 个返回。
测试点 9 是取 $K = 10^5$ 通过的，大约用时 1 分钟。
测试点 8 是取 $K = 10^6$ 通过的，大约用时 10 分钟。
测试点 10 的评分参数是取 $K = 10^5$ 造出的，用这个方法可能要 $K = 10^9$ 以上才能得到精确解，不过也欢迎大家吊打 std。