练习评讲

T1:NKOJ4877 低功率灯泡

何老板装修新房子。
房子有n个房间，何老板安排你在每个房间安装一盏灯。每间房间都需要不少于一定功率的灯泡才可以完全照亮。
何老板事先已经买了n盏灯，你可以直接使用它们。你也可以去附近的商店换新灯泡，商店里所有正整数功率的灯泡都有售。但由于背包空间有限，你至多只能换k个灯泡。
你需要找到一个合理的方案使得每个房间都被完全照亮，何老板想尽可能的省电。所以在每个房间都被完全照亮这个前提下使得n个灯泡的总功率尽可能小。

数据范围

对于 10% 的数据:
$1 <= k <= n <= 1000$
对于 28% 的数据:
$1 <= k <= n <= 5000$
对于 44% 的数据:
$1 <= k <= n <= 50000$
对于 100% 的数据:
$1 <= k <= n <= 500000$
$1 <= p[i] <= 10^9$
$1 <= w[i] <= 10^9$

T1:NKOJ4877 低功率灯泡

解题关键字：堆(后悔堆)

1.声明一个小根堆和一个大根堆
2.将每个房间所需灯按功率由大到小排序；
  将现有灯泡按功率由大到小排序；
3.依次讨论排序后的每个房间和灯泡，对于i号房间：
  将功率>=w[i](房间i的所需最低功率)的灯泡全部放入小根堆
  堆顶即为当前能满足i号房需求的最小功率灯泡，记为Ans[i]
  若堆为空，则表示没有灯泡满足i号房的需求，需要单独购买(记单独购买数量为num)
4.若学要单独购买的灯泡数num>k，则无解。否则有k-num个灯泡可以更换为功率更小的，显然更换差价最大的讨论：
  将差值Ans[i]-w[i]放入大根堆，选取最大的k-num个出来即可。

T2:NKOJ8307 公约数和

重庆南开信竞基础课程

何老板用$[1,K]$内的整数构成一个长度为N的整数数列$A_1, A_2, \ldots, A_N$。

他发现可以构造出 K^N 种不同的数列。

对于其中第i种数列，其公约数 $G_i = gcd(A_1, A_2, \ldots, A_N)$

何老板想请你帮忙计算出每个数列的公约数，并求出这些公约数之和。也就是计算

$G_1 + G_2 + \ldots + G_{K^N}$

答案可能很大 $mod\ (10^9+7)$ 再输出

$2 \le N \le 10^5$

$1 \le K \le 10^5$

T2:NKOJ8307 公约数和

题目考察：数学

总共有K^N(K,N<=10^5)种方案，规模巨大，显然不能直接计算。
注意到1<=K<=10^5，显然公约数也一定在$[1,K]$内，我们考虑$[1,K]$内每个数字对答案的贡献
对于数字X，在$[1,K]$内，只有这些数可以以X为公约数：$1*X,2*X,2*X,\dots,\frac{K}{X}*X$　共$\frac{K}{X}$个。

以X为公约数的数字有$\frac{K}{X}$个，构成的数列有$(\frac{K}{X})^N$种方案。

$Ans[X]$记录以X为最大公约数的方案数，$Ans[X]=(\frac{K}{X})^N$

但$Ans[X]$会把公约数大于X（X的倍数）的方案也算进去。
比如$N=4,K=9,$当$X=3$时，{6,6,6,6}、{9,9,9,9}等方案也会被算进$Ans[X]$里，
必须把以X的倍数为公约数的方案从$Ans[X]$中减掉。

$$Ans[X]=(\frac{K}{X})^N-Ans[2*X]-Ans[3*X]-Ans[4*X]-\cdots-Ans[\frac{K}{X}*X]$$

在算$Ans[X]$的时候，比X大的倍数们对应的$Ans[\ ]$已经事先算好，所以倒序计算$Ans[\ ]$

T2:NKOJ8307 公约数和

重庆南开信竞基础课程

题目考察：数学

```
for(x = k; x >= 1; x--)
{
    Ans[x] = KSM(k/x,n);                  //快速幂计算(k/x)^n % mod
    for(y = 2*x; y <= k; y += x)  //枚举x的倍数
        Ans[x] = Ans[x] - Ans[y];
}
```

$$\frac{n}{1}+\frac{n}{2}+\frac{n}{3}+\cdots+\frac{n}{n}$$

的时间复杂度为"调和级数"，O(logn)
所以，本题时间复杂度为o(nlogn)

$T(n) = n\sum_{i=1}^{n}\frac{1}{n} = O(n\ln n)$

具体过程如下

$$\begin{aligned}
&\sum_{i=1}^{n}\frac{1}{i}\\
&=\sum_{i=1}^{n}\int_{i}^{i+1}\frac{1}{\lfloor x\rfloor}dx\\
&=\int_{1}^{n+1}\frac{1}{x}-\frac{1}{x}+\frac{1}{\lfloor x\rfloor}dx\\
&=\int_{1}^{n+1}\frac{1}{x}dx+\int_{1}^{n+1}\frac{1}{\lfloor x\rfloor}-\frac{1}{x}dx\\
&\approx\ln(n+1)+\gamma
\end{aligned}$$

其中 $\gamma \approx 0.5772156649$

T2:NKOJ8307 公约数和

证明调和级数的时间复杂度约为logn $\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\ldots+\frac{1}{n}\approx log_2^n$

$\sum_{i=1}^{n}\frac{1}{i}=(\frac{1}{1})+(\frac{1}{2}+\frac{1}{3})+(\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7})+(\frac{1}{8}+\frac{1}{9}+\frac{1}{10}+\frac{1}{11}+\frac{1}{12}+\frac{1}{13}+\frac{1}{14}+\frac{1}{15})+(\frac{1}{16}+\ldots$

$\leq(\frac{1}{1})+(\frac{1}{2}+\frac{1}{2})+(\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4})+(\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8})+\ldots$

$=1*(\frac{1}{1})+2*(\frac{1}{2})+4*(\frac{1}{4})+8*(\frac{1}{8})+\ldots+log_2^n*(\frac{1}{log_2^n})$

即共划分为 log_2^n段

对于其中任意一段 $k*(\frac{1}{k})$的值都为1，

所以$1*(\frac{1}{1})+2*(\frac{1}{2})+4*(\frac{1}{4})+8*(\frac{1}{8})+\ldots+log_2^n*(\frac{1}{log_2^n})=1+log_2^n$

所以：$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\ldots+\frac{1}{n}\leq 1+log_2^n$

T3:NKOJ8156 关键点脱环

何老板给你一个n个节点m条边的无向图，无向图中可能存在“环”(回路)。

编号1到k的这k个点，被称为关键点。何老板要你删掉一些边，使得所有k个关键点都不在环上。

问，最少需要删多少条边?

$1 \le n \le 1,000,000$

$1 \le m \le 2,000,000$

$1 \le k \le n$。

T3:NKOJ8156 关键点脱环

第1步：处理编号>k的点

如果一条边的两个端点都>k，这个边一定不删。 我们先把这些边都连上，用并查集维护图的连通性。易知如果删除这些边则一定不是最优解，所以这些边不计入答案中。

```
for(i = 1 ; i <= m ; i ++ )
{
    scanf("%d%d" , &x[i] , &y[i]);
    if(x[i] > k && y[i] > k) fa[getFa(x[i])] = getFa(y[i]);
}
```


第2步：处理编号<=k的点

如果一条边存在编号<=k的端点，且这两个端点都在同一并查集里面，说明连上这条边一定成环。我们把这条边删掉。

```
for(i = 1 ; i <= m ; i ++ )
    if(x[i] <= k || y[i] <= k)
    {
        if(getFa(x[i]) == getFa(y[i])) ans ++ ;
        else fa[fa[x[i]]] = fa[y[i]];
    }
```

T4. cointree

- 题目大意：

  给出一棵树和一个空集合，有三种操作：

  1.将一个节点加入集合；

  2.将一个集合内的节点移出集合；

  3.询问树上的某个点x与集合内的点深度的最大的LCA

- 分析：

  对于集合内的点，我们需要快速加入、快速删除、快速查找。

  使用一个set或者map来作为集合，新增、删除、查询的时间复杂度都是O(logn)的。

T4. cointree

- 解法1：利用第一种欧拉序：

对于点x，找到集合里第一个欧拉序小于等于x的点y（包括x本身），通过rmq求解LCA(x, y);

找到集合里第一个欧拉序大于等于x的点z（包括x本身），通过rmq求解LCA(x, z)，取二者深度的更大值。

需要注意边界问题。时间复杂度O(mlog(2*n-1))

T4. cointree

- 解法2：利用dfs序：

倍增处理出fa[i][j]数组，对于点x，倍增法枚举祖先y。

如果y是集合内某个元素的祖先，则x上跳，否则不跳，最后输出fa[x][0]。

该算法实际与原版倍增法求LCA并无差异，唯一有变化的是，将集合看做了一个整体：对于点u，如果集合内存在一个点v，使得u是v的祖先，则将点u视作集合的祖先。

需要先判断x自己是不是集合的祖先，时间复杂度O(m(logn)^2)

T4. cointree

- 解法3：解法2的拓展：

  求解深度最大的LCA等价于求解所有LCA里区间最小的一个。

  非祖孙关系的两个节点dfs序区间是不相交的，为了使求出来的这个LCA在包含x的基础上尽可能小，可以先找到集合内的点u、v，使in[u]满足in[u]<=in[x]，且in[u]是能取到的最大值；in[v]>=in[x]，且in[v]是能够取到的最小值。

  固定了点之后，只需要求LCA即可。

  时间复杂度O(mlogn)

重庆南开信竞基础课程