1001

签到

1002

因为数组中的值唯一，且在1到n的范围内，而询问的r和k也在1到n的范围内。 所以对于任意一个被操作1修改过的值都不会成为询问的答案，而询问的结果也必然在k到n + 1的范围内。 因为没有被修改过值是唯一的，所以可以建立权值线段树，维护权值区间内的值所在下标的最大值。而询问则转化为不小于k的值里面，下标超过r的最小权值是多少。 如何处理询问呢，一种较为暴力的解法是直接在线段树上询问权值在k到n+1的范围内第一个下标超过r的权值是多少。但复杂度可能会被卡，需要减枝。 再加上一个额外的判断就可以了，就是在递归查询完左子树内存不存在大于r的下标之后，如果不存在，则先看一下右子树内的下标的最大值是否大于r。如果不大于r，则不必再进入右子树内查询，否则答案一定在右子树内。在进左子树之前也利用同样的判断条件来判断是否有必要进入左子树，这样做可以保证单次查询的复杂度是O(log n) 的。 而对于操作1，则等价于修改某个权值的下标为无穷大。操作复杂度也是O（log n)的。 综上所述，得到了一个复杂度为O(m * log n)的在线算法，可以较快地通过此题。

1003

涉及到子串之间的关系的大量查询不难想到后缀自动机/后缀树，这里给出一个后缀自动机的做法。

对串S建后缀自动机，扒出parent树。

考虑将代表第i个前缀$S_1S_2\dots S_i$的点权值设为i，对于后缀自动机上某个点代表的所有字符串，这些串在原串中的第k次出现位置（最后一个字符的下标）即为其在parent树的子树上第k大的权值。

在dfs序上建持久化线段树即将其转化为一个经典问题：静态区间第k大。

在parent树上建倍增数组即可在$O(log(n))$内找到自动机上代表某子串的点，以其为根进行子树询问即可。

1004

先把每条边以(u,v,w)形式放进堆，堆按路径权值从小到大排序，然后每次取出堆顶，用v的出边扩展新的路径。但是一个点的出度可能会非常大（如菊花图），可以发现，将出边排序之后，每次只需要扩展当前点最小的出边，和扩展到当前点的边的下一条边即可。堆中需要记录当前结点，当前距离，上一节点距离，扩展到当前节点时下一条应该扩展的边。（注意，如果一次性扩展当前点连出去的所有权值相同的边，是会TLE的，实际上也是没有必要的。）

复杂度$O(k*\log(m+k))$

1005

solution:

a>b且gcd(a,b)=1,有$gcd(a^n-b^n,a^m-b^m)=a^{gcd(n,m)}-b^{gcd(n,m)}$

证明:

假设$n \geq m, r = n\%m$

有$a^n - b^n = (a^m - b^m)(a^{n-m} + a^{n-2m}b^m + \cdots + a^r b^{n-m-r}) + a^r b^{n-r} - b^n$

$gcd(a^n - b^n, a^m - b^m) = gcd(a^r b^{n-r} - b^n, a^m - b^m) = gcd(b^{n-r}(a^r - b^r), a^m - b^m)$

设$b^{n-r} = b^{m\lfloor n/m \rfloor} = b^{km}$

考虑$gcd(b^{km}, a^m - b^m)$

有$b^{km} = (a^m - b^m)(-b^{(k-1)m} - a^m b^{(k-2)m} - \cdots - a^{(k-1)m}) + a^{km}$

$gcd(b^{km}, a^m - b^m) = gcd(a^{km}, a^m - b^m) = d$

所以$d|b^{km}, d|a^{km}$，$d|gcd(b^{km}, a^{km}) = 1$

即$gcd(b^{n-r}, a^m - b^m) = 1$

所以$gcd(a^n - b^n, a^m - b^m) = gcd(a^{n\%m} - b^{n\%m}, a^m - b^m) = a^{gcd(n,m)} - b^{gcd(n,m)}$

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{i} gcd(i^a - j^a, i^b - j^b)[gcd(i,j) = 1] &= \sum_{i=1}^{n}\sum_{j=1}^{i}(i-j)[gcd(i,j) = 1] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{i} i[gcd(i,j) = 1] - \sum_{i=1}^{n}\sum_{j=1}^{i} j[gcd(i,j) = 1] \\ &= \sum_{i=1}^{n} i\varphi(i) - \sum_{i=1}^{n}(\frac{i\varphi(i)}{2} + [i == 1]) \\ &= \frac{\sum_{i=1}^{n} i\varphi(i) - 1}{2}\end{aligned}$$

设$f(n) = n\varphi(n)$，$f(n)$与$id(n)$作狄利克雷卷积杜教筛即可

1006

签到

1007

签到

1008

每条鱼都需要被抓上来并且煮足够的时间，而我们能在炖鱼的时候抓鱼，所以至少需要抓一条鱼的时间和炖所有鱼的时间，也就是$k + \sum_{i=1}^{n} t_i$。要能在这个时间内就完成，则除了抓第一条鱼以外抓鱼的时候锅里都必须在炖鱼（假设锅里的鱼一旦炖好就自动取出来），显然如果t_i都太小的时候这是不能做到的，会有一些抓鱼的时间锅里没有在炖鱼，我们称锅里没有炖鱼的抓鱼时间为被浪费的时间，所以我们的优化目标就是使被浪费的总时间T最小，答案即为$k + \sum_{i=1}^{n} t_i + T$。

对于第i条鱼，炖它的时候我们可以不浪费时间抓到t_i/k条鱼，或者浪费$k - t_i\%k$的时间抓到$t_i/k + 1$条鱼。所以如果$\sum_{i=1}^{n} t_i/k \geq n - 1$，则可以不浪费时间完成任务；如果$\sum_{i=1}^{n} t_i/k < n - 1$，则差$m$条鱼就选炖$t_i\%k$前$m$大的鱼的时候浪费时间多抓一条鱼。

1009

$dp[i][j][k][l]$表示从1号点开始bfs到第i步, 用了白点j个, 黑点k个, 且第i层个数正好为l的期望。 通过枚举新层的节点个数进行转移。

复杂度$O(n^5)$。

1010

我们设$res=\sum_{i=l}^{r}\sum_{j=l}^{r}\varphi(gcd(a_i,a_j))lcm(a_i,a_j)$，显然这么做不是很好化简，我们考虑直接枚举$i,j$，然后看数组$a$中有多少个值满足$a_k=i$以及有多少个值满足$a_k=j$，那么我们可以看出对于枚举的这对$i,j$，贡献就是乘上一个系数$c_i*c_j$，$c_i$是指$i$这个值在$a$数组的$[l,r]$区间出现的个数。

那么$res=\sum_{i=1}^{n}\sum_{j=1}^{n}\varphi(gcd(i,j))lcm(i,j)c_ic_j$，$n$是$a$数组中出现过的最大的值。然后枚举$gcd(i,j)$的值得到：

$$res=\sum_{d=1}^{n}\varphi(d)d\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{i}{d}\frac{j}{d}c_ic_j[gcd(i,j)=d]$$

$$res=\sum_{d=1}^{n}\varphi(d)d\sum_{i=1}^{\frac{n}{d}}\sum_{j=1}^{\frac{n}{d}}ijc_{id}c_{jd}[gcd(i,j)=1]$$

莫比乌斯反演一下得到：

$$res=\sum_{d=1}^{n}\varphi(d)d\sum_{i=1}^{\frac{n}{d}}\sum_{j=1}^{\frac{n}{d}}ijc_{id}c_{jd}\sum_{k|gcd(i,j)}\mu(k)$$

我们枚举k得到：

$$res=\sum_{d=1}^{n}\varphi(d)d\sum_{k=1}^{\frac{n}{d}}\mu(k)\sum_{i=1}^{\frac{n}{d}}\sum_{j=1}^{\frac{n}{d}}ijc_{id}c_{jd}[k|i][k|j]$$

整除一下范围得到：

$$res=\sum_{d=1}^{n}\varphi(d)d\sum_{k=1}^{\frac{n}{d}}\mu(k)k^2\sum_{i=1}^{\frac{n}{kd}}\sum_{j=1}^{\frac{n}{kd}}ijc_{ikd}c_{jkd}$$

令$T=kd$，我们可以变化res为：

$$res=\sum_{d=1}^{n}\varphi(d)d\sum_{k=1}^{\frac{n}{d}}\mu(k)k^2\sum_{i=1}^{\frac{n}{T}}\sum_{j=1}^{\frac{n}{T}}ijc_{iT}c_{jT}$$

我们改为枚举T可以得到：

$$res=\sum_{T=1}^{n}(\sum_{i=1}^{\frac{n}{T}}ic_{iT})^2\sum_{k|T}\mu(k)k^2\varphi(\frac{T}{k})\frac{T}{k}$$

设$f(T)=\sum_{k|T}\mu(k)k^2\varphi(\frac{T}{k})\frac{T}{k}$，通过对积性函数的分析可以得到$f(T)=\mu(T)T$，也就是说$f(T)$为$T$乘上莫比乌斯函数，因此只有当$T$不含平方因子的时候，才会产生贡献。

对于这个式子，在保证大数据纯随机的情况下，有一个莫队的做法(说不定有其他十分精彩的做法?)。我们发现当我们转移时，只需要枚举a_i的因子T,然后就能对$(\sum_{i=1}^{\frac{n}{T}}ic_{iT})^2$的和进行更新,更新之后由于要乘上一个系数,而当$T$为平方因子时，$f(T)=0$，因此只需要枚举无平方因子，因此对于每个$a_i$我们只需要先处理出质因子个数,然后枚举质因子的子集得到每个无平方因子。由于保证了随机，可以得到质因子的期望值为3，而能产生贡献的因子的期望值为11。最终整体复杂度为$O(11*n\sqrt{n})$。

1011

题目开始给定一个三维空间，空间中每个格点的值为1，之后将第n层的其中m个点从1修改成v_i。之后每一秒，$a_{i,j,k}$将会变成$a^{t1}_{i+1,j+p,k} \times a^{t2}_{i+1,j,k+q} \times a_{i+1,j,k} \times a_{i,j,k}$。最后询问$n$秒后的$a_{0,0,0}$的值。

W先称$a_{i,j,k}$为在第i层的点 观察式子可以发现，最后乘的$a_{i,j,k}$对答案并没有影响，因为$a_{i,j,k}$在每一秒只由第i层的值和第$i+1$层的值决定，所以，在第0秒到第$n-1$秒的时候，$a_{0,0,0}$的值只会被第0层到第n-1层的值影响。而这些层的值都为1，最后乘起来也是1。直到第n秒，第n层的值对$a_{0,0,0}$产生影响。那么我们可以把式子改写成 $a^{t1}_{i+1,j+p,k} \times a^{t2}_{i+1,j,k+q} \times a_{i+1,j,k}$ 因为在最后的结果中，$a_{0,0,0}$一定是由v_i的次方的乘积组成的，所以考虑每个v_i在$a_{0,0,0}$中贡献了几个，即v_i被乘了几次 对于每个v_i，首先需要$p|x_i$同时$q|y_i$，如果$t1=1$且$t2=1$那么v_i的贡献可以看做从(x_i,y_i)，向左跳x_i/p次长度为p的步子，向上跳y_i/q步长度为q的步子，其他步不动，一共n步，最终到达(0,0)的方案数。当$t1,t2$不为1的时候，可以发现对于任意一种方案，向上跳和向左跳的步数相同。$t1$和$t2$的贡献不变，所以对于v_i来说，贡献为

$$t1^{\frac{x_i}{p}} t2^{\frac{y_i}{q}} \binom{n}{\frac{x_i}{p}} \binom{n-\frac{x_i}{p}}{\frac{y_i}{q}}$$

$$v_i$$

由于模数为998244353，所以对于每个v_i，我们只需要求解其上式中的指数模998244352的结果(费马小定理) 对于$t1^{\frac{x_i}{p}} t2^{\frac{y_i}{q}}$，可以使用快速幂计算结果 对于$\binom{n}{\frac{x_i}{p}} \binom{n-\frac{x_i}{p}}{\frac{y_i}{q}}$，我们需要将998244352拆分成$7 \times 17 \times 2^{23}$，模7和17的时候，使用卢卡斯定理解决，模$2^{23}$的时候，使用扩展卢卡斯解决，最后使用中国剩余定理合并。 求出每个v_i的贡献后，乘积起来便是最后的答案 最终的时间复杂度为$O(mlogn)$ tips：在对2^{23}取模的时候，可以使用&运算加快速度。