X



动态规划选讲

mjy0724 (IIIS, @THU)

×



- 给出一棵n个节点的树,每个节点上有点权a_i。
- 求最长的树上路径,满足条件:

路径上经过节点(包括两个端点)点权的gcd和不等于1。

数据范围

$$\circ$$
 n <= 2 * 10^5







- 不互素就ok,不用考虑具体的gcd值,还不用考虑重复计算。
- 对于每个节点u,维护dp_{u,v}表示u往下挂出的点权都能被v整除的最长链。
- v只取整除a_u的质数,个数很少,很容易转移。
- 边转移边更新全局答案。

```
for (int i = lnk[u]; i; i = nxt[i]) if (ter[i]!= las) {
   int v = ter[i];
   dfs(ter[i], u);
   // dp & 更新答案部分
   rep(j, 0, (int) p[u].size() - 1)
       rep(k, 0, (int) p[v].size() - 1) {
       if (p[u][j]!= p[v][k]) continue;
       ans = max(ans, dp[u][j] + dp[v][k]);
       dp[u][j] = max(dp[u][j], dp[v][k] + 1);
}
```









○ 给出一棵n个节点的树,对于1~n间的每一个数k,你需要求出:

最多能选出多少条互不相交的路径,每条路径的长度都为k。

数据范围









- 考虑对于一个给定的k,使用dp计算答案。
- 贪心:在一个子树当中,尽可能最大化完整的路径条数;其次最大化未完成的链的长度。
- 进行合并时,先尝试儿子的dp值中第二维最大值与次大值能否拼成一条完整的链(长度和 >=k),若不行再选取一条最长的链向上延伸。时间复杂度O(n)。
- 令f_k表示单次对于给定的k的答案。
- 显然有: f_k <= n / k</p>
 - ightarrow f中不同的取值个数是 $\mathcal{O}(\sqrt{n})$ 级别的!
- 又, f 显然单调。
- 在每个边界上二分即可。
- 。 时间复杂度 $\mathcal{O}(n\sqrt{n}\log n)$







题目描述

- 给出一棵n个节点的树T。
- 对于其中任意一个非空节点集合X,定义f(X)为包含这些点的最小连通子树的边数。
- 给出一个正整数k, 求:

$$\sum_{X\subseteq\{1,2,\ldots,n\},\ X\neq\emptyset}(f(X))^k.$$

○ 答案对<u>10^9+7</u>取模。

数据范围

- 2 <= n <= 10⁵.
- 1 <= k <= 200.









○ f(X)^k很难直接算,于是我们把它拆开。

$$x^{k} = \sum_{i=0}^{k} {x \choose i} \times i! \times S(k,i)$$

- 其中S(k,i)是第二类斯特林数,表示将k个元素拆分成i个集合的方案数。
- 大家都会算它吧?

```
1  S[0][0] = 1;
2  for (int i = 1; i <= k; ++ i)
3    for (int j = 1; j <= i; ++ j) {
4       S[i][j] = (S[i - 1][j - 1] + (ll) S[i - 1][j] * j % mo) % mo;
5  }</pre>
```

接下来我们尝试用组合意义把式子套回原来的题面里头去。

$$ans = \sum_{X} \sum_{i=0}^{k} {f(X) \choose i} \times i! \times S(k,i)$$
.

$$= \sum_{i=0}^{k} i! \times S(k,i) \sum_{X} {f(X) \choose i}$$











$$ans = \sum_{X} \sum_{i=0}^{k} {f(X) \choose i} \times i! \times S(k, i)$$

$$= \sum_{i=0}^{k} i! \times S(k,i) \sum_{X} {f(X) \choose i}$$



所有非空点集对应的生成树标记了i条边的方案数之和

- 这个东西看上去很树上背包,事实上树上背包也就行了。
- 令dp_{i,j}表示以点i为根的所有生成树当中标记了j条边的方案数,由于点集大小之类的不影响答案, 所以不同的点集对应的生成树可以一并考虑。•
- 对于每一棵生成树,我们都在它最高的节点处累计进答案。
- 大家都会算吗?









```
sz[u] = 1;
     dp[u][0] = 2;
    for (int i = lnk[u] ; i ; i = nxt[i]) if (ter[i] != las) {
         int v = ter[i] ;
        dfs(v, u);
        memset(f, 0, sizeof(f));
         rep(p, 0, min(k, sz[u])) rep(q, 0, min(k - p, sz[v])) {
            upd(f[p + q], (ll) dp[u][p] * dp[v][q] % mo);
10
         rep(j, 0, k) {
11
            upd(g[j], - dp[v][j]);
12
13
        memcpy(dp[u], f, sizeof(f));
        sz[u] += sz[v];
14
15
16
     rep(i, 0, k) {
17
         upd(g[i], dp[u][i]);
18
19
     for (int i = k ; i > 0 ; -- i) {
20
        upd(dp[u][i], dp[u][i-1]);
21
    upd(dp[u][1], -1);
```

○ 时间复杂度O(nk)







- 时间复杂度?
- 分三种情况考虑:
- sz[u] >= k & sz[v] >= k , 单次合并复杂度O(k^2)

此时每次相当于合并了两堆大小为至少为k的石子 至多有n/k堆,因此合并次数为O(n/k),总复杂度O(nk)

○ sz[u] >= k & sz[v] < k , 单次合并复杂度O(k*sz[v])

这种情况下sz[v]之和不超过n,因此总复杂度O(nk)。u和v交换的情况也是如此。

- \circ sz[u] < k & sz[v] < k
 - * 相当于是正常的树上背包,树的大小为O(k),众所周知它的复杂度是O(k^2) (考虑每两个点之间的贡献只会产生于它们的lca处 最多有n/k棵这样的树,因此总复杂度O(nk)









- 如果两棵树可以通过重标号后变为完全相同,那么它们就是同构的。
- 将中间节点定义为度数大于1的节点。
- 计算有n个节点,其中所有的中间节点度数都为d的互不同构的树的数量。
- 答案对大质数取模。

数据范围

- 1 <= n <= 1000.
- 2 <= d <= 10.
- 10^8 <= mod <= 10^9.









Uniformly Branched Trees

X

- 很重要的问题在于如何处理同构的树。
- 首先需要把无根树转为有根树,不妨选取树的重心作为它的根。
 - 性质:任一子树大小不超过n/2。
 - 双重心需要特殊考虑。
- 在确定了根的情况下如何计算满足条件的方案数呢?
- 令dp_{i,j,k}表示节点数为i,共有j棵子树,每棵子树的大小都不超过k的有根树数量。
- 所有子树的大小都小于k。

$$dp_{i,j,k} \longleftarrow dp_{i,j,k-1}$$

- 。 有t棵子树的大小等于k。
 - 不区分子树之间的相对顺序。

$$dp_{i,j,k} \longleftarrow dp_{i-t \times k,j-t,k-1} \times {\binom{dp_{k,d-1,k-1}+t-1}{t}}$$

○ 其中C_{X+t-1,t}表示在X种方案中不分顺序地选取t种,可重复的方案数。可以用组合意义理解。







$$dp_{i,j,k} \longleftarrow dp_{i,j,k-1}$$

$$dp_{i,j,k} \longleftarrow dp_{i-t \times k,j-t,k-1} \times {\binom{dp_{k,d-1,k-1}+t-1}{t}}$$

○ 统计答案:

- \circ 单重心的情况:答案即 $dp_{n,d,\lfloor rac{n}{2}
 floor}$
- 双重心的情况: n为偶数,且两个重心通过一条边相连,各挂着一个大小为n/2的子树。 双重心的情形在上一种情况中恰好被计算了两遍,因此减去一遍即可。

$$\begin{pmatrix} dp_{\frac{n}{2},d-1,\lfloor\frac{n}{2}\rfloor-1}+1 \\ 2 \end{pmatrix}$$

○ 总时间复杂度为O(n^2d^2).







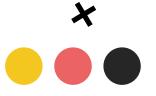


- 有个长度为n的序列a,你需要统计a中有多少个棒棒的子序列。
- 一个序列b被定义为<u>棒棒的</u>,当且仅当: 对于序列中每一个位置i,b_i都能够被i整除。
- 答案对*10^9+7*取模。

数据范围

- o n <= 10⁵
- 1 <= a_i <= 10^6









- 暴力一点嘛
- 考虑到子序列中的元素相互之间是没有影响的,令dp_{i,j}表示使用了a序列的前i个 元素,造了长度恰好为j的子序列的方案数。转移非常显然:

$$dp_{i,j} = dp_{i-1,j} + dp_{i-1,j-1} \times (a_i\%j == 0)$$

- 维护dp_i,dp_{i+1}只需要在dp_{i}的基础上改几个能被a_{i+1}整除的位置。
- 最终的答案就是

$$ans = \sum_{i=1}^{n} dp_{n,i}$$

○ 时间复杂度看实现了,最暴力的根号找因子是O(n\sqrt{A}), 3s 1e8也很轻松。









有个神仙写了个序列求max,它长下面这样:

```
int fast_max(int n, int a[]) {
    int ans = 0;
    int offset = 0;
    for (int i = 0; i < n; ++i)
        if (ans < a[i]) {
            ans = a[i];
            offset = 0;
        } else {
            offset = offset + 1:
            if (offset == k)
                return ans;
    return ans;
```

○ 现在他比较关心的是出错的情况,请你算出有多少个1~n的排列在这个函数的计算下答案不为n。

○ 最终结果对10^9+7取模。

o n,k <= 10^6





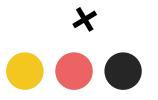




- 考虑暴算,发现基本上枚举个什么东西可行的情况都包含一个前提:在此之前函数并没有退出。
- 那我们不妨来dp这个东西!
- - 令f_i表示1~i的排列当中有多少个是运行完整个循环之后还没有退出的。
 - 怎么算呢?
 - 最大值i必然出现,并且只可能位于[i-k+1,i]
 - 考虑枚举最大值出现的位置j

$$f_{j-1} \times {i-1 \choose i-j} \times (i-j)!$$

= $f_{j-1} \times (i-1)! \times \frac{1}{(j-1)!}$







○ 朋友们,我们再来整理一下!

$$f_i = \sum_{j=i-k+1}^{i} \frac{f_{j-1}}{(j-1)!} \times (i-1)!$$

$$= (i-1)! \sum_{j=i-k}^{i-1} \frac{f_j}{j!}$$

- 边计算f_i边维护一个前缀和就口以啦!
- 最后的答案呢?
- 用相似的方法枚举n所在的位置计算出最终答案为n的排列数,再取个补集就好啦!

$$ans = n! - \sum_{i=1}^{n} f_{i-1} \times {n-1 \choose n-i} \times (n-i)!$$

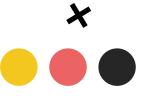
○ 时间复杂度O(n)





- 给出一个长度为n的字符串,如果这个字符串中含有子序列"hard",那么这个字符串就很hard。你不希望这个字符串很hard,所以想要从中删掉几个字符使他不hard。
- o (例: hard, hzazrzd, haaaaard 都很hard, 而har, hart, drah不hard。
- 删掉在<u>原字符串中</u>的第i个字符需要花费a_i的代价,求最小代价。

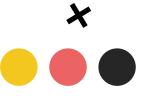
数据范围





- 给出一个长度为n的字符串,如果这个字符串中含有子序列"hard",那么这个字符串就很hard。你不希望这个字符串很hard,所以想要从中删掉几个字符使他不hard。
- o (例: hard, hzazrzd, haaaaard 都很hard, 而har, hart, drah不hard。
- 删掉在<u>原字符串中</u>的第i个字符需要花费a_i的代价,求最小代价。

数据范围







- 有一个n个点的图,有一些点的颜色给定,另一些可以随意确定。另外,还可以在图上 连一些从编号较小的节点向编号较大节点的边。
- 对于一个确定的图,统计图中节点颜色黑白交错的路径条数,如果奇偶性=p,那么该图就被定义为好图。
- 求好图的个数,答案对*10^9+7*取模。

数据范围

○ 1 <= n <= 50, p \in {0,1}</p>







Kuro and Topological Parity

- 先考虑对于一个给定的图如何判断其是否是好图:
 - 按照拓扑序(在本题中即为顺序)dp一下即可。
- 就计数而言,点之间只有颜色、以其为终点的路径的奇偶性两个属性有用。

令dp_{i}{ow}{ob}{ew}{eb}分别表示dp到第i个点,共有奇白、奇黑、偶白、偶黑的点各ow/ob/ew/eb个时的方案数。转移可以做到O(1)。总复杂度O(n^5)。

o ow+ob+ew+eb=i

有一维状态是多余的, 总复杂度O(n^4)。

以其为终点的路径数为偶数的点可以往后随便连。

由于我们只关注奇偶性,同颜色相同的点一样,往后连边时不会影响答案,因此不需要额外记录。 κ 即计算dp_{i}{ow}{ob}即可。总时间复杂度O(n^3)。









- 有一个n个点的图,有一些点的颜色给定,另一些可以随意确定。另外,还可以在图上 连一些从编号较小的节点向编号较大节点的边。
- 对于一个确定的图,统计图中节点颜色黑白交错的路径条数,如果奇偶性=p,那么该图就被定义为好图。
- 求好图的个数,答案对*10^9+7*取模。

数据范围

○ 1 <= n <= 50, p \in {0,1}</p>







Hero Meet Devil

题目描述

- 给出一个字符串S,这个字符串只由'A','C','G','T'四个字母组成。
- 对于每个1~|S|中的每一个i, 求出满足以下条件的字符串T的个数:
 - 长度为m。
 - 只由'A', 'C', 'G', 'T'四个字母组成。
 - LCS(S,T) = i.
- 答案对*10^9+7* 取模后输出。

数据范围

○ |S| <= 15, m <= 1000.





首先考虑LCS的计算过程。



$$\circ$$
 当a_i = b_j时, $f_{i,j}=f_{i-1,j-1}+1$

。 当a_i != b_j时,
$$f_{i,j} = \max(f_{i-1,j}, f_{i,j-1})$$

○ 当i固定时 , f_{i,j-1} <= f_{i,j} <= f_{i,j-1}+1</p>

- 因此可以将差分序列压起来,在本题中,令序列a=T,序列b=S,那么对于一个确定的T的前缀,<mark>当前的</mark> LCS状态可以用一个15位的二进制数来表示。
- 转移时枚举T新添的字母是什么,并使用上面传统的LCS计算方法来求出新的状态。
- 预处理出trans_{st}{ch}表示原来的状态为st,新添了字母ch后的状态。可以在O(|S|2^|S|)内求出。

$$\circ$$
 于是
$$dp_{0,0} = \mathbf{0}$$

$$dp_{i,trans_{st,ch}} + = dp_{i-1,st},$$

$$\forall ch \in \{ \text{'A', 'C', 'G', 'T'} \}$$

○ 时间复杂度O(m2^|S|).









○ 求【L,R】中,各位数字组成的序列的LIS恰好为k的数字个数。

数据范围

- T <= 10⁴.
- 0 < L <= R < 2^63 1.
- *x*1 <= k <= 10.









- 数位dp。
- 。 考虑二分求LIS的dp做法。
 - 其中f_i表示长度为i的最长上升子序列末尾元素最小可以是多少。
 - f_i \in [0,9], 且严格上升子序列长度至多为10, O(10^10)?
 - \circ f_i 单调递增,对差分序列做组合计数, $\mathcal{O}(\sum_{s=1}^{10}\sum_{c=1}^{10}\binom{s+c-1}{c-1})$
 - 仔细思考一下,发现f_i实际上是严格单增的。
 - 因此只需要记录0~9中每个数字在f_i中是否出现过即可。状态数是O(2^10)的。
 - 状态st后新添一个数x获得的新状态可以预处理得到。

```
int calc(int st, int x) {
    int k = -1;
    for (int i = 0; i < x; ++ i)
        if ((st >> i) & 1) k = i;
    if (k != -1) return st ^ (1 << k) | (1 << x);
    return st | (1 << x);
}</pre>
```

○ 外部就是正常的数位dp,相信大家都会。









- 有一个长度为n的序列a,一个a中的子集被定义为好的当且仅当它其中元素的gcd为1。
- 。 求最小的好的子集的大小。
- 或判断不存在这样的子集。

数据范围

- *x* 1 <= n <= 3*10^5.
- 1 <= a_i <= 3*10^5.









- 看上去很不像dp吧......
- 思考答案集合具有什么样的性质。
- 如果有解,那么最优解的大小不超过7。
 - 每新加入一个元素必从gcd中去除一个质因子,否则不起作用可删去。
- 于是可以枚举答案,然后验证是否存在大小为len,且gcd为1的集合。
- 令dp_{i}表示大小为len且gcd为i的子集个数。

$$dp_i = {Cnt_i \choose len} - \sum_{i|j,i\neq j} dp_j$$

- 其中Cnt_i表示序列a中被i整除的元素个数,很容易预处理得到。
- 当dp_1不为0时,此时的len就合法。
- 由予dp值可能很大,需要在模大质数意义下进行计算,如果不放心的话可以多模几个。
- 时间复杂度O(nlogn).



