杂 杂 杂 题 选 讲

mjy0724 (IIIS, @THU)

杂杂杂一

- 给出一个长度为N个序列，序列中的元素取值为1～8. 求一个最长的子序列，满足：
  - 每种取值出现在子序列中的位置连续。
  - 每种取值出现的次数相差不超过1。

- N <= 1000.

杂杂杂一

- Dp当中处理“相差不超过一”的出现次数显然不太现实。
- 考虑枚举这个出现次数L。（部分出现L，部分出现L+1）
- “连续”-> 同种取值放在一起考虑
- 对后面有影响的信息有：目前取到了序列中的什么位置、处理好的颜色集合，在固定这两个信息的情况下，最大化序列的长度。
- 令dp[i][S]表示处理到序列中的第i位，已经处理好的颜色集合为S的最长序列长度。
- 枚举下一个出现的颜色c与其长度l(L|L+1)，一定是贪心地取该颜色最靠前的l次。
- 一种颜色从某一位起连续取k个最后取到的位置可以O(CN^2)预处理出来。
- 时间复杂度O(N^2 2^C + CN^2)。难以通过。

杂杂杂一

- **优化一：**将dp转移时每次枚举下一个处理什么颜色去掉，将当前状态转移到下一位的状态，这样就可以直接选用当前位置的颜色了。 --> O(N^2 2^C / C + CN^2)
- **优化二：**注意到当前复杂度瓶颈上有个N是作为dp外的枚举被乘上去的。但不难发现，不论有多少个取L+1，多少个取L，此时所有合法序列的长度都是大于等于L取更小值的情况的。所以我们需要找到最大的存在合法答案的L，再在里面最大化L+1的颜色个数。
- 而L的取值的合法性显然是**单调**的。（将一个合法的取值为L的答案每种颜色删去一个就得到了一个取值为L-1的合法答案），于是这部分可以改成一个二分。
  - 时间复杂度 -> O(2^C NlogN + CN^2)
- **优化三：**受优化一中的启发，预处理的部分不需要枚举颜色了，颜色被确定为了始发位置的颜色。
  - 时间复杂度 -> O(2^C NlogN + N^2)

杂杂杂二

- 有一棵N个点的树，1号点为根节点。现在你知道M条信息：
  - 某些树上的边
  - 某两个点的LCA
- 现在你需要数满足条件的树形态数。
  - N <= 13, M <= 100.

杂杂杂二

- Dp状态很好设计：dp[i][S]表示以i为根子树中的点集为S的合法方案数。（先不考虑M条限制）
- 枚举i的儿子u，和其状态S’。(对S’求和)

$$dp(i,S)=\sum_{u} dp(u,S')\times dp(i,S\oplus S')$$

- 这样做有什么问题？
- 会算重复。树的儿子之间交换顺序是不改变树的形态的！
- **解决方案：**还记得01背包吗？
- 枚举顺序改成i,u,S’,S即可。这样一来子树必然按照S’的大小依次被考虑进来。

杂杂杂二

- **边的约束：**
  - 显然可以只在端点处处理边。
  - 当我们遇到边(i,v)时，有两种情况：v在子树中、v是父亲。在dp(i)时显然不能处理后一种情况，但可以处理前一种。
  - 于是我们在dp(i)时枚举到每个儿子u时都需要处理u的边。如果另一个端点在u的子树当中，那么一定在dp(u)时被处理好了；否则另一个端点只能是i，否则需要扔掉这个状态。
  - 在处理(i,v)时，如果v在u的子树当中，那么只能是u。否则需要扔掉这个状态。

杂杂杂二

- **LCA的约束：**
  - 显然可以只在LCA处处理约束。
  - 假设约束长这个样子：LCA(x,y)=i
  - 在做完所有的dp(i)之后，将所有S中不包含x或是不包含y的状态全部都扔掉。
  - 那么接下来只需要处理x、y在同一棵子树中的情况。
  - 非常简单，如果x、y都在S’当中，就把dp[u][S’]这个状态扔掉。
- 时间复杂度 O(NM3^N).

杂杂杂三

- 给出平面上的N个点。
  - 你可以删去**至多**k个点。
  - 你想要用**尽可能小**的平行于坐标轴的矩形去覆盖剩下的点。
- N <= 1e5, k <= 10.

杂杂杂三

- 直觉地，我们只会删边界上的点。

- **结论1：** 如果一个点不在任何一条边界的最靠近的k个点当中，那么必然不需要考虑删它。
  - *证明*：删它会产生效用必然在某一条边界上删光了所有和它相等或是更靠近边界的点。而这个数量>k。

- **结论2：** 在某一条边界上我们只需要考虑删一个离它最近的前缀。
  - *证明*：跳着删没必要…我们只关心最靠近的没被删的点。

- 于是我们可以在每个边界上O(k^4)枚举删去的前缀，判断一下点数是否在k的范围内，如果在的话就用面积更新一下答案。

杂杂杂四

- 一开始序列为空。每次你有Pa的概率往序列末尾添加一个a，有Pb的概率往序列末尾添加一个b。当子序列ab的出现次数>=K时停止。
- 问**子序列**ab的期望出现次数。

- K <= 1000.

杂杂杂四

- 想一想对以后有影响的状态，其实跟b没什么关系，只跟目前有多少个a，和有多少个子序列ab有关。

$$dp_{i,j} \cdot P_a \Rightarrow dp_{i+1,j}$$

$$dp_{i,j} \cdot P_b \Rightarrow dp_{i,i+j}$$

- 代码能算的部分(能直接用上面的式子)：i <= K， i+j <= K.

- **问题一**：i+j>k的情况怎么算？
- **问题二**：转移到自身的情况怎么避免？

杂杂杂四

- **问题一**：i+j>k的情况怎么算？
- 换句话来说，就是一旦出现了一个b就立刻停止。
- 原答案为j。在最终出现b之前每出现一个a就给答案额外增加一。我们用X表示产生的额外答案。

杂杂杂四

$$X = \sum_{i=0}^{\infty} i \cdot P_a^i * P_b = P_b \cdot \sum_{i=1}^{\infty} i \cdot P_a^i$$

$$\frac{X}{P_a} = P_b \cdot \sum_{i=1}^{\infty} i \cdot P_a^{i-1} = P_b \cdot \sum_{i=0}^{\infty} (i+1) \cdot P_a^i$$

$$\frac{X}{P_a} - X = P_b \cdot \sum_{i=0}^{\infty} P_a^i = P_b \cdot \frac{1}{1-P_a} = 1$$

$$X = \frac{P_a}{1-P_a} = \frac{P_a}{P_b}$$

杂杂杂四

- **问题二**：转移到自身的情况怎么避免？
- 换句话来说，一堆前缀=b的情况。

- 不难发现，它们完全没有意义…
- 我们完全可以直到它遇到了某个a才开始考虑，也正是这时它对子序列ab的期望出现次数才开始有贡献。

- 从最初的状态到遇到第一个1的状态的概率实际上等于1。
  于是我们只需要求dp(1，0)即可。

杂杂杂五

- 对于一个排列，定义它的权值为前缀GCD当中不同的数字个数。
- 计算有多少个权值最大的1～N的排列。
  - N <= 1e6.

杂杂杂五

- **权值的上界：**
  - 如果排列的第一个数 $X = p_1^{c_1} p_2^{c_2} \ldots p_k^{c_k}$
  - 上界为 $\sum_{i=1}^{k} c_i$
- **事实上，如果我们找到了c之和最大的数作为第一个数，那么总能构造出排列达到上界。**

- 如何确定1～N当中c之和最大的数？它们有什么特点？
- 只可能有质因子2、3！（为什么？）
- 除此之外，3的个数还不超过1个。（为什么？）

杂杂杂五

- **至此，最优情况下前缀gcd的所有可能可以被logN地表示出来。**

- 令dp[i][x][y]表示dp到了排列的第i位，目前的前缀gcd为N(x,y)的方案数。(N(x,y)=2^x·3^y)
- 接下来有可能会发生三种情况：
  - 1. **gcd不变**。此时选的这个数必然是N(x,y)的倍数，一共有n/N(x,y)种取值，**能选的个数要减去i-1**。
  - 2. **gcd中的2减少1**，此时选的这个数必然是N(x-1,y)的倍数，能选的个数为n/N(x-1,y)-n/N(x,y)。
  - 3. **gcd中的3减少1**，此时选的这个数必然是N(x,y-1)的倍数，能选的个数为n/N(x,y-1)-n/N(x,y)。
  - 最后dp[n][0][0]就是答案。
  - 时间复杂度O(nlogn).

杂杂杂六

- 给出一张N个点M条边的连通图，每条边有一个颜色。Q组询问，每次问有多少种颜色可以使得只保留该颜色的边点(x,y)连通。
  - N,M,Q <= 1e5.

杂杂杂六

- 对颜色边数进行分块。

- >= sqrt（n）
  - 此时这样的颜色数不会超过sqrt(n)个。
  - 对于每种颜色可以直接做一次并查集，然后扫一遍所有询问，往询问上累计贡献。

- < sqrt(n)
  - 此时平方和不会超过n*sqrt(n)。
  - 我们在做了并查集之后，枚举连通块里面的点对，往点对上计贡献。

  - 用map实现，总复杂度O(n*sqrt(n)logn).

杂杂杂七

- 给出一个N个点的树，每条树边都是红色或黑色的。一个长度为k的序列a是**小天才**的，当且仅当在a(1)～a(2),…,a(k-1)～a(k)的树上路径中至少有一条黑边。对这样的序列计数。

- N <= 1e5.

杂杂杂七

- 对答案取补集。
- **总方案树 - 只经过红边的序列数。**
- 只能在由红边连接的连通块中任意取点。
- Dfs一下，对连通块大小的k次幂求和即可。
- 你想写并查集小天才也不拦你。

杂杂杂八

- 给出一个R*C的网格，网格中的每一个元素都在1～1e9的范围内。
- 你需要求出元素相同的最大矩形面积。

- R，C <= 5000.

杂杂杂八

- 其实本问题几乎等价于求**最大全1子矩形**的经典问题。
  - 首先可以预处理出每个格子往上最多能延伸的长度(这边的延伸指的是颜色相同)
  - 然后枚举下边界，转成直方图问题。唯一不同的是直方图每一列带有颜色。
  - 拆成若干段连续的颜色相同的直方图进行处理。
- 即，最终的解法是**在列上根据颜色进行一定划分后的求直方图最大子矩形问题**。
  - 可以枚举卡到上边界(高度最小值)的位置。然后左边界最远可以推到它往左遇到的第一个比它小的数的后一位，右边界同理。
  - 我们可以用单调栈预处理出每个位置左、右方向上遇到的第一个比它小的数。
- 时间复杂度$O(R\times C)$。

杂杂杂九

- 给出一个 n 个节点的有根树，对于每个节点 u，我们定义 $f(u) = \sum_{v \in anc(u)} g(v, u)$，其中，$g(a, b)$ 表示 a 的子树当中除 b 以外深度不超过 b 的节点个数。
- 现在对于每个 i 你需要输出 $f(i)$。

- N <= 5e5

杂杂杂九

- 做法一
  - 考虑 bfs 整棵树，每次插入一层之后回答一层的询问。
  - 每个节点都能对它到根的一整条路径上产生贡献，每个节点的答案也是查询它到根的路径上的点权和。
  - 可以使用树链剖分来维护。
  - 时间复杂度 $\mathcal{O}(n\log^2 n)$。

杂杂杂九

- 做法二

  - **除去自身节点**这条限制我们可以暂不考虑，对于求出的答案最后减掉 $dep(i)$ 就可以了。
  - 考虑对于一个节点 a，答案等价于 $\sum_{dep(b)\leqslant dep(a)} dep(lca(a,\ b))$

  - 考虑除 a 外每个深度不超过 a 的节点对 $f(a)$ 的贡献，显然是从 lca 处一直到根节点路径上的节点作为子树的根时才会被统计到。

杂杂杂九

- 做法二
  - 因此$f(a)=f(fa(a))+\sum_{dep(a)=dep(b)} dep(lca(a,b))$，也就是说，
    - 对于每个节点我们现在只需要考虑它同层节点答案的计算。
  - 我们将同层的节点拿出(假设其是按照在树中的dfs序排好的，事实上我们也可以通过预处理得到)。注意，在 dfs 序有序的情况下，一个节点与它前面出现的所有节点的 lca 深度是单调递减的。于是这个问题就可以用单调栈来解决了。

杂杂杂九

- 做法二
  - 考虑在一层中从左往右枚举a，同时维护一个 lca 的单调栈，栈中相邻且相同的元素我们保留一个，并记录个数。
  - 对于新的点a，我们求它与队尾元素的 lca。如果队尾的 lca 深度大于我们求出的深度，就不断地进行合并。 最后再将a自身加入单调栈。
  - 总时间复杂度 $O(n)$。

杂杂杂九

- **做法三：**
  - 每次把一层的点拿出来建一棵虚树。
  - 两两之间lca的深度之和可以用**树形dp**来解决。
  - 比如令f(i)表示i子树外的所有节点与子树中节点间lca的深度之和。
  - 那么i往某个儿子v更新时，
    f(v)=f(i)+dep(i)*(sz(i)-sz(v))
  - 我们的关键点都在叶子。此时的f值就是我们想要统计的答案。
  - 瓶颈在建立虚树，但该做法比较无脑。

杂杂杂十

- 有三个服务器和N个任务，每个任务有一个在任意一个服务器上完成所需的时间t(i)。现在你需要往这三个服务器上分配任务，使得它们工作量的极差尽可能小。

- N <= 400.
- 1 <= t(i) <= 30.

杂杂杂十

- 小x：“这不就是个背包吗？”
- 小tcsn：“你算算复杂度”
- 小x：“我不是只需要dp[i][T1][T2]表示考虑了前i个物品，第一台服务器装了T1，第二台服务器装了T2是否合法就行了吗？”
- 小tcsn：“|T|=T1\T2的上界=1w2。复杂度与状态数同阶=O(N|T|^2)。”

杂杂杂十

- **结论：**不需要考虑每个服务器工作量>4030的情况。
- 原因显然。如果存在一台服务器工作量>4030，那么必然存在一台服务器工作量<4000。此时将任意一个任务移到该低工作量服务器上都不会使结果变差。
- 效果：将1w2*1w2的状态降到了4k*4k.

杂杂杂十

```
24      f[0][0] = 1 ;
25      rep(x, 1, 30) {
26          rep(i, 0, lim) rep(j, 0, lim) {
27              g[i][j] = - 1 ;
28              if (f[i][j]) g[i][j] = sz[x] ;
29          }
30          rep(i, 0, lim) rep(j, 0, lim) {
31              if (i >= x) g[i][j] = max(g[i][j], g[i - x][j] - 1) ;
32              if (j >= x) g[i][j] = max(g[i][j], g[i][j - x] - 1) ;
33          }
34          rep(i, 0, lim) rep(j, 0, lim) if (!f[i][j] && g[i][j] >= 0) {
35              f[i][j] = true ;
36          }
37      }
```

题 题 题 源

- 1. Codeforces 743E
- 2. Codeforces 599E
- 3. Codeforces 595E
- 4. Codeforces 908D
- 5. Codeforces 1174E
- 6. Codeforces 506D
- 7. Codeforces 1139C
- 8. Gym 101102
- 9. Codeforces 860E
- 10. 2015-2016 Petrozavodsk Winter Training Camp, Saratov SU Contest, A