题解 UR 24

出题人：gyh20

2021 年 11 月 4 日

目录

1 串 **2**

1.1 算法 1 . . . 2

1.2 算法 2 . . . 2

1.3 算法 3 . . . 2

2 艺术家 **3**

2.1 算法 1 . . . 3

2.2 算法 2 . . . 3

2.3 算法 3 . . . 3

2.4 算法 4 . . . 3

2.5 算法 5 . . . 3

3 黑白树 **4**

3.1 算法 1 . . . 4

3.2 算法 2 . . . 4

3.3 算法 3 . . . 4

3.4 算法 4 . . . 4

4 测试 **5**

4.1 算法 1 . . . 5

4.2 算法 2 . . . 5

4.3 算法 3 . . . 5

4.4 算法 4 . . . 5

1 串

1.1 算法 1

直接搜索，时间复杂度 $O(2^n n^2)$，期望得分 20。

1.2 算法 2

对于 $k=1$，直接将 1 放在最中间即可，结合之前的算法期望得分 40。

1.3 算法 3

先特判 $k=0$ 的情况。

将数组做前缀和，令前缀和数组中 1 的个数为 X，0 的个数为 Y，则答案为 $X\times Y$，由于 $X+Y=n+1$，所以我们只需要让 X,Y 尽量接近，一种简单的做法是先在最前方放 $k-1$ 个 1，再在后面的位置中选择最优的位置放一个 1，可以发现，这样构造一定能够做到 $X\times Y$ 取到最大，时间复杂度 $O(n)$，期望得分 100。

2 艺术家

2.1 算法 1

直接模拟所有操作，时间复杂度 $O(nmq)$，期望得分 $10 \sim 25$。

2.2 算法 2

对于测试点 $6 \sim 8$，$l_i = 1, r_i = i$，可以发现，若在某一时刻第 i 个区间内部所有颜色不同，则第 $i-1$ 个区间内部所有颜色不同，每次修改可以维护当前一个已经满足条件的前缀，修改后查询下一个区间是否满足条件，做到快速支持修改颜色以及区间判断是否颜色相同可以获得额外 15 分。

2.3 算法 3

我们可以发现，修改一个点时只会影响到所有包含到这个点的区间，每次修改后判断这些区间是否合法即可，可以额外获得 15 分。

2.4 算法 4

对于每次新增的颜色不相同，可以发现一个区间如果在某一时刻区间内部颜色互不相同则之后永远内部颜色互不相同，使用整体二分或者将区间在线段树上拆分为 log 个标记修改时访问标记即可，可以获得额外的 20 分。

2.5 算法 5

发现所有区间不相交也不包含，可以看作一个树形的关系，一个区间的父亲必须在这个区间之后满足要求，同时，所有的叶子节点的区间互不相交，相当于每个点只会在当前的一个区间中，每次修改之后查询这个区间是否合法即可。

查询一个区间是否合法有两种方式：

1. 建出树，相当于一个单点修改，子树数颜色，使用 dfs 序，set，树状数组可以做到 $O(\log n)$。

2. 在原数轴上维护 lst_i 表示 i 之前第一个颜色与 i 相同的点，则一个区间内部颜色互不相同等价于 $\min\{lst_i, l \le i \le r\}$，使用 set 和线段树维护可以做到 $O(\log n)$。

时间复杂度 $O(n + (m+q)\log n)$，期望得分 100。

3 黑白树

3.1 算法 1

枚举，时间复杂度 $O(2^n n^2)$，可以获得 10 分的高分。

3.2 算法 2

对于链，如果不存在颜色的差别，那么到一个点最远的点一定是 1 或者 n，枚举答案 x，可以发现，1 号点的颜色需要和 $x+2 \sim n$ 的所有点的颜色相同，n 号点需要和 $1 \sim n-x-1$ 的所有点颜色相同，由于 $1, n$ 两点颜色不同，所以中间所有点的最远点一定是 $1, n$，所以 $x+1$ 或者 $n-x$ 的其中一个需要满足与另一边的颜色相同，中间的点颜色随意，预处理 2^k 可以做到 $O(n)$，结合算法 1 期望得分 40。

3.3 算法 3

对于菊花图，可以轻易的发现若 $n > 4$，每一棵树的答案一定为 2，因为除了 1 号点其它点至少存在两个颜色相同的点。

同样的，对于测试点 7，我们也有同样的结论，只不过长度不一定为 2，需要先求一遍长度，结合之前的算法期望得分 60。

3.4 算法 4

对于普通的树，到一个点最远的点一定是直径的端点之一，先求出一条直径，若直径的两个端点颜色相同，则最长距离一定为直径，否则，令两个端点分别为 x, y，枚举答案 d，所有到 x 距离 $> d$ 的点颜色必须与 y 一样，所有到 y 距离 $> d$ 的点颜色必须和 x 一样，由于 x, y 是直径的两个端点，可以发现，若一个点 z 到 x, y 的距离都不超过 d，则其到任何一个点的距离不超过 d，所以 z 的黑白并不会对答案产生影响。

具体的，找出直径，预处理直径到每一个点的距离，从大到小枚举 d，维护当前没有被限制的点的个数，当出现 $\min dis(x,i), dis(y,i) > d$ 时说明 d 已经不合法，预处理 2^k 可以做到 $O(n)$。

4 测试

4.1 算法 1

搜索，可以获得 10 分的高分!

4.2 算法 2

树上问题一般考虑子树 DP，但操作是子树操作，子树内部的状态是不足以表示完整的操作，所以我们的状态可以结合子树外部的点。

设 $f_{x,i}$ 表示子树 x，来自外部的加操作的总和是 i，子树内部还需要的最小次数。

则转移分为三步：

$f_{x,i} = f_{ls_i} + f_{rs_i} + w(x,i)$，其中 $w(x,i)$ 表示给 x 加上 i 后是否需要操作，此时还没有进行对 x 的 2 操作。

$f_{x_i} = \min\{f_{x,i}, \min\{f_{x,j}\} + 1\}$，这里表示给 x 位置进行一次 2 操作，相当于一个整体位移。

直接转移可以做到 $O(nK)$ 的复杂度，期望得分 30。

4.3 算法 3

观察转移式，发现对于一个 x，所有的 $f_{x,i}$ 都是 a 或者 $a+1$，并且转移相当于一个对位加，可以优化 DP 的存储状态，使用 bitset，维护出其中所有为 a 的位置，由于是二叉树，转移是三个 bitset 合并，可以在常数时间内求出其中的对位加后的最优位置以及新的最优位置，都可以表示为一些集合的交与并，时间复杂度 $O(\frac{nK}{w})$，期望得分 50。

4.4 算法 4

可以发现，在叶子节点时，$w(x,i)$ 在两个区间为 0，其它区间为 1，每一次的转移对最优决策点的影响都是常数个集合的交并，可以尝试维护每个点最优的区间集合。

维护集合，并且从子树转移，很容易想到启发式合并，现在我们需要在 $\min(|S1|,|S2|)$ 的时间内求出两个集合的交和并，这显然是不可能的，但我们的时间是不用管删除的，因为每个区间只会被删除一遍。

交：枚举小的集合，每次切割另一个集合中当前集合右端点的那一个点，删掉当前左端点与上一个右端点的之间的所有区间。

并：枚举小的集合，每次切割另一个集合中当前集合右端点的那一个点，删掉另一个集合中被包含的区间，再将当前区间加入。

在每一个位置时枚举当前最多合并时能有多少位置取到 1，时间复杂度 $O(n\log^2 n)$，期望得分 100。