题解 2021 CSP/NOIP挑战赛 Contest10

出题人：gyh20

2021 年 11 月 14 日

目录

1 黑白串 **2**

1.1 算法 1 . . . 2
1.2 算法 2 . . . 2
1.3 算法 3 . . . 2

2 黑白树 **3**

2.1 算法 1 . . . 3
2.2 算法 2 . . . 3
2.3 算法 3 . . . 3

3 多彩串 **3**

3.1 算法 1 . . . 3
3.2 算法 2 . . . 3
3.3 算法 3 . . . 3
3.4 算法 4 . . . 4
3.5 算法 5 . . . 4

4 多彩树 **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 黑白串

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 黑白树

2.1 算法 1

直接枚举，时间复杂度 $O(2^n n^2)$，期望得分 20。

2.2 算法 2

考虑拆贡献，可以发现两点间的贡献仅与两点距离相关。

令两点间距离为 k，则答案为 $\sum_{i=0}^{k} \binom{k}{i} i^2$。

$$\sum_{i=0}^{k} \binom{k}{i} i^2$$

$$= k \sum_{i=1}^{k} \binom{k-1}{i-1} i$$

$$= k(\sum_{i=0}^{k-1} \binom{k-1}{i} i + \sum_{i=0}^{k-1} \binom{k-1}{i})$$

$$= k((k-1)2^{k-2} + 2^{k-1})$$

求出距离为 k 的点对数量即可，根据实现方法不同可以得到 $40 \sim 90$ 分。

直接求出距离为 k 的点对数量是不能 $O(n)$ 的，可以换根 DP 直接维护这个式子，时间复杂度 $O(n)$。

2.3 算法 3

考虑平方的实际意义，相当于对于每一条路径求出在其中选出两个黑点的方案，枚举两个点作为黑点，那么方案数为两点的子树大小乘积，对于互为祖先后代的点对特殊计算即可，时间复杂度 $O(n)$。

3 多彩串

3.1 算法 1

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

3.2 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。

3.3 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。

3.4 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。

3.5 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 $\max\{lst_i, l \le i \le r\}$，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。

4 多彩树

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。