P11233 [CSP-S 2024] 染色

题意：

- 给定 n 个数 A_i，对 A_i 进行染色，只有两种颜色。设 C 为 A 染色后的数组。
  - 如果 A_i 左侧没有预期同色的数，则令 $C_i = 0$。否则，记其左侧与其最靠的同色数为 A_j，若 $A_i = A_j$，则令 $C_i = A_i$，否则令 $C_i = 0$。
- 题目求：$\sum_{i=1}^{n} C_i$，需要最大化最终得分，请求出最终得分的**最大值**。

分析：

$20pts$

数据范围：$n \leq 15$，暴力枚举每个数字两种染色的情况 2^n，然后扫描一遍 $O(n)$，找出最大分值，时间复杂度：$O(T \times n \times 2^n) \approx 9.8 \times 10^6 \leq 10^8$ 是可以接受的。

$50pts$

想拿满分，最优性问题要 DP 解法了。

> 二维 DP 作为思路启发：

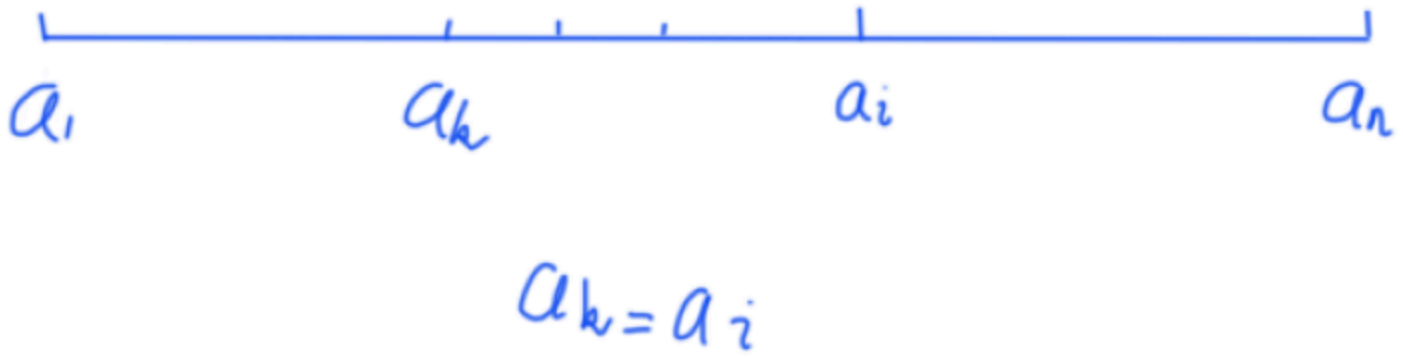
设：$f(i,j)$ 前 i 个数，第 j 个数染色为 j 的最大分值。

分类讨论：

- $1 \sim a_{i-1}$ 都没有相同的数，则贡献 $f(i,0) = f(i,1) = max(f(i-1,0), f(i-1,1))$。
- $1 \sim a_{i-1}$ 前面存在相同的数，假设以染色为 0 来看 $f(i,0)$，前面确定也存在一个染色染色为 0 的数 $(a_i = a_j)$，

  且**这之间所有的数都染色为 1**。确定是前面最靠近相同的数染为相同色，否则在这之间找到相同的数并染色为同色收益也会更大。相同色的数的位置（可预处理出来，$O(n)$），则要分 3 部分进行计算贡献：

  > 设 $l[i]$：表示数字 i 左侧最靠近且相同的数字位置下标。



部分 ①：

位置 $(1, l[a_i]+1)$ 数产生的贡献，其中 $l[a[i]]+1$ 染成 1，即 $f(l[a_i]+1, 1)$ 的贡献。不用担心（没有被染成 0）的问题，因为 $f(l[a_i], 1)$ 会选择最优的策略，那么（此时可以选择不和 $l[a_i]+1$）染为同色。

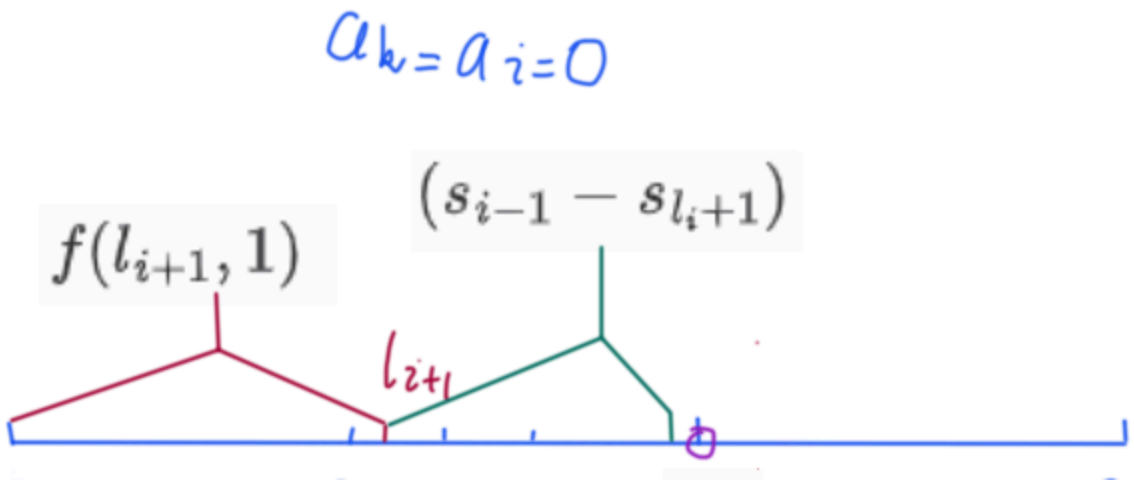
部分 ②：

位置 $ l[a_i]+2 \sim i-1 $ 的**数产生的贡献**，这贡献会把**区间 $l+1 \sim i-1$ 染为同色产生的贡献计算上**，设 s_i 为 $1 \sim i$ 位置上染为同色的贡献。那么 $l+1 \sim i-1$ 全部染为同色所产生的的贡献，或可以表示为 $l+2 \sim i-1$ 数产生的贡献 $\left(s_{i-1} \sim s_{l+1}\right)$。

部分 ③：

在 i 位置上的数，贡献就是 a_i。

> 上述对应的 3 个部分，可以表示如下图所示：



对于前面存在相同数字的贡献，总结：

$$f(i,0)=\begin{cases} f(l_{i+1},1) \\ (s_{i-1}-s_{l_i+1}) \\ a_i \end{cases}$$

特判 $a_i=a_{i-1}$ 的情况，即 $f(i,0)=f(i-1,0)+a_i$ 和 $f(i,1)=f(i-1,1)+a_i$。

在产生贡献和不产生贡献中最大分值中取 max:

$$\begin{cases} f(i,0)=max(f(i-1,0),f(i-1,1),a_i+f(l_i+1,1)+(s_{i-1}-s_{l_i}+1)) \\ f(i,1)=max(f(i-1,0),f(i-1,1),a_i+f(l_i+1,0)+(s_{i-1}-s_{l_i}+1)) \end{cases}$$

$$ans=max(f(n,0),f(n,1))$$

> 最开始就是如上的思路去解题的，但想了一下第二个维度有必要开么？
>
> 感觉可以去掉，再思考一下。

返璞归真，简单的状态表示：

设状态 f_i 表示前 i 位数的最大分值和。显然对于每一个位置 i 可以令 $f_i = f_{i-1}$。

用 l_i 记录 i 上一次出现的位置，初始化令所有的 $l_i = 0$，每遍历到一个位置，动态更新 $l_{a_i} = i$。然后枚举区间更新 f_i，也可以预处理 g 数组 $g[i][j]$ 表示区间 $[i, j]$ 的染色为相同时的最大分值之和，复杂度是 $O(n^2)$，$n \leq 10^3$，$50pts$ 到手。

参考代码：

```cpp
// 50pts
#include <bits/stdc++.h>
const int N = 2e3 + 5;

int n, T;
int a[N], lst[N];  // lst[i]：表示数字 i 最靠近 i 左侧的数字位置
int f[N], g[N][N]; // f[i] ::: 表示考虑前 i 个数贡献的最大分值之和
```

```cpp
int main()
{
    std::cin >> T;
    while (T--)
    {
        memset(a, 0, sizeof a), memset(lst, 0, sizeof lst), memset(f, 0, sizeof f), memset(g, 0, sizeof g);
        std::cin >> n;
        for (int i = 1; i <= n; i++)
            std::cin >> a[i];

        for (int i = 1; i <= n; i++) // 构造 区间 [i,j] 染色相同的最大分值之和
            for (int j = i + 1; j <= n; j++)
            {
                g[i][j] = g[i][j - 1];
                if (a[j] == a[j - 1])
                    g[i][j] += a[j];
            }

        for (int i = 1; i <= n; i++)
        {
            if (lst[a[i]] == 0) // 前 i-1 位不存在相同的数字 a[i]
            {
                lst[a[i]] = i;   // 更新数字 a[i] 最后一次出现的下标
                f[i] = f[i - 1]; // 继承前 i-1 位的最大分值
            }
            else
            {
                f[i] = std::max(f[i - 1], std::max(f[lst[a[i]] + 1], f[lst[a[i]]]) + a[i] + g[lst[a[i]] + 1][i - 1]);
                lst[a[i]] = i; // 更新数字 a[i] 最后一次出现的下标
            }
        }
        std::cout << f[n] << "\n";
    }
    return 0;
}
```

$100pts$

$n \leq 10^6$， $g[i][j]$ 开不下这么大的数组。用**前缀和优化**，每当 $a_i = a_{i-1}$ 时候，更新前缀和数组 s_i。最后对于 a_i 如果 l_{a_i} 存在，则对 f_i 转移：

$$f_i = (1 \leq i \leq n)\ max\{f_{l_{a_i}+1}, a_i, (s_i - s_{l_{a_i}})\}$$

参考代码：

```
#include <bits/stdc++.h>
#define ll long long

const int N = 1e6 + 10;
// f[i] ::: 表示前 i 个数，最大分值之和，lst[i] ::: 表示数字 i 前面最靠近的相同数字位置，s[i] ::: 表示前 i 个数染相同颜色的总贡献
ll f[N], s[N], lst[N], a[N];
ll n, T;
```

```cpp
int main()
{
    std::ios::sync_with_stdio(false), std::cin.tie(nullptr);
    std::cin >> T;
    while (T--)
    {
        memset(f, 0, sizeof f), memset(s, 0, sizeof s), memset(lst, 0, sizeof lst), memset(a, 0, sizeof a);

        std::cin >> n;
        for (int i = 1; i <= n; i++)
            std::cin >> a[i];

        // 构造 1~i 个数，全部数字染色相同时的最大分值之和
        for (int i = 2; i <= n; i++)
            s[i] = (a[i] == a[i - 1] ? s[i - 1] + a[i] : s[i - 1]);

        for (int i = 1; i <= n; i++) // 枚举数字的位置 i
        {
            f[i] = f[i - 1]; // 省去判断，默认继承
            if (lst[a[i]])   // 如果前面存在相同的数字，则将 3 部分的贡献作对比更新
                f[i] = std::max(f[i], f[lst[a[i]] + 1] + a[i] + s[i] - s[lst[a[i]] + 1]);
            lst[a[i]] = i; // 更新数字 a[i] 出现的位置
        }
        std::cout << f[n] << "\n";
    }
    return 0;
}
```