E - Make it Palindrome (atcoder.jp)

简介

对于一个数组。统计所有子数组变成回文数组的最小花费。

solve

对于一个数组，变成回文数组的最优方案是，配对不一样就改一个。

例如: 1 2 3 1 2 3 3

1. 根据容斥，我们假设每一个配对都要改。

```
1 | 然后总贡献为：
```

$$\sum_{i=1}^{n}(N-i+1)\times\left\lfloor\frac{i}{2}\right\rfloor \tag{1}$$

```
1 |     这样直接通过$O(N)$求出即可。
```

2. 同去匹配项的总贡献：从最短的串开始，串两头逐步向两边拓展。贡献次数为

$$\min(l,N-r+1); \tag{2}$$

3. 记录下相同的位置，然后尝试推导出一些结论。快速的计算出每一对的贡献：用双指针对于同一个数字的记录数组上移动。

   然后发现了贡献的统计方法：

$$\begin{gathered} pos[low]<N-pos[high]+1; \\ 说明low....high中的每一个与low对应的数字，贡献都为pow[low] \\ 同理可以推出high的贡献。 \\ 从起点开始，逐渐把所有的贡献计算完成。 \end{gathered} \tag{3}$$

```cpp
#include<bits/stdc++.h>
using namespace std;
using ll = long long;

const int N = 2E5 + 10;

vector<int> pos[N];
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);

    int n ;
    cin >> n;
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        int x; cin >> x;
```

```
        pos[x].push_back(i);
        ans += 1LL * (n - i + 1) * (i / 2);
    }
    for (int i = 1; i <= n; i++) {
        ll low = 0 , high = pos[i].size() - 1;
        while (low < high) {
            if (pos[i][low] < n - pos[i][high] + 1) {
                ans -= (high - low) * pos[i][low];
                low++;
            }
            else {
                ans -= (high - low) * (n - pos[i][high] + 1);
                high --;
            }
        }
    }
    cout << ans << '\n';

}

/* stuff you should look for
* int overflow, array bounds
* special cases (n=1?)
* do smth instead of nothing and stay organized
* WRITE STUFF DOWN
* DON'T GET STUCK ON ONE APPROACH
*/
```