1637B - MEX and Array

Idea: **__JustMe__** and **Mangooste**.

Hint 1

What does happen after replacing a segment of length greater than 1 with segments of length 1?

Hint 2

The cost of the array $b_1, b_2, \dots, b_k$ equals to $k + \sum_{i=1}^{k} \mathrm{mex}(\{b_i\})$.

Tutorial

1637B - MEX and Array

We show, that replacing a segment of length k ($k > 1$) with segments of length 1 does not decrease the *cost* of the partition. Consider two cases:

1. The segment does not contain 0.
2. The segment contains 0.

In the first case the contribution of the segment equals to 1 (because $\mathrm{mex} = 0$), but the contribution of k segments of length 1 equals to k. So the *cost* increased. In the second case the contribution of the segment equals to $1 + \mathrm{mex} <= 1 + k$, but the contribution of the segments of length 1 would be at least $1 + k$, so the *cost* has not decreased.

Then it is possible to replace all segments of length more than 1 by segments of length 1 and not decrease the *cost*. So the *value* of the array $b_1, b_2, \dots, b_k$ equals to $\sum_{i=1}^{k} (1 + \mathrm{mex}(\{b_i\}))$ = k + (the number of zeros in the array).

To calculate the total value of all subsegments, you need to calculate the total length of all subsegments and the contribution of each 0. The total length of all subsegments equals to $\frac{n \cdot (n+1) \cdot (n+2)}{6}$. The contribution of a zero in the position i equals to $i \cdot (n - i + 1)$. This solution works in $O(n)$, but it could be implemented less efficiently.

There is also another solution, which uses dynamic programming: let $dp_{l,r}$ is the *value* of the array $a_l, a_{l+1}, \dots, a_r$. Then $dp_{l,r} = \max(1 + \mathrm{mex}(\{a_l, a_{l+1}, \dots, a_r\}), \max_{c=l}^{r-1}(dp_{l,c} + dp_{c+1,r}))$. This solution can be implemented in $O(n^3)$ or in $O(n^4)$.

Solution

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int t;
    cin >> t;
    for (int i = 0; i < t; i++) {
        int n;
        cin >> n;
        vector<int> a(n);
        for (auto& u : a)
            cin >> u;

        int ans = 0;
        for (int i = 0; i < n; i++) {
            ans += (i + 1) * (n - i);
            if (a[i] == 0)
                ans += (i + 1) * (n - i);
        }
        cout << ans << '\n';
    }
}
```