[Python] 4 lines linear solution, detailed explanation

32

DBabichev 2387 Last Edit: 12 hours ago 823 VIEWS

This is quite difficult problem if you want to have optimal solution. First, I tried several ideas, using queue ar algorithm, where I tried to build sequence symbol by symbol, but it was working very slow and I get TLE. So coding and to think.

So, what is the main trick? First of all notice, that what is matter, is frequence of each letter, not order. For each letter we need to evaluate minimum window size we need to fully use this letter. For example, if we in our tasks and `n = 3`, then minimum window looks like `A...A...A...A`. So, the minimum length in this `= (n+1) * 3 + 1`. Let us call this number 13 **characteristic** of letter `A`. More examples:

1. If `n = 4` and we have `BBBBB`, than **characteristic** of letter `B` is equal to `(n+1) * 4 + 1 = 21`.
2. If `n = 1` and we have `CC`, than **characteristic** of letter `C` is equal to `(n+1) * 1 + 1 = 3`.

So, we need to evaluate **characteristics** of all letters and just choose the maximum one? Similar, but not exa have two letters with the same **characteristic** (it means they have the same frequencies), like we have `AAAA` need to have window `A...A...A...A` and also window `B...B...B...B`, and you can not put one inside a case we need at least one symbol more, and example will be `AB..AB..AB..AB`.

In this problem we are asked, what is the **minimum** number of units we need to use, so, mathematicaly spe do two steps:

1. **Estimation:** prove, that we need at least say **k** units.
2. Give an **example** for this **k** units, how to construct desired sequence. (note, that in this problem you c create example, but we still need to prove, that it exists).

We already considered **Estimation**: we need to find elements with the highest **characteristic** and check how elements we have. So, if `freq = Counter(tasks)` and `Most_freq = freq.most_common()[0][1]` is the ele frequency, than `Found_most = sum([freq[key] == Most_freq for key in freq])` is number of such elen `max(len(tasks), (Most_freq - 1) * (n + 1) + Found_most)`, because we can not be shorter than `len`

The most difficult part is to prove that **example** exists. Let us consider case, where we have `AAAA`, `BBBB`, `n` have some letters. Then first step is to build:

```
AB....AB....AB....AB
```

What we know about other elements? Their frequencies are less than `Most_freq = 4`. So, let us start to fill order:

```
A B 1 4 7 10 A B 2 5 8 11 A B 3 6 9 12 A B
```