Kick Start 2018 - Round F

Analysis: Common Anagrams

Common Anagrams: Analysis

Small and Large dataset

We can create a boolean function f(i, j, k, l) that returns whether the substring from the i-th through the j-th characters of **A** (inclusive) is anagrammatic to the substring from the k-th through the l-th characters of **B** (inclusive).

To do this for the Small dataset, we only need to check that both substrings contain the same number of `A`s and the same number of `B`s.

To do this for the Large dataset, we can loop from i to j to get the number of occurrences of each character in **A**[i .. j]. Similarly, we also loop from k to l to get the occurrences of each character in **B**[k .. l]. We then check whether each character occurs the same number of times in the two substrings.

We want to return the number of ordered pairs (i, j) such that there exists an ordered pair (k, l) satisfying f(i, j, k, l). Since computing a single value of f(i, j, k, l) takes O(**L**), this solution runs in O($\mathbf{L}^5$) time. We can optimize this by observing that if $j - i \neq l - k$, then f(i, j, k, l) will certainly return false. Therefore, for each value of (i, j, k) we only need to check f(i, j, k, k + (j - i)). This removes one O(**L**) loop and improves the algorithm to O($\mathbf{L}^4$) time.

Further improvements

Although O($\mathbf{L}^4$) is fast enough, we can improve the algorithm further. Instead of looping through all substrings of **B** for each substring of **A**, we can do some preprocessing. Before we even process **A**, we can loop through all substrings of **B**. For each substring, we count the occurrences of each character and store it in a hashset. Note that this can be done in O($\mathbf{L}^2$) time, since we can use the character occurrences of B[i .. j] to compute the character occurrences of B[i .. j+1].

Once the preprocess is complete, we can loop through all substrings of **A**. Simiarly, for each substring, we count the occurrences of each character. We then check whether these occurrences of characters are present in our set. If they are, we can increment our answer counter. We can also do this in O($\mathbf{L}^2$) time.

Therefore, this solution runs in O($\mathbf{L}^2$) time overall.