LC-2062 Count Vowel Substrings

Let j → starting of a substring that contains all vowels

i → current position

and k → an index b/w j and i, such that always the substring b/w k and i contains an all vowel substring.

Between j and i → there will be a valid substring.

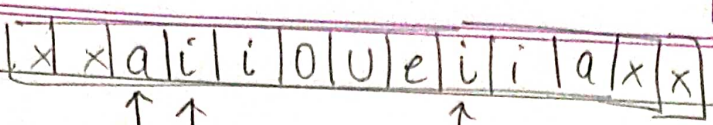
So, for every position i in the array, we have (k-j) valid substrings.

e.g., | x | x | a | i | i | o | u | e | i | i | a | x | x |

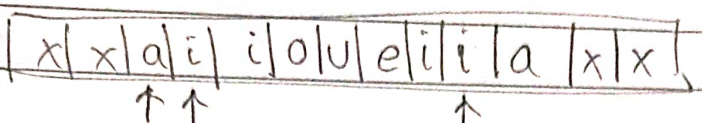
j k i

k-j = 1 substring b/w j and i

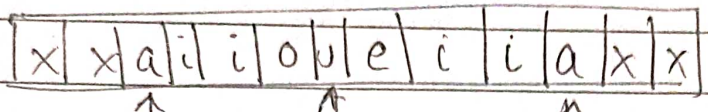
The substring b/w k and i doesn't contain all 5 vowels so we can't forward k yet, We keep forwarding i. And for every index i, we add (k-j) to the result.



k-j = 1 valid substring b/w j and i



k-j = 1 valid substring b/w j and i



k-j = 4 valid substrings b/w j and i.

moving k ahead until we have all vowel substring b/w k and i. existing.

CODE:=

```
int j=0, k=0, vowels=0, ans=0;
unordered_map<char, int> mp; mp['a']=0, mp['e']=0, ..., mp['u']=0;

for(int i=0; i<word.size(); i++)
{ if(mp.find(word[i])!=mp.end()) // is a vowel
  { vowels += (++mp[word[i]]==1); //only if new vowel found

    while(vowels==5) // loop forwards k until when valid substring exists b/w k and i.
    { vowels -= (--mp[word[k++]]==0);
      ans += (k-j); // ans += (k-j)
    }
  }
  else
  { vowels=0; mp['a']=mp['e']=...=mp['u']=0;
    j=k=i+1;
  }
}
return ans;
```