Question statement (5):

Return the longest contiguous substring of 2 distinct characters from an input string.

Example

input: abbaacab
output : abbaa

input: abcefabbabaabefghghfa
return: abbabaab

input: aabcedddcdccecabceftg
return: ddddcdcc

input: acbabbcbca
return : bbcbc

.) understanding:

Question (5)

I-dentify the prob

•) finding the longest ~~repeating~~ substring

Define the goal

•) the returned string has to have these characteristics:

- -) contain 2 distinct characters
- -) the characters must be adjacent to one another in original string
- -) if there are multiple substrings with the above 2 qualities return the longest substring

•) This problem is ~~really~~ composed of 2 sub-problems

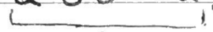
-) finding particular kinds of substrings

-) of all the substrings found return the longest

E-xplore
A-nticipate
-ct
through Duke's
Approach

1) Examples by hand

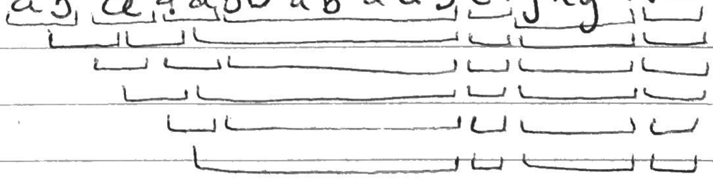
-) abbaacab



->

abbaa

-) abcefabbabaabefghghfa

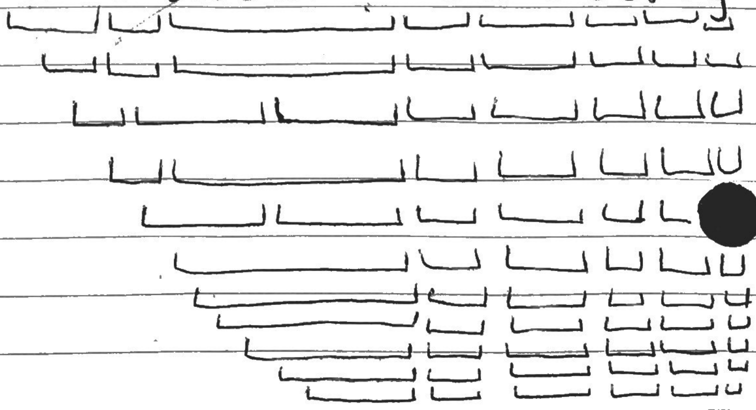


->

abbabaab

* w/ every pass looking for a 3rd new letter

-) aabceddddcdccecabceftg



2) write down what I did

- ·) made multiple passes through original to create all the possible sub strings
  - →) created substrings based on the 2 criteria
- ·) looked for longest substring

2) needs to be teased out further

- ·) accepted the 1st 2 chars encountered and ~~stop~~ "stored" that substring
  - " created new substring starting @ the index of the new char
- ·) repeated the process above at (Ø+i) until, moving the beginning point forward into the array until @ the end of the original string

3) find patterns

- .) if a substring was already in the set of substrings then didn't "store" it
- .) if a substring is a substring of another susbtring then it can also be ignored

!!! New Approach

.) comparing strings lengths

- -) the sub algo to construct substrings can be modified
- -) still, repeated increment through the array by advancing the starting point through the index
- -) instead of constructing all substrings do this:
  - o) make a ~~string~~ ~~int~~ okay variable

    if the new substring is longer than the curr maxSubtring then replace the currMaxSub

Step 2 & 1, & 3 & 4
Redux

☐1 Examples by Hand

→ abbaacab

curr max Substring = ~~""~~

newSubstring = "abbaa"

currMaxSub.length() < newSub.length()

then currmaxSub = newSub

move pointer frward

newSubString = "bbaa"

not longer than currMaxSub

nSS = "baa"

not longer

nss = ~~[illegible]~~ "aaca"

not longer

nss = "aca"

not longer

nss = "ca"

not longer

nss = "ab"

not longer

move pointer frward

5) translate to code

6) ~~Debu~~ Run test cases

7) ~~Debug test cases~~