CS170–Fall 2014 — Solutions to Homework 9

Quoc Thai Nguyen Truong, SID 24547327, `cs170-ig`

November 10, 2014

Collaborators: God

DNA Subsequence

Main idea.
The main idea is to have a OverLap function which take 2 sub-strings and return a tuple of a overlap between 2 sub-strings and the maximum overlap of 2 sub-strings. Then, we read the text file and append it to the array (using close set so the array won't have duplicate sub-string).
There are one while loop (outer) and one for loop (inner), call OverLap function between s1, and s2, keep track of the maximum overlap which store a tuple of 3 elements (overlap string, number of maximum overlap, index of s2 in the array) . s1 is alway the 1st element in the array and s2 is the rest of the element in the lines array. If all the overlaps are NOT none (k[1] $\neq 0$), then remove s1 (1st element in the array) and s2 from the array , and append the most overlap string between 2 sub-strings. If all the overlaps are none which we know from k (k[1] = 0 overlap), then add remove s1(1st element in the array) and append s1 to the array (end of the array)

Let M(i,j) = the string which has the maximum number of overlap of sub-string lines[i] and sub-string lines$[i+1, i+2, \cdots, j]$

$$M(i,k) = Max \begin{cases} OverLap(lines[i], lines[i+1]) \\ OverLap(lines[i], lines[i+2]) \\ \cdots \\ OverLap(lines[i], lines[j]) \end{cases}$$

$$If\ M(i,j)\ is\ None\ ,\ remove\ lines[i]\ and\ append\ it\ to\ lines$$

$$Else:\ remove\ 2\ substring\ of\ max\ overlap\ and\ append\ the\ new\ sub-string\ to\ lines$$

$$RESULT = M(i,j)_{i:=0\ to\ n-1}^{j:=1\ to\ n}$$

Pseudocode.
Line 0: OverLap(s1,s2):
Line 1: If len(s1) < len(s2): s1,s2 = s2,s1
Line 2: If len(s2) == 0: return s1
Line 3: k = (””,0,0) keep track of the most number of over lap between 2 substrings
Line 4: Loop 1: If s2 is complete substring of s1, update k the most number of over lap
Line 5: Loop 2: If s1 and s2 are overlap on the tail, update k the most number of over lap
Line 6: Loop 3: If s2 and s2 are overlap on the head, update k the most number of over lap

Line 7: If k[1] == 0: return None ← there is no overlap between 2 sub-string
Line 8: return (k[0],k[1]) ← tuple (string, number of overlap)

Line 9: lines = []
Line 10: lines = read all the lines from the text file and append each substring(using close set so the array won't have duplicate sub-string)
Line 11: While(len(lines) > 2):
Line 12: s1 = lines[0]
Line 13: k = ("",0,0) Line 14: for $i := 1$ to len(lines):
Line 15: S = OverLap(s1,line[i])
Line 16: If S $\neq$ None:
Line 17: If k[1] < S[1]:
Line 18: update k = (S[0],S[1],i)
Line 18: If (k[1] $\neq$ 0):
Line 19: remove element s1 and element at index k[2] from lines
Line 20: append the new overlap string k[0] to lines
Line 21: Else: remove s1 (1st element) from lines and append s1 to lines

Line 22: If len(lines) == 2: s1 = overlap(line[0], lines[1]), and output(s1[0])
Line 23: Else : output(lines[0])

Running time.

$$\boxed{T(n,k) = \Theta(n^2k^2)}$$

Justification of running time.
Let T(k,n) be the run time of the algorithm with n lines of substring at the most length of k.
From line 0 to line 8: It's the OverLap function, there are 3 different cases, and each case is the loop of checking between 2 substring which take $\Theta(k^2)$, and return the tuple of (string, number of overlap)
From line 11 to line 22: has 1 while loop and one for loop. the inner loop (for loop) call OverLap function between s1, and s2. s1 is alway the 1st element in the lines array and s2 is the rest of the element in the lines array. There are n of substrings, and we do this for every single substring. Hence, it will call Overlap function $\Theta(n^2)times$, and each time calling overlap take $\Theta(k^2)$. Therefore, the total run time would be:

$$\boxed{T(n,k) = \Theta(n^2k^2)}$$