CS311 - Analysis of Algorithms

Term Project

Milestone # 3: Complexity analysis and correctness of algorithm:

We will determine the running time of our algorithm. Here each instruction takes unit time (1).

1. str1[],str2[] ----------(1)
2. ArrayOfFile[] ----------(1)
3. for t=0 to ArrayOfFile.length ----------()
4. while(!(end of file[t])) ----------(n+1)
5. ArrayOfFile[t]<<str[0] ----------(n)
6. i=0; ----------(n)
7. while(str1[i]!=".") ----------(n/li+1)
8. ArrayOfFile[t]<<str1[i]; ----------(n/li)
9. i++ ----------(n/li)
10. while(!(end of file2)) ----------(n+1)
11. ArrayOfFile[t+1]>>str2[0]; ----------(n)
12. j=1 ----------(n)
13. while(str2[j]!=0) ----------(n/li+1)
14. ArrayOfFile[t+1]<<str2[i] ----------(n/li)
15. j++ ----------(n/li)
16. temp=0; ----------(n)
17. if((LongestCommonSubstring(str1,str2,lengthOfString1, lengthOfString2)>temp)
18. temp=LongestCommonSubstring(str1,str2,lengthOfString1, lengthOfString2) ----------(n)
19. resultArray[]=str2; ----------(n)

Here loop in the line 3 iterates equal to length of the number of files user enter. Like if user enter 3 it will executes 3 times. Loop in the lines number 4 iterates n+1 times.

li is equal to the number of lines or number of statements the file has.

Now,

Running time is =1+1+ (n+1) + n + n+ (n+1)*n/li + n/li + n/li +(n+1)*(n + 1) +n + (n+1)*n/li + n/li + n/li + n(LCS time) + n + n

Now we have to calculate the running time of LCS.

20. constant MAX_SIZE=10000
    ----------(1)
21. resultArray[MAX_SIZE]
    ----------(1)
22. memoizedArray[MAX_SIZE][MAX_SIZE]
    ----------(1)
23. function LongestCommonSubString(string string1[],string string2[], lengthOfString1, lengthOfString2)
24.
25. if(lengthOfString1==0 OR lengthOfString2==0)
    ----------(1)
26. return 0
27. //if the data is already in the table then return it instead of recomputing it
28. if(memoizedArray[m-1][n-1]!=-1)
    ----------(1)
29. return memoizedArray[m-1][n-1]
30. //case 1: if matches
31. if(string1[m-1]==string2[n-1])
32.
33. return 1+LongestCommonSubstring(string1,string2, lengthOfString1-1, lengthOfString2-1) (i)
34.
35. //if do not match
36. else
37. return max(LongestCommonSubstring(string1,string2, lengthOfString1-1, lengthOfString2), (ii)

38. LongestCommonSubsequence(string1,string2, lengthOfString1, lengthOfString2-1))

Recurrence equation for equation (i) is:

$$T(n) = \begin{cases} 0 \\ T(n1-1, n2-1)+1 \\ \max(T(n1-1, n2), T(n1, n2-1))+1 \end{cases}$$

it is 0 when n1=0 or n2=0 ……it is T(n1-1,n2-2) + 1 when matches ……max(T(n1-1,n2),T(n1,n2-1)+1 when do not matches.

T(n1-1,n2-1)+1

↓

T(n1-2,n2-2)+2

↓

T(n1-3,n2-3)+3

↓

:

:

:

↓

T(n1-k,n2-k)+k

Assume, n1-k=0 => n1=k similarly, n2-k=0 => n2=k

As k=n1 as well as k=n2 so let k=n

So, T(n1-n1,n2-n2)+n ➔ T(0) + n ➔ we know T(0) = 0 which is base case so is it n.

Do same for other recursive equation.

T(n1-1,n2)+1 ↓ T(n1-2,n2)+2 ↓ T(n1-3,n2)+3 ↓ : : : ↓ T(n1-k,n2)+k Assume, n1-k=0 => n1=k	T(n1,n2-1)+1 ↓ T(n1,n2-2)+2 ↓ T(n1,n2-3)+3 ↓ : : : ↓ T(n1,n2-k)+k Assume,n2-k=0 => n2=k

As k=n1 as well as k=n2 so let k=n So, T(n1-n1,n2)+n ➔ T(0,n2) + n ➔ we know T(0) = 0 which is base case so is it n.	As k=n1 as well as k=n2 so let k=n So, T(n1,n2-n2)+n ➔ T(0,n2) + n ➔ we know T(0) = 0 which is base case so is it n.

So

1+1+1+1+1+n+n ➔ 5+2n ➔ O(n)

Put it into the equation we are solving for finding the running time of algorithm.

Running time is =1+1+ (n+1) + n + n+ (n+1)*n/li + n/li + n/li +(n+1)*(n + 1) +n + (n+1)*n/li + n/li + n/li + n(LCS time) + n + n

=1+1+ (n+1) + n + n+ (n+1)*n/li + n/li + n/li +(n+1)*(n + 1) +n + (n+1)*n/li + n/li + n/li + n(n) + n + n

= 2 + n +1 +2n+ n^2/li + n/li + 2n/li + n^2 +2n +1 + n^2/li + n/li + 2n/li +n^2 +2n

=4+ 7n + 6n/li + $2n^2$/li + $2n^2$

= O(n^2)

The outer for loop iterates equal to the number of files given. So let number of files given is f then the total running time is = O(fn^2)

Correctness of algorithm:

Correctness of algorithm can be shown suing contradiction and inductive hypothesis. But we will show that our algorithm is correct using loop invariant. For this we will claim that before and after each iteration of loop it(result array) has the longest common matched substring.

Initialization:

Before the first iteration, lcs was not called and temp =0 which means number of matched words is 0. Which is the correct answer.

Maintenance:

Line one of file1 will be selected and lcs was called on line 1 of file1 with each line of file 2 one by one. Simply, line1 of file matches with line1 of file 2 then matched words length stored in temp. again line 1 of file 1 matches with line 2 of file2 and match words length store in temp if it is greater than previous value of temp. this procedure continouse until the file2 ends. Hence the line of file2 which matches with greatest matched words with line 1 of file1 then stored in resultArray. Here loop invariant maintains because after the forst iteration result array has the longest common words from both files.

Termination:

As we explained in maintenance how our algo works for one line of file 1. It will do the same for each and every line of file1. In the end result array wil have al the longest common substrings or words.

Lets suppose two files.

File1:	File2:
Lorem ipsum is simply dummy text. There are many variations of passages available.	Lorem ipsum is not simply random text. Here are some variations of pages available.

In the beginning temp=0 because no files text gets compare with other. And this is correct result. Then our algorithm will take line of file which is “lorem ipsum is simply dummy text” and take line no 1 of file 2 which is “lorem ipsum is not simply random text” and then apply lcs on both of these. It returns 5 as 5 strings are common and store this value in temp. in the next iteration it took second line of file2 which is ‘here are some variations of pages available” and apply lcn on them. It returns 0 as no string is common. And this inner loop terminates and result array has “lorem ipsum is simply text”. After first iteration outer loop takes line 2 of file and coampre it with whole file2 and store 5 in temp and finally the outer loop also terminates as file2 ended.

In the result array has “ lorem ipsum is simply text are variations of passages available”. And length =10.

So our algorithm is correct.