



UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES (UITs)

LAB REPORT - 4

IT-214 : ALGORITHM LAB

LCS, LIS, LPS

Submitted To:

Sumaiya Akhtar Mitu
Lecturer,
Department of IT, UITs

Submitted By:

Name: Nazmul Zaman
Student ID:2014755055
Department of IT UITs

24 MAY 2022

Contents

1	Abstrac	2
2	Introduction	2
3	Working methods	3
3.1	(LCS)	3
3.2	(LIS)	4
3.3	(LPS)	6
4	Conclusion	8
5	References	8

1 Abstrac

In mathematics and computer science, an algorithm is a finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing.

2 Introduction

In this task we are going to learn about LCS, LIS, LPS and there different methods.

LCS: The longest common subsequence (LCS) is defined as the longest subsequence that is common to all the given sequences, provided that the elements of the subsequence are not required to occupy consecutive positions within the original sequences.

LIS : Longest increasing subsequence problem is to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, lowest to highest, and in which the subsequence is as long as possible.

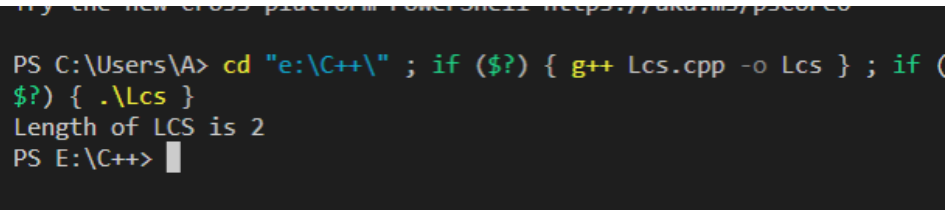
LPS : The longest palindromic substring problem is the problem of finding a maximum-length contiguous substring of a given string that is also a palindrome.

3 Working methods

3.1 (LCS)

```
1
2 //NAZMUL ZAMAN BSC IN (IT)
3
4 /* A Naive recursive implementation of LCS problem */
5 #include <bits/stdc++.h>
6 using namespace std;
7
8
9
10 /* Returns length of LCS for X[0..m-1], Y[0..n-1] */
11 int lcs( char *X, char *Y, int m, int n )
12 {
13     if (m == 0 || n == 0)
14         return 0;
15     if (X[m-1] == Y[n-1])
16         return 1 + lcs(X, Y, m-1, n-1);
17     else
18         return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
19 }
20
21
22
23 /* Driver code */
24 int main()
25 {
26     char X[] = "NAZMUL";
27     char Y[] = "ZAMAN";
28
29     int m = strlen(X);
30     int n = strlen(Y);
31
32     cout<<"Length of LCS is "<< lcs( X, Y, m, n ) ;
33
34     return 0;
35 }
36
37 // This code is contributed by rathbhupendra
```

OUTPUT ?



```

PS C:\Users\A> cd "e:\C++\" ; if ($?) { g++ Lcs.cpp -o Lcs } ; if ($?) { .\Lcs }
Length of LCS is 2
PS E:\C++>

```

Figure 1: OUTPUT

3.2 (LIS)

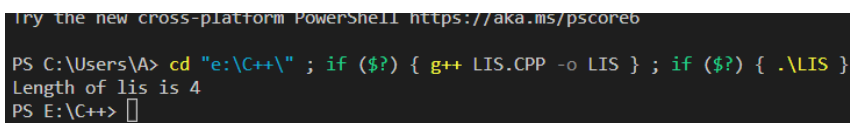
```

1  /* A Naive C++ recursive implementation
2  of LIS problem */
3  #include <iostream>
4  using namespace std;
5
6  /* To make use of recursive calls, this
7  function must return two things:
8  1) Length of LIS ending with element arr[n-1].
9     We use max_ending_here for this purpose
10  2) Overall maximum as the LIS may end with
11     an element before arr[n-1] max_ref is
12     used this purpose.
13  The value of LIS of full array of size n
14  is stored in *max_ref which is our final result
15  */
16  int _lis(int arr[], int n, int* max_ref)
17  {
18      /* Base case */
19      if (n == 1)
20          return 1;
21
22      // 'max_ending_here' is length of LIS
23      // ending with arr[n-1]
24      int res, max_ending_here = 1;
25
26      /* Recursively get all LIS ending with arr[0],
27      arr[1] ... arr[n-2]. If arr[i-1] is smaller
28      than arr[n-1], and max ending with arr[n-1]
29      needs to be updated, then update it */
30      for (int i = 1; i < n; i++) {
31          res = _lis(arr, i, max_ref);
32          if (arr[i - 1] < arr[n - 1]
33              && res + 1 > max_ending_here)
34              max_ending_here = res + 1;
35      }
36
37      // Compare max_ending_here with the overall
38      // max. And update the overall max if needed
39      if (*max_ref < max_ending_here)

```

```
40     *max_ref = max_ending_here;
41
42     // Return length of LIS ending with arr[n-1]
43     return max_ending_here;
44 }
45
46 // The wrapper function for _lis()
47 int lis(int arr[], int n)
48 {
49     // The max variable holds the result
50     int max = 1;
51
52     // The function _lis() stores its result in max
53     _lis(arr, n, &max);
54
55     // returns max
56     return max;
57 }
58
59 /* Driver program to test above function */
60 int main()
61 {
62     int arr[] = { 24, 22, 9, 53, 41, 52, 31, 70 };
63     int n = sizeof(arr) / sizeof(arr[0]);
64     cout << "Length of lis is " << lis(arr, n);
65     return 0;
66 }
67 // This code is contributed by shivanisinghss2110
```

?



```
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\A> cd "e:\C++\" ; if ($?) { g++ LIS.CPP -o LIS } ; if ($?) { .\LIS }
Length of lis is 4
PS E:\C++> 
```

Figure 2: OUTPUT

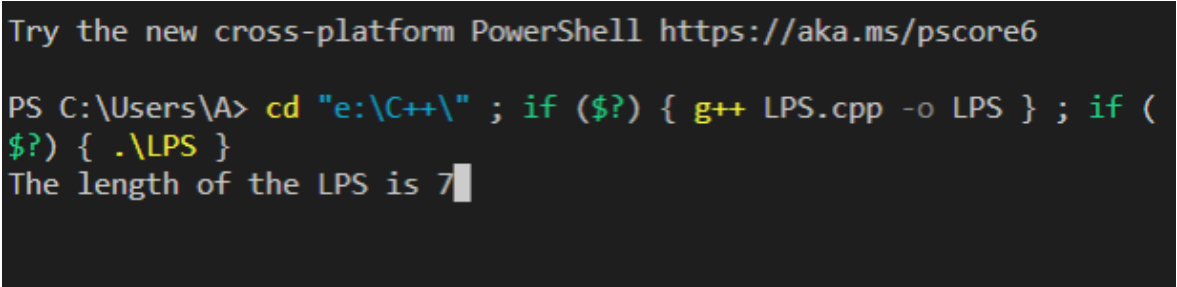
3.3 (LPS)

```

1 // NAZMUL ZAMAN BSC IN (IT)
2
3
4 // A Dynamic Programming based C++ program for LPS problem
5 // Returns the length of the longest palindromic subsequence in seq
6 #include<stdio.h>
7 #include<string.h>
8
9 // A utility function to get max of two integers
10 int max (int x, int y) { return (x > y)? x : y; }
11
12 // Returns the length of the longest palindromic subsequence in seq
13 int lps(char *str)
14 {
15     int n = strlen(str);
16     int i, j, cl;
17     int L[n][n]; // Create a table to store results of subproblems
18
19
20     // Strings of length 1 are palindrome of length 1
21     for (i = 0; i < n; i++)
22         L[i][i] = 1;
23
24
25     for (cl=2; cl<=n; cl++)
26     {
27         for (i=0; i<n-cl+1; i++)
28         {
29             j = i+cl-1;
30             if (str[i] == str[j] && cl == 2)
31                 L[i][j] = 2;
32             else if (str[i] == str[j])
33                 L[i][j] = L[i+1][j-1] + 2;
34             else
35                 L[i][j] = max(L[i][j-1], L[i+1][j]);
36         }
37     }
38
39     return L[0][n-1];
40 }
41
42 /* Driver program to test above functions */
43 int main()
44 {
45     char seq[] = "NAZMUL ZAMAN SHAEL";
46     int n = strlen(seq);
47     printf ("The length of the LPS is %d", lps(seq));
48     getchar();
49     return 0;
50 }

```

?



```
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\A> cd "e:\C++\" ; if ($?) { g++ LPS.cpp -o LPS } ; if (
$?) { .\LPS }
The length of the LPS is 7
```

Figure 3: OUTPUT

4 Conclusion

In this lab report we learn about different type of algorithm method that help us to find the common sub sequence, longest increasing sub sequence ,longest palindromic sub sequence from any string .

5 References

1.https://www.w3schools.com/algorithm/algorithm_intro.asp

2.https://www.javatpoint.com/algorithm_intro.asp

3.<https://github.com/ZamanNazmul>