**Report**

**Course code: CSE 214**

**CSE 215**

**Course title: Algorithm** **Submitted to:**

**Subroto Nag Pinku**

**Department of CSE**

**Daffodil International University**

**Submitted by:**

**Name: Zannatun Nayem Vasha**

**Id     :191-15-12939**

**Name: Bidyut sharma**

**Id:     191-15-12720**

**Name: Israt Jahan Esha**

**Id: 191-15-12921**

**Name: Md. Amirul Islam**

**Id :     191-15-12123**

# Comparative study on LIS and LCS

## Introduction:

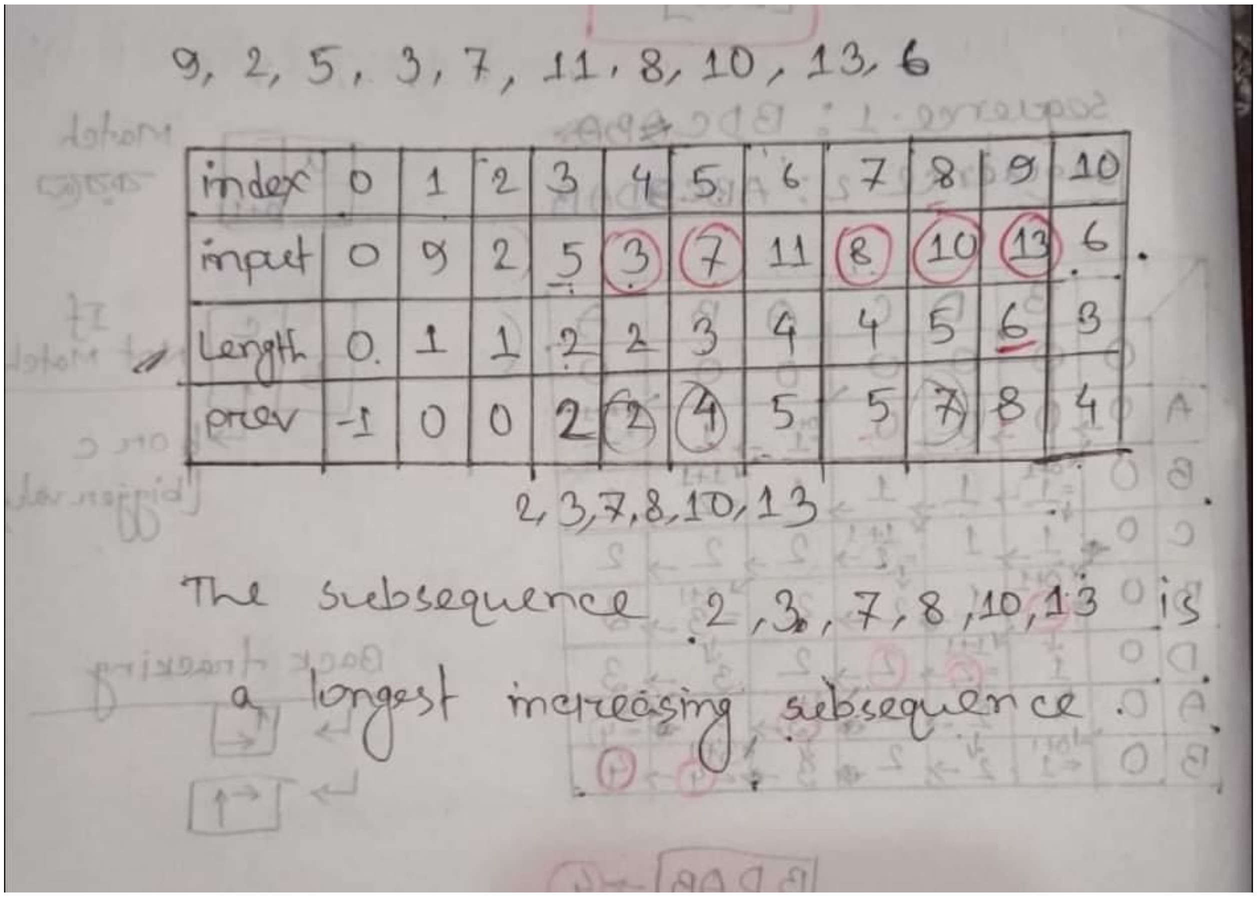
The longest Increasing subsequence (LIS) problem is to find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order. For example, the length of LIS for {10, 22, 9, 33, 21, 50, 41, 60, 80} is 6 and LIS is {10, 22, 33, 50, 60, 80}

The longest Common subsequence (LCS)  problem is the problem of finding the longest subsequence in a set of sequences. It differs from the longest common substring problem.

For example, LCS  input subsequence “ABCDGH” and “ AEDFHR” is “ADH” of length 3.

## Analysis:

LIS:



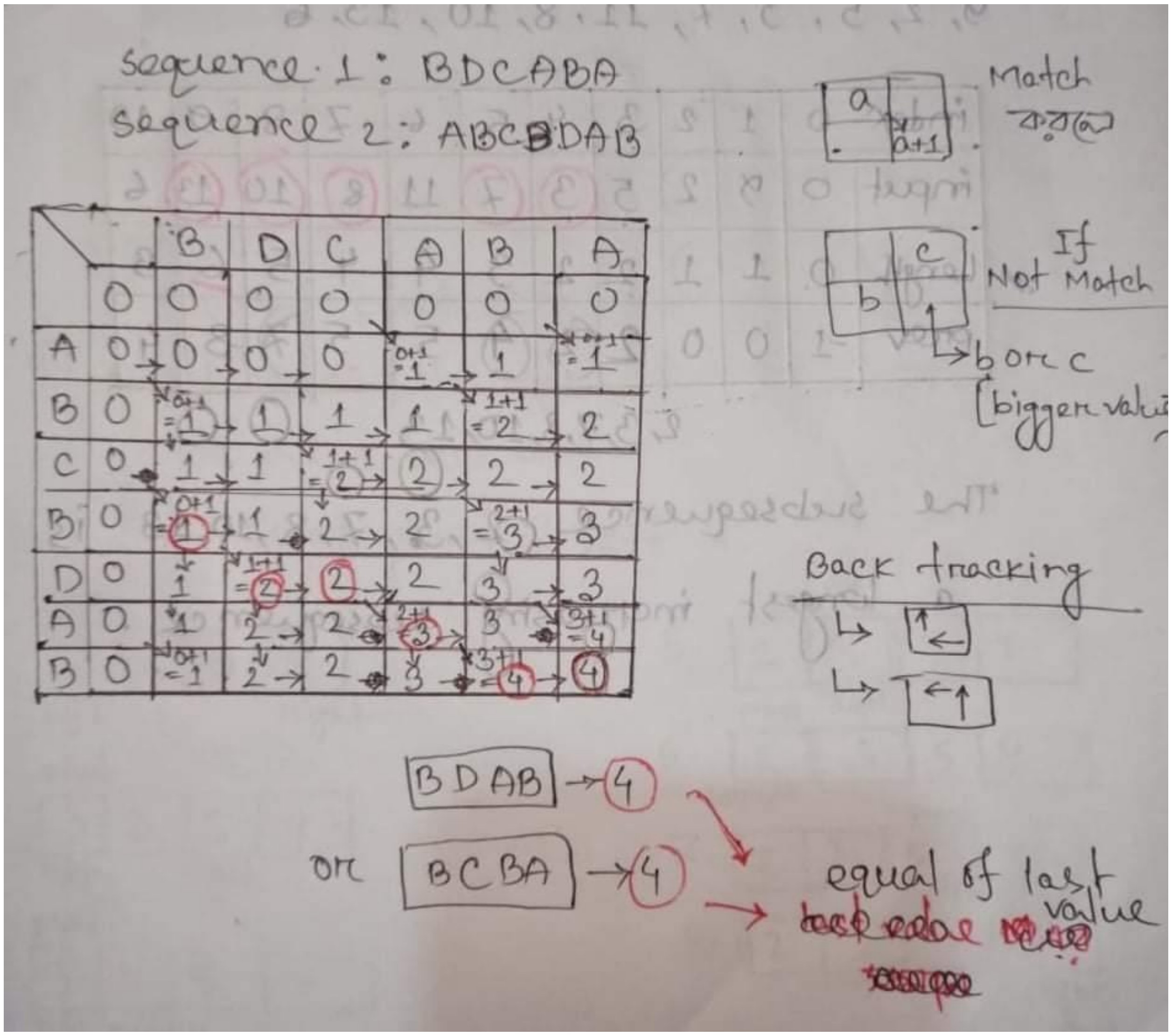
9, 2, 5, 3, 7, 11, 8, 10, 13, 6

| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| input | 0 | 9 | 2 | 5 | 3 | 7 | 11 | 8 | 10 | 13 | 6 |
| Length | 0 | 1 | 1 | 2 | 2 | 3 | 4 | 4 | 5 | 6 | 3 |
| prev | -1 | 0 | 0 | 2 | 2 | 4 | 5 | 5 | 7 | 8 | 4 |

2, 3, 7, 8, 10, 13

The subsequence 2, 3, 7, 8, 10, 13 is a longest increasing subsequence.

LCS:



|   |   | B | D | C | A | B | A |
|---|---|---|---|---|---|---|---|
|   | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 0 | 0 | 0 | 0+1 =1 | 1 | 0+1 =1 |
| B | 0 | 0+1 =1 | 1 | 1 | 1 | 1+1 =2 | 2 |
| C | 0 | 1 | 1 | 1+1 =2 | 2 | 2 | 2 |
| B | 0 | 0+1 =1 | 1 | 2 | 2 | 2+1 =3 | 3 |
| D | 0 | 1 | 1+1 =2 | 2 | 2 | 3 | 3 |
| A | 0 | 1 | 2 | 2 | 2+1 =3 | 3 | 3+1 =4 |
| B | 0 | 0+1 =1 | 2 | 2 | 3 | 3+1 =4 | 4 |

## Pseudocode:

LIS:

1. Describe an array of values we want to compute.

For 1 <= i <= n, let A(i) be the length of a longest increasing sequence of input.

2. Give a recurrence.

   For1 <= i <=n, A(i)= 1+ max{A(j)|1 <=j < i and input(j) < input(i)}

3. Compute the valus of A.
4. Find the optimal solution.

The following program uses A to compute an optimal solution. The first part computes a value m such that A(m) is the length of an optimal increasing subsequence of input. The second part computes an optimal increasing subsequence, but for convenience we print it out in reverse order. The program runs in time O(n), so the entire algorithm runs in time O(n^2).

LCS:

Take X = < x_1,...x_m> and Y = <y_1,…y_n> as input . stores c[I,j] into table c[0…m,0..n]in row major order. The array b[I,j] points to the table entry for optimal subproblem solution when computing c[I,j].

```
LCS-length(X,Y)

m<-length[x] n<-

length[y] for i <-1

to m c[I,0] <-0 for

j <-1 to n c[0,j] <-

0 for i <- 1 to m

for j <-1 to n if
```

```
(x_i == y_j){ c[i,j]
<-c[i-1,j-1]+1

b[i,j] < -NW } else if
(c[i-1,j]>=c[i,j-1]{ c[I,j]
<- c[i-1,j] b[I,j]<- N }
else{

c[i,j] <- c[i,j-1] b[i,j]
<- W
}
```

## **Time complexity:**

The general algorithms which are followed to solve the longest common subsequence (LCS) problems have both time complexity of space complexity of O(m*n). to reduce this complexity, two ways are proposed in this work.

Longest increasing subsequences(LIS) are studied in the context of various disciplines related mathematics, including algorithmics, random matrix theory, representation theory and physics. The longest increasing subsequence problem is solvable in time O(n log n), where n denotes the length of the input sequence.

**<u>Code:</u>**

LCS

```
#include<stdio.h>

#include<string.h>

Int I,j,m,n,c[20][20];

Char x[20],y[20],b[20][20];

Void print(int I,int j)

{

 If(i==0 || j==0)

 {

 Return;

 }

 If(b[i][j]=='c')

 {

 Print(i-1,j-1);

 Printf("%c",x[i-1]);

}

 Else if(b[i][j]=='u')
 {
 Print(i-1,j);
 }
 Else
 {
 Print(I,j-1);
 }
}
Void lcs()
{
 M=strlen(x);
 N=strlen(y);
 For(i=0; i<=m; i++)
 {
 C[i][0]=0;
 }
 For(i=0; i<=n; i++)
 {
```

```
C[0][i]=0;

}

For(i=1; i<=m; i++)

{

For(j=1; j<=n; j++)

{

If(x[i-1]==y[j-1])

{

C[i][j]=c[i-1][j-1]+1;

B[i][j]='c';

}

Else if(c[i-1][j]>=c[i][j-1])

{

C[i][j]=c[i-1][j];

B[i][j]='u';

}

Else

{
 C[i][j]=c[i][j-1];
 B[i][j]=’l’;
 }
 }
 }
}
Int main()
{
 Printf(“Enter 1st sequence:”);
 Scanf(“%s”,x);
 Printf(“Enter 2nd sequence:”);
 Scanf(“%s”,y);
 Printf(“\nThe Longest Common Subsequence is “);
 Lcs();
 Print(m,n);
 Return 0;
}
```

**Lis**

```
#include <stdio.h>

#include <stdlib.h>

#define INT_INF 10000

Int search_replace(int *lis, int left, int right, int key)

{

 Int mid;

 For (mid = (left+right)/2; left <= right; mid = (left+right)/2)

 {

 If (lis[mid] > key)

 {

 Right = mid – 1;
 }
 Else if (lis[mid] == key)
 {
Return mid;
}
Else if (mid+1 <= right && lis[mid+1] >= key)
 {
 Lis[mid+1] = key;
 Return mid+1;
 }
 Else
 {
 Left = mid + 1;
 }
 }
 If (mid == left)
 {
 Lis[mid] = key;
 Return mid;

 }

 Lis[mid+1] = key;

 Return mid+1;

}

Int main(void)

{

 Int I, tmp,lis_length = -1;

 Int *answer;

 Int A[100];

 Int size;

 Printf(“Enter Total Number of Element : “);

 Scanf(“%d”,&size);

 Printf(“Enter Elements Sequence :\n”);

 Int LIS[size],index[100];

 For(int i=0; i<size; i++)

 {

 Scanf(“%d”,&A[i]);
 }
 LIS[0] = A[0];
 For (I = 1; I < size; ++i)
{
LIS[i] = INT_INF;
}
 For (I = 1; I < size; ++i)
 {
 Index[i] = search_replace(LIS, 0, I, A[i]);
 If (lis_length < index[i])
 {
 Lis_length = index[i];
 }
 }
 Answer = (int*) malloc((lis_length+1) * sizeof(int));
 For (I = size-1, tmp = lis_length; I >= 0; --i)
 {
 If (index[i] == tmp)
 {
 Answer[tmp] = A[i];
 --tmp;
}
}
Printf("LIS: ");
 For (I = 0; I < lis_length+1; ++i)
 {
 Printf("%d ", answer[i]);
 }
 Printf("\n");
 Return 0;
}
```