Name: 陳玉山
ID： 0710185

Report Homework 6

I. Introduction

- In this homework, we have to implement an algorithm to find out the Longest Common Subsequences.

II. Methodology

- There are three main parts in this homework, building table c, table b, and print out process.
- Table c is used to track the length of the common subsequence of the two sequences.
- During building table c, we also can build table b which contains 3 kinds of arrows, left ($\boxed{\leftarrow}$), up($\boxed{\uparrow}$), and up-left ($\boxed{\nwarrow}$).
- The final function is print, this function will take advantage of table c to find out the common number recursively. By comparing the value of b[i][j] with up-left, if they are equal, it means that x[i] is a common number.

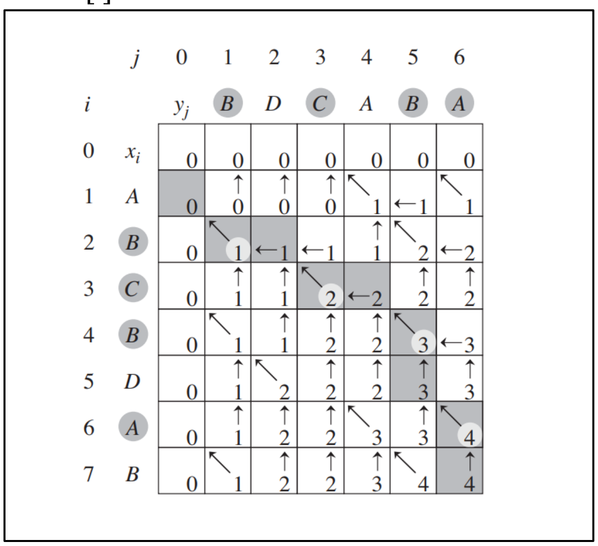

Figure 1: The figure shows how the algorithm works.

Function Print()

```
void print_out(int i, int j){
    if (i==0 || j==0)
        return;
    if (b[i][j]==upleft)
    {
        print_out(i-1,j-1);
        if ((i==1 && flag == false) || (j==1 && flag ==false))
            {
                cout << x[0] << " ";
                flag = true;
                cout << x[i] << " ";
            }

        else
            cout << x[i] << " ";
    }

    else if (b[i][j]==up)
        print_out(i-1,j);
    else if (b[i][j] == _left)
        print_out(i,j-1);

    }
```

Figure 2: Print_out function.

Name: 陳玉山
ID： 0710185

Building table c and b

```
void initial(){
    for (int i=0; i<=n; i++)
    {
        c[i][0] = 0;
    }
    for (int j=0;j<=m; j++)
    {
        c[0][j] = 0;
    }

    for (int i =1; i<=n; i++)
    {
        for (int j=1; j<=m;j++)
        {
            if(x[i]==y[j])
            {
                c[i][j] = c[i-1][j-1]+1;
                b[i][j] = upleft;
            }
            else if(c[i-1][j]>=c[i][j-1])
            {
                c[i][j] = c[i-1][j];
                b[i][j] = up;
            }
            else
            {
                c[i][j] = c[i][j-1];
                b[i][j] = _left;
            }
        }
```

Figure 3: Building tables b and c.

Results

```
X = 1 4 6 2 8 1 5 6
Y = 1 6 9 8 5 1 6 3 2
Z= 1 6 8 1 6
Length = 5
```

```
X = 6 6 6 0 8 9 4 5 7 9 4 5 3
Y = 1 6 8 0 4 8 7 5 4 8 7 7 4 1 4
Z= 6 0 8 4 7 4
Length = 6
```

Name: 陳玉山
ID：0710185

```
X = 4 6
Y = 6 0
Z=
Length = 0
Running time (n = 2) (m = 2) = 3545 microseconds
```

```
X = 9 1 0
Y = 0 4 0
Z= 0
Length = 1
Running time (n = 3) (m = 3) = 1994 microseconds
```

```
X = 6 0
Y = 6 5 6 4 8 2 0
Z= 6 0
Length = 2
Running time (n = 2) (m = 7) = 3921 microseconds
```

```
X = 8 7 1 1 9 0 2 7 6 0
Y = 3 5 0 5 8 7 0
Z= 0 7 0
Length = 3
Running time (n = 10) (m = 7) = 4102 microseconds
```

```
X = 6 2 7 7 4 8 5 8 8 0
Y = 0 5 7 0 3 0
Z= 7 0
Length = 2
Running time (n = 10) (m = 6) = 4985 microseconds
```

```
X = 4 4 6 5 3 1 6 4 4 6 0
Y = 1 9 9 7 3 7 5 2 8 0 0
Z= 5 0
Length = 2
Running time (n = 11) (m = 11) = 6096 microseconds
```

Name: 陳玉山
ID：0710185

From the results above, the longest time of running is when n=11 and m=11 with two numbers in common. The fastest result is n=3, m = 3 with 1 in number in common. Although the lower number of n and m could send out a shorter time of finding, the two sequences that have no number in common also spend quite a bit long time to finish the algorithm.

- With the help of this algorithm, the time complexity is decreasing from $O(2^n)$ to $O(m*n)$.