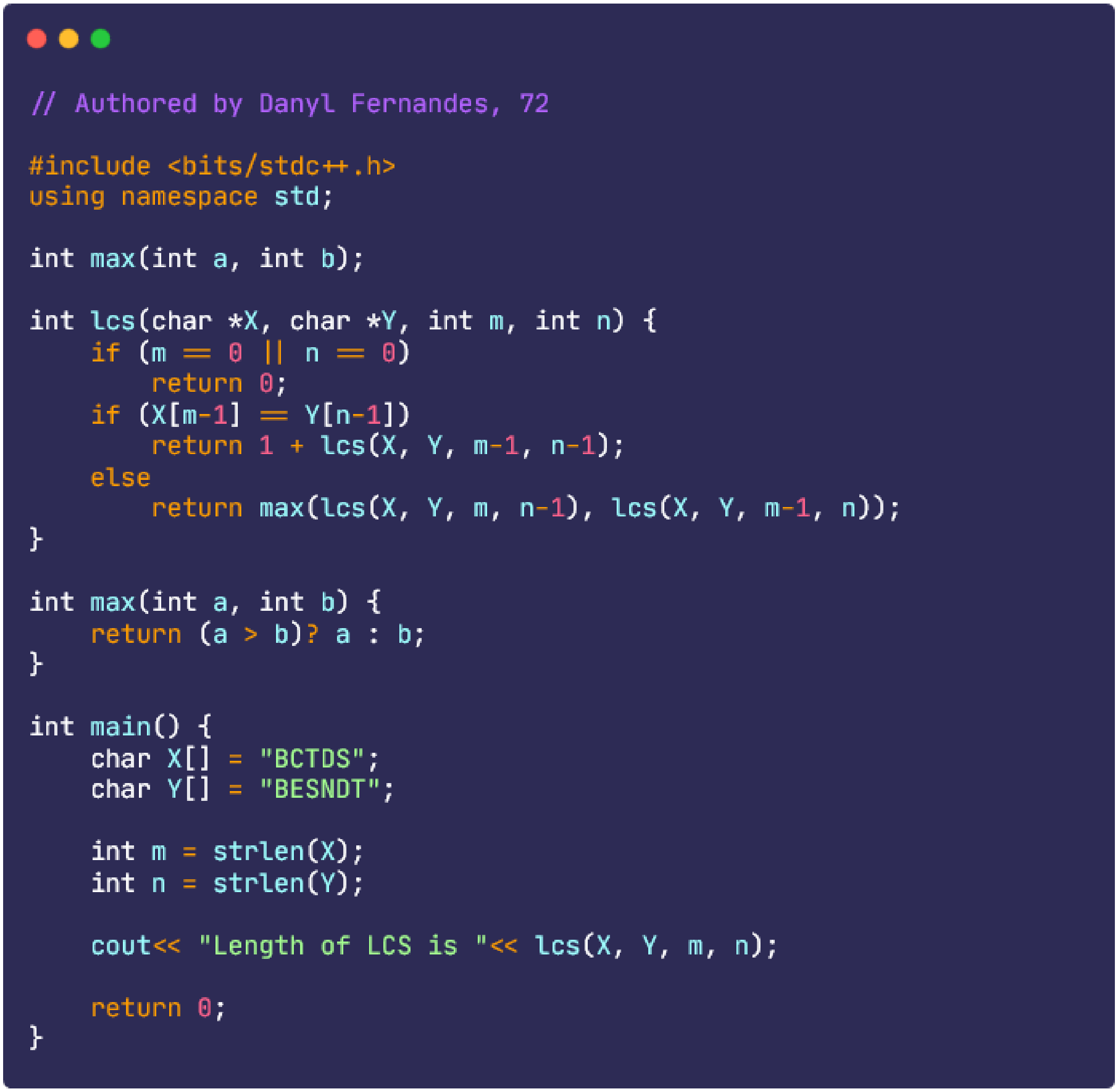
Danyl Fernandes
2020012004 (72)
14-04-2021

AOA Experiment 6

Aim:

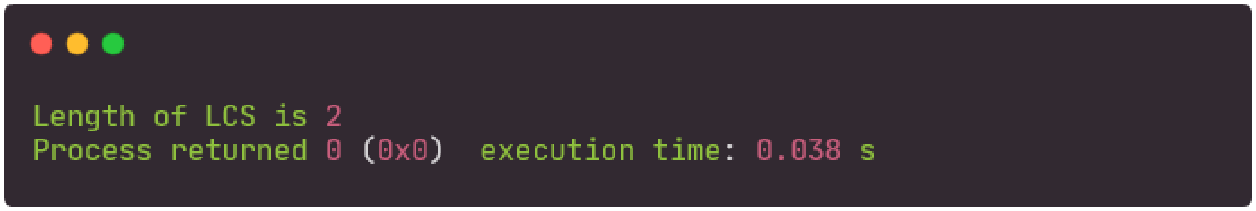
To implement & analyze Longest Common Subsequence algorithm:

Implementation:



```cpp
// Authored by Danyl Fernandes, 72

#include <bits/stdc++.h>
using namespace std;

int max(int a, int b);

int lcs(char *X, char *Y, int m, int n) {
    if (m == 0 || n == 0)
        return 0;
    if (X[m-1] == Y[n-1])
        return 1 + lcs(X, Y, m-1, n-1);
    else
        return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
}

int max(int a, int b) {
    return (a > b)? a : b;
}

int main() {
    char X[] = "BCTDS";
    char Y[] = "BESNDT";

    int m = strlen(X);
    int n = strlen(Y);

    cout<< "Length of LCS is "<< lcs(X, Y, m, n);

    return 0;
}
```

Output:



```
Length of LCS is 2
Process returned 0 (0x0)  execution time: 0.038 s
```

Danyl Fernandes
2020012004 (72)

Exp 06

Theory:

- It is the problem of finding the largest subsequence common to all sequences in a set of sequences

- It differs from the longest common substrin problem unlike substrings, subsequences are not required to occupy consequtive positions with in the original sequence.

- It starts comparing strings in reverse order one character at a time.

Now, we have 2 cases:

Case 1: If both character are samethen add 1 to the result & remove the last character from both the strings and make recursive call to the modified strings.

Case 2: If both characters different then remove the last character of string 1 & make a recursive call & remove the last character from string 2 & make a recursive & then return the max of both recursive calls.

Danyl Fernandes
2020012004 (72)

X = ABCBDAB
Y = BDCABA

i ↓	j →	0	1	2	3	4	5	6	
		yi	B	D	C	A	B	A	
0	xi	0	0	0	0	0	0	0	
1	A	0	0↑	0↑	0↑	1↖	1←	1↖	
2	B	0	1↖	1←	1←	1↑	2↖	2←	B
3	C	0	1↖	1↑	2↖	2←	2↑	2↑	C
4	B	0	1↑	1↑	2↑	2↑	3↖	3←	B
5	D	0	1↑	2↖	2↑	2↑	3↑	3↑	
6	A	0	1↑	2↑	2↑	3↖	3↑	4↖	A
7	B	0	1↖	2↑	2↑	3↑	4↖	4↑	

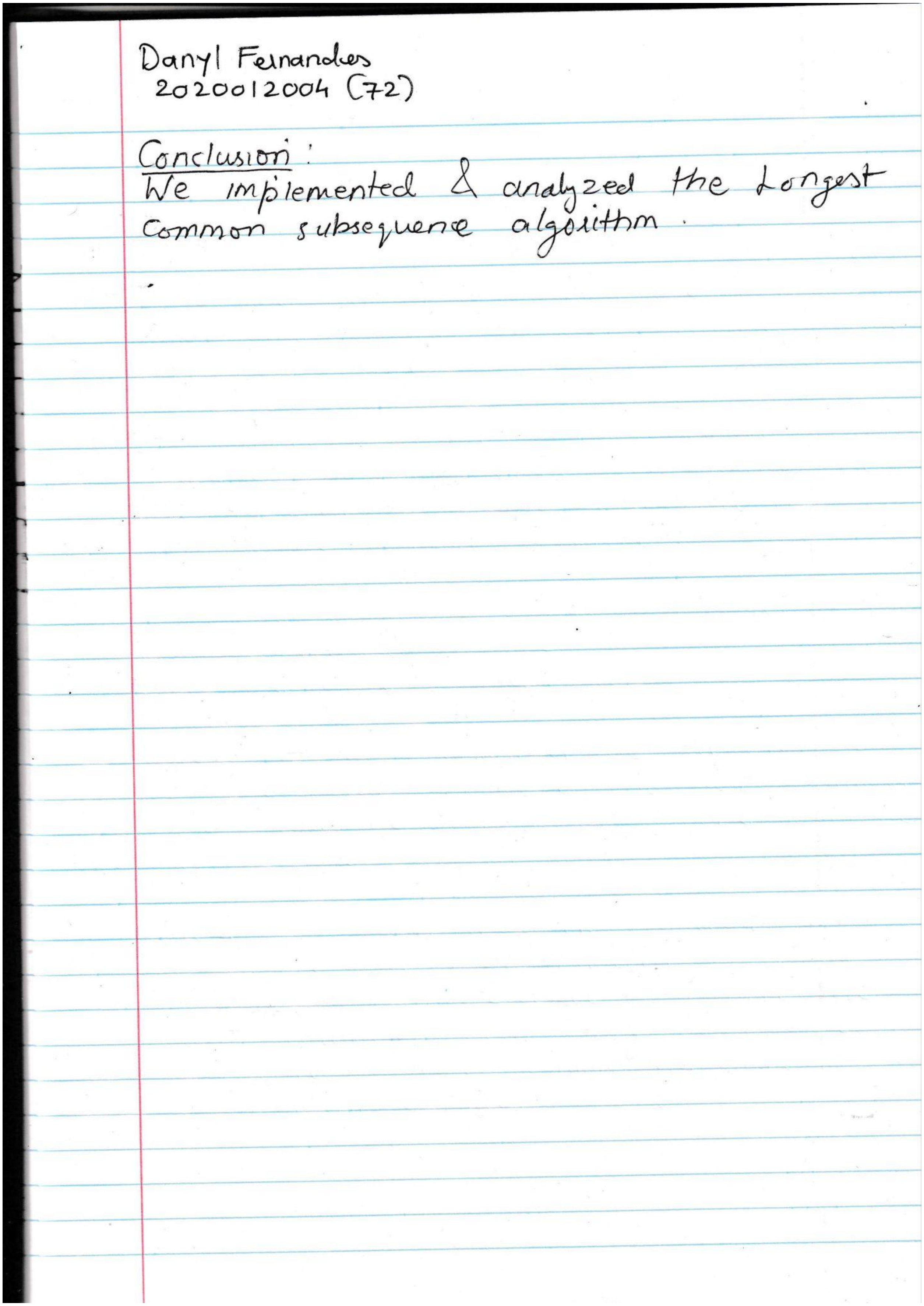
The LCS is BCBA

LCS Complexity:

- In brute force, we need to perform checks on every Subsequene of P[1 .. m] to see if it is also a subsequence of Q[1 .. n] checking membeship of one subsequnce P[1 .. m] into Q[1 .. n] takes O(n) time

Thus worse case time complexity is ~~$O(n^2$~~ $O(2^n)$

- In dynamic Programming, the only table of size mxn is filled up using nested for loops. So running time of dynam approach would take O(m, n)



Danyl Fernandes
2020012004 (72)

Conclusion :
We implemented & analyzed the Longest common subsequene algorithm.