ASSIGNMENT #5 – Comparing LCS and LPS

C Code of Longest Common Subsequence

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
void LCS() {
	string x = " ABCBDAB";
	string y = " BDCABA";
	vector < vector <int> > C(x.size(), vector <int>(y.size())),
movement(x.size(), vector <int>(y.size()));
	int m = x.size();
	int n = y.size();
	for (int i = 0; i < m; i++)
		C[i][0] = 0;
	for (int j = 0; j < n; j++)
		C[0][j] = 0;
	for (int i = 1; i < m; i++) {
		for (int j = 1; j < n; j++) {
			if (x[i] == y[j]) {
				C[i][j] = C[i - 1][j - 1] + 1;
				movement[i][j] = 2; //Recording the movement
to backtrack later on
			}
			else {
				if (C[i - 1][j] >= C[i][j - 1]) {
					C[i][j] = C[i - 1][j];
					movement[i][j] = 3; //Recording the
movement to backtrack later on
				}
				else {
					C[i][j] = C[i][j - 1];
					movement[i][j] = 1; //Recording the
movement to backtrack later on
				}
			}
		}
	}
	for (int i = 0; i < C.size(); i++) {  //displaying the matrix
		for (int j = 0; j < C[i].size(); j++) {
			cout << C[i][j] << " \t";
		}
		cout << endl;
	}
	cout << "\n\nLCS result : " << C[m - 1][n - 1] << endl;
//Displaying the bottom right corner value
	vector <char> answer;
	int i = m - 1;
	int j = n - 1;
	bool found = false;
	while (!(i <= 0 && j <=0)) { //Finding out the letters for LCS
		if (found == true)
			break;
		switch (movement[i][j]) {
		case 2:  //2 : left-top corner
			if (x[i] == y[j]) {
				answer.push_back(x[i]);
			}
			i = i - 1;
			j = j - 1;
			break;
		case 3: //3 : up
			if (x[i] == y[j]) {
				answer.push_back(x[i]);
			}
			i = i - 1;
			break;
		case 1: // 1 : left
```

```
                        if (x[i] == y[j]) {
                                answer.push_back(x[i]);
                        }
                        j = j - 1;
                        break;
                default:
                        found = true;
                        break;
                }
        }
        cout << answer.size()<< " - LCS output : ";
        for (auto it = answer.rbegin(); it != answer.rend(); it++) {  //
Outputting the vector from reverse to get the correct answer
                cout << *it;
        }
}
int main()
{
        LCS();
}
```

C Code of Longest Palindromic Subsequence

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

void display(std::vector < std::vector<int> > v) {
        for (int i = 0; i < v.size(); i++) {
                for (int j = 0; j < v[i].size(); j++) { //Printing the row
of the d table
                        std::cout << v[i][j] << "\t";
                }
                std::cout << "\n";
        }
}
int max(int a, int b) {
        if (a < b)
                return b;
        return a;
}
void polindromic(std::string text, std::vector < std::vector <int>  > L) {
        for (int i = 0; i < L.size(); i++) {
                for (int j = 0; j < L[i].size(); j++) {
                        if (i == j)
                                L[i][j] = 1;
                }
        }
        int i = 0, j = 1, counter = 2;
        while (true) {
                if (i == L.size())
                        break;
                if (j == L[i].size())
                {
                        if (counter == L[i].size())
                                break;
                        j = counter;
                        counter++;
                        i = 0;
                }
                if (text[i] != text[j])
                        L[i][j] = max(L[i + 1][j], L[i][j - 1]);
                else if ((text[i] == text[j]) && (j == i + 1))
                        L[i][j] = 2;
                else
                        L[i][j] = L[i + 1][j - 1] + 2;
```

```
                i++;
                j++;
        }
        display(L);
        std::cout << "\t\n\nResult = " << L[0][L[0].size() - 1] << "\n";
        i = 0;
        j = L[0].size() - 1;
        std::cout << "Polindromic = ";
        while (true) {
                if (L[i + 1][j] == L[i][j - 1] && L[i + 1][j] > L[i][j - 1])
{
                        i++;
                }
                else if (L[i + 1][j] < L[i][j - 1]){
                        j--;
                }
                else {
                        i++;
                        j--;
                }
                std::cout << text[i];
                if (L[i][j] == 0)
                        break;
        }
}
int main()
{
        string text = "ABCBDAB";
        vector < std::vector <int>  > L (text.size(), std::vector <int>
(text.size()));
        polindromic(text, L);
}
```

Big O notation LCS

```
8   string x = "ABCBDAB";
9   string y = "BDCABA";
10  vector < vector <int> > C(x.size(), vector <int>(y.size())), movement(x.size(), vector <int>(y.size()));
11  int m = x.size();
12  int n = y.size();
13
14  for (int i = 0; i < m; i++)
15    C[i][0] = 0;
16  for (int j = 0; j < n; j++)
17    C[0][j] = 0;
18
19  for (int i = 1; i < m; i++) {
20    for (int j = 1; j < n; j++) {
21      if (x[i] == y[j]) {
22        C[i][j] = C[i - 1][j - 1] + 1;
23        movement[i][j] = 2; //Recording the movement to backtrack later on
24      }
25      else {
26        if (C[i - 1][j] >= C[i][j - 1]) {
27          C[i][j] = C[i - 1][j];
28          movement[i][j] = 3; //Recording the movement to backtrack later on
29        }
30        else {
31          C[i][j] = C[i][j - 1];
32          movement[i][j] = 1; //Recording the movement to backtrack later on
33        }
34      }
35    }
36  }
```

O(n)

O(n)

O(2n) => O (n)

$O(n^2) \leq O(n+n^2) \leq O(n^2 + n^2)$

O(2n) => O (n) + $O(n^2)$ = $O(n + n^2)$

$O(n + n^2) \rightarrow O(n^2)$

$O(n^2)$

```
38  for (int i = 0; i < C.size(); i++) {  //displaying the matrix
39    for (int j = 0; j < C[i].size(); j++) {
40      cout << C[i][j] << " \t";
41    }
42    cout << endl;
43  }
```

$O(n^2)$

```
51  while (!(i <= 0 && j <=0)) { //Finding out the letters for LCS
52    if (found == true)
53      break;
54    switch (movement[i][j]) {
55    case 2:  //2 : left-top corner
56      if (x[i] == y[j]) {
57        answer.push_back(x[i]);
58      }
59      i = i - 1;
60      j = j - 1;
61      break;
62    case 3: //3 : up
63      if (x[i] == y[j]) {
64        answer.push_back(x[i]);
65      }
66      i = i - 1;
67      break;
68    case 1: // 1 : left
69      if (x[i] == y[j]) {
70        answer.push_back(x[i]);
71      }
72      j = j - 1;
73      break;
74    default:
75      found = true;
76      break;
77    }
78  }
```

$O(n^2)$

Finally, Big O Notation result for LCS = $O(n^2)$

Big O notation LPS

```
8   void display(std::vector < std::vector<int> > v) {
9
10    for (int i = 0; i < v.size(); i++) {
11      for (int j = 0; j < v[i].size(); j++) { // Printing the row of the d table
12          std::cout << v[i][j] << "\t";
13      }
14      std::cout << "\n";
15    }
```

$O(n^2)$

```
23  void polindromic(std::string text, std::vector < std::vector <int>  > L) {
24
25    for (int i = 0; i < L.size(); i++) {
26      for (int j = 0; j < L[i].size(); j++) {
27        if (i == j)
28          L[i][j] = 1;
29      }
30    }
31
```

$O(n^2)$

```
34    while (true) {
35      if (i == L.size())
36        break;
37      if (j == L[i].size())
38      {
39        if (counter == L[i].size())
40          break;
41        j = counter;
42        counter++;
43        i = 0;
44      }
45      if (text[i] != text[j])
46        L[i][j] = max(L[i + 1][j], L[i][j - 1]);
47      else if ((text[i] == text[j]) && (j == i + 1))
48        L[i][j] = 2;
49      else
50        L[i][j] = L[i + 1][j - 1] + 2;
51      i++;
52      j++;
53    }
```

$O(n^2)$

```
61    while (true) {
62      if (L[i + 1][j] == L[i][j - 1] && L[i + 1][j] > L[i][j - 1]) {
63        i++;
64      }
65      else if (L[i + 1][j] < L[i][j - 1]){
66        j--;
67      }
68      else {
69        i++;
70        j--;
71      }
72      std::cout << text[i];
73      if (L[i][j] == 0)
74        break;
75    }
```

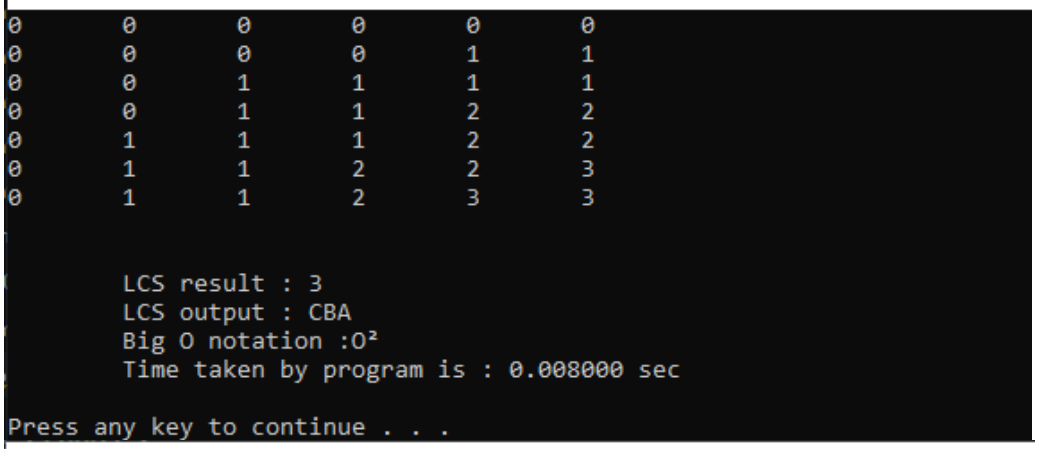
$O(n^2)$

$O(4n^2) \Rightarrow O(n^2)$

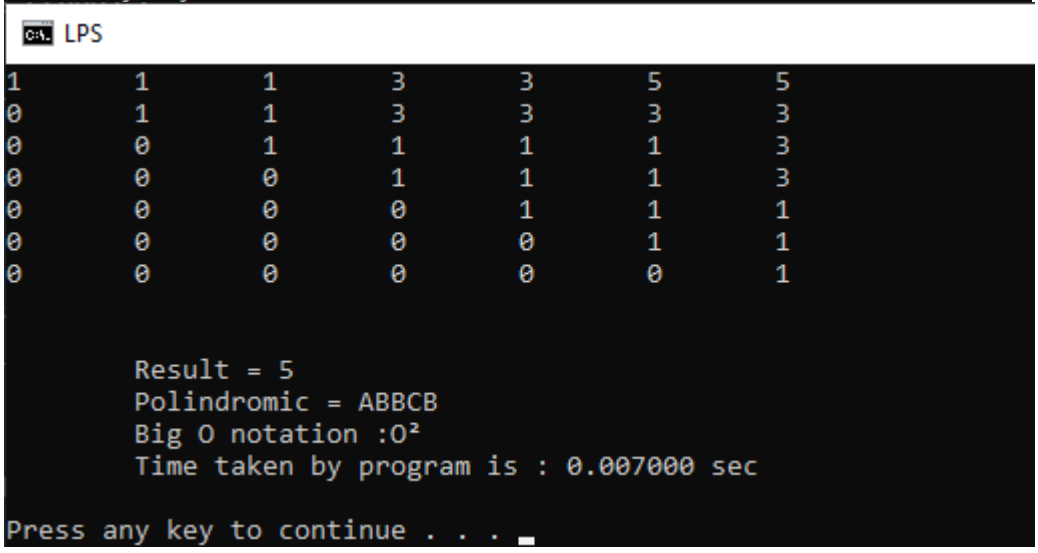
Finally, Big O Notation result for LPS = $O(n^2)$

Output of Program

LCS



LPS



[Download the C Code File (until 3 January.)]