
알 고 리 즘

Homework 3

Bottom-Up Dynamic Programming



Professor	황호영 교수님
Department	컴퓨터공학과
Student ID	2012722028
Name	장 한 별
Date	2018. 11. 04.

Introduction.

이번 프로젝트의 목표는 dynamic programming algorithm을 이해하는 것이다.
주어진 2가지 sequence 의 가장 긴 공통 subsequence 인 LCS 의 길이를 구하는 방법과
LCS가 어떻게 이루어지는지 bottom-up dynamic programming algorithm 으로 파악해
직접 구현해서 이를 확인하도록 한다.

1. To compute the length of the longest common subsequence(LCS) of two sequences in $\theta(mn)$ space, find a bottom-up dynamic programming algorithm.

(a) Write your pseudocode.

```
x <- 1st Sequence
y <- 2nd Sequence
ComputeLength( x , y)
  m <- x.length
  n <- y.length
  i <- 1, j <- 1
  while( j <= n )
  {
    While( i <= m )
    {
      If ( x[i] = y[j])
        c[j][i] <- c[j-1][i-1] + 1
      else
        c[j][i] <- max { c[j-1][i] , c[j][i-1] }
    }
  }
print( c[n][m] )
```

(b) Write your program with your comments.

```
prob3-1.cpp* x
prob3-1 (전역 범위)
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int ComputeLength(string x, string y);    // LCS의 길이를 계산하는 함수
6
7  int main(void)
8  {
9      string x = "";    // string형 변수 선언 (Sequence 2개)
10     string y = "";
11
12     cout << "Enter 1st Sequence:" << endl;
13     getline(cin, x);    // 첫 번째 Sequence 입력받아 x 에 저장
14     x = " " + x;    // table 가장자리를 0으로 채우기 위해 공백문자 삽입
15     cout << "Enter 2nd Sequence:" << endl;
16     getline(cin, y);    // 두 번째 Sequence 입력받아 y 에 저장
17     y = " " + y;    // 공백문자 삽입
18
19     cout << endl;
20     cout << "LCS Length: " << ComputeLength(x, y) << endl;    // LCS의 길이 출력
21     return 0;
22 }
23
24 int ComputeLength(string x, string y)
25 {
26     int m = 0, n = 0;    // Sequence 의 길이를 넣을 변수 선언 및 초기화
27     int i = 1, j = 1;    // Sequence 의 각 요소들을 순차적으로 비교하기 위한 변수
28     m = x.length() - 1;    // main 함수에서 공백 문자를 추가했으므로, string 길이에서 -1
29     n = y.length() - 1;
30
31     int c[100][100] = { 0, };    // 100 X 100 table, 모든 요소를 0으로 초기화
32
33     for (int a = 0; a <= m; a++)
34         cout << c[0][a] << " ";    // 1행 0으로 채움(출력)
35     cout << endl;
36
37     for (j = 1; j <= n; j++)    // y 요소 하나씩 비교
38     {
39         cout << c[j][0] << " ";    // 1열 0으로 채움(출력)
40         for (i = 1; i <= m; i++)    // x 요소 하나씩 비교
41         {
42             if (x[i] == y[j])    // 같은 문자라면,
43             {
44                 c[j][i] = c[j-1][i - 1] + 1;    // 현재 위치에서 행-1, 열-1 값에서 +1 한 후 넣어줌
45                 cout << c[j][i] << " ";
46             }
47         }
48     }
49 }
```

```

47         else // 다른 문자라면, 현재 값의 위 값과 왼쪽 값 중 큰 값을 넣어줌
48         {
49             if (c[j-1][i] > c[j][i-1]) // 위 값이 왼쪽 값보다 클 때
50             {
51                 c[j][i] = c[j-1][i]; // 위 값(현재 위치에서 행-1 값)을 넣어줌
52                 cout << c[j][i] << " ";
53             }
54             else // 왼쪽 값이 위 값보다 클 때 or 같을 때
55             {
56                 c[j][i] = c[j][i-1]; // 왼쪽 값(현재위치에서 열-1)을 넣어줌
57                 cout << c[j][i] << " ";
58             }
59         }
60     }
61     cout << endl; // y요소가 다음 문자로 넘어가기전 줄바꿈
62 }
63 cout << endl;
64
65 return c[n][m]; // LCS의 길이 반환
66 }
67

```

(c) For at least one example, show the simulation results.

Microsoft Visual Studio 디버그 콘솔	Microsoft Visual Studio 디버그 콘솔
<pre> Enter 1st Sequence: ABCBADAB Enter 2nd Sequence: BDCABA 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 2 2 2 0 0 1 2 2 2 2 2 0 1 1 2 2 2 3 3 0 1 2 2 3 3 3 4 0 1 2 2 3 3 4 4 LCS Length: 4 C:\Users\stanley\source\repos\prob3-1\ 이 창을 닫으려면 아무 키나 누르세요. </pre>	<pre> Enter 1st Sequence: BDCABA Enter 2nd Sequence: ABCBADAB 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 2 2 2 0 1 1 2 2 2 2 2 0 1 1 2 2 3 3 3 0 1 2 2 2 3 3 3 0 1 2 2 3 3 4 4 0 1 2 2 3 4 4 4 LCS Length: 4 C:\Users\stanley\source\repos\prob3-1\ 이 창을 닫으려면 아무 키나 누르세요. </pre>

(d) Explain the program and the simulation results (at least five lines).

table을 구성하는 algorithm은 재귀적으로 함수를 호출하는 방법, 반복문을 이중으로 사용하는 방법으로 크게 2가지가 있는데 필자는 후자를 선택했다. 이중 for문을 사용해서 algorithm을 직관적으로 이해할 수 있기 때문이다. 먼저 string형 변수 x, y를 각각 첫번째, 두번째 Sequence를 입력받은 후, LCS의 길이를 계산하는 함수인 ComputeLength() 함수에 인자로 넣어 호출한다. (단순히 LCS의 길이만 출력하는 것이 아닌 table 전체를 확인하기 위해 table 상태도 같이 출력한다. 이때, table의 최대 크기는 100 X 100)

먼저, 1행을 모두 0으로 채운다. 그 후, x의 문자들과 y의 문자들을 하나씩 비교해야하는데 y의 첫번째 문자를 x의 첫번째문자부터 끝까지, 그 다음 y의 두번째 문자를 다시 x의 첫번째문자부터 끝까지 비교해 y의 마지막 문자와 x의 마지막 문자를 비교할 때 까지 수행한다. 이를 위해 필자

이중 for문을 사용했다. 문자를 비교중에 같은 문자를 찾는다면 현재 i 와 j 에서 각각 1을 빼서 i-1, j-1 좌표(즉, 왼쪽 위 대각선 위치) 의 값에서 +1을 한후, (i, j) 좌표값에 넣어준다. 다른 문자라면 현재 좌표에서 왼쪽 값과 위 값을 비교해 둘 중 큰 값을 넣어준다. 이렇게 한 단계씩 수행할 때 마다 해당 좌표의 값을 출력해 나간다면 두 문자열의 끝에 이르렀을 때 table이 완성된다. 그 끝의 값이 바로 LCS의 길이가 되어, 이 값을 반환시켜 main() 함수에서 출력했다.

	A	B	C	B	D	A	B
B	0	0	0	0	0	0	0
D	0	0	1	1	1	2	2
C	0	0	1	2	2	2	2
A	0	1	1	2	2	3	3
B	0	1	2	2	3	3	4
A	0	1	2	2	3	3	4

강의자료에 예시처럼 두 sequence에 'ABCBADAB'와 'BDCABA'를 입력했다. table 의 상태와 LCS의 길이인 4 모두 문제없이 출력됨을 확인할 수 있었고, 이 두 sequence의 순서를 반대로 입력했을 때에도 역시 문제없이 LCS의 길이인 4가 출력됨을 확인할 수 있었다.

2. Using the results of Problem 1, find an algorithm to print an LCS of two sequences in $O(m+n)$ time.

(a) Write your pseudocode.

```

x <- 1st Sequence
y <- 2nd Sequence
ComputeLength( x , y)

i <- x.length, j <- y.length

while
  if ( i = 0 or j = 0 )
    break
  if ( c[j][i] = c[j][i-1] )
    i <- i -1, continue
  else if ( c[j][i] = c[j-1][i] )
    j <- j -1, continue
  else

```

```

LCS <- LCS + x[i]
i <- i -1, j <- j -1
continue

```

```

LCS <- Reverse( LCS )
Print( LCS )

```

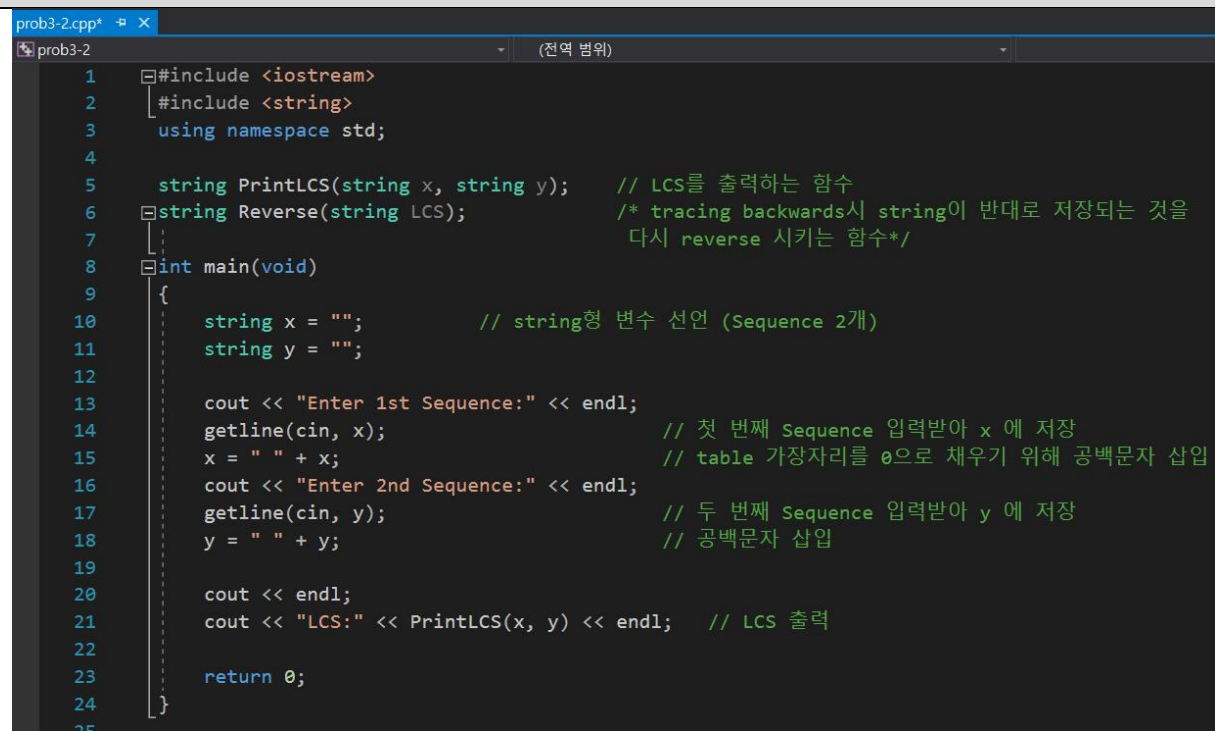
Reverse (string)

```

len = string.length
While ( i <= len )
    temp <- temp + LCS[ len - i ]
    i <- i +1
return temp

```

(b) Write your program with your comments.



```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  string PrintLCS(string x, string y);    // LCS를 출력하는 함수
6  string Reverse(string LCS);           /* tracing backwards시 string이 반대로 저장되는 것을
7                                         다시 reverse 시키는 함수*/
8  int main(void)
9  {
10     string x = "";                    // string형 변수 선언 (Sequence 2개)
11     string y = "";
12
13     cout << "Enter 1st Sequence:" << endl;
14     getline(cin, x);                  // 첫 번째 Sequence 입력받아 x 에 저장
15     x = " " + x;                      // table 가장자리를 0으로 채우기 위해 공백문자 삽입
16     cout << "Enter 2nd Sequence:" << endl;
17     getline(cin, y);                  // 두 번째 Sequence 입력받아 y 에 저장
18     y = " " + y;                      // 공백문자 삽입
19
20     cout << endl;
21     cout << "LCS:" << PrintLCS(x, y) << endl;    // LCS 출력
22
23     return 0;
24 }
25

```

```

26 string PrintLCS(string x, string y)
27 {
28     string LCS="";          // LCS를 담을 string 선언
29     int m = 0, n = 0;        // Sequence 의 길이를 넣을 변수 선언 및 초기화
30     int i = 1, j = 1;        // Sequence 의 각 요소들을 순차적으로 비교하기 위한 변수
31     m = x.length() - 1;      // main 함수에서 공백 문자를 추가했으므로, string 길이에서 -1
32     n = y.length() - 1;
33
34     int c[100][100] = { 0, }; // 100 X 100 table, 모든 요소를 0으로 초기화
35
36     for (j = 1; j <= n; j++) // y 요소 하나씩 비교
37     {
38         for (i = 1; i <= m; i++) // x 요소 하나씩 비교
39         {
40             if (x[i] == y[j]) // 같은 문자라면,
41             {
42                 c[j][i] = c[j - 1][i - 1] + 1; // 현재 위치에서 행-1, 열-1 값에서 +1 한 후 넣어줌
43             }
44             else // 다른 문자라면, 현재 값의 위 값과 왼쪽 값 중 큰 값을 넣어줌
45             {
46                 if (c[j - 1][i] > c[j][i - 1]) // 위 값이 왼쪽 값보다 클 때
47                 {
48                     c[j][i] = c[j - 1][i]; // 위 값(현재 위치에서 행-1 값)을 넣어줌
49                 }
50                 else // 왼쪽 값이 위 값보다 클 때 or 같을 때
51                 {
52                     c[j][i] = c[j][i - 1]; // 왼쪽 값(현재위치에서 열-1)을 넣어줌
53                 }
54             }
55         }
56     }
57
58     i = m; // tracing backwards 전 변수 초기화
59     j = n; // : 배열의 끝에서 부터 확인해야 하므로, 배열의 길이를 넣어줌
60
61     while (1) // 2 Sequence의 모든 요소들을 비교하는 Loop
62     {
63         if (i == 0 || j == 0) // 모두 확인 했으면 Loop를 빠져나감
64             break;
65
66         if (c[j][i] == c[j][i - 1]) // 현재 값이 왼쪽 값과 같다면
67         {
68             i--; // 왼쪽으로 이동
69             continue;
70         }
71         else if (c[j][i] == c[j - 1][i]) // 현재 값이 위 값과 같다면
72         {
73             j--; // 위로 이동
74             continue;
75         }
76     }

```



```

76         else // 현재 값이 왼쪽 값과도 다르고 위쪽 값과도 다르다
77         { // 즉, 대각선 + 1 한 경우이다
78             LCS = LCS + x[i]; // 대각선 + 1 한 경우는 2 Sequence의 요소를 비교시,
79             i--; // 두 문자가 같을 때 이므로, 그 값을 LCS에 넣어준다
80             j--; // 그 후, 대각선으로 이동
81             continue;
82         }
83     }
84     LCS = Reverse(LCS); // tracing backwards 했으므로 string 을 reverse 시킨다
85
86     return LCS; // 완성된 LCS 반환
87 }
88

```

```

89 string Reverse(string LCS)
90 {
91     string temp = ""; // 임시로 사용할 string 선언
92     int len = 0; // string의 길이를 담을 변수 선언
93     len = LCS.length()-1; // 배열의 index는 0부터 시작하므로 길이-1 해줌
94
95     for (int i = 0; i <= len; i++)
96     {
97         temp = temp + LCS[len - i]; // string의 마지막 문자부터 시작해 차례로 넣어준다
98     }
99
100    return temp; // reverse 된 string 반환
101 }
102

```

(c) For at least one example, show the simulation results.

Microsoft Visual Studio 디버그 콘솔	Microsoft Visual Studio 디버그 콘솔
<pre> Enter 1st Sequence: ABCBDBAB Enter 2nd Sequence: BDCABA LCS:BCBA C:\Users\stanley\source\repos\prob3-2\ 이 창을 닫으려면 아무 키나 누르세요. </pre>	<pre> Enter 1st Sequence: BDCABA Enter 2nd Sequence: ABCBDBAB LCS:BDAB C:\Users\stanley\source\repos\prob3-2\ 이 창을 닫으려면 아무 키나 누르세요. </pre>

(d) Explain the program and the simulation results (at least five lines).

1에서 만든 2차원 배열 c 를 이용해 LCS 를 구한다. 이때 tracing backwards 한다. 필자는 이 과정을 while 문으로 수행했다. 현재 좌표가 비교하는 좌표에 의해 왼쪽 혹은 위쪽으로 이동한다. 계속 해서 이동하게 된다면, $(0, 0)$ 좌표($i=0, j=0$)에 도달하게 되는데 이때 while문을 빠져나온다. 상세히 설명하자면, while 문 안에 상태는 총 4가지로 나뉘게 된다. 첫번째, 현재 값이 왼쪽 값과 같을 때($c[j][i] == c[j][i-1]$) 이때는 단순히 왼쪽으로 이동($i--$) 하고 continue 하면 된다. 두번째, 현재 값이 위 값과 같을 때($c[j][i] == c[j-1][i]$) 는 위로 이동($j--$) 하고 continue 한다. 세번째, 첫번째의 경우도 아니고 두번째 경우도 아닌 경우, 즉, 현재 값이 왼쪽 값과도 다르고 위 값과도 다른 경우 이므로, 대각선 방향에서 +1 한 경우이다. 이때가 문자가 같은 경우 이므로 그 문자를 LCS 에 삽입하도록 한다. 현재 좌표를 대각선으로 이동($i--, j--$) 후 continue 한다. 1,2,3 과정을 모두 거치다 보면 마지막으로 4번째 상황이 온다. 이 경우는 모든 문자의 비교를 완료한 상태이므로

break 로 while문을 빠져나가도록 한다.

이렇게 완성된 LCS는 tracing backwards로 수행되었기 때문에 문자열이 뒤집어져 있는 상태이다. 이 문자열의 각각의 index들의 순서를 역순으로 정렬하여 LCS를 구한다. 그후 그 LCS를 반환하여 main() 함수에서 출력한다.

강의자료에 예시처럼 두 sequence에 'ABCB~~D~~AB'와 'BDCABA'를 입력했다. 그 후 여러 LCS 중 하나인 'BCBA'가 출력된 것을 확인할 수 있다. 두 sequence 'ABCB~~D~~AB'와 'BDCABA'의 LCS는 'BCBA', 'BCBA', 'BDAB' 가 있지만 프로그래머의 코드에 따라 이 중 하나가 출력된다. 두 문자열의 순서를 바꿔서 입력한 경우는, 'BDAB'가 출력된 것을 확인할 수 있다.

Conclusion.

이번 과제는 Dynamic programming Algorithm 을 LCS의 길이와 LCS를 찾는 것에 적용했다. LCS의 길이는 두 sequence 의 문자를 하나씩 비교하며, 같은 문자라면 대각선 위치에서 +1 을 하고, 다른 문자라면 왼쪽값 과 위 값 중 더 큰 값을 넣어서 두 문자열의 길이에 이를 때 까지 수행하여, 최종적으로 그 값이 무엇인지 파악하는 것이다. LCS를 찾는 것은 이렇게 이루어진 table을 tracing backwards 하여 찾는다. 2차원배열의 끝에서부터 확인하는데, 현재 값이 왼쪽 값과 같다면 왼쪽으로 이동, 위 값과 같다면 위로 이동, 둘 다 다를 경우는 두 문자가 같을 경우 이므로 해당 문자를 LCS에 삽입 후 대각선으로 이동하여 다시 배열의 처음에 이를 때 까지 이를 수행하도록 한다. 이렇게 만들어진 LCS는 reverse 된 string 이므로 배열의 index를 역순으로 정렬하여 최종적으로 LCS를 완성하도록 한다.

이번 과제를 수행하면서 어려웠던 점은 없었다. 다만 기본적인 문법에서 약간의 문제가 있었는데 string형 관련 문제, 2차원 배열의 행, 열 문제이다. 항상 char형 배열에 문자열을 넣어주는 방식으로 수행했지만, 이번 과제는 string 으로 수행했다. String을 입력받는 getline(), 길이를 반환해주는 length() 를 사용했고, 이외에도 string 이 문자열 관련된 일을 수행할 때 char 형 문자열보다 훨씬 사용법도 쉽고, 유용한 함수들을 제공한다는 것을 새롭게 알게 됐다.

다음으로, 행과 열의 순서를 2차원배열의 인자로 적절하게 넣어야 하는데 처음에는 (x, y) 좌표 순서로 넣어서 계속 엉뚱한 결과가 만들어졌다. 아무리 확인해도 원인을 알 수 없었지만, 디버깅 과정에서 엉뚱한 위치의 값이 새롭게 들어가는 것을 확인해 원인을 파악했다. 그렇게 2차원 배열의 문법

을 다시 파악했고, 단순히 행과 열의 위치를 다시 바꿔주는 것으로 이 문제를 해결했다.

Reference.

2차원 배열 참고 "<http://recloud.tistory.com/72>"