Crocus 구독하기

# LCS(Longest Common Subsequence) 알고리즘

2017. 4. 19. 19:55

## 목록



## 1. LCS(Longest Common Subsequence) 알고리즘이란?

LCS는 번역하면 **최장 공통 부분 문자열**이라고 할 수 있다.

부분 문자열이라는 개념을 생각해보자.

Substring이라는 개념과, Subsequence라는 개념이 있다.

우리가 늘 단어가 일치하는지 확인한다는 개념은 **연속된 부분 문자열인 Substring**이라는 개념이고,
어떤 단어에서 **연속되지 않은 부분 문자열이 있다면 그것은 Subsequence**이다.

예를 들어보자.
가나다라마바사와
가아나자마바사에서

**가장 긴 Substring**은 다음과 같다.
가나다라**마바사**
가아나자**마바사**

반면 **가장 긴 Subsequence**는 다음과 같다.
**가나**다라**마바사**
**가**아**나**자**마바사**

어떤 문제가 다음과 같이 주어졌다고 생각해보자.



```
두개의 String이 주어졌을 때 LCS를 구하시오.
ACAYKP
CAPCAK
```

답은 무엇일까?

ACAYKP

CAPCAK

즉, ACAK이다.

## 2. LCS(Longest Common Subsequence) 알고리즘 구현 과정

### - LCS 길이 찾는 방법

이제 위에서 계속 읽어온 LCS(Longest Common Subsequence) 알고리즘을 어떻게 구현하는지 생각해보자.

ACAYKP

CAPCAK

이 두 단어를 통해 한번 확인해보자.

| | 0 | A | C | A | Y | K | P |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | | | | | | |

LCS 알고리즘을 구현할 때는, LCS 테이블의 첫번째 행과 열을 항상 0으로 맞추어준다.(관례이기도 하고, 편의를 위해 이렇게 쓴다.)

| | 0 | A | C | A | Y | K | P |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

이제 CAPCAK의 첫번째 단어인 C를 이용하여 테이블을 채우면 다음과 같다.

위의 1이 의미하고 있는 것은 무엇이냐면,

A까지의 LCS는 0, AC까지의 LCS는 1 ACA까지의 LCS는 1 ... ACAYKP까지의 LCS는 1이라는 것이다.

즉, **ACA까지 LCS가 1이라는 것은 ACA와 C가 공통 부분 문자열이 1개 있다는 뜻이다.**

| | 0 | A | C | A | Y | K | P |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 2 | 2 | 2 | 2 |

다음으로 ACAYKP와 CA를 비교해보자.

당연히 LCS는 2가 나올 것이다. (ACAYKP, CA)

이 과정이 위의 테이블에서 작성되고 있다.

| | 0 | A | C | A | Y | K | P |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| P | 0 | 1 | 1 | 2 | 2 | 2 | 3 |

이제 어떻게 구현되는지는 감은 오지만, 테이블에 숫자 넣는 방법이 어떻게 되는지 조금 애매하기도 하다.

**어떤 기준으로 넣는 것일까?**

| | 0 | A | C | A | Y | K | P |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| P | 0 | 1 | 1 | 2 | 2 | 2 | 3 |

**위의 푸른색 부분을 보자.**

C와 AC에서 이미 1이 됐으니, C와 ACA에서는 마지막 단어가 C와 A로 서로 다르지만,
LCS가 1인 이유는 왼쪽에서 값이 1이었으므로 1이 가장 최장 부분 문자열이다.

**분홍색 부분을 보자.**

이 부분은 CAP와 ACA에서 마지막 단어가 P와 A로 서로 다르지만,
LCS가 2인 이유는 이전 CAP의 CA와 ACA의 CA가 서로 2이기에 누적된 값이 내려왔음을 알 수 있다.

**즉, 현재 서로 다른 문자열일 때는, 현재 테이블에 들어갈 수는 위쪽, 왼쪽중 큰 값을 따름을 알 수 있다.**

마지막으로 회색 부분을 보자.

우선 3이 된 이유는 CAP와 ACAYKP의 서로 마지막 위치의 단어가 P로 같기 때문이다.

이 3은 위의 2에서 1더해진 3인가, 왼쪽에서 1더해진 3인가?

정답은 왼쪽 위 2에서 1더해진 3이다.

이 이유는 CA와 ACAYK다음 P와 P를 봤을 때 LCS가 3이 된 것이 정확한 의미이기에

대각선(왼쪽 위) 방향인 값에서 1을 더해야 한다.

**즉, 현재 서로 같은 문자열일 때는, 현재 테이블에 들어갈 수는 대각선(왼쪽 위)의 값 + 1이다.**

최종적으로 LCS 테이블이 어떻게 구성되는지 확인해보자.

| | 0 | A | C | A | Y | K | P |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| P | 0 | 1 | 1 | 2 | 2 | 2 | 3 |
| C | 0 | 1 | 2 | 2 | 2 | 2 | 3 |
| A | 0 | 1 | 2 | 3 | 3 | 3 | 3 |
| K | 0 | 1 | 2 | 3 | 3 | 4 | 4 |

## 3. LCS(Longest Common Subsequence) 소스 코드

### - LCS 길이 찾는 방법

**이 코드는 [9251번] LCS :: https://www.acmicpc.net/problem/9251 를 기준으로 제작하였습니다.**

Crocus

```cpp
#include <iostream>
#include <cstdio>
#include <string>

using namespace std;

string str1, str2;

int lcs[1001][1001];

int main()
{
    string tmp1, tmp2;
    cin >> tmp1 >> tmp2;

    // LCS 알고리즘을 위해 앞에 '0'을 붙여준다.
    str1 = '0' + tmp1;
    str2 = '0' + tmp2;

    int len1 = str1.size();
    int len2 = str2.size();

    for (int i = 0; i < len1; i++)
    {
        for (int j = 0; j < len2; j++)
        {
            if (i == 0 || j == 0)
            {
                lcs[i][j] = 0;
                continue;
            }

            // 현재 비교하는 값이 서로 같다면, lcs는 + 1
            if (str1[i] == str2[j])
                lcs[i][j] = lcs[i - 1][j - 1] + 1;

            // 서로 다르다면 LCS의 값을 왼쪽 혹은 위에서 가져온다.
            else
            {
                if (lcs[i - 1][j] > lcs[i][j - 1])
                    lcs[i][j] = lcs[i - 1][j];
                else
                    lcs[i][j] = lcs[i][j - 1];
            }
        }
    }

```

```
48      /*
49      // 검증 코드
50      for (int i = 0; i < len1; i++)
51      {
52          for (int j = 0; j < len2; j++)
53              cout << lcs[i][j] << " ";
54          cout << endl;
55      }
56      */
57
58      cout << lcs[len1-1][len2-1] << endl;
59
60      return 0;
61  }
62
63  //                                                  This source code Copyright belongs to Crocus
64  //                                                   If you want to see more? click here >>
```

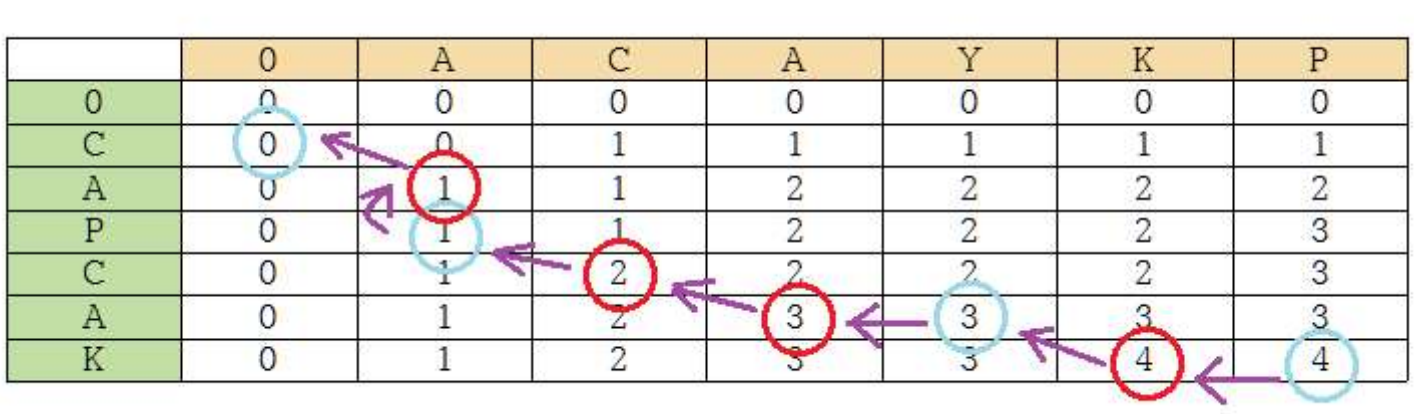


## 4. LCS(Longest Common Subsequence) 알고리즘 구현 과정

### - LCS 실제 단어 찾는 방법

| | 0 | A | C | A | Y | K | P |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| P | 0 | 1 | 1 | 2 | 2 | 2 | 3 |
| C | 0 | 1 | 2 | 2 | 2 | 2 | 3 |
| A | 0 | 1 | 2 | 3 | 3 | 3 | 3 |
| K | 0 | 1 | 2 | 3 | 3 | 4 | 4 |

이 표를 이용하여 실제 단어를 찾아볼 것이다.




가장 끝자리부터 시작하여 자신과 같은 숫자가 있는곳까지 따라간다.

그리고 왼쪽, 위쪽 둘중 둘다 같은 수가 없다면, 대각선 방향 값이 현재 값 - 1인지 확인해보고 그 수를 따라간다.

위에서 표시된 빨간색 동그라미가 추적되고있는 단어를 의미하고있다.(빨간 동그라미의 행,열을 보면 같은 단어임을 알 수 있다.)

이것을 테이블에서 0이 나타날 때 까지 계속해서 반복한다.

## 5. LCS(Longest Common Subsequence) 소스 코드
## - LCS 실제 단어 찾는 방법

**이 코드는 [9252번] LCS :: https://www.acmicpc.net/problem/9252 를 기준으로 제작하였습니다.**

```cpp
#include <iostream>
#include <cstdio>
#include <string>
#include <stack>

using namespace std;

string str1, str2;

int lcs[1001][1001];

int main()
{
    string tmp1, tmp2;
    cin >> tmp1 >> tmp2;

    // LCS 알고리즘을 위해 앞에 '0'을 붙여준다.
    str1 = '0' + tmp1;
    str2 = '0' + tmp2;

    int len1 = str1.size();
    int len2 = str2.size();

    for (int i = 0; i < len1; i++)
    {
        for (int j = 0; j < len2; j++)
        {
            if (i == 0 || j == 0)
            {
                lcs[i][j] = 0;
                continue;
            }

            // 현재 비교하는 값이 서로 같다면, lcs는 + 1
            if (str1[i] == str2[j])
                lcs[i][j] = lcs[i - 1][j - 1] + 1;

            // 서로 다르다면 LCS의 값을 왼쪽 혹은 위에서 가져온다.
            else
            {
                if (lcs[i - 1][j] > lcs[i][j - 1])
                    lcs[i][j] = lcs[i - 1][j];
                else
                    lcs[i][j] = lcs[i][j - 1];
            }
        }
    }

    /*
    // 검증 코드
    for (int i = 0; i < len1; i++)
    {
        for (int j = 0; j < len2; j++)
            cout << lcs[i][j] << " ";
```

```
        cout << endl;
    }
    */


    int i = len1-1;
    int j = len2-1;
    stack<int> st; // 거꾸로 담기니 stack을 이용

    while (lcs[i][j] != 0)
    {
        // 경로 추적
        // cout << " i :: " << i << " j :: " << j << endl;

        // 테이블이 같은 넘버링이라면
        // 왼쪽으로 이동
        if (lcs[i][j] == lcs[i][j - 1])
            j--;

        // 위쪽으로 이동
        else if (lcs[i][j] == lcs[i - 1][j])
            i--;

        // 왼쪽 위쪽 모두 다른 넘버링이라면 대각선 방향으로 이동
        else if (lcs[i][j] - 1 == lcs[i - 1][j - 1])
        {
            st.push(i);
            i--;
            j--;
        }
    }

    // 길이 출력
    cout << lcs[len1-1][len2-1] << endl;

    // 단어 출력
    while (!st.empty())
    {
        cout << str1[st.top()];
        st.pop();
    }
    return 0;
}

//                                                  This source code Copyright belongs to Crocus
//                                                   If you want to see more? click here >>
```

## 5. LCS 관련 문제

http://www.crocus.co.kr/search/lcs



'Applied > 알고리즘' 카테고리의 다른 글

최적화된 에라토스테네스의 체(Eratosthenes' sieve) (0) 2017.04.21

단어를 찾아 지우거나, 치환하는 알고리즘 (0) 2017.04.21

LCS(Longest Common Subsequence) 알고리즘 (11) 2017.04.19

최소 버텍스 커버(Minimum Vertex Cover) (0)

최소 컷(Minimum Cut) (2) 2017.04.12

이분 매칭(Bipartite Matching) 시간 단축 방법 (2) 2017.04.10



NAME

PASSWORD

HOMEPAGE http://

SECRET ☐ WRITE

---

hoon222y PROFILE 2017.06.15 17:27 신고

덕분에 잘 공부하고 갑니다 ㅎㅎ

# Delete Reply

가누 PROFILE 2017.06.15 20:40 신고

감사합니다~~

# Delete

---

바지 2018.03.26 16:45

친절한 설명 감사합니다^^

# Delete Reply

가누 PROFILE 2018.03.26 17:58 신고

감사합니다

# Delete

---

Student1 2018.10.05 21:05

당신의 명확성에 무릎을 탁 치고 갑니다.

# Delete Reply

가누 PROFILE 2018.10.05 21:08 신고

감사합니다~~!

# Delete

---

cykei 2020.02.16 21:38 신고

많이 배워갑니다!



# Delete Reply

가누 PROFILE 2020.02.17 20:22 신고

감사합니다~

# Delete

---

arkingco 2020.08.30 16:33

배우다 갑니다잉

# Delete Reply

---

2je0 2021.04.04 01:17

혹시 관련 문제 질문 하나만 할수 있을까요 ..?

# Delete Reply

가누 2021.04.04 20:53

네네 어떤건가요

# Delete



crocus@kakao.com