알고리즘 과제(08)

201404377 진승언

1. 과제 목표 및 해결 방법

➔ 이번 과제는 최장 공통 부분 순서(Longest Common Subsequence)를 구하는 것이었다. 예를 들어 ABCDAB 와 BDCABA 문자열의 LCS를 구하면 BCAB이다. 즉 두 문자열의 공통적으로 나타나는 부분 순서(공통 부분 순서)이면서 가장 긴 공통 부분순서를 구하는 거였다.

- 두 문자열 $X_m = \langle x_1 x_2 \dots x_m \rangle$과 $Y_n = \langle y_1 y_2 \dots y_n \rangle$에 대해
  - $x_m = y_n$이면
    X_m과 Y_n의 LCS의 길이는 X_{m-1}과 Y_{n-1}의 LCS의 길이보다 1이 크다
  - $x_m \neq y_n$이면
    X_m과 Y_n의 LCS의 길이는
    X_m과 Y_{n-1}의 LCS의 길이와 X_{m-1}과 Y_n의 LCS의 길이 중 큰 것과 같다

- $$c_{ij} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c_{i-1,j-1} + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max\{c_{i-1,j}, c_{i,j-1}\} & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

✓ c_{ij} : 두 문자열 $X_i = \langle x_1 x_2 \dots x_i \rangle$과 $Y_j = \langle y_1 y_2 \dots y_j \rangle$의 LCS 길이

수업시간 때 배운 위와 같은 정의를 참고해서 구현하였다. 배열에 값 뿐만 아니라 방향도 저장했어야 했으므로 객체를 생성해 객체이차배열을 사용하였다. 그리고 방향은 대각선 위, 옆 3가지가 필요한데 이것은 숫자 0, 1, 2로 나타냈다. 첨에 배열에 값을 저장하고 방향을 마킹하는데 경우의 수는 위와 같이 3가지이므로 3가지 경우의 수로 나누어서 구현하였다. 그리고 마킹을 해논 또 3가지 경우의 수로 재귀호출 해서 LCS를 출력할 수 있게 했다.

2. 주요 부분 코드 설명(알고리즘 부분 코드 캡쳐)

Arr.java

```java

public class Arr {

    int num;
    int direction;
}

```

위와 같이 숫자(값)과 방향을 담을 객체(구조체)를 생성하였다.

```java
public class LCS {

    //Direction
    final int LEFT = 0;
    final int UP = 1;
    final int DIAGONAL = 2;

    StringBuffer result = new StringBuffer();    //LCS 결과값 저장

    //Longest Common Subsequence
    Arr[][] LCS_length(String x, String y) {
        int m = x.length()+1;  //x의 문자열 길이 (문자열의 길이보다 1더 필요하므로 1더함)
        int n = y.length()+1;  // ''''
        Arr[][] arr = new Arr[m][n];

        //배열 초기화 생성해줌
        for(int i = 0; i<m; i++) {
            for(int j =0; j<n; j++) {
                arr[i][j] = new Arr();
            }
        }

        //배열 행, 열 첫줄들을 0으로 초기화 (=행 0번 줄 0으로 초기화, 열 0번 줄 0으로 초기화)
        for(int i =1; i<m; i++) {
            arr[i][0].num = 0;
        }
        for(int j =0; j < n; j++) {
            arr[0][j].num = 0;
        }

```

(주석에도 자세히 설명 적어놨습니다.)

방향은 위와 같이 LEFT(왼쪽), UP(위), DIAGONAL(대각선)을 각각 0 , 1, 2로 하였다. 그리고 LCS의 결과값을 리턴 하기 위해 StringBuffer로 result를 생성하였다. 문자를 계속해서 덧붙이고 바뀌는 문자열이어야 하므로 StringBuffer로 하였다.

LCS_length함수는 앞서 설명한 Arr객체를 이차원배열로 생성해서 값과 방향 마킹을 한 후 해당 Arr객체배열을 반환해준다. 먼저 이차원배열의 행, 열 길이가 문자열의 길이보다

한 칸 더 필요하므로 두 문자열의 길이에 1을 더해서 m, n에 저장하고 이것으로 m행과 n행을 가진 Arr객체 이차원배열을 생성한다. 그리고 이 객체배열 사용하기 위해서 초기화를 해준다. 그리고 LCS 알고리즘에 맞게 첫 행과 열 줄의 값을 0으로 해준다.

```
32          //LCS 마킹
33          for(int i =1; i<m; i++) {  // 위에서 0으로 초기화 한 위치를 제외한 칸들을 for문 돌려 마킹한다
34              for(int j=1; j<n; j++) { // ''
35                  if(x.charAt(i-1) == y.charAt(j-1)) {    //x문자열의 i-1번째 문자와 y문자열의 j-1번째 문자가 같은 경우(문자열의 시작은 0부터 시작하므로 i,j보다 1작아야 매칭되므로 charAt에 -1해줌)
36                      arr[i][j].num = arr[i-1][j-1].num + 1;  //현재 i,j칸의 왼쪽대각선위칸([i-1,j-1])에 1을 더한 값을 저장
37                      arr[i][j].direction = DIAGONAL;     //방향에 대각선 마킹
38                  }
39                  else if(arr[i-1][j].num >= arr[i][j-1].num) {   //문자가 같지않고 현재위치의 위쪽값이 왼쪽칸 값보다 큰 경우
40                      arr[i][j].num = arr[i-1][j].num;    //현재위치의 위칸값 저장
41                      arr[i][j].direction =  UP;          //방향에 UP 마킹
42                  }
43                  else {      //문자가 같지않고 현재위치의 왼쪽칸 값이 위쪽칸 값보다 큰 경우
44                      arr[i][j].num = arr[i][j-1].num;    //현재위치의 왼쪽칸값 저장
45                      arr[i][j].direction = LEFT;     //방향에 LEFT 마킹
46                  }
47              }
48          }
49          return arr;
50      }
```

그 후 LCS알고리즘에 알맞게 3가지 경우의 수로 나누어 각 객체 배열에 알맞은 값과 방향을 저장해 나간다. 첫 번째 경우의 수는 조사하는 인덱스에 매칭되는 행의 문자와 열의 문자가 같으면 해당 인덱스의 객체 값은 왼쪽 위 대각선 인덱스의 값에 1을 더한 값을 저장하고 방향은 대각선(DIAGONAL)을 저장한다. 두 번째 경우의 수와 세 번째 경우의 수는 그에 행과 열에 해당되는 문자가 같지 않을 경우, 현재 인덱스 값을 위와 왼쪽에 있는 인덱스 값을 비교해서 더 큰 값을 가지게 한다. 그리고 해당 방향도 저장한다.

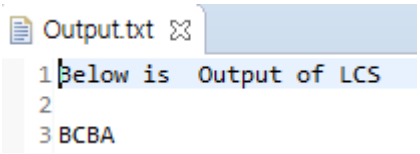
```
52    StringBuffer LCS_print(Arr[][] b, String x, int i, int j ) {
53        if(i==0 || j == 0) {     //i,j가 0인 경우 탐색을 다한것이므로 지금까지 저장한 결과를 리턴
54            return result;
55        }
56        if(b[i][j].direction == DIAGONAL) {      //DIAGONAL 마킹인 경우
57            LCS_print(b,x, i-1, j-1);        //왼쪽대각선 위 칸으로 재귀
58            System.out.print(x.charAt(i-1));
59            result.append(x.charAt(i-1));    //해당 문자 결과값에 저장
60        }
61        else if(b[i][j].direction == UP) {  //UP 마킹인 경우
62            LCS_print(b,x, i-1, j);      //  //왼쪽 칸으로 재귀
63        }
64        else {  //LEFT 마킹인 경우
65            LCS_print(b,x, i, j-1); //오른쪽 칸으로 재귀
66        }
67
68        return result;
69
70    }
```

LCS의 문자열을 재귀호출을 이용해서 반환하는 함수이다. 이전에 값과 방향을 저장해 논 객체배열을 매개변수로 준다. 그리고 행에 속했던 x문자열도 매개변수로 준다. i와 j는 행과 열의 길이이다. (즉 문자열의 길이 이다.) 이 객체배열 b를 주석에 써있는 것과 같은 경우의 수 별로 재귀호출을 해서 LCS에 속하는 문자들을 더해가고 반환해주면 된다. LCS에 해당하는 문자 하나씩 더해서 문자열을 반환하므로 StringBuffer를 사용하였다.

```
43        LCS lcs = new LCS();
44        //LCS 마킹
45        Arr[][] arr = lcs.LCS_length(data1[0], data1[1]);
46        //LCS 순회
47        result = lcs.LCS_print(arr , data1[0], data2[0], data2[1]);
```

그래서 정리하자면 LCS객체를 생성하고 값과 방향을 저장하는 즉 마킹하는 함수인 LCS_Length함수를 호출해서 Arr객체에 저장한다. (data1에는 입력받은 문자열이 들어있다.) 그리고 이 arr을 LCS_print함수에 사용하여 LCS인 문자열을 result에 저장해서 파일 출력시키면 되었다. (data2는 문자열의 길이 입력이 저장되있다.)

결과(시간 복잡도 포함)



LCS에 속하는 BCBA가 잘 출력됨을 볼 수 있었다.

LCS는 이차원 배열에 각 부분 문제의 답을 저장하면서 풀어나가는 방식이다. 배열의 총 원소는 (m+1)(n+1)개이고, LCS의 값을 저장하는 LCS_Length함수의 2중 for문에서 for루프가 총 m*n번을 반복하면서 원소를 하나씩 계산한다. 각 원소를 계산하는 데는 상수 시간이 든다. 따라서 총 수행 시간 즉 시간복잡도는 θ(mn)이다.