201201871 서현택

1.목표

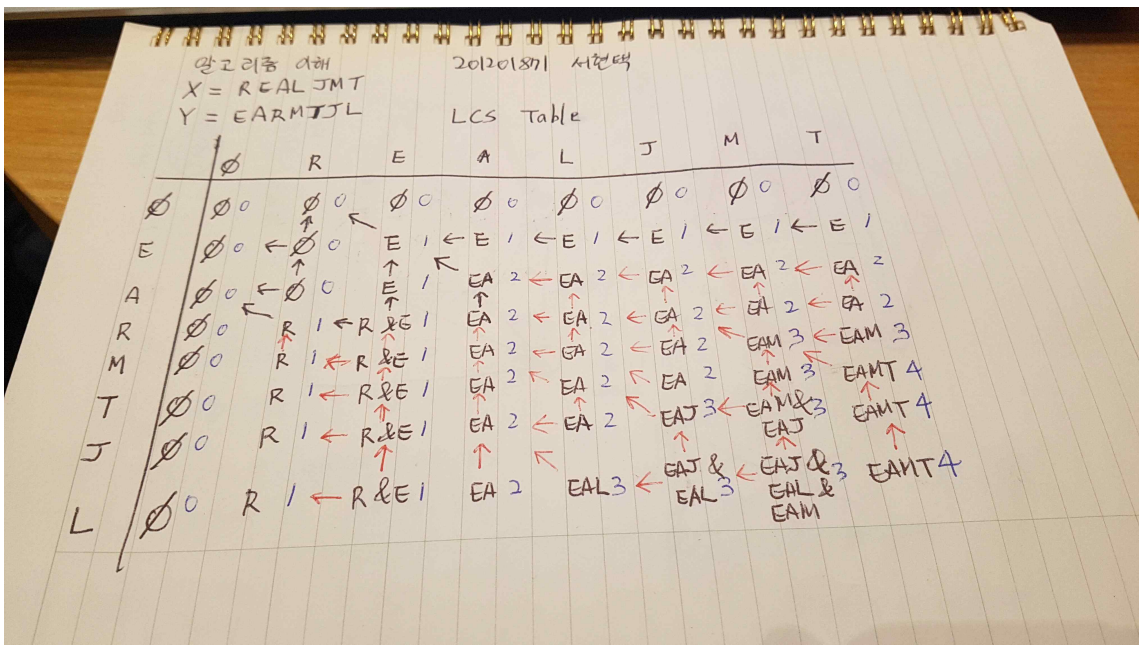
1) Longest Common Subsequence(최장 공통 부분 수열) 구현
2) 비속어 차단 프로그램 구현

2.과제를 해결하는 방법

1) LCS에 대한 이해
2) 손으로 그려보기

3.과제를 해결한 방법

1) 손으로 직접 그려보기



직접 lcs table을 그려보았다.
비교하려는 행과 열의 값이 같다면 대각선 왼쪽 위 값에 현재 값을 붙이고, 그렇지 않다면 현재 위치 위와 왼쪽 값을 비교하여 lcs 값이 더 큰 값을 그대로 가져왔다. 이를 통해 최종적으로 나온 글자가 EAMT 임을 알 수 있었다.

2) 비속어 탐지 프로그램 구현
input 함수를 통하여 비속어를 입력 받고 word에 미리 저장된 비속어들과 입력된 값을 backtrack 함수에 넣어 나온 값이 word와 같다면 비속어가 있음을 나타내고, 아니라면 아님을 나타낸다.

4.결과화면

1) LCS 값
X = REALJMT

Y = EARMTJL

```
Some LCS: 'EAMT'
All LCSs: {'EAMT'}
```

2) 비속어 차단 프로그램

(1) 시바

```
비속어인지 확인할 단어를 입력하세요: 시부라럼ㄴㅇ라ㅓㅣ바ㅋㅊㅌ쩜니마러비ㅏ
비속어가 있습니다. {'시바'}
```

(2) 개아이

```
비속어인지 확인할 단어를 입력하세요: 개ㅁㄴ얼만어마ㅁㄴ머리ㅏㅁ넣리ㅓㅣㅁㅇㄴ라ㅓ이ㅁㄴㅇ레ㄴㅇ
비속어가 있습니다. {'개아이'}
```

5.소감

1학년 때 아무것도 모를 때부터 졸업 전 마지막 수업까지 도움 많이 주셔서 감사하고, 한 학기 동안 열심히 수업해주셔서 감사합니다!

1 개요

컴퓨터공학은 기계, 항공, 우주, 전기, 전자 공학뿐 아니라 생명 공학과도 많은 영향을 주고 받으며 성장하고 있다. 특히, 생명 공학의 염기 서열 분석은 오늘날 급속도로 향상된 컴퓨팅 성능으로 그 성장이 가속화되었다. 염기 서열 분석에서 중요한 것 중 하나는 두 염기 서열이 얼마나 비슷한지 찾는 것이다. 이럴 때 주로 사용하는 알고리즘이 '최장 공통 부분 수열'찾기 알고리즘이다. 이번 실습과 과제에서는 'Longest common subsequence'를 찾는 알고리즘을 학습한다.

2 최장 공통 부분 수열(Common Subsequence)

공통 부분 수열은 두개의 수열(혹은 문자열)에서 공통으로 들어가있는 부분 수열을 의미한다. 이 때 각 수 혹은 문자가 연속해서 나타날 필요는 없고 순서만 일치하면 된다. 예를 들어 ABCD와 ACBD의 공통 부분 수열은 AB, AC, AD, BD, CD, ABD, ACD다.

최장 공통 부분 수열은 공통 부분 수열 중에서 가장 긴 수열들을 말한다. 위의 예에서는 ABD, ACD가 LCS, 즉 최장 공통 부분 수열이다.

2.1 구현 - 수도코드

알고리즘은 모든 언어에서 구현하기 쉽게 수도코드로 자주 표현된다. LCS를 구할 때에는 그 복잡도를 줄이기 위하여 다이나믹프로그래밍이 주로 사용된다. 그림 1은 AGCAT와 GAC의 공통 부분 수열을 찾는 알고리즘 결과로 생성되는 테이블이다. 이 테이블이 LCS를 구하기 위하여 꼭 채워져야하는 데이터이므로 테이블을 채워넣는 수도코드먼저 확인하도록 하자.

Listing 1: LCS 테이블 제작 수도 코드

Completed LCS Table

	Ø	A	G	C	A	T
Ø	Ø	Ø	Ø	Ø	Ø	Ø
G	Ø	↑Ø	↖(G)	←(G)	←(G)	←(G)
A	Ø	↖(A)	↑(A) & ←(G)	↑(A) & ←(G)	↖(GA)	←(GA)
C	Ø	↑(A)	↑(A) & ←(G)	↖(AC) & ←(GC)	↑(AC) & ←(GC) & ←(GA)	↑(AC) & ←(GC) & ←(GA)

Figure 1: AGCAT, GAC 최장 공통 부분 수열 테이블(결과)

```
1  function LCSLength(X[1..m], Y[1..n])
2      C = array(0..m, 0..n)
3      for i := 0..m
4          C[i,0] = 0
5      for j := 0..n
6          C[0,j] = 0
7      for i := 1..m
8          for j := 1..n
9              if X[i] = Y[j]
10                 C[i,j] := C[i-1,j-1] + 1
11             else
12                 C[i,j] := max(C[i,j-1], C[i-1,j])
13     return C[m,n]
```

2.2 LCS 테이블 backtrack

LCS 테이블을 읽어 최장 공통 부분 수열을 찾아내는 작업을 backtrack이라고 부른다. 모든 최장 공통 부분 수열을 찾아내기 전에 하나를 출력하는 수도코드를 살펴보자. 그림 ??은 그림 ??로 생각하여 계산한다.

Storing length, rather than sequences

	Ø	A	G	C	A	T
Ø	0	0	0	0	0	0
G	0	↑←0	↖1	←1	←1	←1
A	0	↖1	↑←1	↑←1	↖2	←2
C	0	↑1	↑←1	↖2	↑←2	↑←2

Figure 2: AGCAT, GAC 최장 공통 부분 수열 테이블(결과)의 backtrack 버전

Listing 2: LCS 테이블 제작 수도 코드

```
1  function backtrack(C[0..m,0..n], X[1..m], Y[1..n], i, j)
2      if i = 0 or j = 0
3          return ""
4      if  X[i] = Y[j]
5          return backtrack(C, X, Y, i-1, j-1) + X[i]
6      if C[i,j-1] > C[i-1,j]
7          return backtrack(C, X, Y, i, j-1)
8      return backtrack(C, X, Y, i-1, j)
```

2.3 backtrack-all

최장 공통 부분 수열 중 하나를 출력해보았으니 모든 최장 공통 부분 수열을 출력해보자.

Listing 3: LCS 테이블 제작 수도 코드

```
1  function backtrackAll(C[0..m,0..n], X[1..m], Y[1..n], i,
       j)
2      if i = 0 or j = 0
3          return {""}
4      if X[i] = Y[j]
5          return {Z + X[i] for all Z in backtrackAll(C, X,
               Y, i-1, j-1)}
6      R := {}
7      if C[i,j-1] >= C[i-1,j]
8          R := R U backtrackAll(C, X, Y, i, j-1)
9      if C[i-1,j] >= C[i,j-1]
10         R := R U backtrackAll(C, X, Y, i-1, j)
11     return R
```