알고리즘 실습

181129 - Longest Common Subsequence

오늘의 목표

- Longest Common Subsequence(최장 공통 부분 수열)

Feedback

지난 시간: 그래프 응용(Dijkstra, Prim)

- 제출률: 68.29%(28/41)
  - 100% 완료: 31.71(13/41)
- 질문: 이메일 67건, 연구실 방문 4건

공통 부분 수열

부분 수열(Subsequence)

- ABCD
  - A, B, C, D
  - AB, AC, AD, BC, BD, CD
  - ABC, ABD, ACD, BCD
- ACBD
  - A, C, B, D
  - AC, AB, AD, CB, CD, BD
  - ACB, ACD, ABD, ABD

공통 부분 수열(Common Subsequence)

- ABCD
  - A, B, C, D
  - **AB**, **AC**, **AD**, BC, **BD**, **CD**
  - ABC, **ABD**, **ACD**, BCD
- ACBD
  - A, C, B, D
  - **AC**, **AB**, **AD**, CB, **CD**, **BD**
  - ACB, **ACD**, **ABD**, ABD

최장 공통 부분 수열(Longest Common Subsequence)

- ABCD
  - A, B, C, D
  - AB, AC, AD, BC, BD, CD
  - ABC, **ABD**, **ACD**, BCD
- ACBD
  - A, C, B, D
  - AC, AB, AD, CB, CD, BD
  - ACB, **ACD**, **ABD**, ABD

LCS 구현

LCS Table

- 이전까지의 결과를 이용한다!
  - 채울 수 있는 부분부터 차례차례 접근

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)

LCS 수도코드: LCS Table

- https://en.wikipedia.org/wiki/Longest_common_subsequence_problem
- LCS Table 만들기

```
function LCSLength(X[1..m], Y[1..n])
    C = array(0..m, 0..n)
    for i := 0..m
       C[i,0] = 0
    for j := 0..n
       C[0,j] = 0
    for i := 1..m
        for j := 1..n
            if X[i] = Y[j]
                C[i,j] := C[i-1,j-1] + 1
            else
                C[i,j] := max(C[i,j-1], C[i-1,j])
    return C[m,n]
```

LCS 수도코드: Backtrack

- https://en.wikipedia.org/wiki/Longest_common_subsequence_problem
- Backtrack 1개

```
function backtrack(C[0..m,0..n], X[1..m], Y[1..n], i, j)
    if i = 0 or j = 0
        return ""
    else if  X[i] = Y[j]
        return backtrack(C, X, Y, i-1, j-1) + X[i]
    else
        if C[i,j-1] > C[i-1,j]
            return backtrack(C, X, Y, i, j-1)
        else
            return backtrack(C, X, Y, i-1, j)
```

LCS 수도코드: BacktrackAll

- https://en.wikipedia.org/wiki/Longest_common_subsequence_problem
- Backtrack 모두

```
function backtrackAll(C[0..m,0..n], X[1..m], Y[1..n], i, j)
    if i = 0 or j = 0
        return {""}
    else if X[i] = Y[j]
        return {Z + X[i] for all Z in backtrackAll(C, X, Y, i-1, j-1)}
    else
        R := {}
        if C[i,j-1] ≥ C[i-1,j]
            R := R ∪ backtrackAll(C, X, Y, i, j-1)
        if C[i-1,j] ≥ C[i,j-1]
            R := R ∪ backtrackAll(C, X, Y, i-1, j)
        return R
```

LCS 구현의 실제

LCS

- LCS 테이블 만들기

```
def LCS(X, Y):
    m = len(X)
    n = len(Y)
    # An (m+1) times (n+1) matrix
    C = [[0] * (n + 1) for _ in range(m + 1)]
    for i in range(1, m+1):
        for j in range(1, n+1):
            if X[i-1] == Y[j-1]:
                C[i][j] = C[i-1][j-1] + 1
            else:
                C[i][j] = max(C[i][j-1], C[i-1][j])
    return C
```

LCS backtrack

- LCS 1개 출력하기

```
def backTrack(C, X, Y, i, j):
    if i == 0 or j == 0:
        return ""
    elif X[i-1] == Y[j-1]:
        return backTrack(C, X, Y, i-1, j-1) + X[i-1]
    else:
        if C[i][j-1] > C[i-1][j]:
            return backTrack(C, X, Y, i, j-1)
        else:
            return backTrack(C, X, Y, i-1, j)
```

LCS backtrack all

- LCS 모두 출력하기

```
def backTrackAll(C, X, Y, i, j):
    if i == 0 or j == 0:
        return set([""])
    elif X[i-1] == Y[j-1]:
        return set([Z + X[i-1] for Z in backTrackAll(C, X, Y, i-1, j-1)])
    else:
        R = set()
        if C[i][j-1] >= C[i-1][j]:
            R.update(backTrackAll(C, X, Y, i, j-1))
        if C[i-1][j] >= C[i][j-1]:
            R.update(backTrackAll(C, X, Y, i-1, j))
        return R
```

LCS 실행 결과

- 실행 결과: https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Longest_common_subsequence

```
X = "AATCC"
Y = "ACACG"
m = len(X)
n = len(Y)
C = LCS(X, Y)

print "Some LCS: '%s'" % backTrack(C, X, Y, m, n)
print "All LCSs: %s" % backTrackAll(C, X, Y, m, n)
```

```
Some LCS: 'AAC'
All LCSs: set(['ACC', 'AAC'])
```

어디서 사용할까?

- git과 같은 version control system (documentation diff)

```
27  28 +    # check chnnel file exist?
28     -    file_path = file_dir + channel_url
    29 +    file_path = file_dir + channelnumber + '_' + time.strftime('%Y%m%d', stime) + '.html'
29  30      if not os.path.exists(file_path): # If not, download channel info.
    31 +        time.sleep(3)
30  32          response = urllib.request.urlopen(channel_url)
31     -        html = response.readall()
    33 +        html = response.read()
32  34          html_file = open(file_path, 'w')
33     -        html_file.write(html)
    35 +        html_file.write(html.decode('euc-kr'))
34  36          html_file.close()
```

LCS Print Diff

- 다른 부분을 표시해서 출력하는 방법

```
def printDiff(C, X, Y, i, j):
    if i > 0 and j > 0 and X[i-1] == Y[j-1]:
        printDiff(C, X, Y, i-1, j-1)
        print "   " + X[i-1]
    else:
        if j > 0 and (i == 0 or C[i][j-1] >= C[i-1][j]):
            printDiff(C, X, Y, i, j-1)
            print "+ " + Y[j-1]
        elif i > 0 and (j == 0 or C[i][j-1] < C[i-1][j]):
            printDiff(C, X, Y, i-1, j)
            print "- " + X[i-1]
```

과제1: LCS 손으로 따라가기

과제1) 알고리즘 이해

- LCS 알고리즘 손으로 풀어보기
  - LCS 구현물에 입력해서 비교
  - 과정 필수로 포함! (어떻게 이런 결과가 나오는가?)
  - Backtracking 도 해볼 것
- XMJYAUZ & MZJAWXU 처럼 긴 단어로 해볼 것

과제2: 비속어 탐지 프로그램

과제2) 비속어 탐지 프로그램

- 2018년 가을학기 SCSC 알고리즘 마지막 과제
- 자유롭게 설계!
- 필수 조건
  - 욕설 중간에 다른 글자가 들어가도 식별할 수 있어야 함
  - 오탐지(욕설이 아니지만 욕설로 탐지)는 해도 괜찮음

```
/home/harny/PycharmProjects/SCSC/venv/bin/python
  /home/harny/PycharmProjects/SCSC/week12/word_banner.py
비속어인지 확인할 단어를 입력하세요: 개
비속어가 아닙니다.
비속어인지 확인할 단어를 입력하세요: 개복치
비속어가 아닙니다.
비속어인지 확인할 단어를 입력하세요: 개아이
비속어가 있습니다. {'개아이'}
비속어인지 확인할 단어를 입력하세요: 시쓰고 싶다.
비속어가 아닙니다.
비속어인지 확인할 단어를 입력하세요: 시123바!!!
비속어가 있습니다. {'시바'}
비속어인지 확인할 단어를 입력하세요: 시바견은 귀엽다.
비속어가 있습니다. {'시바'}
비속어인지 확인할 단어를 입력하세요: 종료합니다.

Process finished with exit code 0
```

과제2) 하드코어 비속어 탐지 프로그램

- 선택 과제로 안 해도 됨: 추가점수 보장되는 것은 아님!
- 자모음으로 끊어서 입력해도 식별하는 방법?
  - 예시) ㅅ ㅣ ㅂ ㅏ ⇒ 시바 욕설 탐지
- 참고 자료(by 튜터)
  - https://github.com/bluedisk/hangul-toolkit
  - https://lovit.github.io/nlp/2018/08/28/levenshtein_hangle/

기타 유용한 정보

과제 요약: LCS 와 LCS 응용 프로그램

- Longest Common Subsequence 설명
  - LCS 구현물에 입력해서 비교
  - 과정 필수로 포함! (어떻게 이런 결과가 나오는가?)
  - Backtracking 도 해볼 것
- 비속어 탐지 프로그램
  - 자유롭게 설계해서 구현해볼 것!

실습 숙제 제출

- 숙제 제출 기한: 2018. 12. 05. 23:59:59
  - 실습 전 날
- 파일 제목: AL_학번_이름_12.zip
  - 파일 제목 다를 시 채점 안 합니다.
  - .egg 안 됨!

실습 숙제 제출할 것

- 2가지 파일을 제출
- AL_학번_이름_숙제번호.zip
  - 소스코드
    - Pycharm을 사용했을 경우 Project 디렉터리에 .idea, venv 같은 디렉터리는 제외
    - Jupyter + IPython을 사용했을 경우 ‘File - Download as’ 에서 .py 다운로드 가능
  - AL_학번_이름_숙제번호.pdf
    - 보고서는 무조건 .pdf
    - .hwp, .doc 등 채점 안 함

실습 보고서에 들어가야할 것

- 목표(할 일)
- 과제를 해결하는 방법
  - 알아야 할 것
- 과제를 해결한 방법
  - 주요 소스코드: 굳이 소스코드 전체를 붙일 필요는 없음
- 결과화면
  - 결과화면 설명(해석), 테스트코드 통과
- 보고서는 기본적으로 '내가 숙제를 했음'을 보이는 것
  - 지나치게 대충 작성하면 의심하게 됨

출석부 및 실습 점수가 궁금하다면?

- 출석부 및 실습 채점표
  - 수업 시작 후 30분까지 지각, 이후 결석
  - 실습 딜레이 1일당: -2점
    - 딜레이 2일까지: -2
    - 이후 -1씩 추가
- 튜터의 테스트 결과

질문이 생기면?

- 이름: 문현수
- 전공: 통신및보안
- 과정: 석박사통합과정 8학기
- 연구실: 데이터네트워크연구실(공5633)
- 메일: munhyunsu@cs-cnu.org
- 알고리즘은 함께 해결해가는 과목이므로 과감하게 연락
- 이메일로 처리가 안 되는 급한일: 문자/전화 등