알고리즘 실습

180927 - 재귀, 복잡도

오늘의 목표

- Recursion 연습
  - 거스름돈 반환 예제

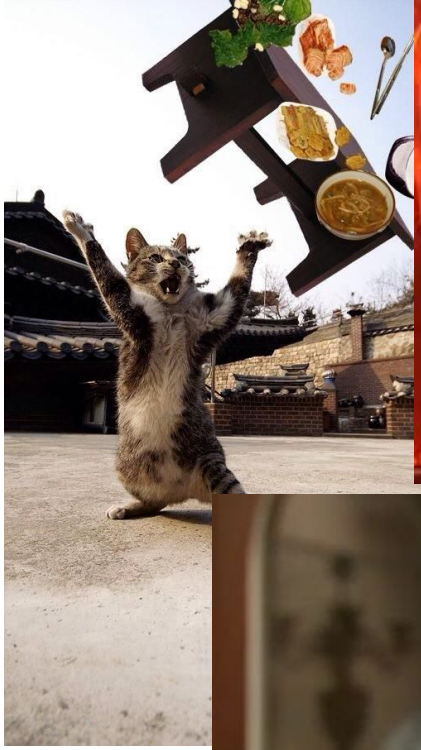
Feedback

지난 과제: Queue 응용 프로그램

- 제출: 40 / 41 (97.56%)
  - 시뮬레이션 구현 못 한 사람을 제외: 33/41 (80.49%)
- 질문: 145개

Feedback

- 추석이라 쉽게 낸다면서...
  - 실습 코드에 대한 자세한 설명이 필요함
  - 시뮬레이션 코드는 사막에서 바늘 찾기 아님?
- 실력이 있어서가 아니라 스터디를 통해 간신히 제출률 95% 맞추는 거임
- 변수/함수 이름으로 직관이 없음: 설명이 더 필요
- Python 3 공부할 수 있는 시간(실습)



(간단한) 실습: 잔돈 거슬러주기

잔돈 거슬러주기: 목표

- 잔돈을 최소한의 동전 수로 되돌려주자!

잔돈 거슬러주기 알고리즘

- 잔돈이랑 같은 동전이 있으면 그대로 반환
- 잔돈보다 작은 동전을 하나 선택해서 순회

잔돈 거슬러주기 코드!

- 코딩해보자!

0.7 Make change with recursion (1,5,10, 25 cents) for 63 cents

```
In [15]: def recMC(coinValueList,change):
             global counter
             counter = counter + 1
             minCoins = change
             if change in coinValueList:
               return 1
             else:
                for i in [c for c in coinValueList if c <= change]:
                   numCoins = 1 + recMC(coinValueList,change-i)
                   if numCoins < minCoins:
                      minCoins = numCoins
             return minCoins

         counter = 0
         print(recMC([1,5,10,25],63))
         print(counter)
```

```
6
67716925
```

잔돈 거슬러주기: code

- inline code 등을 제거한 버전
- 손으로 그림을 그려봐야 이해가 쉬움

```
거스름돈: 63
3
33160648
Time:  18.364
```

```python
def make_change(coin_value_list, coin_change):
    """
    잔돈
    """
    global counter
    counter = counter + 1
    min_coins = coin_change

    if coin_change in coin_value_list:
        return 1
    else:
        coins = list()
        for coin in coin_value_list:
            if coin <= coin_change:
                coins.append(coin)
        for coin in coins:
            num_coins = 1 + make_change(coin_value_list, coin_change - coin)
            if num_coins < min_coins:
                min_coins = num_coins
    return min_coins
```

좀 더 빠르게!

- 현재 코드의 문제점: 계산했던 것을 또 계산!
- 계산한 것을 저장해두자!

0.8 Make change with memory or caching (1,5,10, 25 cents) for 63 cents

```
In [19]: def recDC(coinValueList,change,knownResults):
             global counter
             counter = counter + 1
             # print(knownResults)

             minCoins = change
             if change in coinValueList:
                knownResults[change] = 1
                return 1

             # known results are saved and not computed again !!!
             elif knownResults[change] > 0:

                return knownResults[change]
             else:
                 for i in [c for c in coinValueList if c <= change]:
                   numCoins = 1 + recDC(coinValueList, change-i,
                                        knownResults)
                   if numCoins < minCoins:
                      minCoins = numCoins
                      knownResults[change] = minCoins
             return minCoins

         counter = 0
         print(recDC([1,5,10,25],63,[0]*64))
         print(counter)
```

잔돈 거슬러주기 개선된 코드! with cache

- inline code 등을 제거한 버전
- 손으로 그림을 그려봐야 이해가 쉬움
- known_result는 어떻게 변하는가?

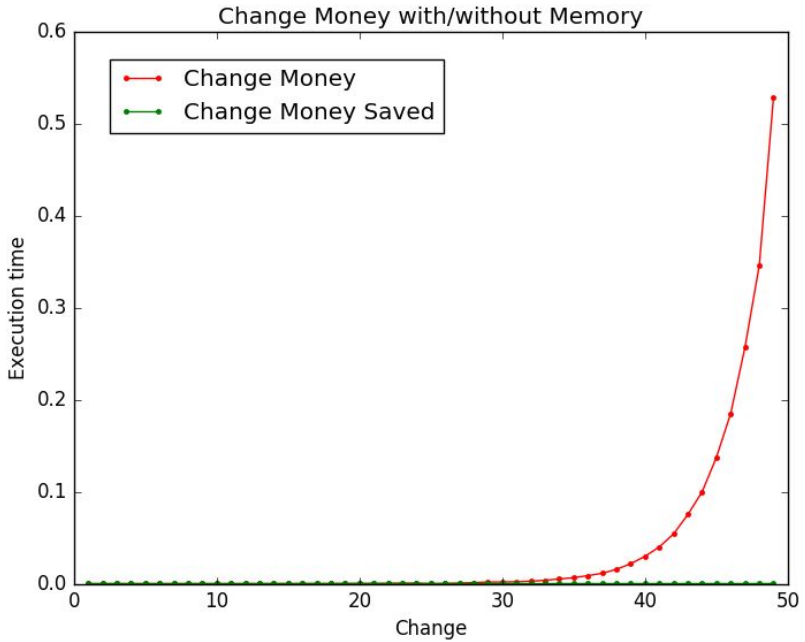
```
거스름돈: 63
3
262
Time:   0.002
```

```python
def make_change(coin_value_list, coin_change, known_result):
    """
    잔돈
    """
    global counter
    counter = counter + 1
    min_coins = coin_change

    if coin_change in coin_value_list:
        known_result[coin_change] = 1
        return 1
    elif known_result[coin_change] > 0:
        return known_result[coin_change]
    else:
        coins = list()
        for coin in coin_value_list:
            if coin <= coin_change:
                coins.append(coin)
        for coin in coins:
            num_coins = 1 + make_change(coin_value_list, coin_change - coin, known_result)
            if num_coins < min_coins:
                min_coins = num_coins
                known_result[coin_change] = min_coins
    return min_coins
```

얼마나 빨라질까?

- 메모리 유무에 따른 실행 시간 차이 차트
- 본래 실습/과제로 해보려 했으나 실습 기간이 부족하므로 패스
    - 나중에 해볼 시간이 옴



편의성(기능)을 개선해보자!

- '어떤' 동전을 '몇 개' 줘야하는지는 아직 모름
  - 동전을 '몇 개' 줘야하는지만 출력됨
- How?

```
거스름돈: 63
(3, 33160648, [21, 21, 21])
Time:  27.278
```

```python
def main():
    """
    잔돈 거스르기
    """
    coin_value_list = [1, 5, 10, 21, 25]  # 동전의 종류
    # coin_value_list = [1, 5, 10, 25]  # 동전의 종류
    random.shuffle(coin_value_list)
    coin_change = input('거스름돈: ')  # 거스름돈
    coin_change = int(coin_change)  # 인트형으로 바꾸자!

    start = time.time()
    print(make_change(coin_value_list, coin_change))
    end = time.time()
    return_time = end-start
    print('Time: {0:7.3f}'.format(return_time))


if __name__ == '__main__':
    main()
```

편의성(기능)을 개선해보자! with cache

- 메모리 기능이 있는 버전도 개선해보자!
- 2개 모두 make_change()를 구현하면 됨

```
거스름돈: 63
(3, 244, [21, 21, 21])
Time:   0.002
```

```python
def main():
    """
    잔돈 거스르기
    """
    coin_value_list = [1, 5, 10, 21, 25]  # 동전의 종류
    # coin_value_list = [1, 5, 10, 25]  # 동전의 종류
    random.shuffle(coin_value_list)
    coin_change = input('거스름돈: ')  # 거스름돈
    coin_change = int(coin_change)  # 인트형으로 바꾸자!
    known_result = list()
    for index in range(0, coin_change+1):
        known_result.append(None)

    start = time.time()
    print(make_change(coin_value_list, coin_change, known_result))
    end = time.time()
    return_time = end-start
    print('Time: {0:7.3f}'.format(return_time))


if __name__ == '__main__':
    main()
```

테스트코드

- 잔돈거스르기 테스트코드
  - 입력 잔돈을 무작위로 섞어서 넣어줌 (기존 전역변수를 통한 해결 불가)
  - 0~50원까지 테스트
- 주의
  - 자신의 코드 반환값에 따라서 result[2]의 인덱스를 알맞은 인덱스로 변경해줘야함

```python
import unittest
import random

from change_money_list import make_change


class TestMakechange(unittest.TestCase):
    def test_makechange(self):
        coin_value_list = [1, 5, 10, 21, 25]
        for change in range(0, 51, 1):
            random.shuffle(coin_value_list)
            result = make_change(coin_value_list, change)
            print(sum(result[2]) == change, result[2], change)
            self.assertEqual(change, sum(result[2]))
```

```python
import unittest
import random

from change_money_saved_list import make_change


class TestMakechange(unittest.TestCase):
    def test_makechange(self):
        coin_value_list = [1, 5, 10, 21, 25]
        for change in range(0, 101, 1):
            random.shuffle(coin_value_list)
            known_result = list()
            for index in range(0, change + 1):
                known_result.append(None)
            result = make_change(coin_value_list, change, known_result)
            print(sum(result[2]) == change, result[2], change)
            self.assertEqual(sum(result[2]), change)
```

테스트코드 결과

- 0부터 50(혹은 100)까지 차례대로 결과가 맞을 경우 True와 함께 결과 출력
- 마지막 Ran 3 test OK 가 뜨면 성공
- 그.런.데
  - 이것이 정확하지 않은 테스트 일 수 있음
  - Why?
  - 그래서 ‘case’ 2개를 넣어둠

```
True [10, 5, 25, 25, 21] 86
True [10, 10, 21, 21, 25] 87
True [25, 21, 21, 21] 88
True [21, 21, 21, 21, 5] 89
True [10, 5, 25, 25, 25] 90
True [10, 10, 21, 25, 25] 91
True [21, 21, 25, 25] 92
True [25, 25, 21, 21, 1] 93
True [10, 21, 21, 21, 21] 94
True [10, 10, 25, 25, 25] 95
True [21, 25, 25, 25] 96
True [1, 21, 25, 25, 25] 97
True [10, 21, 21, 21, 25] 98
True [5, 21, 21, 21, 21, 10] 99
True [25, 25, 25, 25] 100


Ran 3 tests in 0.060s

OK
```

과제) 기능이 개선된 재귀(with/without M.)

1. change_money_list.py 구현
    - change_money_list_test.py 로 테스트
2. change_money_saved_list.py 구현
    - change_money_saved_list_test.py 로 테스트

기타 유용한 정보

실습 숙제 제출

- 숙제 제출 기한: 2018. 10. 03. 23:59:59
  - 실습 전 날
- 파일 제목: AL_학번_이름_04.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