Dynamic Programming

(동적 계획법)

dynamic programming

- 문제 해결 전략 중 하나
- 문제를 더 작은 부분 문제로 나누고, 작은 문제를 해결함으로써 최종 문제를 해결
  - divide and conquer 또는 decrease and conquer와 유사
- 핵심적 특징
  - 반복해서 풀어야 하는 부분 문제가 있음
  - 부분 문제의 해답을 어딘가에 저장하여 활용 (memoization)
    - 메모리를 활용함으로써 해답을 구하는 시간을 단축함
  - 예> 피보나치 수열

dynamic programming - 피보나치 수열

- f(n) = f(n-1) + f(n-2)
- 피보나치 수열을 재귀적 기법을 통해 구한다면...
  - f(8) = f(7) + f(6)

    = f(6) + f(5) + (f5) + f(4)

    = f(5) + f(4) + f(4) + f(3) + f(4) + f(3) + f(3) + f(2)

    = ...
  - 동일한 부분 문제를 반복하여 풀고 있음을 알 수 있음.. 가령 f(8)을 구하기 위해서는 f(5)를 여러 번 계산하게 됨.

dynamic programming - 피보나치 수열

- 피보나치 수열을 재귀적 기법을 통해 구한다면...
  - n이 커질수록 성능이 급격히 저하
    - 동일한 부분 문제를 반복해서 풀고 있기 때문.
  - 이미 구한 f(k)를 메모리에 저장하고 추후 필요할 때마다 이를 활용하면.. 성능을 크게 개선할 수 있음 => dynamic programming 기법 활용

문제1

- 피보나치 수열을 재귀적 기법으로 구하시오. (without dynamic programming)
  - fibo(n) = fibo(n-1) + fibo(n-2)

  - base case?
    fibo(1) = 1
    fibo(2) = 1

문제1

- 파이썬 코드

```
def fibo(num) :
    if (num == 1 or num == 2) :
        return 1
    return (fibo(num - 1) + fibo(num - 2))

print(fibo(10))
```

문제2

- 피보나치 수열을 dynamic programming을 활용하여 재귀적 기법으로 구하시오.
  - fibo(n) = fibo(n-1) + fibo(n-2)

  - base case?
    fibo(1) = 1
    fibo(2) = 1

  - memoization?
    - 한번 수행한 연산 결과를 어딘가에 저장 => dictionary, list 등

문제2

```
def fibo(num, flist) :
    if (num == 1 or num == 2) :
        return 1

    if (이미 구했다면.. 즉 리스트에 답이 저장되어 있다면) :
        return flist[num]
    else :
        …….

print(fibo(10))
```

문제2

- 파이썬 코드 (list를 이용한 memoization)

문제2

- 파이썬 코드 (list를 이용한 memoization)
  - 리스트를 초기화하는 선행 작업이 필요함

```
fibos = []
for i in range(1001):
    fibos.append(-1)

print(fibo_dyn(1000, fibos))
```

파이썬 dictionary

- 리스트와 유사하게 여러 item들을 저장
- 개별 아이템은 key와 value로 구성됨
- key를 통해 indexing 수행

파이썬 dictionary

- 변수 정의
  - {} 사용
  - mydict = {‘key0’:’value0’, ‘key1’:’value1’}
- key를 이용한 인덱싱
  - [] 사용
  - print(mydict[key0])
    - value0 출력됨
  - 만약 key가 발견되지 않으면 KeyError 예외 발생함

파이썬 - 예외처리

- 예외처리
  - try, except, else, finally 구문
  - try
    - 예외 발생 가능한 구문
  - except
    - try 구문에서 예외 발생 시 실행
  - else
    - try 구문에서 예외 발생하지 않은 경우 실행
  - finally
    - 항상 실행

파이썬 - 예외처리

```
fibos = {0:'a'}
try :
    print(fibos[0])
except KeyError :
    print('no value for key 0')
else :
    print('no error')
finally :
    print('This is always printed')
```

파이썬 - 예외처리

```
fibos = {0:'a'}
try :
    print(fibos[1])
except KeyError :
    print('no value for key 1')
else :
    print('no error')
finally :
    print('This is always printed')
```

문제2

- 파이썬 코드 (dictionary를 이용한 memoization)

Summary

- dynamic programming
  - 어떤 문제의 해답이 부분 문제들의 해답을 통해 구해지고, 동일한 부분 문제들이 반복될 때 적용 가능함.
  - 추가 메모리를 사용하여 이미 해결한 부분 문제들의 해답을 기억하고 (memoization), 동일한 부분 문제 해결 시 저장된 해답을 활용함으로써 해답을 찾는 시간을 단축.
  - 예> 피보나치 수열