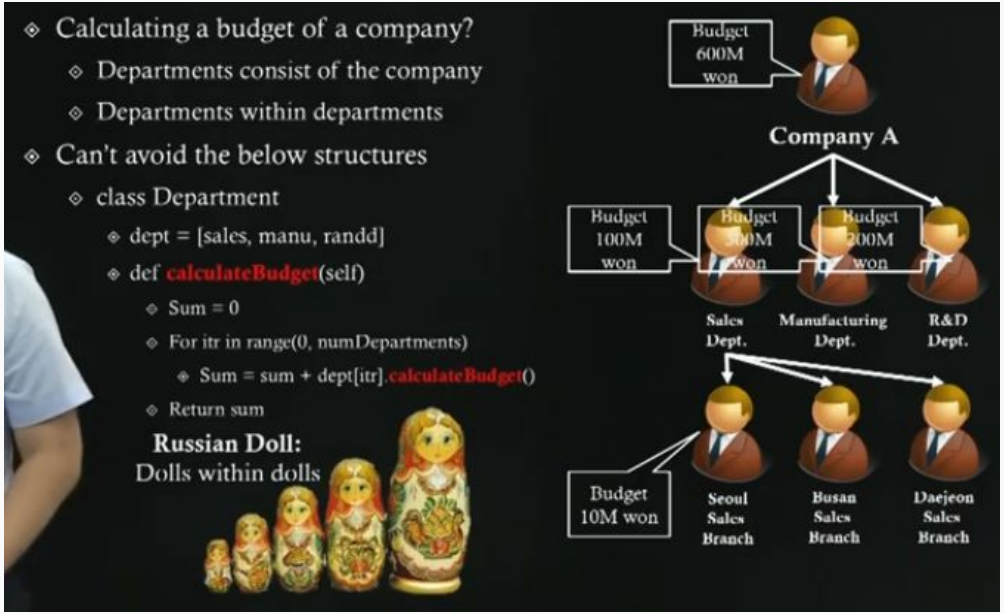
4장. Recursions and Dynamin Programming

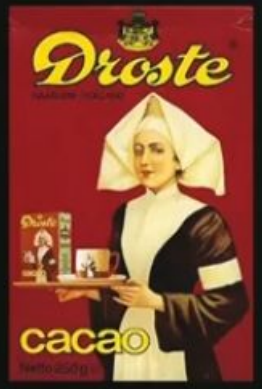
Repeating Problems and Divide and Conquer



- 계층적으로 조직이 있어서 어떻게 쪼개지는지를볼수 있다.
- 위에 뷰와 아래의 뷰는 유사하다. 달라진것은 예산의 크기가 줄어들어다
- repeation problem은 이렇게 큰 하나의 문제가 정의 되면 그 문제를 숙의해서 다시 반복되는 또 다른 문제로 만든다는 것이다. 다만 그 문제가 작아지게끔 만드는 것이다.
- 문제를 잘게 쪼개어서 문제를 해결해 나가는 과정을 divide and conquer라고 한다.
- 상위에도 burget 계산이 있고 밑에도 burget 계산이 있어서 고 상위 함수를 하위에서 동일한 함수를 콜해서 사용한다. 다만 둔화된 디바이드된 작은 문제로 동일한 함수를 콜하는 것이 recursion이 되고 그게 리피팅 프라블럼을 프로그램으로써 풀어보는 과정이 되고 그리고 그 컨셉은 디바이드 컨컬을 따르는 것이다.

More examples

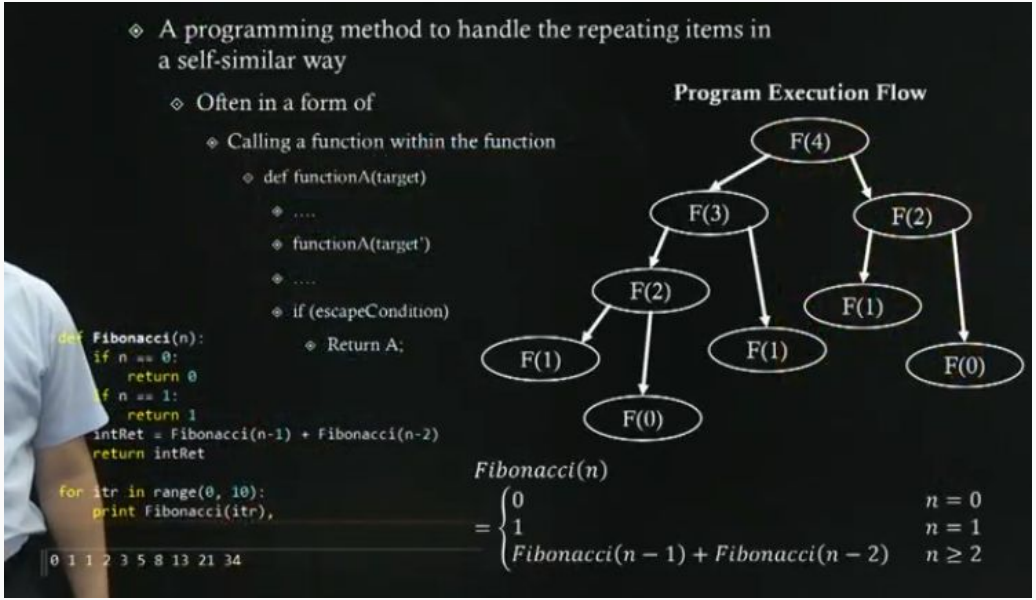
- Factorial
  - $Factorial(n) = \begin{cases} 1 & if\ n = 0 \\ n \times (n-1) \times \cdots \times 2 \times 1 & if\ n > 0 \end{cases}$
  - Repeating problems?
    - $Factorial(n) = \begin{cases} 1 & if\ n = 0 \\ n \times Factorial(n-1) & if\ n > 0 \end{cases}$
- Great Common Divisor
  - GCD(32,24) = 8
  - Euclid's algorithm
    - GCD(A, B)=GCD(B, A mod B)
    - GCD(A, 0)=A
- Commonality
  - Repeating function calls
  - Reducing parameters
  - Just like the mathematical induction



Self-Similar

- Factorial !
- 팩토리아 속에 펑션이 n-1이 되면 또 같은 펑션이 나온다.
- great common divisor 최대 공약수
- gcd(32, 24)에서 32를 24로 나누는 몫은 1이고 나머지는 8
- gcd(24, 8)dptj 24를 8로 나누면 몫은 3이고 나머지는 0
- gcd(8, 0) -> 최대공약수는 8
- 사이즈는 작아지고 있고 펑션은 동일하다
- 공통점은 펑션콜은 반복되고 사이즈는 줄어든다.

Recursion



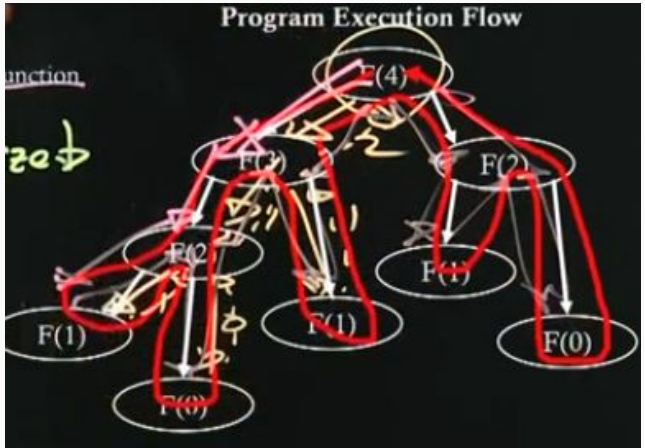
- recursion은 리피팅하는 같은 방법으로 핸들하는것이다.
- sewdo code 실제로 안돌아가는데 이렇게하는것이다라고 간략히 적어논것이다.
- fuctionA를 콜해서 사용하지만 target은 사이즈가 작아지는 것이다.

Recursion

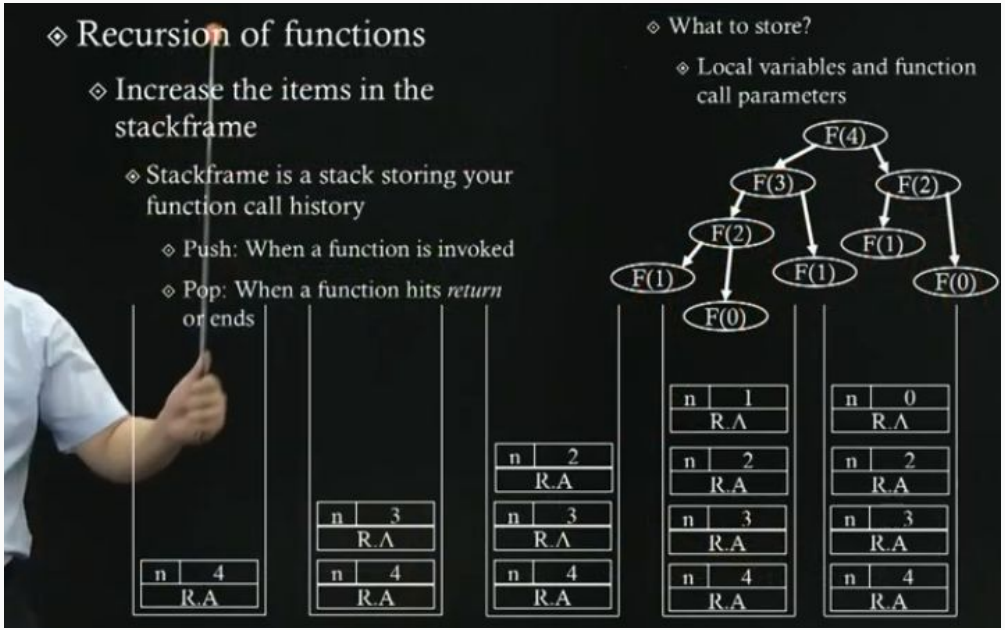
```python
def Fibonacci(n):
    if n == 0:
        return 0
    if n ==1:
        return 1
    intRet = Fibonacci(n-1) + Fibonacci(n-2)
    return intRet
```

```
In[36]:
   ...: for itr in range(0, 10):
   ...:     print(Fibonacci(itr))
   ...:
0
1
1
2
3
5
8
13
21
34
```

- 0 1 1 2 5 8 13 21 34
- 앞에 0과 1은 규칙에 따라서 작성한다.
- 두개를 가지고 다음 값을 만든다.
- 정의한 함수를 두번 콜하고 있다.
- 사이즈는 계속 죽어든다.
- 탈출구문은 n이 0이거나 1이면 더이상 자기 자신을 재귀 리컬전을 하지 않고 탈출하게 된다.



Recursions and Stackframe



- recursion은 재귀호출이 계속 일어나는것
- stackframe은 stack인데 특별한 목적을 가진 스택이다. 펑션콜의 히스토리의 진행된 역사를 기록하고 있는 것을 스택프레임이라고한다.
- 스택에 대해서 할 수 있는 오퍼레이션은 푸쉬와 팝 두가지만 있다.
- 푸쉬라는 것은 펑션이라는 것이 콜되면(invoked)
- 팝은 펑션이 리턴을 타 가지고 펑션을 받고 나가게 되면 그러면 탑이 일어난다.
- 어떤것을 저장해 놓느냐가 또한 핵심인데 이것은 로컬 베리어블과 펑션 파라미터를 저장해놓고 해야 된다.
- 로컬 버라이어블은 펑션 속에서만 접근 가능한 within 펑션 버라이블이 된다.
- 펑션 콜 파라미터는 특정 펑션 콜 인스턴스에 할당된 파라미터 예를 들어 f4라고 하면 이자가 바로 펑션 콜의 파라미터가 된다.