자료구조 (2 분반) – 2 주차

Jong-Kyou Kim, PhD

2016-03-10

Fibonacci 관련 질문

- f_n 을 계산하는데 2^n 번 더하기 하는 것이 맞나요?
  - F_n: f_n 을 계산하는데 더하기 횟수
  - $F_0 = 0$
  - $F_1 = 0$
  - $F_2 = 1$
  - $F_3 = 2 = F_2 + F_1 + 1$
  - $F_n = F_{n-1} + F_{n-2} + 1$

Fibonacci

- 이산수학에서 배울 내용

$$f_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

Python 예제관련 질문

```
def fib(n):
  a,b = 0,1
  for k in range(n):
    a,b=b,a+b
  return a
print(fib(10))
```


Jong-Kyou Kim, PhD

알고리즘의 선택

- `fib` 와 `fib_fast` 중 어떤 것을 선택할까?
- f_n 을 빠르게 계산하는 알고리즘
  - n 값이 클 때 f_n 을 빠르게 계산하는 알고리즘

$\longrightarrow$ 얼마나 큰 것이 큰 것일까?

Growth function

- 두 개의 알고리즘 `f`, `g`
  - 알고리즘을 수행하는데 걸리는 시간: $f(n), g(n) > 0$
  - n 에 관계 없이 $f(n) < g(n)$ 이라면 어떤 알고리즘을 선택할까?
  - $n > n_0$ 라는 조건을 만족하기만 하면 $f(n) < g(n)$ 이라고 보장할 수 있다. 어떤 알고리즘을 선택할까?

Mathmatical definition

- 두 개의 알고리리즘을 수행하는데 걸리는 시간
  - $f(n) = 1,000,000,000 \times n$
  - $g(n) = 0.000000001 \times 2^n$
- 어떤 알고리즘이 좋을까?
  - $n_0 > 65$ 이고 $n > n_0$ 라면?
  - $n = 66$:
    - $f(n) = 66,000,000,000$
    - $g(n) = 73,786,976,295$

Big-O: $O(g(n))$

- $O(n)$: 여러 함수의 집합
  - 어떤 상수 c, n_0 를 지정하여
  - $0 \leq f(n) \leq c \cdot g(n), n > n_0$ 를 만족시키도록

$\longrightarrow$ $O(g(n)) = \{f(n) | 0 \leq f(n) \leq c \cdot g(n), n > n_0\}$

- 예
  - $f(n) = n$, $g(n) = 1,000,000,000 \times n$
  - $f(n), g(n) \in O(n)$
- 그러나
  - $h(n) = n^2$ 라면 $h(n) \notin O(n)$
  - $f(n), g(n) \in O(n^2)$

$\longrightarrow$ 이보다 나쁠 수는 없다는 정도는 보장

Big-O: $O(g(n))$

- 어떤 알고리즘이 $O(n^2)$ 라는 사실을 증명하였다. 다음 중 거짓인 것은?
  - 이 알고리즘은 $g(n) = n^2$ 의 시간보다 빨리 결과를 도출할 것이다
  - 이 알고리즘이 $g(n) = n$ 의 시간이 걸리는 알고리즘보다 빨리 결과를 도출하는 경우는 절대 없다
  - 이 알고리즘은 $O(n^3)$ 에도 속한다
  - 이 알고리즘은 $O(2^n)$ 에도 속한다

Big-Omega: $\Omega(g(n))$

- $\Omega(n)$: 여러 함수의 집합
  - 어떤 상수 c, n_0 를 지정하여
  - $f(n) \geq c \cdot g(n) \geq 0, n > n_0$ 를 만족시키도록

  $\longrightarrow$ $\Omega(g(n)) = \{f(n) | f(n) \geq c \cdot g(n) \geq 0, n > n_0\}$

$\longrightarrow$ 이보다 좋을 수는 없다는 것을 보장

Big-Theta: $\Theta(g(n))$

- Big-O, Big-Omega 를 모두 증명한 경우

Sorting concept

- 다음 중 가장 작은 수는?

```
x = [40694 ,
73593 ,
13612 ,
65541 ,
19386 ,
2347 ,
26723 ,
42533 ,
27999 ,
96272 ]
```

- 네 번째로 작은 수는?

Sorting concept

- 네 번째로 작은 수는?

```
x = [ 2347 ,
13612 ,
19386 ,
26723 ,
27999 ,
40694 ,
42533 ,
65541 ,
73593 ,
96272 ]
```

$\longrightarrow$ x.sort()

Sorting algorithm

- Bubble sort
  - 인접한 두 개를 비교하여 교환
- Insertion sort
  - 앞에서 차곡 차곡 정렬해 가는 것

Bubble sort

- bubble sort

```
4     3  2  1
3  4     2  1
3  2  4     1
3  2  1  4
```

- insertion sort

```
   4     3  2  1
3  4     2  1
   3  4  2  1
2  3  4     1
```

Reading

- 이번 주: Chapter 1-3
- 다음 주: Chapter 10

Wrap-up

- Regarding performance of algorithms, we are interested in solving large problems
- For sufficiently large problem, the execution time is dominated by its growth and denote it as a growth function
- We prefer slow growing algorithms to fast growing ones
- We can classify growth functions using Big-O, Big-Omega and Big-Theta analysis
- Each gives upper, lower and exact bounds for growth functions