\*점근적 표기법

알고리즘이 얼마나 오래 걸리는가?

알고리즘 실행시간

1.입력값의 크기에 따른 알고리즘의 실행 시간을 알아야함(배열이 커지면 선형 검색과 이진 검색의 최대 추측 횟수가 함께 증가함)

2.입력값의 크기에 따라 이 함수가 얼마나 빨리 커지는지(실행 시간의 성장률)

-불필요한 부분은 버리고 가장 중요한 부분만 추려내서 함수를 간소화해야함

Ex) 입력값 크기가 n인 알고리즘이 6n^2+100n+300이라는 기계 명령을 받는다

->알고리즘의 실행시간은 n^2으로 기하급수적으로 커짐(계수는 중요하지 않다)

- 상수 계수와 불필요한 항목을 제거하면 점근적 표기법을 사용함

=> 점근적 표기법의 세 가지 형태(big-오 표기법, big-세타 표기법,big-오메가 표기법)

\*시간 복잡도 분석

-input n에 대해 알고리즘이 문제를 해결하는데 얼마나 오랜 시간이 걸리는지를 분석하는 것

-‘실행시간’이라는 관점에서 알고리즘의 효율을 측정

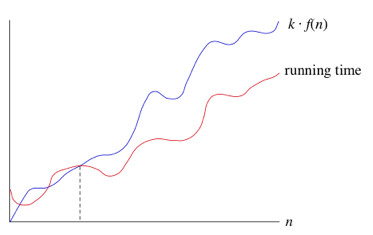
-연산자의 숫자 or 연산과정의 수에 주목

\*빅 오 표기법

-빅-o표기법 정의 :

함수 f(n)과 g(n)의 자연수를 양의 실수로 대응시키는 함수라고 할 때, 다음을 만족시키는 양의 상수 c>0와 정수n0>=1이 존재하면 “f(n) is O(g(n))” 혹은 “f(n) ∈O(g(n))”이라고 쓰고, “f(n) is big-Oh of g(n)”혹은 “f(n) is order of g(n)”으로 읽는다.

f(n)<=cg(n), for n>=n0 를 만족하는 c를 찾을 수 있는가



-점근적 상한선에 대해서는 big-O표기법을 사용하는데 이는 충분히 큰 입력 크기에 대하여 실행 시간에 상한값을 두고 제한하기 때문

-big-o표기법은 점근 상한만 두고 점근적으로 상한 하한이 좁지 않음

1) O(1) – constant time

Ex1)

public void print\_FirstItem(int [] arrayOfItems){

System.out.println(arrayOfItems[0]);

}->아무리 많은 item이 있어도 O(1) (한 개의 index에만 접근하므로)

-Push,pop on stack , access hash table..

2) O(logn) – log time

-Binary search,tree access, insertion, deletion, search..

3) O(n) – linear time

Ex1)

인자로 크기가 n인 하나의 배열을 받고 모든 배열 요소를 print-> loop(n번 접근)

-traverse tree, traverse linked list

4) O(nlogn) – log linear time

-Quick sort, heap sort..

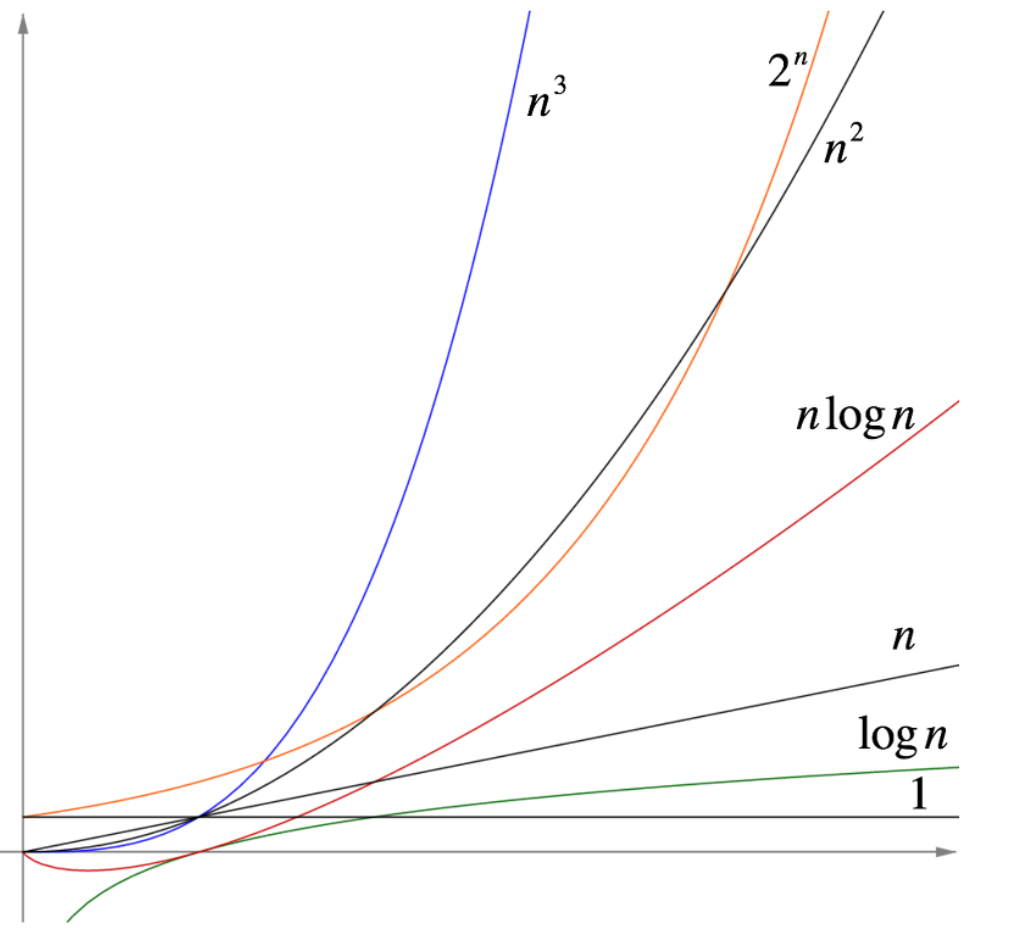
5)O(n^2) – quadratic time

Ex1)

이중 for문 -> 배열 크기가 10이라면 100번 access(n^2번)

-insertion sort, bubble sort, selection sort

Cf) O(n^3) – cubic time , O(2^n) – exponential time



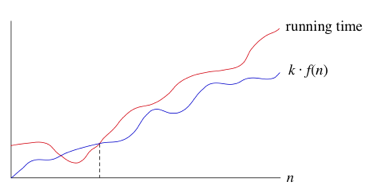
\*빅 오메가 표기법

빅-오메가 정의:

함수 f(n)과 g(n)의 자연수를 양의 실수로 대응시키는 함수라고 할 때, 다음을 만족시키는 양의 상수 c>0와 정수 n0>=1이 존재하면 “f(n) is 오메가(g(n))”이라고 쓰고, “f(n) is big-Omega of g(n)”으로 읽는다.

f(n)>=cg(n), for n>=n0

-알고리즘이 상한선 없이 최소한 어느정도 걸린다를 표현할 때



-실행시간이 Ω(f(n))이라면 n이 충분히 클 때 실행 시간은 어떤 상수 k에 대해 최소 k\*f(n)이다.

-점근적 하한선을 표현하기 위해 big-오메가 표기법을 사용하는데, 그 이유는 점근적 하한선은 충분히 큰 입력크기에서 실행시간을 밑에서 제한하기 때문

\*빅 세타 표기법

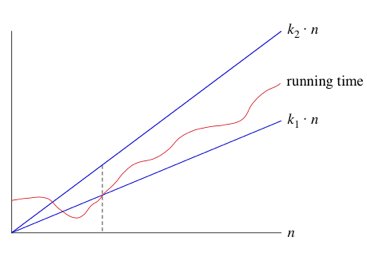
빅-세타 정의:

함수 f(n)과 g(n)의 자연수를 양의 실수로 대응시키는 함수라고 할 때, 다음을 만족시키는 양의 상수 c’>0, c”>0와 정수n0>=1이 존재하면 “f(n) is 세타(g(n))”이라고 쓰고, “f(n) is big-Theta of g(n)”으로 읽는다.

c’g(n)<=f(n)<=c”g(n), for n>=n0 (빅 O, 빅 오메가가 동시에 참이 성립하면 Big 세타가 성립한다)

-선형 검색의 최악의 경우의 실행시간은 배열 크기인 n에 따라 커진다. 이때 실행 시간을 표시하는 표기법은 세타(n)임

-세타(n) : n이 충분히 크다면 실행 시간이 어떤 상수 k1과 k2에 대해서 최소k1\*n이며 최대 k2\*n이 된다



-Big-세타 표기법을 사용하는 것은 실행 시간에 대해 점근적으로 근접한 한계값이 있다고 표현하는 것

-“점근적으로”라는 말을 쓰는 이유는 큰 값의 n에 대해서만 적용되기 때문

* 빅오는 알고리즘의 최악의 성능을, 빅오메가는 알고리즘 최고의 성능을, 빅쎄타는 좀더 정확한 알고리즘의 성능을 표시해줌.
* 이 중에서 빅오표기법을 주로 사용하는 이유는 ‘가장 최악인 상황에서도 최소 이 정도의 성능은 보장된다’ 라는 의미를 담고 있기 때문