SCCC 스터디

시간복잡도+구현

시간복잡도란?

- input 에서 output 까지 걸리는 시간
- 알고리즘의 연산을 얼마나 하는가
- == 알고리즘이 얼마나 걸리는가

시간복잡도란?

- 왜 필요한가?
  - 최적화를 위해 필요 -> 효율적
- 수행 횟수 구하는 방법
  - 가장 간단한 연산(더하기,빼기 등)의 횟수를 구한다.

```
void  Func(int *a, n)

{
    int i=0, j=0;                          1
    for (i = 0 ; i < n-1 ; i++)              n            [i => 0 to n-1 ∴ n times called]
      for(j=i+1; j<n ; j++)                  (n-1) * n    [j => 1 to n   ∴ n times called]
        if (a[i] == a[j]) a[j] = 0 ;         (n-1) * (n-1)  [if 문은 결국 n-1 번만 실행 ]
}
```

$2n^2-2n+2$

시간복잡도 표기법

- 세가지 표기법
  - Big-O 빅-오 표기법
    - 최악의 경우, 가장 보편적인 표기법

      두 함수 $f(n)$, $g(n)$이 있을 때, $n_1 \leq n$, $f(n) \leq C * g(n)$이 성립하는

      상수 C, n_1이 존재하면 $f(n) = O(g(n))$이다.

    - 시간 복잡도에 가장 영향을 미치는 차항만 뺀다) $2n^2-2n+2$ -> n^2
  - Big-Ω 빅-오메가 표기법
    - 최선의 경우

      두 함수 $f(n)$, $g(n)$이 있을 때, $n_1 \leq n$, $f(n) \geq C * g(n)$이 성립하는

      상수 C, n_1이 존재하면 $f(n) = \Omega(g(n))$이다.

      $2n^2-2n+2$ -> Ω(n^2), Ω(n), Ω(1)

시간복잡도 표기법

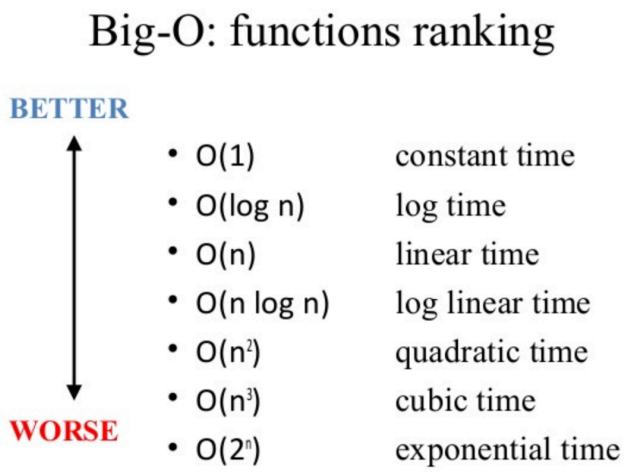
- Big-θ 빅-세타 표기법
  - 평균적인 복잡도

    두 함수 $f(n)$, $g(n)$이 있을 때, $n_1 \leq n$, $C_1 * g(n) \leq f(n) \leq C_2 * g(n)$이 성립하는

    상수 C_1, C_2, n_1이 존재하면 $f(n) = \Theta(g(n))$이다.
  - 예) 퀵소트는 평균의 경우 NLogN이므로 θ(NLogN)시간복잡도를 가진다

Big-O 표기법의 종류

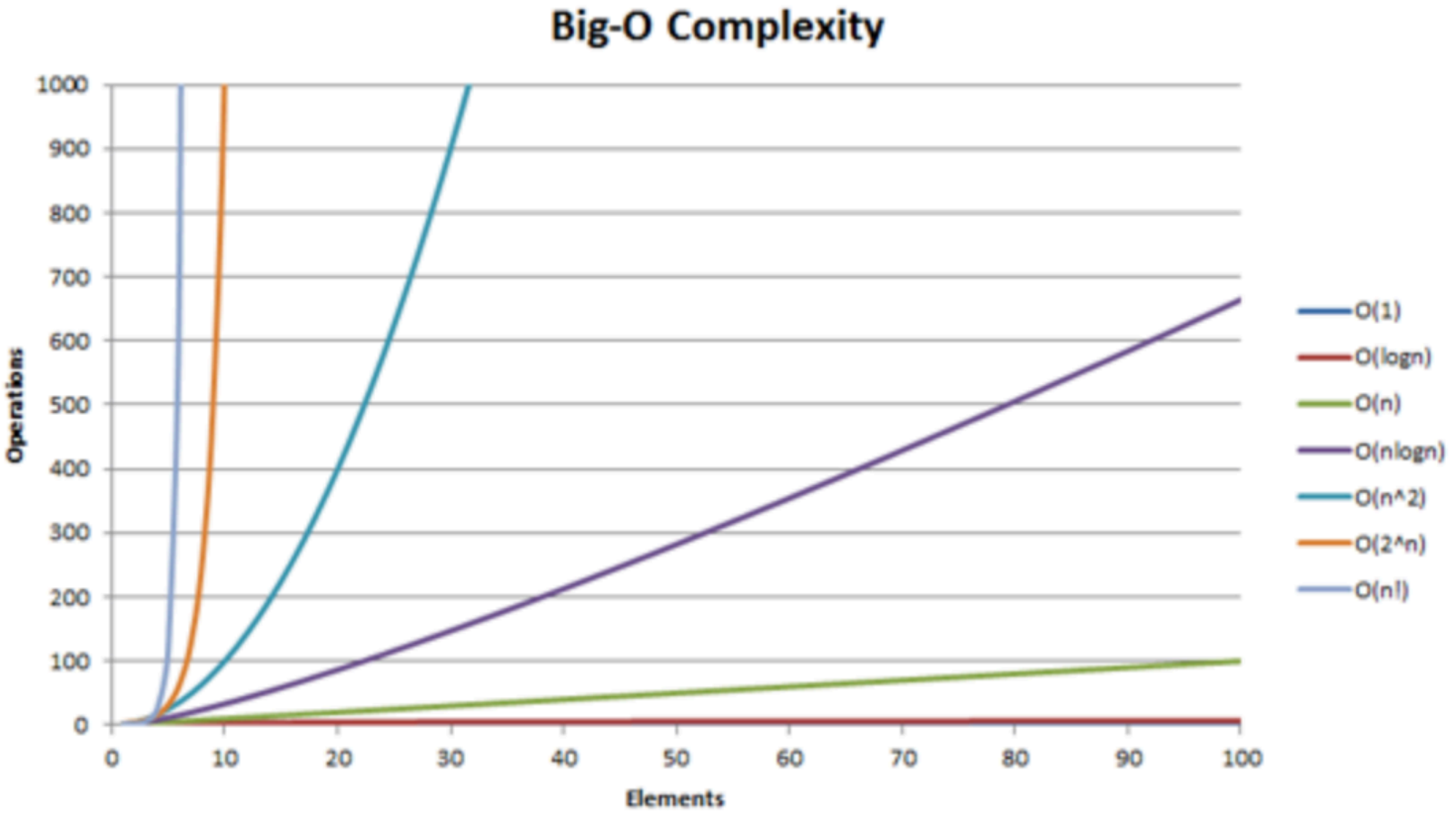
f(n)	Name
1	Constant
$\log n$	Logarithmic
n	Linear
$n \log n$	Log Linear
n^2	Quadratic
n^3	Cubic
2^n	Exponential



Big-O 표기법의 종류

- O(1) : 입력자료의 수에 관계없이 일정한 실행시간을 갖음
- O(logn) : 입력자료의 수에 따라 시간이 흐를수록 시간이 조금씩 증가
  EX > 큰 문제를 일정한 크기를 갖는 작은 문제로 쪼갤 때 나타남
  EX > 이진탐색 같은 효율이 좋은 검색 알고리즘
- O(n) : 입력 자료의 수에 따라 선형적인 실행 시간이 걸리는 경우 - 입력자료마다 일정 시간 할당
- O(nlogn) : 큰 문제를 일정 크기 갖는 문제로 쪼개고(logn+logn+ .. + logn) 다시 그것을 모으는 경우
  EX > 효율 좋은 정렬 알고리즘
  EX > quick sorting / heap sorting 등
- O(n^2) : 이중 루프내에서 입력 자료를 처리할 때
  EX > 인접행렬이용한 bfs/dfs 알고리즘
- O(n^3) : 삼중 루프 내에서 입력자료 처리할 때

Big-O 표기법의 종류


Big-O Complexity
Operations
Elements
1000
900
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
O(1)
O(logn)
O(n)
O(nlogn)
O(n^2)
O(2^n)
O(n!)

시간복잡도

- 대부분의 대회 문제의 수행 시간 제한은 1초
  - 1초 ~= 1억 번의 연산 수행
  - 적절한 알고리즘을 선택해 수행 시간 내에 답안이 출력하도록 짠다.
  - 1초를 넘기면... 시간 초과!

시간 초과

대중적인 알고리즘의 시간복잡도

- 단순 연결리스트
  - 조회 – O(N)
  - 삽입 – O(N)
  - 삭제 – O(N)
- 정렬
  - 선택정렬 – O(N^2)
  - 버블정렬 - O(N^2)
  - 삽입정렬 - O(N^2)
  - 퀵정렬 – O(N^2)
  - etc

대중적인 알고리즘의 시간복잡도

- 스택,큐
  - 삽입 – O(1)
  - 삭제 – O(1)
- 이분 탐색
  - 탐색 – O(logN)
- 힙
  - 삽입 – O(logN)
  - 삭제 – O(logN)

구현

- 시간복잡도를 생각하며 적절한 방안을 찾아 문제를 풀어봅시다.
- STL을 적절히 활용해서 시간단축