4. 알고리즘 성능 분석

알고리즘 성능을 보다 객관적으로 평가하기 위해서 다음 두 가지 측면을 고려한다.

시간복잡도 (time complexity)	알고리즘 수행이 완료되는데 필요한 컴퓨터 **시간의 양**을 의미한다.
공간복잡도 (space complexity)	알고리즘 수행이 완료되는데 필요한 컴퓨터 **기억공간의 양**을 의미한다.

[예] 다음 코드를 이용하여 a++;의 실행 횟수와 빅오 함수 표현의 관계를 살펴본다.

C 코드	수행 횟수(a++;)	빅오 표현
① a++;	1번	$O(1)$
② for(i=1; i<=n; i++) a++;	n번	$O(n)$

③ for(i=1; i<=n; i++)
for(j=1; j<=n; j++)
a++;
	n^2번	$O(n^2)$

④ for(i=1; i<=n; i++)
for(j=1; j<=i; j++)
a++;
	$1 + 2 + 3 + ... + n$
	$= \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$
		$O(n^2)$

- 위에서, 3번째와 4번째는 a++;의 실행 횟수가 각각 다르다.
- 하지만, 빅오 함수로는 똑같이 표현된다.
- 빅오 함수 표현은 **최대 차수**를 중시하기 때문이다. 상수는 필요 없다.

◈ **복합 알고리즘의 연산 차수**

① $5n^3 + 4n^2 + 3n + 2 = O(n^3)$

② $n + \log_2 n = O(n)$, $n \times \log_2 n = O(n \log_2 n)$

③ $\log(n^2 + 1) = O(\log n)$

④ $(n^2 + 3)\log n = O(n^2 \log n)$

⑤ $(nlogn + n^2)(n^3 + 1) = O(n^5)$

⑥ $365 = O(1)$ → 상수는 아무리 큰 값이라도 무조건 $O(1)$이다.

기출문제 분석

1. 다음 알고리즘의 시간복잡도를 빅오(O) 표기법으로 옳은 것은? [2015년 국가 7급]

```
----------------------------------------
 Procedure calculate(int n)
 {
    int i, k, m, x = 0;

    for (i = 0; i < (n - 5); i++)
      for (k = 0; k < 100; k++)
        for (m = 0; m < 1000; m++)
          x++;
 }
----------------------------------------
```

① O(log n)
② O(n)
③ $O(n^2)$
④ $O(n^3)$

♤ **시간복잡도 – 빅오(O)**

```
Procedure calculate(int n)
{
    int i, k, m, x = 0;

    for (i = 0; i < (n - 5); i++)        → O(n)
      for (k = 0; k < 100; k++)          → O(1)
        for (m = 0; m < 1000; m++)       → O(1)
          x++;
}
```

• 시간복잡도 = O(n) × O(1) × O(1) = **O(n) → 상수시간은 제거**된다.

정답 : ②

2. C언어로 작성된 함수의 시간복잡도를 빅오표기법으로 표현할 때, 가장 낮은 시간복잡도를 가지는 것은? [2021년 서울 7급]

(가)
```
void func1(int n)
{
    int i, j, k=0;
    for (i=0; i<n; i++)
      for (j=i; j<n; j++)
        k = i * j;
}
```

(나)
```
void func2(int n)
{
  int i, j, a=0;
  for(i=1; i<=n; i=i*2)
    for (j=1; j<=n; j++)
      a = i + j;
}
```

(다)
```
void func3(int n)
{
  int i, j, k=0;
  for (i=0; i<n; ++i)
    for (j=1; j<1000; ++j)
      ++k;
}
```

(라)
```
int func4(int n) {
  int i, k=0;
  while (n>2){
    n--;  k++;
    for (i=0; i<n; i++) k++;
  }
  return k;
}
```

① (가)　② (나)
③ (다)　④ (라)

♤ 시간복잡도

(가)
```
void func1(int n)
{
    int i, j, k=0;
    for (i=0; i<n; i++)
      for (j=i; j<n; j++)
        k = i * j;
}
```
$O(n^2)$

n + (n−1) + (n−2) + ··· 2 + 1 = $O(n^2)$

(나)
```
void func2(int n)
{
  int i, j, a=0;
  for(i=1; i<=n; i=i*2)
    for (j=1; j<=n; j++)
      a = i + j;
}
```
$O(nlog_2 n)$

for(i=1; i<=n; i=i*2) : 2배씩 증가하므로 $\log_2 n$

(다)
```
void func3(int n)
{
  int i, j, k=0;
  for (i=0; i<n; ++i)
    for (j=1; j<1000; ++j)
      ++k;
}
```
O(n)

// 상수 1000은 O(1)이다.

(라)
```
int func4(int n) {
  int i, k=0;
  while (n>2){
    n--;  k++;
    for (i=0; i<n; i++) k++;
  }
  return k;
}
```
$O(n^2)$

정답 : ③

3. 아래의 알고리즘 mystery1과 mystery2의 시간복잡도를 빅오 표기법으로 나타낸 것으로 가장 옳은 것은? [2022년 군무원 7급]

```
--------------------------------
void mystery1(int n)
{
  int a[n][n], b[n][n], i, j;
  b[0][0]=a[n-1][n-1];
}
void mystery2(int n)
{
  int a[n][n], b[n][n], i, j;
  for(i = 1; i <= n; i++)
  {
    for(j = 1; j <= 1000; j++)
      b[i][j]=a[i][j];
  }
}
--------------------------------
```

① O(1), O(n)　② O(1), $O(n^2)$　③ O(n), $O(n^2)$　④ O(1), O(1)

♤ 시간복잡도

```
void mystery1(int n)
{
  int a[n][n], b[n][n], i, j;
  b[0][0]=a[n-1][n-1];              //O(1) - 상수시간
}
void mystery2(int n)
{
  int a[n][n], b[n][n], i, j;
  for(i = 1; i <= n; i++)           //O(n)
  {
    for(j = 1; j <= 1000; j++)      //O(1) - 숫자는 상수시간
      b[i][j]=a[i][j];
  }
}
```

정답 : ①

4. 시간복잡도 함수에 대한 점근표기법으로 옳은 것만을 모두 고르면? [2019년 국가 7급]

ㄱ. $n^{2n} + 6 \cdot 2^n = O(n^{2n})$

ㄴ. $n^2/\log n = \Theta(n^2)$

ㄷ. $n^3 2^n + 6n^2 3^n = O(n^2 2^n)$

ㄹ. $6n^3 + \log n = O(n^3)$

---

① ㄱ, ㄷ　　② ㄱ, ㄹ

③ ㄴ, ㄷ　　④ ㄴ, ㄹ

♤ **시간복잡도**

ㄴ. $n^2/\log n = \Theta(n^2/\log n)$

ㄷ. $n^3 2^n + 6n^2 3^n = O(n^2 3^n)$

정답 : ②