

시간복잡도 연습문제

made by gardenjun

모든 문제는 big-O 표기법을 사용합니다.

예시

```
a = 0
for i in range(N):
    a += 1
```

정답: $O(N)$

문제 #1

```
print("Hello World1")
print("Hello World2")
print("Hello World3")
```

정답: $O(1)$

문제 #2

```
a = 0
b = 0
for i in range(N):
    a += 1
for j in range(N):
    b += 1
```

정답: $O(N)$

문제 #3

```
a = 0
b = 0
for i in range(N):
    a += 1
for j in range(M):
    b += 1
```

정답: $O(N+M)$

문제 #4

```
a = 0
b = 0
for i in range(N):
    for j in range(N):
        a = a + i + j
for j in range(M):
    b += 1
```

정답: $O(N^2)$

문제 #5

우리는 두가지 알고리즘을 가지고 있습니다, 'Algo A'와 'Algo B' 이지요. 여기서, 'Algo A'가 'Algo B' 보다 점근적으로 더 효과적이라는 말은 무슨 뜻인지 아래 선택 답지에서 골라 보세요.

1. Algo A가 어떤 크기의 입력이 들어와도 항상 빠르다.
2. Algo A가 입력 크기가 클 경우 빠르다.
3. Algo A가 입력 크기가 작을 경우 빠르다.
4. Algo B가 항상 더 빠르다.
5. Algo A가 입력 크기에 따라 메모리 관리능력이 효과적이다.

정답: ②

문제 #6

```
a = 0, i = N
while i > 0:
    a += i
    i = i/2
```

정답: $O(\log N)$

문제 #7

```
def gcd(N, M):
    if N % M == 0:
        return M
    if N < M:
        N, M = M, N
    while M > 0:
        N = N % M
        N, M = M, N
    return N
```

정답: $O(\log N)$

문제 #8

다음 보기중 $O(n^2)$ 가 아닌 것은 무엇인가요?

- 1) $(15^{10}) * n + 12099$
- 2) $n^{1.98}$
- 3) $n^3 / (\text{sqrt}(n))$
- 4) $(2^{20}) * n$

정답: ③

문제 #9

아래 보기 중 가장 빠른 반복문을 고르세요.(n은 양의 정수)

1. for(i = 0; i < n; i++)
2. for(i = 0; i < n; i += 2)
3. for(i = 1; i < n; i *= 2)
4. for(i = n; i > -1; i /= 2)

이 문제는 안배웠지만 알아두셨으면 해서 넣어놨습니다!
답은 3번으로 나누기는 매우매우매우매우 느린 연산입니다!
만약 시간초과가 났을 때 4번처럼 반복해서 나누는 연산이 있다면
3번처럼 곱하기로 바꾸어 해결해보세요!

정답: 3번

문제 #10

다음 함수들을 가장 빠른 순서대로 나열한 것은?

$f_1(n) = 2^n$ ⁴
 $f_2(n) = n^{(3/2)}$ ²
 $f_3(n) = n \log n$ ¹
 $f_4(n) = n^{(\log n)}$ ³
 $f_5(n) = n!$ ⁵

1. f3(), f2(), f4(), f1(), f5()
2. f3(), f2(), f1(), f4(), f5()
3. f2(), f3(), f1(), f4(), f5()
4. f2(), f3(), f4(), f1(), f5()

정답: ①

문제 #11

```
def func(N):  
    m = 0  
    for i in range(N):  
        for j in range(i):  
            m += 1  
    return m
```

정답: $O(N \log N)$

문제 #12

```
import math

def func(N):
    m = 0
    for i in range(N):
        for j in range(math.sqrt(N)):
            m += 1
    return m
```

정답: $O(N\log N)$

문제 #13

```
def func(N):
    m = 0
    i = N
    while i > 0:
        for j in range(i):
            m += 1
        i = i / 2
    return m
```

정답: $O(N\log N)$

문제 #14

```
def binarySearch(array, value, low, high):
    if low > high:
        return False
    mid = (low+high) / 2
    if array[mid] > value:
        return binarySearch(array, value, low, mid-1)
    elif array[mid] < value:
        return binarySearch(array, value, mid+1, high)
    else:
        return mid
```

정답: $O(\log N)$

문제 #15

```
def binarySearch(array, value, low, high):
    if low > high:
        return False
    mid = (low+high) / 2
    if array[mid] > value:
        return binarySearch(array, value, low, mid-1)
    elif array[mid] < value:
        return binarySearch(array, value, mid+1, high)
    else:
        return mid
```

정답: $\log(N)$

문제 #16 (서술형 풀이 필요!)

다음 재귀식을 $O(n)$ 표현 수준으로 푸시오:

$$T(n) = 2T(n/4) + n$$

정답:

16번은 적당히 고민하다가 밑에 풀이를 넣어 봤으니 보셔도 됩니다!

시간 복잡도가 어떻게 계산되는지 어렵קות하게 말하는 것보단 증명할 수 있다는 것도 알아두었으면 해서 넣어봤습니다!

16번 풀이

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{4}\right) + n \\
&= 2 \cdot \left\{ 2T\left(\frac{n}{4^2}\right) + \frac{n}{4} \right\} + n \\
&= 2^2 T\left(\frac{n}{4^3}\right) + \frac{n}{2} + n \\
&= 2^2 \left\{ 2T\left(\frac{n}{4^3}\right) + \frac{n}{4^3} \right\} + \frac{n}{2} + n \\
&= 2^3 T\left(\frac{n}{4^3}\right) + \frac{n}{4} + \frac{n}{2} + n \\
&\vdots \\
&= 2^k T(a) + \sum_{i=0}^k \frac{n}{2^i} \\
&= T(a) \cdot 2^k + \frac{n \cdot (1 - \frac{1}{2^{k+1}})}{1 - \frac{1}{2}} \quad \begin{array}{l} k = \lfloor \log_4 n \rfloor \\ 0 < a < 4 \end{array} \\
&= \underbrace{T(a) \cdot \sqrt{n}}_{\substack{0 < a < 4 \text{ 이므로} \\ \text{상속시간}}} + \underbrace{2n - \sqrt{n}}_{\text{가장 높은 차수}} \quad \begin{array}{l} 2^k \approx \sqrt{n} \end{array} \\
\therefore O(n)
\end{aligned}$$