3장

알고리즘의 성능

Eunji Lee

연습문제 01

- 입력의 크기가 n일 때 다음 알고리즘의 수행 시간을 Θ- 표기법으로 나타내시오.

```
matrixMult(A[][], B[][], M[][], n):
     for i ← 1 to n
        for j ← 1 to n
           M[i, j] ← 0
           for k ← 1 to n
              M[i, j] ← M[I, j] + A[i, k] * B[k, j]
```

연습문제 02

- 입력의 크기가 n일 때 다음 알고리즘의 점근적 수행 시간을 O-, Θ- 표기법으로 각각 나타내시오. 단, random(1, 100)은 1부터 100까지의 정수 중 하나를 임의로 리턴한다.

```
sample(A[], n):
     for i ← 1 to n
       if (random(1, 100) <= 50)
           sum ← 0
           for i ← 1 to n
              sum ← sum + A[i]
```

연습문제 03

- 입력의 크기가 n일 때 다음 알고리즘의 점근적 수행 시간을 O-, Θ- 표기법으로 각각 나타내시오. 단, random(1, 100)은 1부터 100까지의 정수 중 하나를 임의로 리턴한다.

```
sample(A[], n):
     if (n = 1) return 1
     else if (random(1, 100) <= 50)
        sum ← 0
        for i ← 1 to n
           sum ← sum + A[i]
     sample(A, n-1)
```

연습문제 04

- 입력 크기가 n일 때 다음 알고리즘의 점근적 수행 시간은 얼마인가?

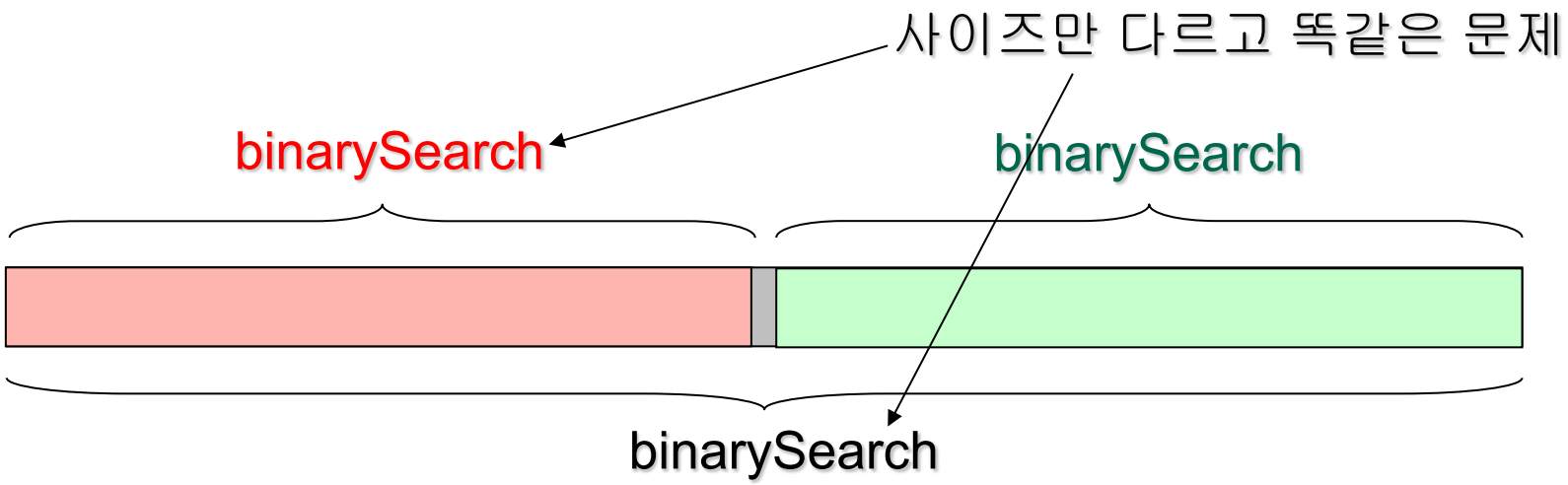
```
sample(A[], n):
     if (n = 1) return 1
     sum ← 0
     for i ← 1 to n
        sum ← sum + A[i]
     tmp ← sum + sample(A, n-1)
     return tmp
```

연습문제 05

```
binarySearch (A[], x, low, high) :
        // A: array,  x: search key,  low, high: array bounds
        if (low > high) return "Not found"
        mid ← (low + high)/2
        if (A[mid] < x) return binarySearch(A, x, mid+1, high)
        else if (A[mid] > x) return binarySearch(A, x, low, mid-1)
        else return mid
```

배열의 중앙에 있는 원소와 비교하고 나면
자신과 똑같지만 크기가 반이 되는 문제를
만난다. 즉, T(n) = c + T(n/2)
이런 식으로 크기를 반씩 줄여나가면
~$\log_2 n$번 만에 크기 1인 문제를 만나게 된다.
문제의 크기를 반으로 줄이는데 필요한 작업은
상수 시간이므로 최대 $\log_2 n$에 비례하는 시간에 끝난다.

수행 시간:
최악의 경우 Θ(log n),
최선의 경우 Θ(1),
앞에 수식어 없이 그냥 말하면 O(logn)



연습문제 06

- 하노이 타워 시간복잡도는?

```
[4]  def move(n, src, tmp, dest):
       if n == 0:
         return
       move(n-1, src, dest, tmp)
       print("move %d from %c to %c" % (n, src, dest))
       move(n-1, tmp, src, dest)
```

연습문제 06

- 하노이 타워 시간복잡도는?

$T(n) = 2T(n-1) + 1 \quad \dots (1)$
$T(n-1) = 2T(n-2) + 1 \quad \dots (2)$

(2)를 (1)에 대입하면
$T(n) = 2^2T(n-2)+(2+1)$

위와 같은 방식으로 계속해서 정리하면,
$T(n) = 2^{n-1}T(1) + (2^{n-2} + 2^{n-3} + \dots 2 + 1)$

$T(1) = 1$ 이므로
$T(n) = 2^{n-1} + 2^{n-2} + \dots 2 + 1 \quad \dots (3)$

(3)은 공비가 2인 등비수열이 됨.
$T(n) = 2^n - 1 = O(2^n)$

연습문제 07

- 피보나치 수열의 시간복잡도는?

알고리즘 2-2 피보나치 수열(재귀 버전)

```
fib(n):
    if (n = 1 or n = 2)
        return 1
    else
        return fib(n-1) + fib(n-2)
```

연습문제 07

- 피보나치 수열의 시간복잡도는?