2. Divide-and-conquer algorithms

: Breaking into subproblems and recursively solving these subproblems

2.1 Multiplication

Figure 2.1 A divide-and-conquer algorithm for integer multiplication.

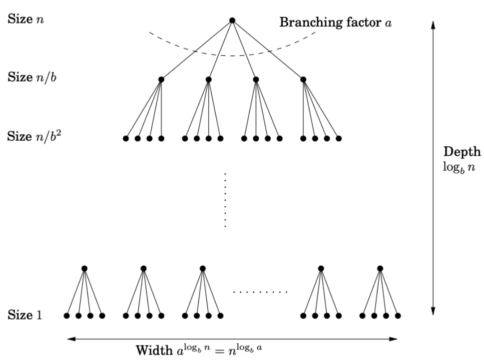
```
function multiply(x, y)
Input:  Positive integers x and y, in binary
Output:  Their product

n = max(size of x, size of y)
if n = 1:  return xy

x_L, x_R = leftmost ⌈n/2⌉, rightmost ⌊n/2⌋ bits of x
y_L, y_R = leftmost ⌈n/2⌉, rightmost ⌊n/2⌋ bits of y

P_1 = multiply(x_L, y_L)
P_2 = multiply(x_R, y_R)
P_3 = multiply(x_L + x_R, y_L + y_R)
return P_1 × 2^n + (P_3 − P_1 − P_2) × 2^(n/2) + P_2
```

2.2 Recurrence Relations

$$T(n) = aT(\lceil n/b \rceil) + O(n^d)$$

분할되어 적용되어 위의 공식을 따르게 되고,
recursive한 형식을 가진다.

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a . \end{cases}$$



적용예시

2.3 Merge sort

```
function iterative-mergesort(a[1...n])
Input:  elements a_1, a_2, ..., a_n to be sorted

Q = [ ] (empty queue)
for i = 1 to n:
   inject(Q, [a_i])
while |Q| > 1:
   inject(Q, merge(eject(Q), eject(Q)))
return eject(Q)
```

2.4 Medians

$$\text{selection}(S,k) = \begin{cases} \text{selection}(S_L, k) & \text{if } k \le |S_L| \\ v & \text{if } |S_L| < k \le |S_L| + |S_v| \\ \text{selection}(S_R, k - |S_L| - |S_v|) & \text{if } k > |S_L| + |S_v|. \end{cases}$$

2.5 Matrix Multiplication / 2.6 The fast Fourier transform

Multiplication 원리가

> 2.1 Multiplication인 subproblem으로 적용되어