3.1 Recurrence relation

- Recurrence relation
  - An equation in which each term of the sequence is defined as a function of the preceding terms
  - Examples
    - $T(n) = T(n/2) + 1$
    - $T(n) = 2T(n/2) + n$
    - $T(n) = T(n-1) + T(n-2)$
    - $T(n) = T(n/2) + n$
  - Solutions
    - Characteristic equation
    - Repeated substitution or telescoping
    - Master theorem

- Performance comparison

Function	n = 10	100	1,000	10,000	100,000	1,000,000
1	1	1	1	1	1	1
$\log_2 n$	3	6	9	13	16	19
n	10	10^2	10^3	10^4	10^5	10^6
$n * \log_2 n$	30	664	9,965	10^5	10^6	10^7
n^2	10^2	10^4	10^6	10^8	10^{10}	10^{12}
n^3	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
2^n	10^3	10^{30}	10^{301}	$10^{3,010}$	$10^{30,103}$	$10^{301,030}$

3.1 Recurrence relation

- Exercise 2.5 (P72)

(j) $T(n) = 2T(n-1) + 1$

(k) $T(n) = T(\sqrt{n}) + 1$

- Exercise 2.13 (P73)

2.13. A binary tree is *full* if all of its vertices have either zero or two children. Let B_n denote the number of full binary trees with n vertices.

(a) By drawing out all full binary trees with 3, 5, or 7 vertices, determine the exact values of B_3, B_5, and B_7. Why have we left out even numbers of vertices, like B_4?

(b) For general n, derive a recurrence relation for B_n.

(c) Show by induction that B_n is $2^{\Omega(n)}$.

3.2 Multiplication

- Multiplying two integers of n-digit.
  - u & v: n-digit integers
  - Time for adding u & v: O(n)

Carry:	1			1	1	1		
		1	1	0	1	0	1	(53)
		1	0	0	0	1	1	(35)
	1	0	1	1	0	0	0	(88)

3.2 Multiplication

- Multiplying two integers of n-digit.
  - Time for multiplying u & v: $O(n^2)$

$$
\begin{array}{ccccccccl}
 & & & & 1 & 1 & 0 & 1 & \\
 & & & \times & 1 & 0 & 1 & 1 & \\
\hline
 & & & & 1 & 1 & 0 & 1 & \text{(1101 times 1)} \\
 & & & 1 & 1 & 0 & 1 & & \text{(1101 times 1, shifted once)} \\
 & & 0 & 0 & 0 & 0 & & & \text{(1101 times 0, shifted twice)} \\
+ & 1 & 1 & 0 & 1 & & & & \text{(1101 times 1, shifted thrice)} \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \text{(binary 143)}
\end{array}
$$

  - Can we improve it by divide & conquer?

3.2 Multiplication

- Gauss's original suggestion
  - $(a + b\,i)\,(c + d\,i) = ac - bd + (ad + bc)\,i$
  - How many multiplications?
  - Actually, 3 instead of 4
    - $ad + bc = (a + b)(c + d) - ac - bd.$

3.2 Multiplication

- Two binary numbers x & y with length n

$$x = [x_L][x_R] = 2^s x_L + x_R$$

$$y = [y_L][y_R] = 2^s y_L + y_R, \text{ where } s = \left\lceil n/2 \right\rceil$$

$$xy = (2^s x_L + x_R)(2^s y_L + y_R)$$

$$= 2^n x_L y_L + 2^s(x_L y_R + x_R y_L) + x_R y_R$$

– T(n) ➔ 4 T(n/2) + O(n)

$$x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$$

– T(n) ➔ 3 T(n/2) + O(n)

3.2 Multiplication

- Algorithm

```
void multiply ( int u, int v )
{
    Degenerate case

    Divide

    Conquer

    Combine
}
```

- Algorithm

```
void multiply ( int u, int v )
{
    n ← min ( digit of u, digit of v );
    if ( n is small enough )
        return u * v;
    s ← n div 2;
    w ← u div 2^s;
    x ← u mod 2^s;
    y ← v div 2^s;
    z ← v mod 2^s;

    r ← multiply ( w + x, y + z );
    p ← multiply ( w, y );
    q ← multiply ( x, z );

    return p * 2^2s + (r – p – q) * 2^s + q;
}
```

3.2 Multiplication

- Algorithm → Check three points

```
void multiply ( int u, int v )
{
    n ← min ( digit of u, digit of v );
    if ( n is small enough )
        return u * v;
    s ← n div 2;
    w ← u div 2^s;
    x ← u mod 2^s;
    y ← v div 2^s;
    z ← v mod 2^s;

    r ← multiply ( w + x, y + z );
    p ← multiply ( w, y );
    q ← multiply ( x, z );

    return p * 2^2s + (r – p – q) * 2^s + q;
}
```

3.2 Multiplication

- Performance analysis
  - Recurrence relation

$$T(n) = 3T(n/2) + O(n)$$

    - a = 3, b = 2, k = 1.
    - a = 3 > $b^k = 2^1$,

$$T(n) = O(n^{\log_2 3})$$

3.2 Multiplication

- Comparison

	degenerate case	divide	conquer	combine	performance
tournament	n = 1 (s = e)	m = (s+e)/2	champ (s,m); champ (m+1,e);	win (LW, RW);	2T(n/2) + O(1) = O(n)
binary search	n = 1 (s = e)	m = (s+e)/2	bs (s, m-1); or bs (m+1, e);	-	T(n/2) + O(1) = O(log n)
integer multiplication	n = 1	s = n/2; w = u div 2^s;	mult (w+x, y+z); mult (w, y); mult (x, z);	p 2^n + (r – p – q) 2^s + q;	**$3T(n/2) + O(n) = O(n^{\log_2 3})$**
merge sort					
quick sort					
median					
matrix multiplication					

퀴즈 2

- n 자리의 두 수를 곱하는 연산에 대한 설명이다 올바른 것을 모두 고르시오.

  (a) n 자리의 두 수의 곱을 n/2 자리의 두 수의 곱으로 분할해서 문제를 해결한다.

  (b) n/2 자리 수의 곱을 2번 수행한다.

  (c) divide 과정에서는 O(n)번 연산이 수행된다.

  (d) combine 과정에서는 O(1)번 연산이 수행된다.