알고리즘

- 어떤 값이나 값의 집합을 입력으로 받아 또 다른 값이나 값의 집합을 출력하는, 잘 정의된 계산 절차
- 잘 정의된 계산 문제를 풀기 위한 **도구**로 사용

Correct Algorithm(타당한 알고리즘)

- 문제를 풀 수 있는 알고리즘
  - 타당한 알고리즘인지 확인하기 위해, 루프 불변성을 보여야 함(Loop invariants)
    - Initialization : 루프가 첫번째 반복을 시작하기 전에 루프 불변성이 참이어야 한다
    - Maintenance : 루프의 반복이 시작되기 전에 루프 불변성이 참이었다면 다음 반복이 시작되기 전까지도 계속 참이어야 한다
    - Termination : 루프가 종료될 때 그 루프 불변식이 알고리즘의 타당성을 보이는데 도움이 될 유용한 특성을 가져야 한다

알고리즘의 효율성

- 알고리즘의 효율성 : input size n에 대하여 알고리즘의 실행 시간을 분석
  - RAM(Random Access Machine) Model
    - 기본적인 산술연산, 데이터 이동 연산, 제어 연산을 수행하는 명령어들을 가지고 있으며 이 명령어들을 **한 개씩 상수 시간만큼 걸려서 수행**한다고 가정한 모델
    - 캐쉬나 가상 메모리와 같은 메모리 계층 구조가 없다고 가정

Order of growth(증가 차수)

- 수행시간 분석을 할 때 함수를 단순화 한다
  - 최고차항만 고려하고, 상수 계수는 무시한다

- $T(n) = an^2 + bn + c$
- → $T(n)$ equals to n^2
- → $T(n)$ grows like n^2
- → order of growth is n^2
- → $T(n)$ is $\Theta(n^2)$

Sorting Problem

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$.

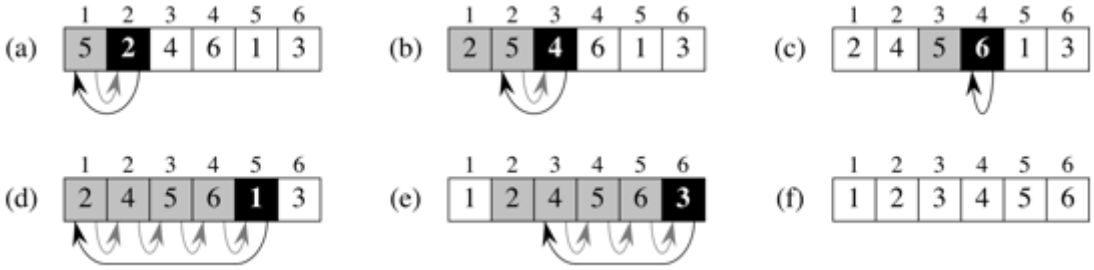
Output: A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

- Sequence는 주로 배열에 저장, a_i는 key이고 실제로는 각 key로 대표되는 satelite data가 있다

Insertion Sort(삽입 정렬)

- **Incremental Approach**

- 단계



- 의사코드

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

- Loop Invariants

"1-8행의 부분 배열 *A[i.. j-1]* 은 원래 *A[i.. j-1]* 의 원소지만 for loop가 반복을 시작할 때마다 정렬된 순서로 구성된다."

1. Initialization : j=2 일 때, 부분 배열은 A[1]이고 정렬되어 있다.
2. Maintenance : 4-7행에서 *A[j-1], A[j-2], A[j-3]…* 을 오른쪽으로 한 자리씩 이동시키고 *A[j]*를 적절한 위치에 삽입한다. *A[i.. j-1]* 가 정렬되어 있었기 때문에 *A[i.. j]* 는 정렬된 상태가 된다.
3. Termination : j=A.length+1 일 때 끝나는데 이 때 유지조건에 의하여 *A[i.. A.length]* 는 정렬되어 있다.

- 분석
  - 알고리즘의 수행시간 = 수행된 기본연산의 갯수를 입력크기 n에 대한 함수로 나타냄

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

t_j : j값에 대한 while loop 의 반복 횟수
(input sequence 에 따라 다름)

- 수행시간

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

- Best Case
  - 배열이 이미 정렬되어 있는 경우

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

$= an + b$ ← n 에 대한 선형 함수

- Worst Case
  - 배열이 역순으로 정렬되어 있는 경우

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

배열이 역순으로 정렬되어 있는 경우 $t_j = j$

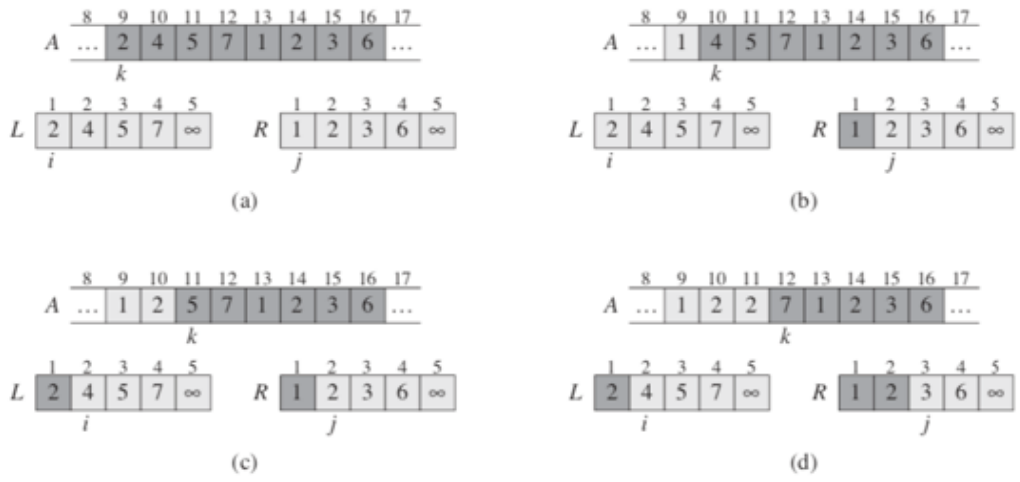
$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \left(\frac{n(n+1)}{2} - 1 \right) \\ &\quad + c_6 \left(\frac{n(n-1)}{2} \right) + c_7 \left(\frac{n(n-1)}{2} \right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

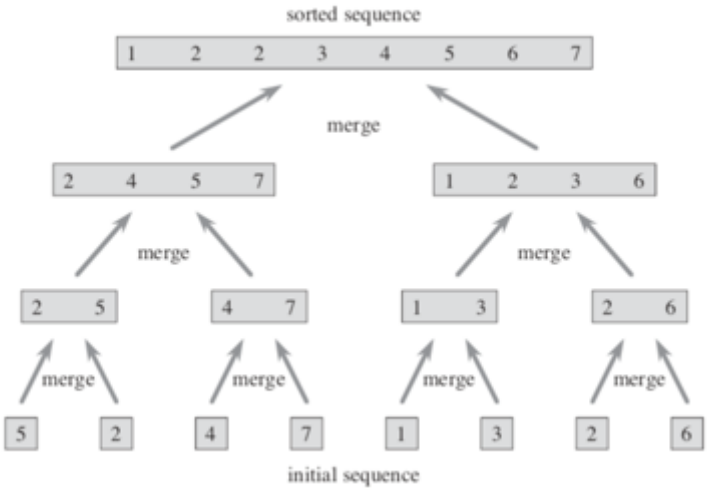
$= an^2 + bn + c$ ← n 에 대한 이차 함수

- **In-place Sorting** / Out-of-place Sorting
  - 다른 배열을 만들 필요가 없음

Merge Sort

- **Divide And Conquer Approach**
  - Divide : 정렬할 n개 원소의 배열을 n/2개씩 부분 수열 2개로 분할
  - Conquer : 병합 정렬을 이요해 두 부분 배열을 재귀적으로 정렬
  - Combine : 정렬된 두 개의 부분 배열을 병합해 정렬된 배열 하나로 만든다
- 단계





- 의사코드

```
MERGE(A, p, q, r)
 1  n1 = q − p + 1
 2  n2 = r − q
 3  let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
 4  for i = 1 to n1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n2
 7      R[j] = A[q + j]
 8  L[n1 + 1] = ∞
 9  R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

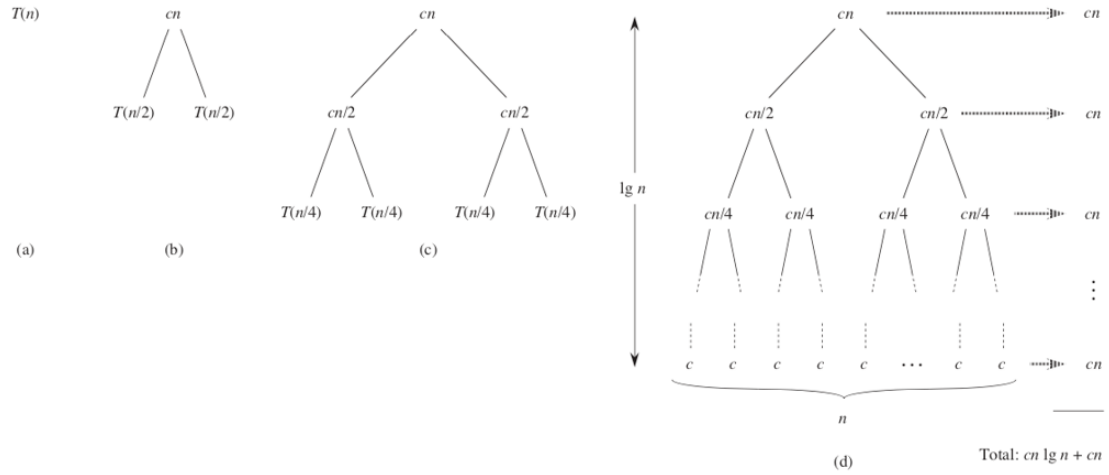
MERGE-SORT(A,1,A.length)

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

- In-place Sorting / **Out-of-place Sorting**
  - 다른 배열을 만들어서 값을 옮겨야 함
- 분석

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c\,, \\ aT(n/b) + D(n) + C(n) & \text{otherwise}\,. \end{cases}$$

$$T(n) = \begin{cases} c & \text{if } n = 1\,, \\ 2T(n/2) + cn & \text{if } n > 1\,, \end{cases}$$



- 시간복잡도

$$\Theta(n \lg n)$$