Ch 9. 정렬 (Sorting)

배열 A[1..N] N numbers to be sorted 아무리 좋아도 NlgN

✱ 비교기반 (comparison-based) sorting = Ω(NlgN)

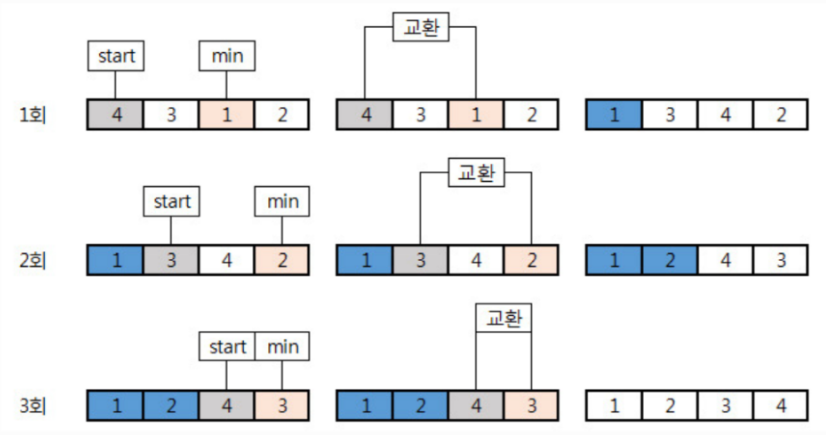
- 선택 (selection), 삽입 (Insertion), 버블 (bubble): O(N²)
- 쉘 (Shell): O(N√N), 최악 O(N²)

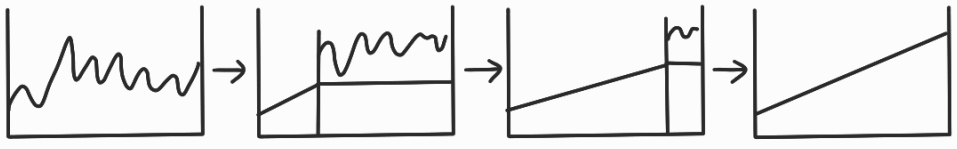
→ in-place (입력리스트 내부에서 정렬)

- 합병 (merge): O(NlgN) → in-place X (별도의 저장공간 필요)
- 퀵 (quick): O(NlgN) → in-place
- BST (이진 탐색 트리): O(NlgN)
- 기수 (radix): O(N) : 비교 기반 아님 → in-place X

1. Selection (선택) sort

```
for (i=1 to N-1)
{   m=i
    for (j=i+1 to N)
        if (A[j]<A[m]) m=j
    swap (A[i], A[m])
}
```



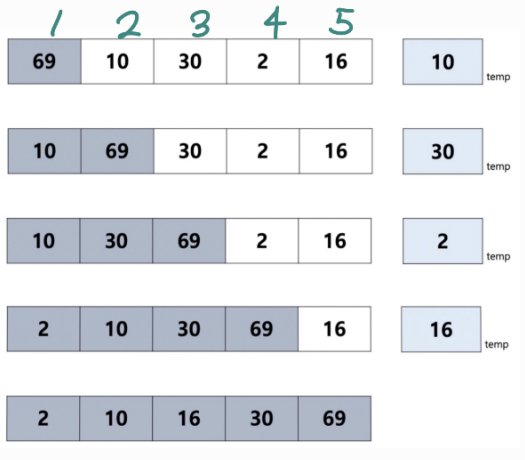


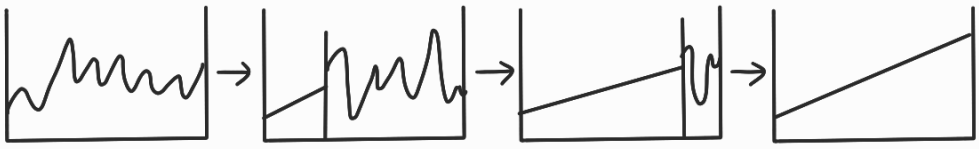
시간 = 비교횟수

= (N-1) + (N-2) + ⋯ + 2 + 1 = O(N²)

2. Insertion (삽입) sort ← sorted data에서는 빠르게 마무리

```
for (k=2 to N) // [1~k-1] sorted에 k를 삽입
{   m = A[k]
    for(i=k-1 down to 1)
        if(A[i]<m) break
        else  A[i+1]=A[i]  //shift right
    A[i+1]=m
}
```





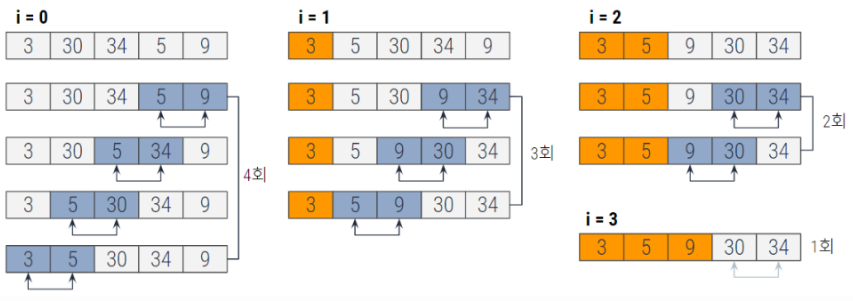
시간 = 비교횟수

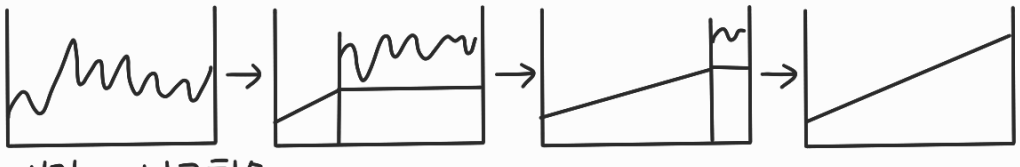
$= 1 + 2 + 3 + \cdots + (N-1) = O(N^2)$

3. Bubble sort : 인접한 두 원소를 비교하여 교환을 반복

```
for (k=1 to N-1)
{   count=0
    for (j=N downto k+1)
    {  if (A[j-1] > A[j])
       {  swap (A[j-1], A[j])
          count++
       }
       if (count==0) break;
    }
}
```

(Sorted data에는 빠르게 마무리
그러나 역순인 data에는 최악





시간 = 비교횟수

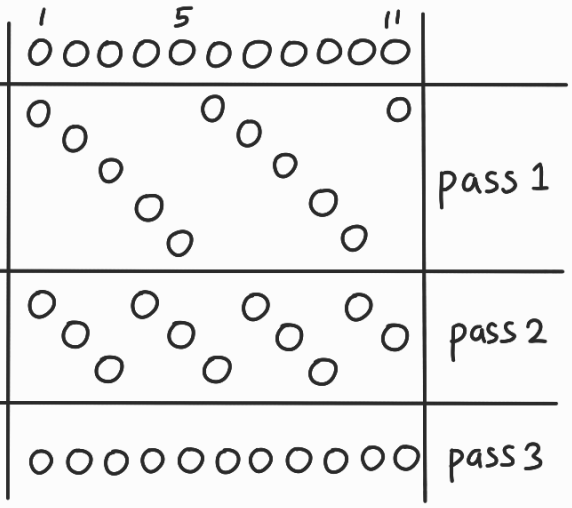
$= (N-1) + (N-2) + \cdots + 2 + 1 = O(N^2)$

4. Shell sort : 간격별 sub-list를 삽입정렬

약간 sorted data에는 빠르게 마무리

```
for (gap = N/2 ; gap>0; gap = gap/2)  // passes
{   if (gap = 짝수) gap++;
    for (i=1 to gap) insertion_sort()
     // gap 만큼의 sub-list 존재 → 삽입정렬
}
```

ex) N=11



	최선	평균	최악
선택	$O(N^2)$	$O(N^2)$	$O(N^2)$
삽입	$O(N)$	$O(N^2)$	$O(N^2)$
bubble	$O(N)$	$O(N^2)$	$O(N^2)$
shell	$O(N)$	$O(N\sqrt{N})$	$O(N^2)$

⊛ Devide & Conquer (분할정복법)

1. Devide (분할) - 동일한 유형의 작은 크기의 문제(들)로 분할
2. Conquer (정복) - 순환 (재귀) 호출로
3. Combine (병합)

5. Merge (합병) sort

MS(1, N)

MS(l, h)

{ if(l≥h) ret // size≤1

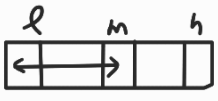
m = L(l+h)/2]

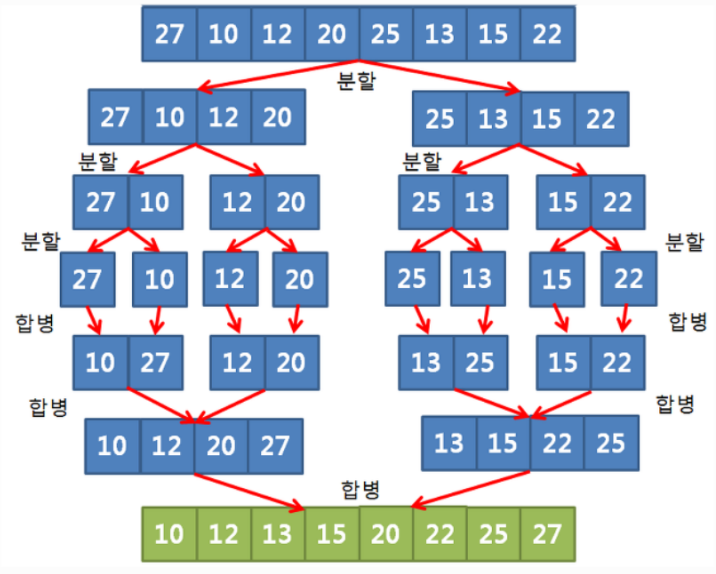
MS(l, m)

MS(m+1, h)

merge(l, m, h) → in-space O(N)

}





$$T(N)=\begin{cases}0 & \text{if } N\le 1\\ 2T(N/2)+N & \text{if } N\ge 2\end{cases}$$

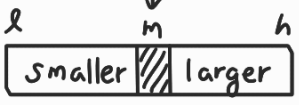
$= k \cdot N \quad (N = 2^k \text{ 가정})$

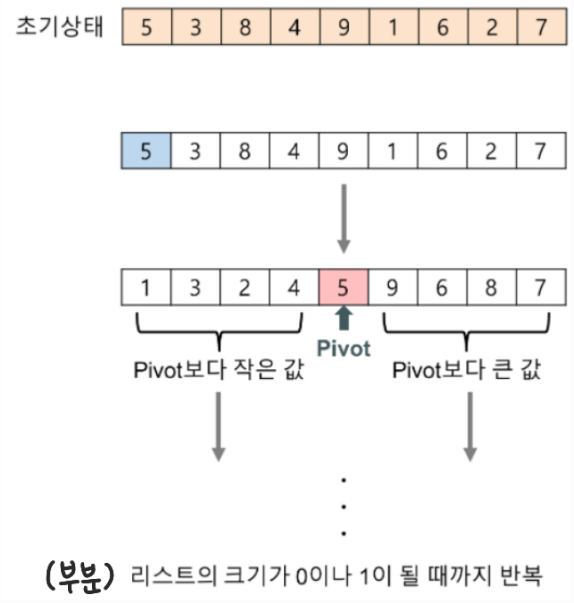
$= O(N \lg N)$

6. 퀵(Quick) 정렬 → 최악 $O(N^2)$

```
QS(l, N)
{ if(l≥h) ret    // ≤1
   m = partition(l,h)   → O(N)
   QS(l, m-1)
   QS(m+1, h)
}
```

임의의(보통 1st) pivot 원소





(부분) 리스트의 크기가 0이나 1이 될 때까지 반복

시간 분석

① pivot 원소의 위치에 따라 시간이 달라진다.

$T(N) = 2 \cdot T(\frac{N}{2}) + N$ → m이 절반일 때 최선

→ $O(N \lg N)$

② 실제 m의 위치 random이며 $m = 1 \sim N$ 이 동일한 확률 $(= \frac{1}{N})$

$$T(N) = \sum_{k=1}^{N} \frac{1}{N} \underbrace{[T(K) + T(N-K)]}_{\text{기댓값}} + N$$

$$= O(N \lg N) \leftarrow \text{average}$$

③ 이미 sorted data에는 최악 → $O(N^2)$

④ 동급 $O(N \lg N)$ 중에서 가장 빠르다!!

7. 힙(Heap) 정렬

: heap 구성 후 root를 N-1번 삭제하여 나열함

⇒ $O(N \lg N)$

8. BST sorting

① A[1..N]의 N개의 data를 이진탐색트리(BST)를 구성 : $N \lg N$

② 이 BST를 in-order로 방문하며 data를 A에 재수거 : $O(N)$

⇒ $O(N \lg N) + O(N) = O(N \lg N)$

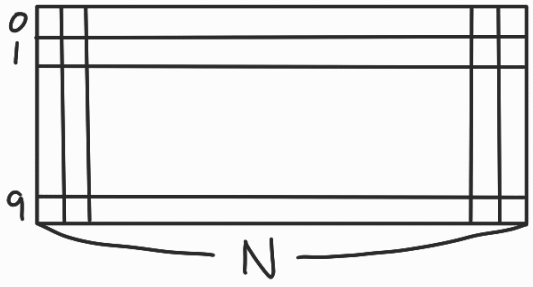
9. Linear sorting ← O(N)에 Sorting

① Radix(기수) sort (= Bucket sort)

A[1..N]에 N개의 K-digit 10진 정수 (중복 X → $K \le 10^K$)

(i.e $A[i] = d_{iK} \cdots d_{i2}\, d_{i1}$)

B[10][N] ← 10개의 size N인 bucket



pass 1 to pass k // 자릿수만큼

pass j = 1 to k (좌←우)

{ 1. clear all bucket B[][]

2. 모든 A[i], i = 1~N를 buck $B[d_{ij}][]$에 추가

3. B[0] ~ B[9]의 순서로 A[1..N]에 재수거

}

⇒ O(K·2N) = O(N)

ex)

2 5 1 **4**	3 | 0 0 **2** 3	1 | 2 **5** 1 4	0 | **0** 0 2 3
0 3 7 **5**	4 | 2 5 **1** 4	2 | 0 **0** 2 3	3 | **0** 3 7 5
4 4 3 **7**	5 | 0 3 **7** 5	3 | 4 **4** 3 7	4 | **4** 4 3 7
0 0 2 **3**	7 | 4 4 **3** 7	7 | 0 **3** 7 5	5 | **3** 5 1 4

⇒

0	0023, 0375
2	2514
4	4437

⇒

0 0 2 3
0 3 7 5
2 5 1 4
4 4 3 7

A

② Hashing에 의한 Sorting

: N개의 정수를 T[0..M-1]에 저장 후 재수거하는 방식

(M >>> N → M = k·N)

1) for all integer K, T[k] = k ⇒ O(N)

2) T[i], i = 0 ~ M-1 순으로 재수거 ⇒ O(M)

⇒ O(M) = O(K·N) = O(N)