Revisiting Sort Algorithm

충남대학교 컴퓨터공학과
이영석

문제1

- https://www.acmic

문제

알파벳 소문자로 이루어진 N개의 단어가 들어오면 아래와 같은 조건에 따라 정렬하는 프로그램을 작성하시오.

1. 길이가 짧은 것부터
2. 길이가 같으면 사전 순으로

입력

첫째 줄에 단어의 개수 N이 주어진다. (1≤N≤20,000) 둘째 줄부터 N개의 줄에 걸쳐 알파벳 소문자로 이루어진 단어가 한 줄에 하나씩 주어진다. 주어지는 문자열의 길이는 50을 넘지 않는다.

출력

조건에 따라 정렬하여 단어들을 출력한다. 단, 같은 단어가 여러 번 입력된 경우에는 한 번씩만 출력한다.

예제 입력 1 복사

```
13
but
i
wont
hesitate
no
more
no
more
it
cannot
wait
im
yours
```

예제 출력 1 복사

```
i
im
it
no
but
more
wait
wont
yours
cannot
hesitate
```

문제2

- https://tech.kakao.com/2017/11/14/kakao-blind-recruitment-round-3/

문제3. 파일명 정렬

세 차례의 코딩 테스트와 두 차례의 면접이라는 기나긴 블라인드 공채를 무사히 통과해 카카오에 입사한 무지는 파일 저장소 서버 관리를 맡게 되었다.

저장소 서버에는 프로그램의 과거 버전을 모두 담고 있어, 이름 순으로 정렬된 파일 목록은 보기가 불편했다. 파일을 이름 순으로 정렬하면 나중에 만들어진 ver-10.zip이 ver-9.zip보다 먼저 표시되기 때문이다.

버전 번호 외에도 숫자가 포함된 파일 목록은 여러 면에서 관리하기 불편했다. 예컨대 파일 목록이 ["img12.png", "img10.png", "img2.png", "img1.png"]일 경우, 일반적인 정렬은 ["img1.png", "img10.png", "img12.png", "img2.png"] 순이 되지만, 숫자 순으로 정렬된 ["img1.png", "img2.png", "img10.png", img12.png"] 순이 훨씬 자연스럽다.

Sort Libraries

- C++ STL
  - Introsort: quick + heap sort
- Java/Python
  - Timsort: mergesort + insertion sort

```
2  print(sorted([5, 2, 3, 1, 4]))
3  a = [5, 2, 3, 1, 4]
4  print(a)
5  a.sort()
6  print(a)
7
8  print(sorted("This is a test string from Andrew".split(), key=str.lower))
9
10 student_tuples = [
11       ('john', 'A', 15),
12       ('jane', 'B', 12),
13       ('dave', 'B', 10),
14   ]
15 print(sorted(student_tuples, key=lambda student: student[2]))   # sort by age
```

속도

- 10,000,000 개 정수

랜덤 데이터 (#12 ~ #15)

- #12 ~ #15: 1부터 N까지의 정수를 랜덤하게 섞어놓은 데이터

순위	언어	출력 방법	평균 (MS)
1	C++17	`std::sort (array)`	775.225
2	C++17	`std::sort (vector)`	804.8
3	C++17	`std::stable_sort (array)`	961.9
4	C++17	`std::stable_sort (vector)`	966.625
5	Java	`Arrays.sort`	983.55
6	C11	`qsort`	1529.025
7	C++17	`std::make_heap, std::sort_heap`	1695.075
8	PyPy3	`a.sort()`	1779.3
9	PyPy	`a.sort()`	1781.25
10	PyPy3	`a = sorted(a)`	1804.775
11	PyPy	`a = sorted(a)`	1810.45
12	C# 6.0	`Array.Sort`	2732.255675
13	Java	`Collections.sort`	3308
14	Python 2	`a.sort()`	5979.025
15	Python 2	`a = sorted(a)`	6036.675
16	Python 3	`a.sort()`	7174.65
17	Python 3	`a = sorted(a)`	7285.95
18	C# 6.0	`ArrayList.Sort`	9216.67275
19	C++17	`std::multiset`	11074.925
20	C++17	`std::map`	11470.475
21	Java	`TreeMap<Integer, Integer>`	18109.1

https://www.acmicpc.net/blog/view/58

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)

Merge Sort

- Divide and conquer strategy
  - N, M 개 정렬 리스트 합치기
  - 1. 주어진 리스트 2개로 나누기
  - 2. 나눈 리스트 2개 정렬
  - 3. 정렬 2개 리스트 합치기
- Complexity ?
  - O(n log n)
  - 대략 10,000 개 이상 정렬 문제
- Space ?
  - Extra space to hold the two halves
- Stable sorting

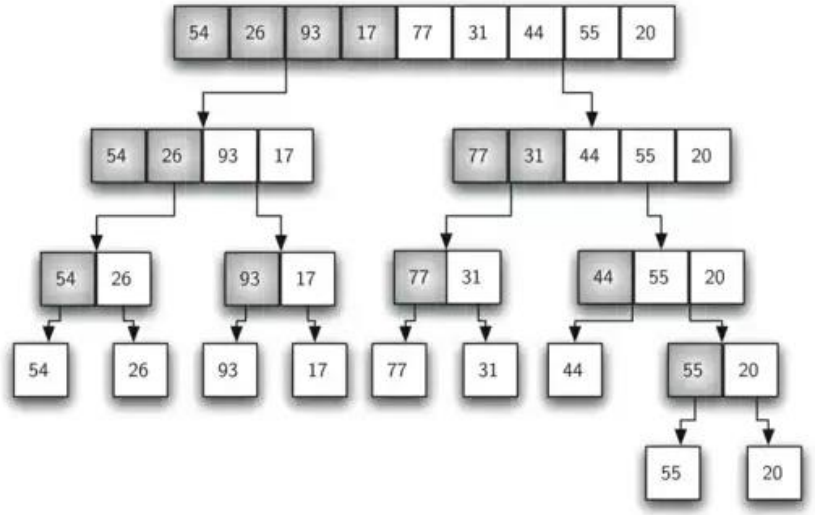

Figure 10: Splitting the List in a Merge Sort

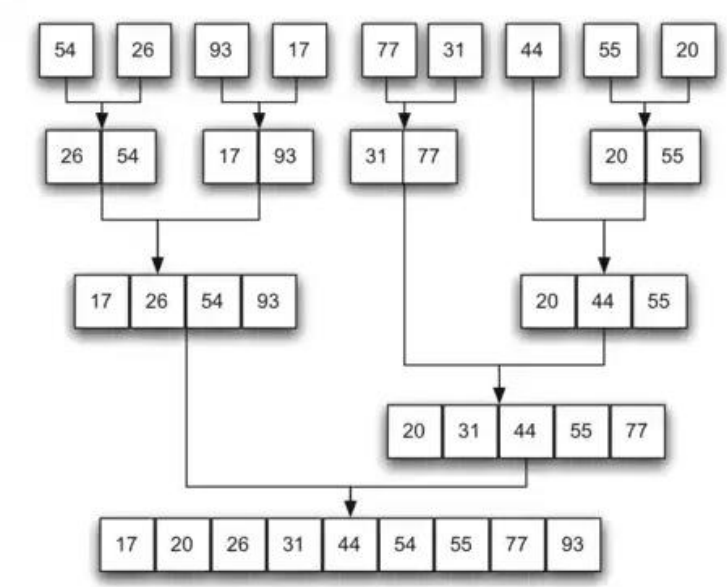

Figure 11: Lists as They Are Merged Together

Quick Sort

- Pivot기준 나눈 후 각각 정렬
- 복잡도
  - Average case: O(n log n)
  - Worst case: O(n^2)
    - Skewed to the left or right
- Unstable sorting

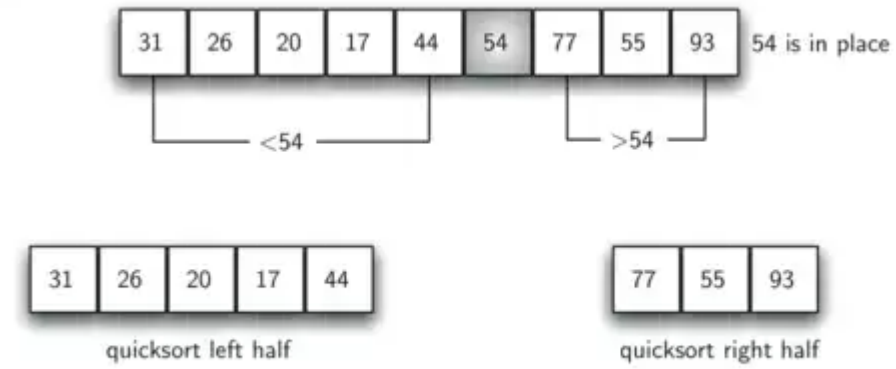

Figure 14: Completing the Partition Process to Find the Split Point for 54

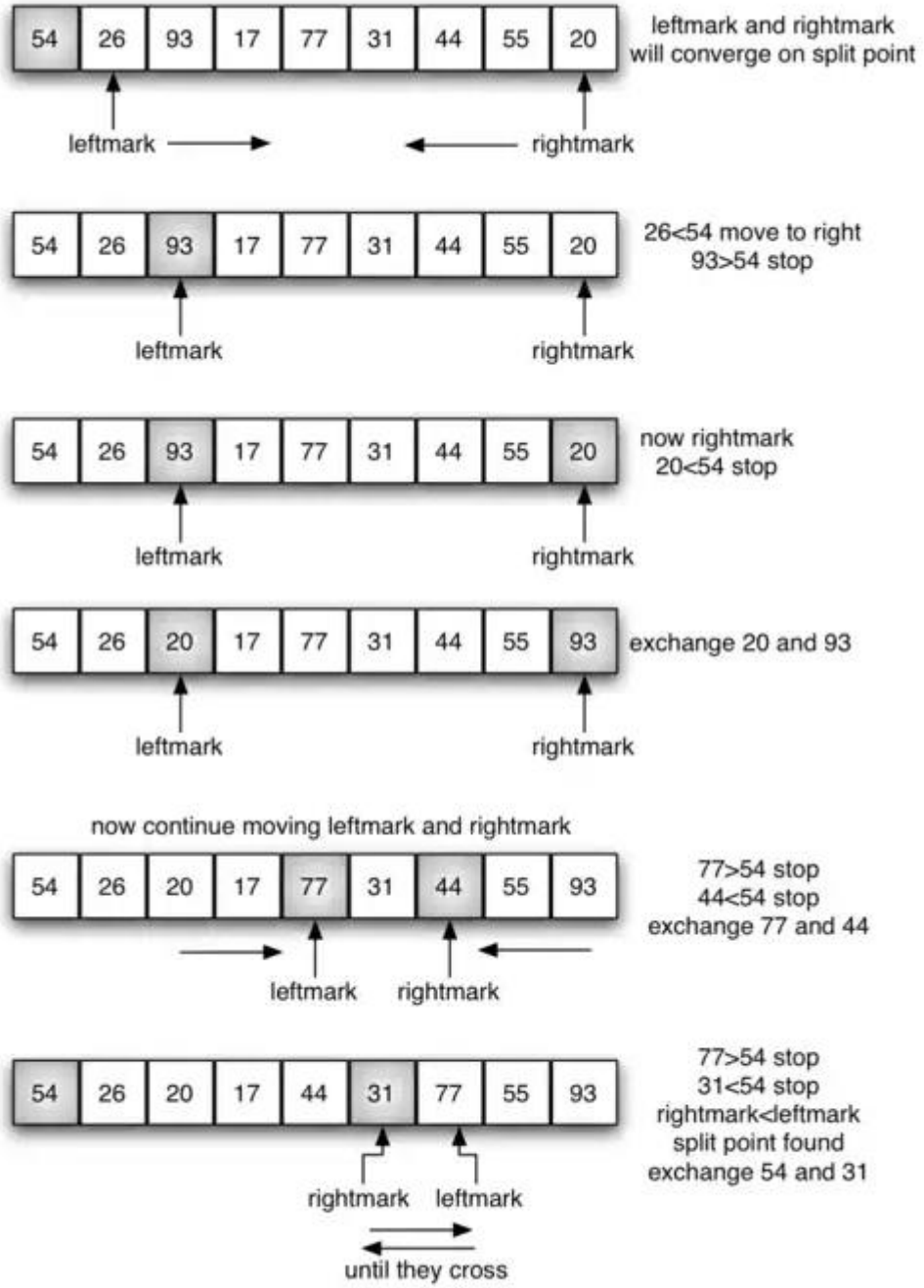

Figure 13: Finding the Split Point for 54

Counting Sort

- 원소의 종류가 k개
  - 숫자 1,2,3,4,5로 구성된 수열의 정렬문제
  - 1 5 4 2 3 1 4 3 정렬!
    - 1이 2번
    - 2가 1번
    - 3이 2번
    - 4가 2번
    - 5가 1번
- 복잡도
  - O(n+k)

- 512MB
  - int 변수 1.2억개

- http://www.cs.miami.edu/home/burt/learning/Csc517.091/workbook/countingsort.html

Radix Sorting

- Counting sort 응용 버전
- 자릿수가 있는 데이터 (정수, 문자열)

- Worst case: O(w*n)
  - w: word size (자릿수)
- Worst case space complexity O(w + n)

- 입력
  - 170, 45, 75, 90, 802, 2, 24, 66
- 1의 자리 정렬
  - 170, 90, 802, 2, 24, 45, 75, 66
- 10의 자리 정렬
  - 802, 2, 24, 45, 66, 170, 75, 90
- 100의 자리 정렬
  - 2, 24, 45, 66, 75, 90, 170, 802

Heap Sort

- Average/Worst case complexity
  - O(n log n)

1. n개의 노드에 대한 완전 이진 트리 구성
2. 최대 힙 구성
3. 루트를 배열의 마지막 노드와 교환
4. 2와 3을 반복

Bubble Sort

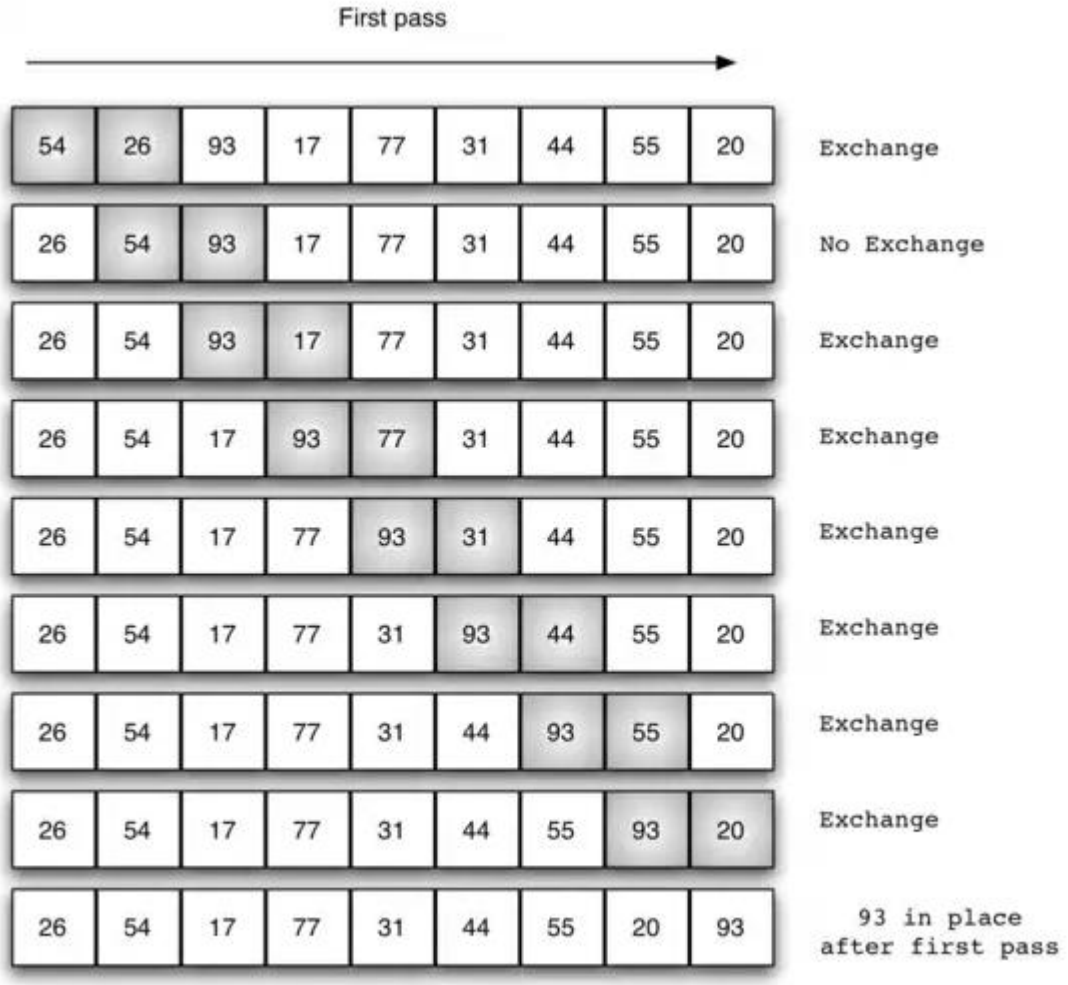

Figure 1: `bubbleSort`: The First Pass

Complexity ?

- O(n^2)
  - Comparison and swaps
- Unstable

Table 1: Comparisons for Each Pass of Bubble Sort

Pass	Comparisons
1	$n-1$
2	$n-2$
3	$n-3$
...	...
$n-1$	1

Selection Sort

- Bubble sorting 개선 ?
  - 간 단계(pass)별로 1회 교환만

- Complexity
  - O(n^2) comparisons
  - O(n) swaps

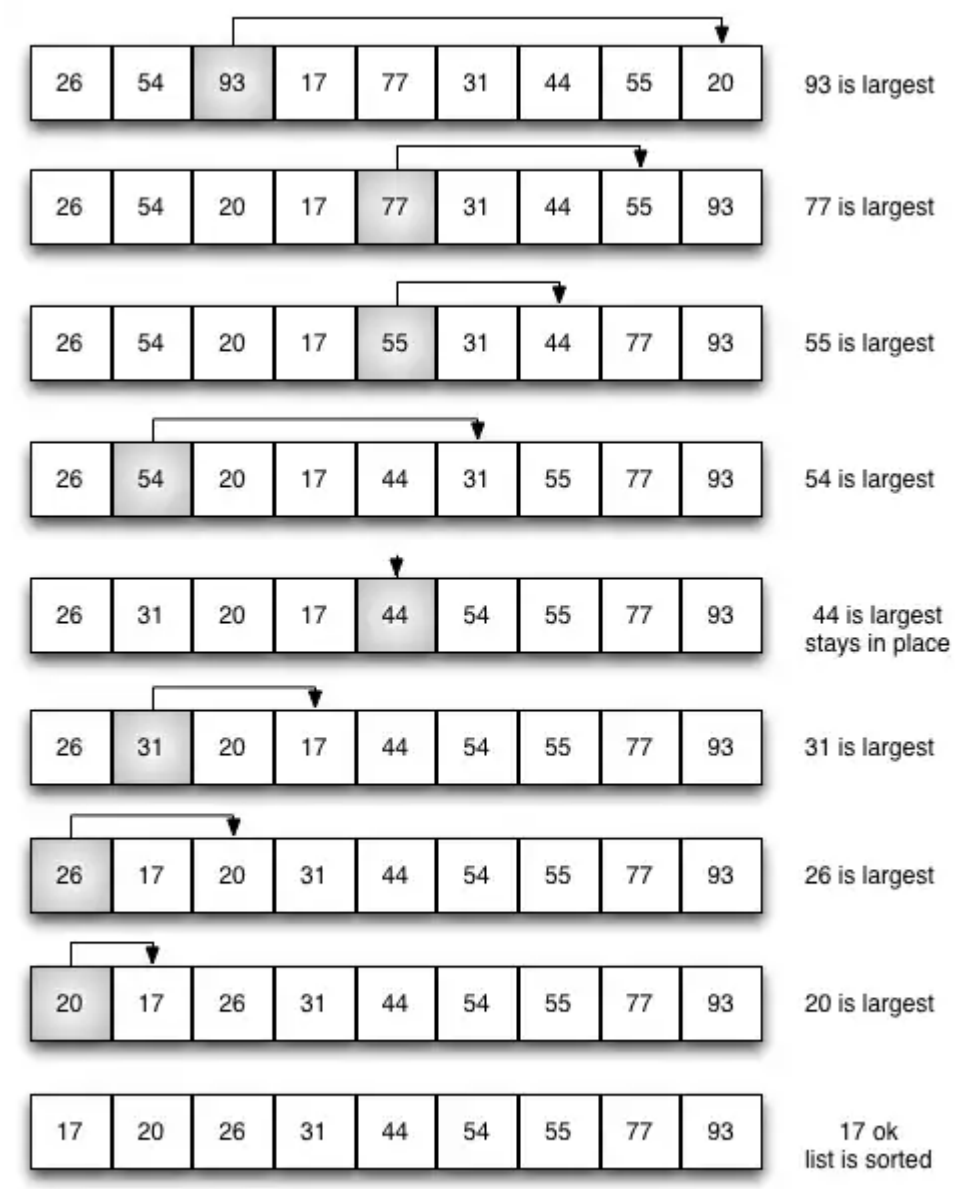

Figure 3: `selectionSort`

Insertion Sort

- always maintains a sorted sublist in the lower positions of the list

- O(n^2)
  - Comparisons and swaps

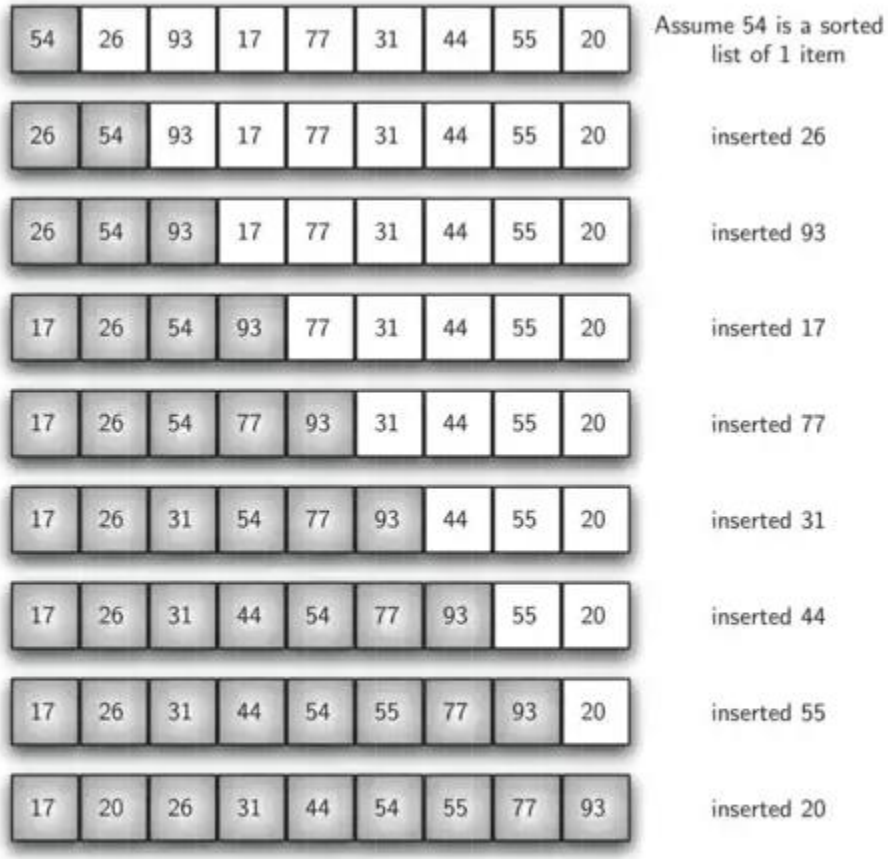

Figure 4: `insertionSort`

학습방법

1. C++/Java/Python sort 활용법을 익히자!!!
2. 다음으로
    1. Merge sort, counting sort, bubble sort는 구현할 수 있어야함!
3. 면접/시험을 위해 (코딩 + 칠판 설명)
    1. Merge sort, quick sort, counting sort, radix sort
    2. (bubble, insert, select)