CSE-5311 Algorithms

* Big-O notation

$$\underbrace{T(n)}_{\text{Real Function}} \leq C \times \underbrace{F(n)}_{\text{O functions}}$$

1) Find constant C

2) Find the smallest n (n_0)

* Big-θ (theta) Notation

* Big-Ω (Omega) Notation

* Array Operations
- lookup
- append
- insert
- delete
- extend
- slice

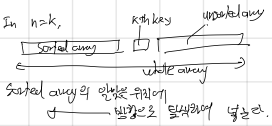
* Dynamic Array Operations
- lookup
- append
- insert
- delete

* Searching Algorithms

1. Linear Search
2. Binary Search

* Sorting Algorithms

1. Insertion Sort - n^2
2. Merge Sort - $n \cdot \log n$
3. Selection Sort - n^2
4. Bubble Sort - n^2
5. Quick Sort - n^2
   Three-way Quick Sort - n^2
   Median-of-Three Quick Sort - n^2



Sorted array의 알맞은 위치에 ← 방향으로 탐색하여 넣는다.

*Quicksort의 키워드는 pivot

* K-th smallest selection

1. Sort the array - $n \cdot \log n$.
2. Find the smallest in each iteration — $O(n \cdot k)$
3. Quick Select Algorithm — $O(n^2)$
4. Median of Medians — $O(n)$

* Inversion Count Problem

1. Brute Force — $O(n^2)$
2. Merge Sort — $O(n \cdot \log n)$
3. Bubble Sort — $O(n^2)$
4. Quick Sort — $O(n \cdot \log n)$

* Tree (Graph) Traversal

1) Breadth First Traversal

2) Depth First Traversal
- Pre-order Traversal — Graph default.
- In-order Traversal
- Post-order Traversal

* Tree

1) Binary Tree

2) Complete Binary Tree - Heap
- $i \to \lfloor \frac{i}{2} \rfloor, 2i, 2i+1$

3) Disjoint Set

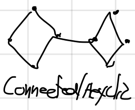
4) Binary Search Tree
- AVL Tree
- Red Black Tree — LR, RR 등 rotation 대신

B
R R — 색으로 recolor
B

* Splay Tree

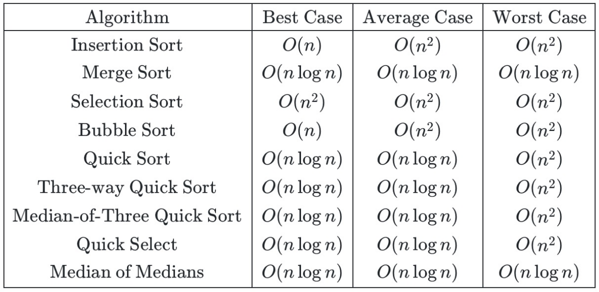
* Graph Theory
- Distance
- Eccentricity
- Degree (of Vertex)

- Simple Graph
  - Connected / Disconnected Graph
  - Complete Graph
  - Cycle / Acyclic Graph
  - (Complete) Bipartite Graph
  - Complement Graph



- Spanning Tree (Minimum Spanning Tree)

- Cut Vertex, edge, and Set



Algorithm	Best Case	Average Case	Worst Case
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Three-way Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Median-of-Three Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Quick Select	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Median of Medians	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Algorithms

1.

```
def search (arr, x):
    for i, element in enumerate (arr):
        if element == x:
            return i
    return -1
```

2.

0	1	2	···	30	31

$32 = 2^5 \rightarrow k=5$ $\quad \log_2 2^{k+1} = 6$

직접 세보는 것도..

✲ The last element which binary search needs to find.

3.

A

| 1 | 1 ·· | n |

a. $A[n]$ — $O(1)$

b. add all values while iterating to the array and divide n. $O(n)$

c. n : odd

$\frac{n+1}{2} \rightarrow O(1)$

4. (a) : $O(n^3)$ (b) : $O(n^2)$ (c) : $O(\log n)$

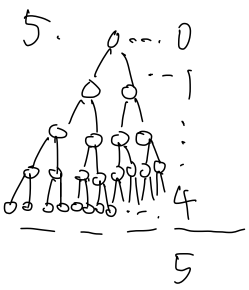
(d) : $O(\log^2 n)$ (e) : $O(2^n)$ (f) : $O(n^5)$

$x = \log n$

$\log x + x^2$

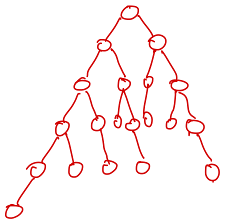
(g) : $O(n^2 \cdot \log n)$

5.



~~$2^5 - 1 = 31$~~

$2^6 - 1 = 63$



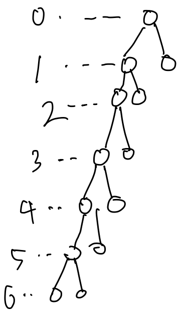
MIN: 양쪽으로 크게 먼저 그리고,

나머지는 AVL 조건에 맞게 채우기

6. ✲ Full Tree : Every node except leaf has two children w/o complete graph condition

✲ Perfect tree : All internal nodes have two children and all leaves are in the same level.

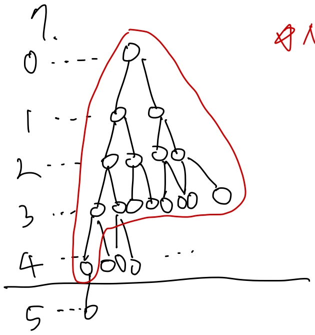
(→ complete & full tree)



⇒ MIN : 13

Max : $2^n - 1 = 127$ (perfect tree)

7.



✲ Non-leaf node

MIN : $2^4 - 1 + 1 = 16$

Max : $2^5 - 1 = 31$

8. ✲ Depth of Root = 0

9. J, Y, W, Z, P, A, X, C

10.

Best	Worst
logn	n
logn	logn
logn	logn

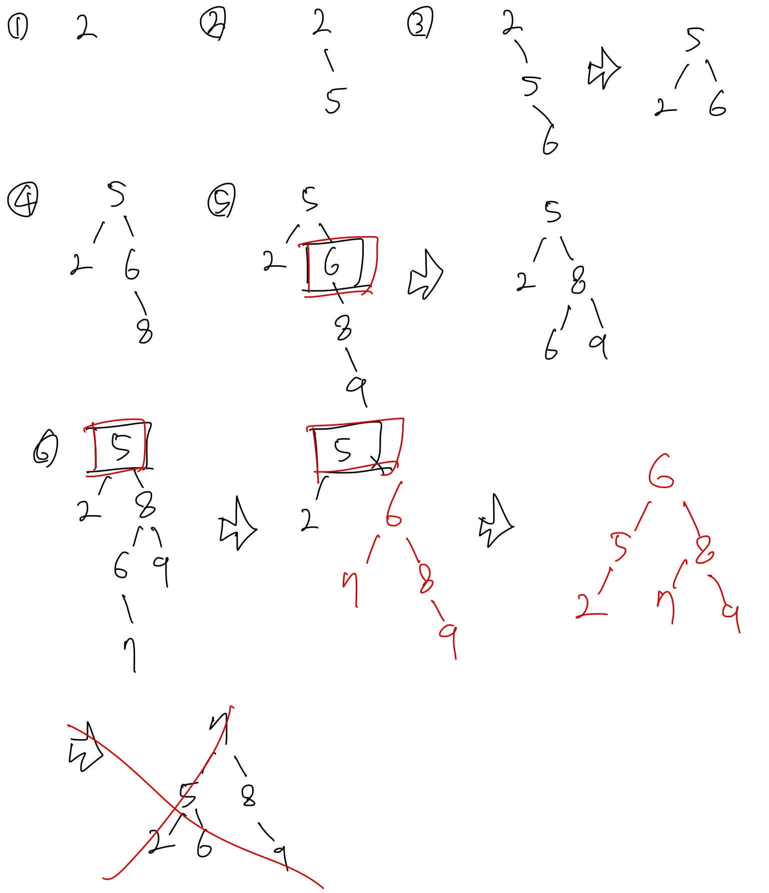
11.

Full	: a, c, f
Complete	: a, f
AVL balanced	: a, c, e, f
Perfect	: a,

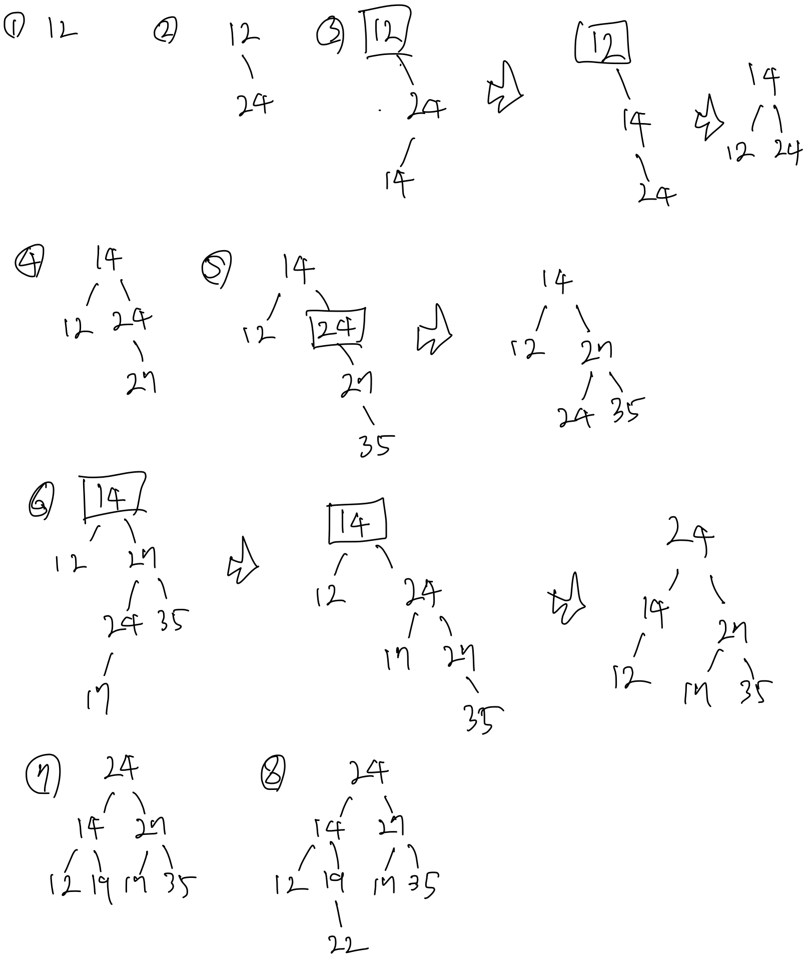
12.

	sorted array	reversed ··
Quick	n^2	n^2
Heap	$n \cdot \log n$	$n \cdot \log n$
Insertion	✲ n	n^2
Selection	n^2	n^2

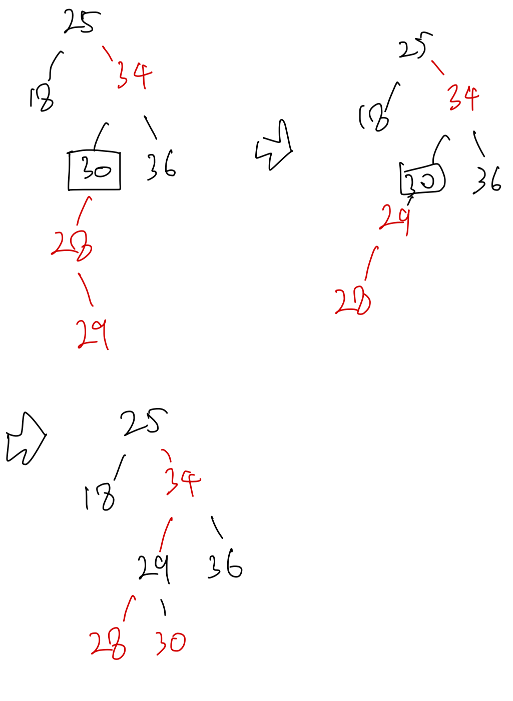
13.



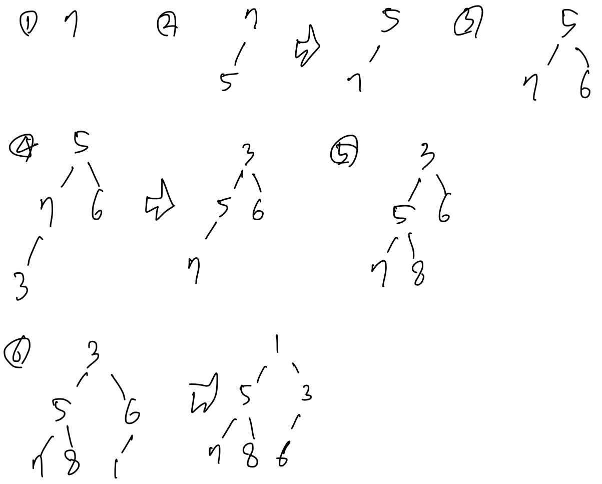
14.



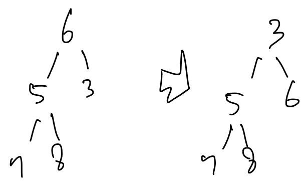
15.



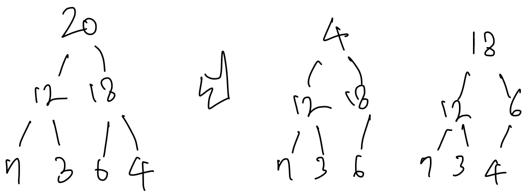
16.



17.



18.



18 12 6 7 3 4

19. Y X H G T C A F B Q R

20. Y X P G T H A F B Q R C

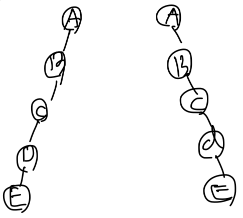
21.

H, ~~G~~ ≤ ? ≤ ~~O~~ N

H, I, J, K, L, M, N, ~~O, P, Q, R, S, T, U~~

✻ 기존 heap에서의 서열 관계도 고려해야 함!

22.



23.

* Sorted Set 2개 합치기 — 두 set에서 하나씩 pop 해가며 비교

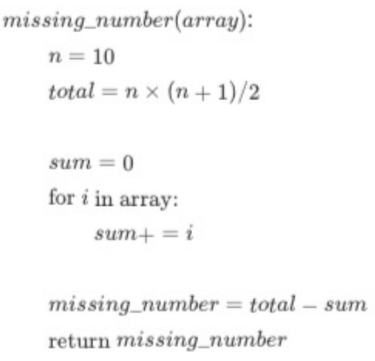
: $O(2n) = O(n)$

24.

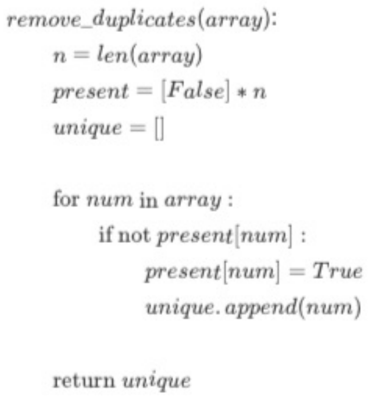
* All possible spanning trees of graph G

have the same number of edges and vertices.

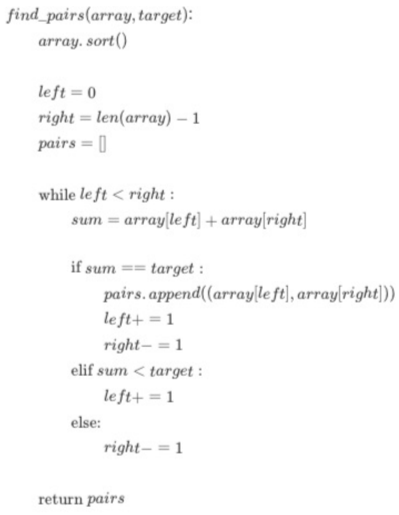
(n vertices and n-1 edges)



```
missing_number(array):
    n = 10
    total = n × (n + 1)/2

    sum = 0
    for i in array:
        sum+ = i

    missing_number = total − sum
    return missing_number
```

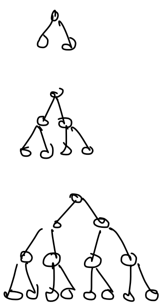


```
remove_duplicates(array):
    n = len(array)
    present = [False] ∗ n
    unique = []

    for num in array :
        if not present[num] :
            present[num] = True
            unique.append(num)

    return unique
```



```
find_pairs(array, target):
    array.sort()

    left = 0
    right = len(array) − 1
    pairs = []

    while left < right :
        sum = array[left] + array[right]

        if sum == target :
            pairs.append((array[left], array[right]))
            left+ = 1
            right− = 1
        elif sum < target :
            left+ = 1
        else:
            right− = 1

    return pairs
```

25.



N=1 : 1

N=3 : 2

N=7 : 4

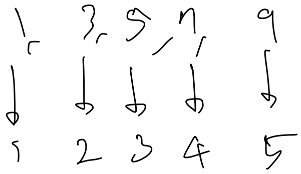
N=15 : 8

$2^{h+1} - 1 = N$ 2^h 개

$2^{h+1} = N+1$

$2^h = \left\lfloor \frac{N+1}{2} \right\rfloor$ 개

26.



$2^k - 1$: # of nodes

K : # of leaf nodes.

27.

24 < ~~68~~ ~~62~~ ~~58~~ 54

28.

* The maximum difference in Leaves of AVL

: $h - \frac{h}{2} = \frac{h}{2}$

where $h = \log n$

29.