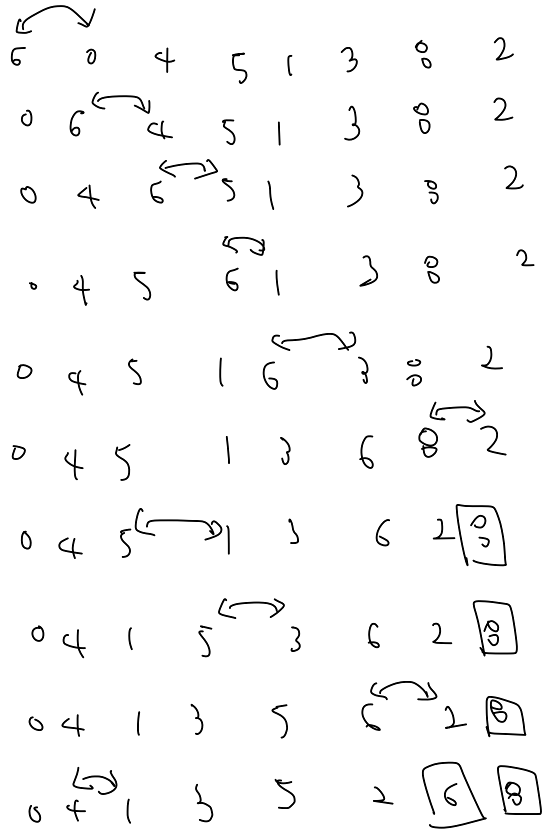
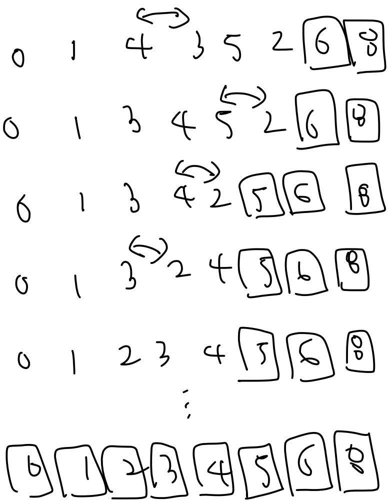
20210878 김윤서

```
1. void bubble_sort(int list[], int n)
   {
     for(i=n-1; i>0; i--)
       for(j=0; j<i; j++)
       {
         j번째 항목과 j+1번째 항목 비교
          j번째 항목이 더 크면 2개 항목 교환
       }
       교환이 더 이상 발생하지 않으면 stop
       }
   }
```





* avg: $O(n^2)$
worst: $O(n^2)$
space: $O(1)$

```
2. for i←0 to n-2 do
       least ← A[i], A[i+1], ···, A[n-1] 중에서 가장 작은 값의 인덱스;
       A[i]와 A[least]의 교환;
       i++;
```

$$\overset{\leftrightarrows}{\boxed{6}\quad\boxed{0}}\quad 4\quad 5\quad 1\quad 3\quad 8\quad 2$$

$$0\quad \boxed{6}\quad 4\quad 5\quad \boxed{1}\quad 3\quad 8\quad 2$$

$$0\quad 1\quad \boxed{4}\quad 5\quad 6\quad 3\quad 8\quad \boxed{2}$$

$$0\quad 1\quad 2\quad \boxed{5}\quad 6\quad \boxed{3}\quad 8\quad 4$$

$$0\quad 1\quad 2\quad 3\quad \boxed{6}\quad 5\quad 8\quad \boxed{4}$$

$$\vdots$$

$$0\quad 1\quad 2\quad 3\quad 4\quad \underline{5}\quad \boxed{8}\quad \boxed{6}$$

$$0\quad 1\quad 2\quad 3\quad 4\quad 5\quad 6\quad 8$$

* avg: $O(n^2)$

worst: $O(n^2)$

space: $O(1)$

3.
```
for i ← 1 to n-1 do
    key ← A[i];
    j ← i-1;
    while (j ≥ 0 and A[j] > key) do
        A[j+1] ← A[j];
        j ← j-1;
    A[j+1] ← key
```

$$\begin{tikzcd}6 & \boxed{0} \arrow[l, bend right] & 4 & 5 & 1 & 3 & 8 & 2\end{tikzcd}$$

$$\begin{tikzcd}0 & 6 & \boxed{4} \arrow[l, bend right] & 5 & 1 & 3 & 8 & 2\end{tikzcd}$$

$$\begin{tikzcd}0 & 4 & 6 & \boxed{5} \arrow[l, bend right] & 1 & 3 & 8 & 2\end{tikzcd}$$

$$\begin{tikzcd}6 & 4 & 5 & 6 & \boxed{1} \arrow[lll, bend right] & 3 & 8 & 2\end{tikzcd}$$

$$\begin{tikzcd}0 & 1 & 4 & 5 & 6 & \boxed{3} \arrow[lll, bend right] & 8 & 2\end{tikzcd}$$

$$\begin{tikzcd}0 & 1 & 3 & 4 & 5 & 6 & 8 & \boxed{2} \arrow[lllll, bend right]\end{tikzcd}$$

$$\begin{tikzcd}0 & 1 & 2 & 3 & 4 & 5 & 6 & 8\end{tikzcd}$$

*avg: $O(n^2)$

worst: $O(n^2)$

space: $O(1)$

4.

```
quickSort(left, right)
    if right-left <= 0
        return
    else
        pivot = A[right]
        partition = partitionFunc(left, right, pivot)
        quickSort(left, partition-1)
        quicksort(partition+1, right)
```

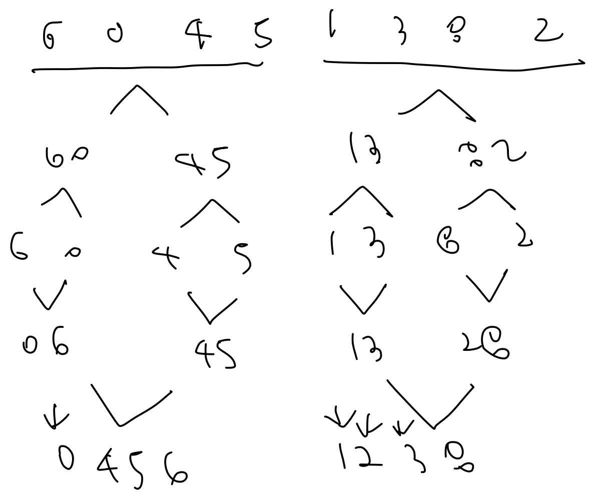
$$\begin{array}{cccccccc}
\downarrow & & & & & & \text{low} & \text{high} \\
6 & 0 & 4 & 5 & 1 & 3 & 8 & 2 \\
6 & 0 & 4 & 5 & 1 & 3 & 2 \,\vdots & 8 \\
\downarrow & & & & & & & \\
0 & 4 & 5 & 1 & 3 & 2 & \boxed{6} & 8 \\
 & \downarrow & \text{low} & & & \text{high} & & \\
\boxed{0} & 4 & 5 & 1 & 3 & 2 & \boxed{6} & 8 \\
\boxed{0} & 4 & 2 & 1 & 3 \,\vdots & 5 & \boxed{6} & 8 \\
 & \downarrow & & & & & & \\
\boxed{0} & 2 & 1 \,\vdots & 3 & \boxed{4} & 5 & \boxed{6} & 8 \\
\boxed{0} & \boxed{1} & \boxed{2} & \boxed{3} & \boxed{4} & \boxed{5} & \boxed{6} & \boxed{8}
\end{array}$$

* avg : $O(n\log n)$

worst : $O(n^2)$

space : $O(\log n)$

5.
```
if left < right
    mid = (left + right) / 2;
    merge_sort(list, left, mid);
    merge_sort(list, mid+1, right);
    merge(list, left, mid, right);
```



0

0 1

0 1 2

0 1 2 3

0 1 2 3 4

0 1 2 3 4 5

0 1 2 3 4 5 6

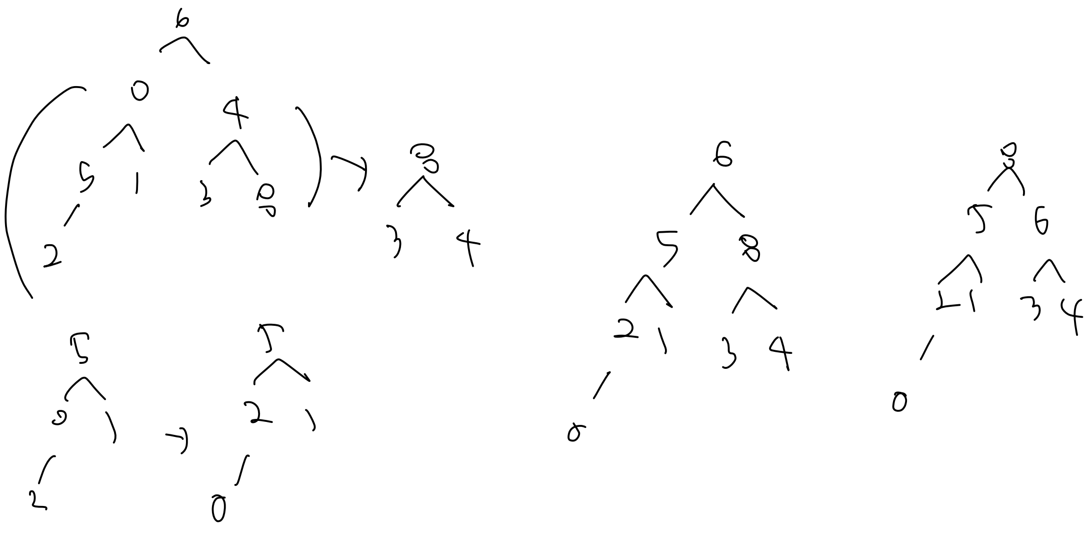
0 1 2 3 4 5 6 8

* avg: O(nlogn)

worst: O(nlogn)

space: O(n)

6.



8 5 6 2 1 3 4 0

$$\underbrace{0\quad 5\quad 6\quad 2\quad 1\quad 3\quad 4}_{\text{다시힙}}\quad \boxed{8}$$

⋮

0 1 2 3 4 5 6 8

* avg: O(nlogn)

worst: O(nlogn)

space: O(1)

```
HeapSort(A)
    BuildMaxHeap(A)
    for i=A length downto 2
        exchange A[1] with A[i]
        A.heapsize=A.heapsize-1
        maxHeapify(A,1)
```