---------------------------- **1 & 2** ----------------------------

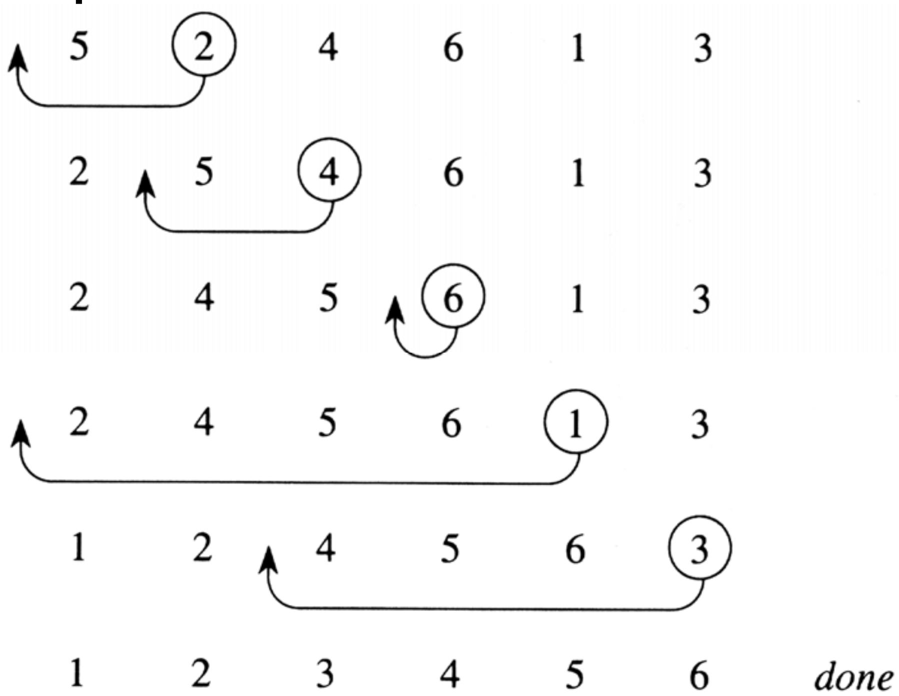
Introduction & Getting Started

1.1 Algorithms

Algorithm**:** A sequence of computational steps that transform the ***input*** of a ***computational problem*** into the ***output***.

2.1 Insertion Sort: An efficient algorithm for sorting a small number of elements.

Example:



2.2 Analyzing Algorithms

RAM**:** *Random-access machine*, in which each memory access takes unit time and instructions are executed one by one.

Running time**:** number of steps, which is a function of the ***input size***.

Example: Insertion Sort

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

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

***Best-case*:**

Each t_j=1. (The input A is sorted.)

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

(***rate of growth, order of growth***)

***Worst-case*: (upper bound)**

Each $t_j = j$.

$$\begin{aligned} T(n) &= k_1 n^2 + k_2 n + k_3 \\ &= \Theta(n^2) \end{aligned}$$

***Average-case*: (Expected running time)**

Each $t_j = j/2$.

$$\begin{aligned} T(n) &= t_1 n^2 + t_2 n + t_3 \\ &= \Theta(n^2) \end{aligned}$$

2.3 Designing Algorithms

***Divide-and-Conquer*:**

Divide: **(into the same problems of smaller size)**

Conquer:

Combine:

Example: Merge Sort

```
MERGE-SORT(A, p, r)
if p < r
   then q ← ⌊(p + r)/2⌋
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q + 1, r)
        MERGE(A, p, q, r)
```

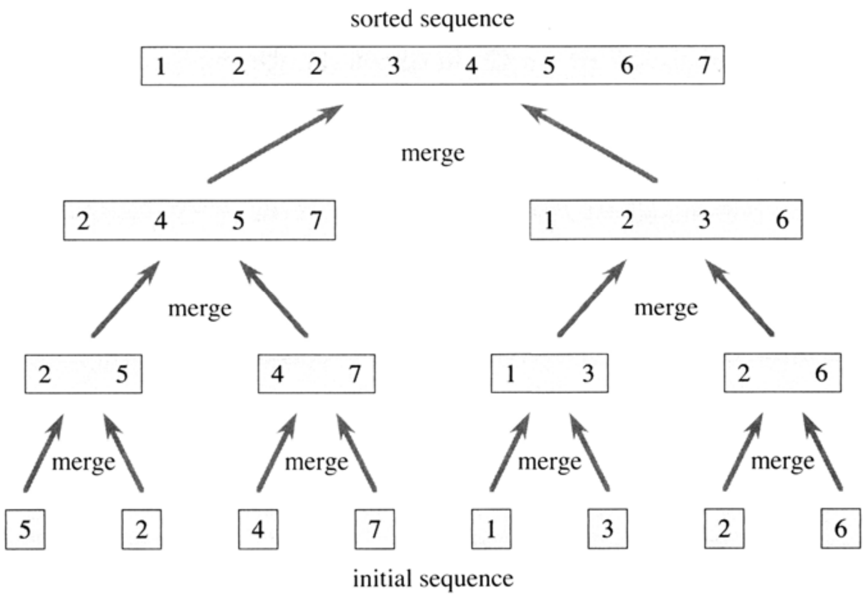


Figure 2.4 The operation of merge sort on the array $A = \langle 5, 2, 4, 7, 1, 3, 2, 6\rangle$. The lengths of the sorted sequences being merged increase as the algorithm progresses from bottom to top.

Analysis: (*recurrence*)

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

$$= \Theta(n \log n)$$

Homework: Pro. 2-1 and 2-4.