**In-Place Sorting**

**Knuth Vol 3**

"Art of Computer Programming"

Theorichian's Dream

- Merge Sort
- Divide and Conquer

\[
T(n) = \begin{cases} 1 & \text{if } n = 1 \\ \Theta(n) + 2T(n/2) & \text{otherwise} \end{cases}
\]

Time to split + recursive sort = time merge

\[
T(n) = 2T(n/2) + \Theta(n)
\]

\[
T(n) = \frac{2^k T(n/2^k) + n}{K}
\]

\[
T'(n) = 2T'(n/2) + n
\]

\[
T(n) = \sum_{k=0}^{\log n} 2^k T(n/2^k) + \Theta(n)
\]
$T(1) = O(1)$

$T(n) = 2T(n/2) + n$

$t_h = O(2^h)$

$t_h = 2t_{h-1} + 2^h$

$(\varepsilon - 2) \Rightarrow t_h = O(2^h)$

$\Rightarrow T(n) = \Theta(n \log n)$
**In-Place**

\[ \text{Insertion Sort} \]

1. \[ j = 2 \]
2. \[ 3 \ldots n \]
3. Sort and insert

**Average Case Behavior**

- \[ n! \text{ permutations equally likely} \]
- \[ M = \# \text{ of comparisons} \]
- \[ M + n - 1 \]

\[ d_j = \frac{\text{Inversions negative of } a_j}{\# a_j} \]

\[ d_1 = 0 \]

\[ g(m) = \frac{1}{n!} \sum_{\pi \in \text{permute}} \sum_{i=1}^{n} [d_i \neq \pi] \]

\[ 0 \leq d_i < i \]

\[ \pi_1, \pi_2, \ldots, \pi_m \]

- \[ \text{How many permutations have } d_i = m ? \]

\[ \# [d_i = m] \]

\[ 0 \leq m < i \]

\[ \sum_{i=1}^{n} \sum_{m=0}^{i-1} m \times \# [d_i = m] \]

\[ \frac{n!}{c(i-1)! (n-i)!} \]

\[ a, a_2, a_3 \ldots a_i, a_{i+1} \ldots a_n \]

\[ \text{meets } a \]

\[ \frac{c(i-1)!}{c(i-1)! (n-i)!} \]
Figure 2.2 The operation of INSERTION-SORT on the array $A = \{5, 2, 4, 6, 1, 3\}$. Array indices appear above the rectangles, and values stored in the array positions appear within the rectangles. (a)–(e) The iterations of the for loop of lines 1–8. In each iteration, the black rectangle holds the key taken from $A[j]$, which is compared with the values in shaded rectangles to its left in the test of line 5. Shaded arrows show array values moved one position to the right in line 6, and black arrows indicate where the key moves to in line 8. (f) The final sorted array.

**INSERTION-SORT**

1  for $j = 2$ to $A.length$
2  \hspace{1em} key = A[j]
3  \hspace{1em} // Insert $A[j]$ into the sorted sequence $A[1 \ldots j - 1]$.
4  \hspace{1em} i = j - 1
5  while $i > 0$ and $A[i] > key$
6  \hspace{2em} $A[i + 1] = A[i]$
7  \hspace{2em} $i = i - 1$
8  \hspace{1em} $A[i + 1] = key$

**Loop invariants and the correctness of insertion sort**

Figure 2.2 shows how this algorithm works for $A = \{5, 2, 4, 6, 1, 3\}$. The index $j$ indicates the "current card" being inserted into the hand. At the beginning of each iteration of the for loop, which is indexed by $j$, the subarray consisting of elements $A[1 \ldots j - 1]$ constitutes the currently sorted hand, and the remaining subarray $A[j + 1 \ldots n]$ corresponds to the pile of cards still on the table. In fact, elements $A[1 \ldots j - 1]$ are the elements originally in positions 1 through $j - 1$, but now in sorted order. We state these properties of $A[1 \ldots j - 1]$ formally as a loop invariant:

At the start of each iteration of the for loop of lines 1–8, the subarray $A[1 \ldots j - 1]$ consists of the elements originally in $A[1 \ldots j - 1]$, but in sorted order.

We use loop invariants to help us understand why an algorithm is correct. We must show three things about a loop invariant:
\[ E(M) = \sum_{i=1}^{n} \frac{1}{i} \sum_{m=0}^{i-1} E^m \]
\[ = \sum_{i=1}^{n} \frac{\xi^\prime(i-1)}{2} \]
\[ = \frac{1}{2} \sum_{i=1}^{n} (i-1) \]
\[ = \frac{1}{2} \cdot \frac{n(n-1)}{2} \]
\[ \boxed{\sum_{i=1}^{n} \frac{1}{i} = \frac{n}{2} + \Theta(n)} \]
Heap Sort

\[ a, a_2, \ldots, a_n \]

\text{parent of } a_i = a[i/2] \\
\text{left}(a_i) = a_{2i} \\
\text{right}(a_i) = a_{2i+1}

\sum_{i=1}^{n} \Theta(i) \approx \int_{0}^{n} \log(x) dx \\
\Theta(n) + \Theta(n) = \Theta(n)