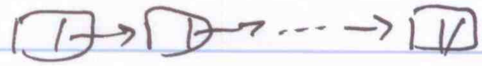
SORTING — IN PLACE

a_1	a_2	a_3	$\cdots$	a_n

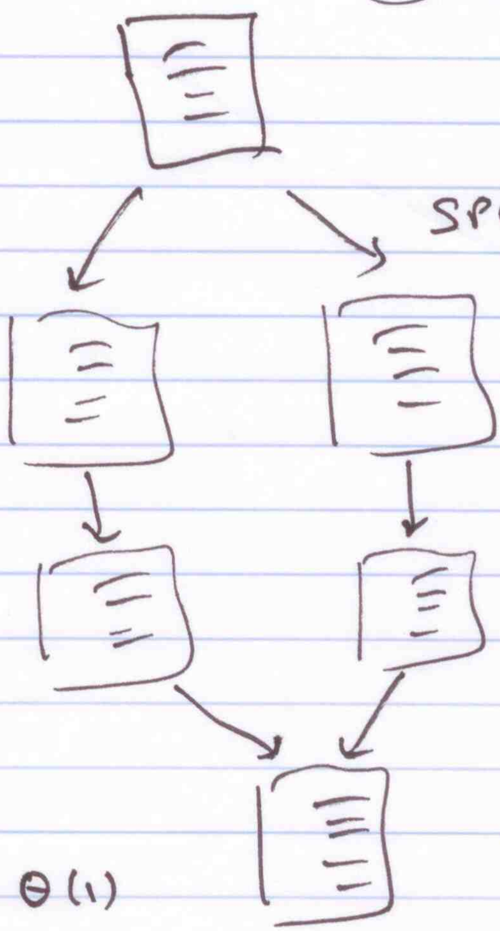
$$\Downarrow$$

$$a_1 \le a_2 \le a_3 \le \cdots \le a_n$$

Merge Sort (D+C)



$\boxed{T(n)}$



SPLIT $\boxed{\Theta(n)}$

SORT RECURSIVELY $\boxed{2T(n/2)}$

$T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor)$

MERGE $\boxed{\Theta(n)}$

KNUTH - VOL 3

$$T(1) = \Theta(1)$$

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

$T(1) = 1$

$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n$

$\boxed{n = 2^k}$

$\Downarrow$

$k = \lg n$

$$T(2^0) = \Theta(1)$$

$$T(2^k) = 2T(2^{k-1}) + \Theta(2^k)$$

$t_k = T(2^k)$

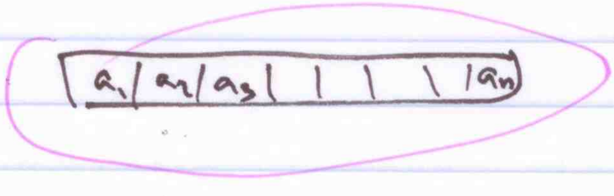
$$\begin{cases} t_k = 2t_{k-1} + \Theta(2^k) \\ t_0 = \Theta(1) \end{cases}$$

$(E-2)(E-2)$

$= (E-2)^2$

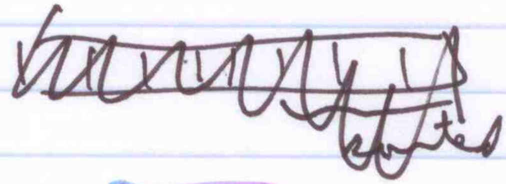
$\Theta(k 2^k) = \Theta(n \lg n)$

INSERTION SORT

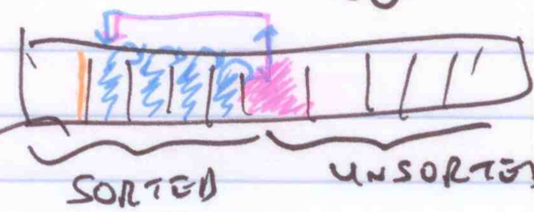


Random – Permutation of $1, 2, 3, \ldots n$

$n!$



To the left of [pink] and larger



(cost(perm) × prob(perm))

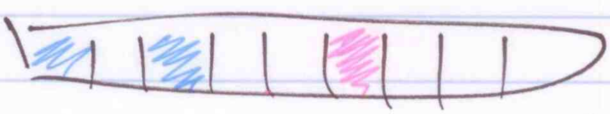
$$E(M) = \sum_{n!\ \text{permutations}} (\#\text{ of times 6 \& 7 are executed})/n!$$

$\frac{n^2}{4}$

# of blue items left of pink item is called the number of

INVERSIONS

w.r.t. pink $\boxed{a_i}$ — d_i



$$E(M) = \frac{1}{n!} \sum_{\substack{\text{permutations} \\ \pi}} \sum_{i=1}^{n} (d_i \text{ for } \pi)$$

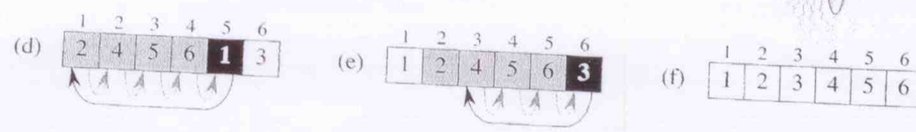


Figure 2.2 The operation of INSERTION-SORT on the array $A = \langle 5, 2, 4, 6, 1, 3 \rangle$. Array indices appear above the rectangles, and values stored in the array positions appear within the rectangles. **(a)–(e)** The iterations of the **for** loop of lines 1–8. In each iteration, the black rectangle 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.

execute n-1 times

INSERTION-SORT(A)

```
                                               n
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j - 1].
4      i = j - 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i - 1
8      A[i + 1] = key
```

n-1 Times

# of times successfully

$\Theta(n^2)$

?

Sorted array decreasing order

$\sum_{i=n}(i-1) = \frac{n(n-1)}{2}$

AVG ? $\frac{n^2}{4}$

Loop invariants and the correctness of insertion sort

Figure 2.2 shows how this algorithm works for $A = \langle 5, 2, 4, 6, 1, 3 \rangle$. 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 $A[1 \,..\, j-1]$ constitutes the currently sorted hand, and the remaining subarray $A[j+1 \,..\, n]$ corresponds to the pile of cards still on the table. In fact, elements $A[1 \,..\, j-1]$ are the elements *originally* in positions 1 through $j-1$, but now in sorted order. We state these properties of $A[1 \,..\, j-1]$ formally as a ***loop invariant***:

> At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1 \,..\, j-1]$ consists of the elements originally in $A[1 \,..\, j-1]$, 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:

$$
n!\left\{
\begin{array}{ll}
\quad\quad\quad\; a_i \downarrow & \\
1\ 2\ 3\ \cdots\ n & d_i \\
2\ 1\ 3\ \cdots\ n & d_i \\
3\ 1\ 2\ \cdots\ n & \vdots \\
\vdots & \vdots \\
n\ \ n-1\ \cdots\ 1 & d_i
\end{array}
\right.
$$

$d_i = 0$ $\qquad$ $\#[d_i = 0] \times 0$

$d_i = 1$ $\qquad$ $\#[d_i = 1] \times 1$

$\vdots$

$d_i = i-1$ $\qquad$ $\#[d_i = i-1] \times (i-1)$

$$E(M) = \frac{1}{n!}\sum_{i=1}^{n}\sum_{m=0}^{i-1} m\,\boxed{\#[d_i = m]} = \sum_{i=1}^{n}\frac{1}{i}\underbrace{\sum_{m=0}^{i-1} m}_{\frac{i(i-1)}{2}}$$

# of permutations in which a_i has m larger elts to its left.

$\binom{n}{i}(i-1)!\,(n-i)!$

$\cancel{\frac{n!}{i!(n-i)!}}\,(i-1)!\,\cancel{(n-i)!}$

a_i

$\underbrace{\qquad}_{\binom{n}{i}}$ $\qquad$ $\underbrace{\qquad}_{n-i}$ $\qquad$ $(n-i)!$

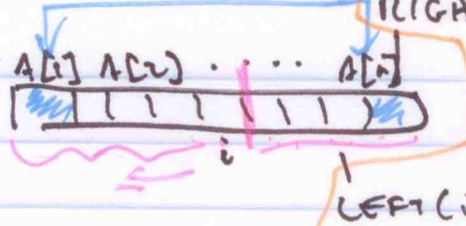
$\underbrace{\qquad}_{i-1}$ $\qquad$ $(i-1)!$

$$\frac{1}{2}\sum_{i=1}^{n}\frac{(i-1)}{\cancel{2}}$$

$$\frac{1}{2}\underbrace{(0+1+2+\cdots+n-1)}_{\frac{n(n-1)}{4}}$$

HEAPSORT

SWAP



RIGHT(i) = 2i+1

LEFT(i) = 2i

PARENT(i) = ⌊i/2⌋

BINARY HEAP

a_i

a_{2i} $\quad$ a_{2i+1}

CREATE — $\Theta(n)$

$\frac{n}{2^{h+1}}$ $\quad$ h

ORDERED HEAP

A [PARENT(i)] ≥ A[i]

PARENTS ≥ CHILDREN — RECURSIVELY

h

$\Theta(h)$

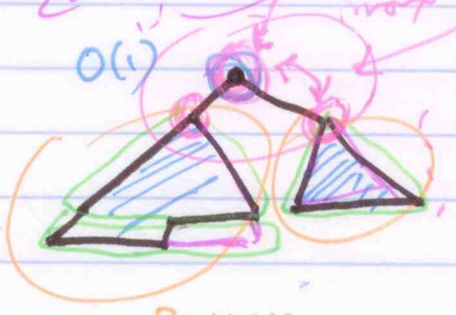
$\Theta\left(\sum_{h=1}^{\lg n} h \cdot \frac{n}{2^{h+1}}\right)$

$\sum \frac{h}{2^h}$

$\sum i x^{i-1}$ $\quad x = \frac{1}{2}$

swap w/ larger child

$O(1)$



NOT BINARY ORDERED HEAP

BINARY ORDERED HEAP

repeat n-1 times

$\Theta(n \lg n)$

$$T(n) = T\left(\overset{\frac{2}{3}n}{n/2}\right) + O(1)$$

$$\underbrace{t_h}_{E-1} = t_{h-1} + \underbrace{O(1)}_{E-1} \Rightarrow (E-1)^2$$

$t_h = \Theta(h)$

$n = \left(\frac{3}{2}\right)^h$

$h = \lg n \quad \log_{3/2} n$

$\boxed{T(n) = \Theta(\lg n)}$