---------------------------- 8 ----------------------------

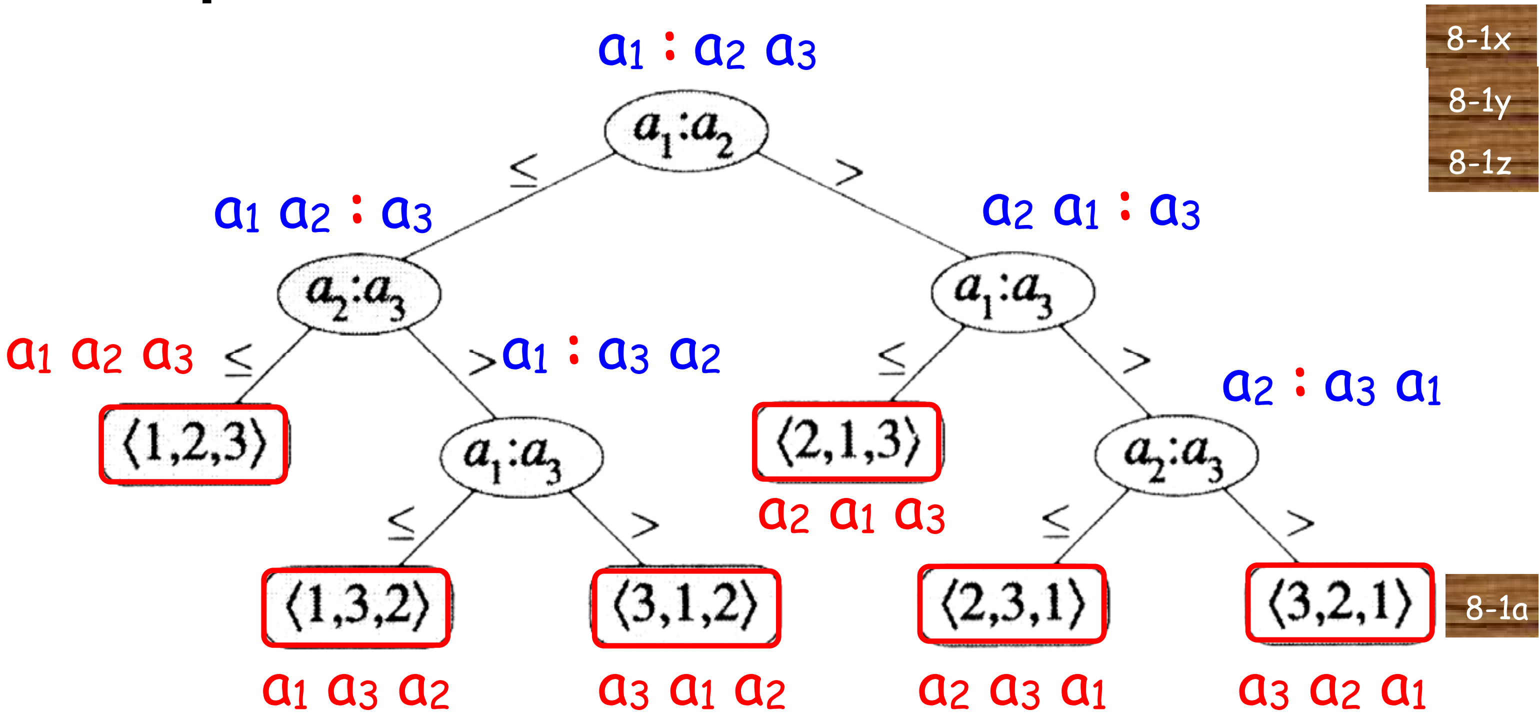
# Sorting in Linear Time

## 8.1 Lower bounds for sorting

***Comparison sorts***: Determine the sorted order based only on comparisons between the input elements (<, >, =, ≤, ≥). We may not inspect the values of the elements or gain order information about them in any other way.

***The decision-tree model***: A decision tree is ① a full binary tree that represents the comparisons performed by a sorting algorithm that operates on an input of a given size. In a decision tree, each ② internal node is annotated by $a_i : a_j$, and each ③ leaf is annotated by a permutation $\langle \pi(1), \pi(2), ..., \pi(n) \rangle$.

**Example:** Decision tree of insertion sort with $n$=3.



## Lower bound for the worst case

- Each of the $n$! permutations on $n$ elements must appear as a leaf. (n = 3, 3! = 6 leaves)

- Worst case number of comparisons is equal to the height (the longest path from root to a leaf).

8-2x

- A binary tree of height $h$ contains at most $2^h$ leaves. We have $n! \le 2^h$, which implies

  $h \ge \lg(n!)$.

  Using Stirling's approximation (3.18):

  $$n! = \underbrace{\sqrt{2\pi n}}_{①}\underbrace{(n/e)^n}_{②}\underbrace{(1+\Theta(1/n))}_{③},$$

  we have

  $$h \ge \underbrace{\lg(n/e)^n}_{②} = \Theta(n \lg n)$$

**Theorem 8.1** Any decision tree that sorts $n$ elements has height $\Omega(n \lg n)$.

at least

8-2a

**Corollary 8.2** Heapsort and merge sort are asymptotically optimal comparison sorts.

## 8.2 Counting sort

(Assume that each input is an integer in [0..$k$-1].)

$A$[1..$n$]: input  $C$[0..$k$-1]: counter  $B$[1..$n$]: output

Counting-Sort($A$, $B$, $k$)
**for** $i \leftarrow 0$ **to** $k$-1 **do** $C[i] \leftarrow 0$ O(k) /* Reset counters */
**for** $i \leftarrow 1$ **to** $n$ **do** $C[A[i]] \leftarrow C[A[i]]+1$ O(n) /* counting */
**for** $i \leftarrow 1$ **to** $k$-1 **do** $C[i] \leftarrow C[i]+C[i-1]$ /* prefix sums */ O(k)
**for** $i \leftarrow n$ **downto** 1 **do** O(n) /* output */
  $B[C[A[i]]] \leftarrow A[i]$
  $C[A[i]] \leftarrow C[A[i]]-1$

* positions for "3" are 5, 6, 7

* position for the last "3"

**Example:** $n$=8 and $k$=6.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| A | 2 | 5 | 3 | 0 | 2 | 3 | 0 | 3 |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 2 | 0 | 2 | 3 | 0 | 1 |

(a)

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 2 | 2 | 4 | 7 | 7 | 8 |

prefix sums

(b)

8-3x

A[8] = 3

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | | | | | | 3 | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 2 | 2 | 4 | 6 | 7 | 8 |

(c) position of the next "3"

A[7] = 0

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | 0 | | | | | 3 | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 1 | 2 | 4 | 6 | 7 | 8 |

(d)

A[6] = 3

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| *B* | | 0 | | | | 3 | 3 | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *C* | 1 | 2 | 4 | 5 | 7 | 8 |

(e)

A[5] = 2

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| *B* | 0 | 0 | 2 | 2 | 3 | 3 | 3 | 5 |

(f)

- not a comparison sort.
- **Time:** $T(n)=O(n+k)$  (=$O(n)$ if $k=O(n)$.)
- ***Stable sort***: numbers of the same value appear in the output array in the same order as they do in the input array.
- Counting sort is stable.

Heapsort & Quicksort are NOT !

**8.3 Radix sort:** (stable sort) on each digit *i* (*i*=1 to *d*)
(Every element consists of *d* digits each of which is an integer in the range [0..*k*-1].)

[k = 26 for strings]
[d = string length]

**Example:** *n*=7, *d*=3 and *k*=10

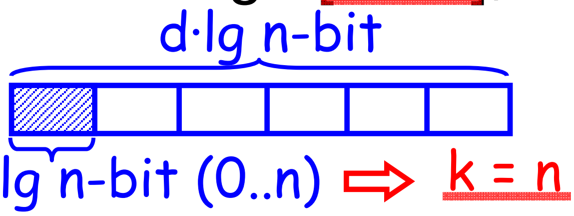
8-4x

| | i = 1 stable sort | i = 2 stable sort | i = 3 stable sort |
|---|---|---|---|
| 329 | 720 | 720 | 329 |
| 457 | 355 | 329 | 355 |
| 657 | 436 | 436 | 436 |
| 839 | 457 | 839 | 457 |
| 436 | 657 | 355 | 657 |
| 720 | 329 | 457 | 720 |
| 355 | 839 | 657 | 839 |
| $d_1$ | $d_2$ | $d_3$ | |

8-5x

8-5a

- $T(n)=O(d(n+k))$   ($=O(n)$ if $k=O(n)$ & $d=O(1)$.)
- $O(n)$ for sorting $n$ elements in the range $[0..n^d]$, where $d$ is a constant.




## 8.4 Bucket sort 8-5b

(The input distributes uniformly over the interval [0, 1).)

$A[1..n]$:input   $B[0..n-1]$: buckets

take k = n

Why insertion sort?

**Bucket-sort(*A*)** * initialize buckets: O(k)

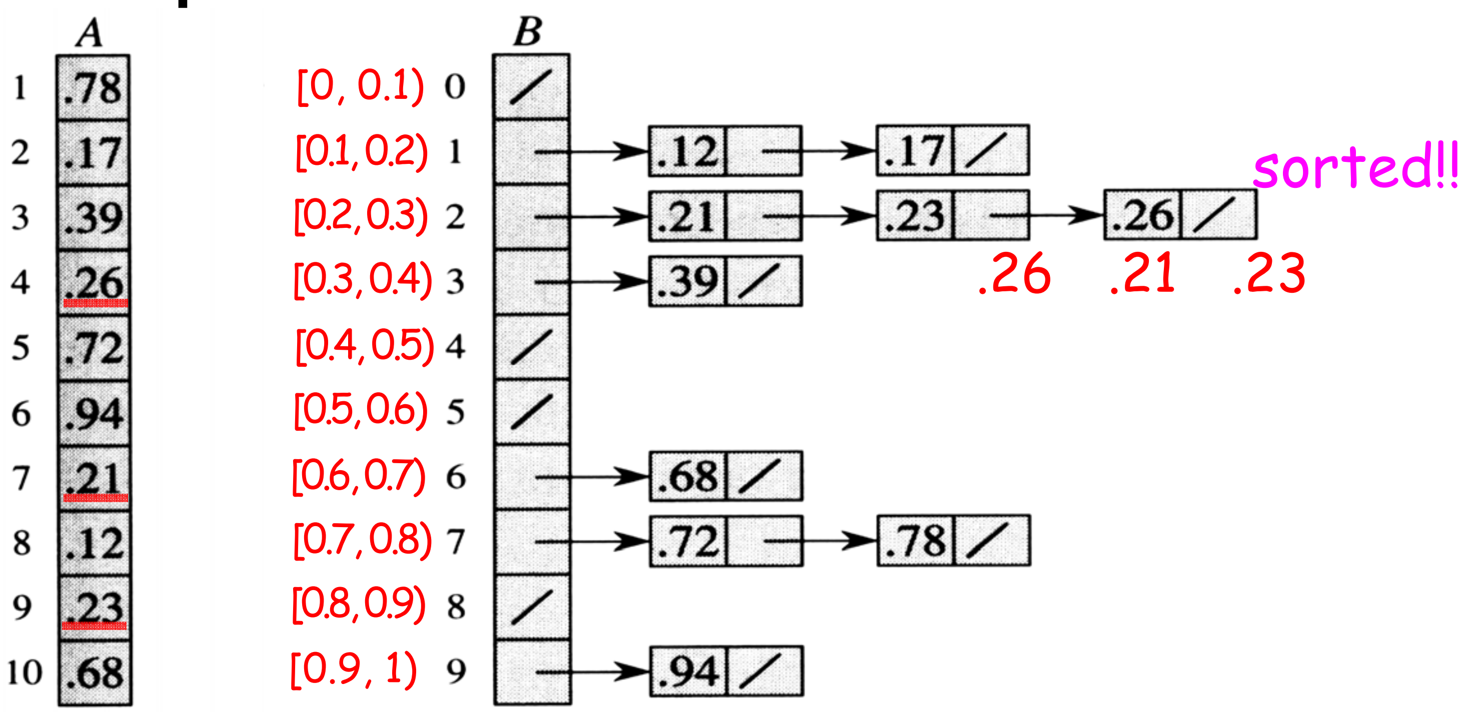
**for** $i \leftarrow 1$ **to** $n$ **do** insert $A[i]$ into list $B[\lfloor nA[i] \rfloor]$ O(n)

**for** $i \leftarrow 0$ **to** $n$-1 **do** sort list $B[i]$ by insertion sort O(?)

concatenate the lists $B[0]$, $B[1]$, ..., $B[n-1]$ O(k)

convert the list into an array O(n)

**Example:** $n=10$ → 10 buckets



- Worst case: $T(n) \;=\; O(n) + \sum_{0 \le i \le n-1} O(n_i^2)$

  $\sum n_i = n$

  $= \; O(n^2).$

- Average case: $T(n) = O(n) + \sum_{0 \le i \le n-1} O(E[n_i^2])$

  $\approx 2$

  $= O(n) + \sum_{0 \le i \le n-1} O(1)$

  $= O(n)$

  (See the textbook for $E[n_i^2] = \Theta(1)$.)

  $\approx 2$

**Homework:** Ex. 8.2-4, 8.3-2, 8.4-2, Prob. 8-3, 8-6.

To make all sort stable:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | | 2 | 1 | 3 | 4 |
| 4 | 2 | 4 | 4 | $\Rightarrow$ | 2 | 4 | 4 | 4 |

$\Downarrow$

(4, 1) (2, 2) (4, 3) (4, 4)