6.006 Introduction to Algorithms
Spring 2008

# Lecture 10: Sorting III: Linear Bounds Linear-Time Sorting

### Lecture Overview

- Sorting lower bounds
  - Decision Trees
- Linear-Time Sorting
  - Counting Sort

### Readings

CLRS 8.1-8.4

### Comparison Sorting

Insertion sort, merge sort and heap sort are all comparison sorts.
The best worst case running time we know is $O(n \lg n)$. Can we do better?

### Decision-Tree Example

Sort $< a_1, a_2, \cdots a_n >$.

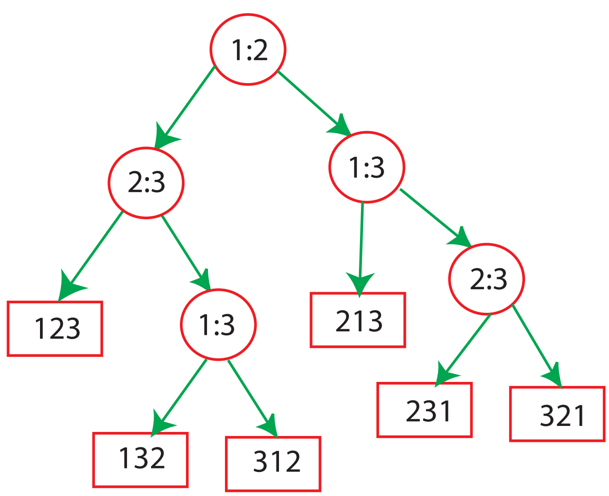


Figure 1: Decision Tree

Each internal node labeled $i : j$, compare $a_i$ and $a_j$, go left if $a_i \leq a_j$, go right otherwise.

**Example**

Sort $< a_1, a_2, a_3 >=< 9, 4, 6 >$ Each leaf contains a permutation, i.e., a total ordering.

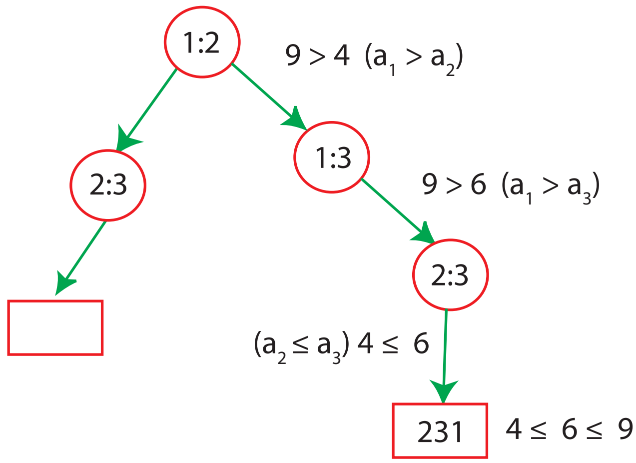


Figure 2: Decision Tree Execution

**Decision Tree Model**

Can model execution of any comparison sort. In order to sort, we need to generate a total ordering of elements.

- One tree size for each input size $n$
- Running time of algo: length of path taken
- Worst-case running time: height of the tree

**Theorem**

Any decision tree that can sort $n$ elements must have height $\Omega(n \lg n)$.

**Proof:** Tree must contain $\geq n!$ leaves since there are $n!$ possible permutations. A height-$h$ binary tree has $\leq 2^h$ leaves. Thus,

$$\begin{aligned} n! &\leq 2^h \\ \Longrightarrow h &\geq \lg(n!) \quad (\geq \lg((\frac{n}{e})^n) \text{ Stirling}) \\ &\geq n \lg n - n \lg e \\ &= \Omega(n \lg n) \end{aligned}$$

### Sorting in Linear Time

Counting Sort: no comparisons between elements

Input: $A[1 \dots n]$ where $A[j] \epsilon \{1, 2, \cdots, k\}$

Output: $B[1 \dots n]$ sorted

Auxiliary Storage: $C[1 \dots k]$

### Intuition

Since elements are in the range $\{1, 2, \cdots, k\}$, imagine collecting all the $j$'s such that $A[j] = 1$, then the $j$'s such that $A[j] = 2$, etc.

Don't compare elements, so it is not a comparison sort!

$A[j]$'s index into appropriate positions.

### Pseudo Code and Analysis

```
θ(k)  { for i ← 1 to k
            do C [i] = 0

θ(n)  { for j ← 1 to n
            do C [A[j]] = C [A[j]] + 1

θ(k)  { for i ← 2 to k
            do C [i] = C [i] + C [i-1]

θ(n)  { for j ← n downto 1
            do B[C [A[j]]] = A[j]
               C [A[j]] = C [A[j]] - 1
------
θ(n+k)
```

Figure 3: Counting Sort

**Example**

Note: Records may be associated with the $A[i]$'s.

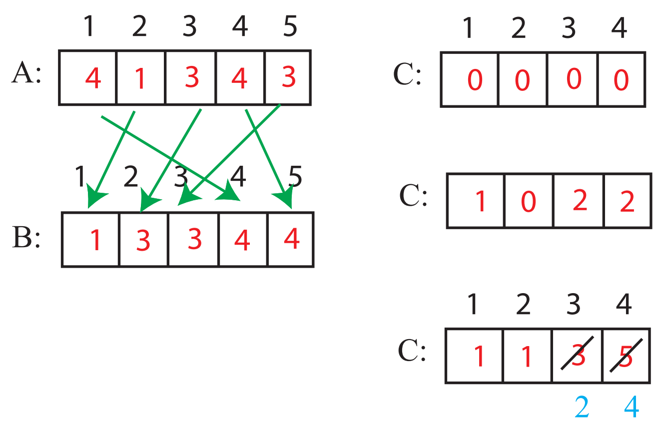


Figure 4: Counting Sort Execution

$$\begin{aligned}
A[n] &= A[5] = 3 \\
C[3] &= 3 \\
B[3] &= A[5] = 3, C[3] \text{ decr.} \\
A[4] &= 4 \\
C[4] &= 5 \\
B[5] &= A[4] = 4, C[4] \text{ decr.} \qquad \text{and so on \ldots}
\end{aligned}$$