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

![Decision Tree Diagram](image)

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.
Lower Bounds and Linear time sorting

Example

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

![Decision Tree Execution Diagram](image)

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,

\[
\begin{align*}
n! & \leq 2^h \\
\implies h & \geq \lg(n!) \quad (\geq \lg((\frac{n}{e})^n) \text{ Stirling}) \\
& \geq n \lg n - n \lg e \\
& = \Omega(n \lg n)
\end{align*}
\]
Sorting in Linear Time

**Counting Sort:** no comparisons between elements

**Input:** $A[1 \ldots n]$ where $A[j] \in \{1, 2, \cdots, k\}$

**Output:** $B[1 \ldots n]$ sorted

**Auxiliary Storage:** $C[1 \ldots k]$

**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**

```plaintext
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.

\[ A: \begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
4 & 1 & 3 & 4 & 3
\end{array} \]

\[ B: \begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
1 & 3 & 3 & 4 & 4
\end{array} \]

\[ C: \begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
0 & 0 & 0 & 0
\end{array} \]

\[ C: \begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
1 & 0 & 2 & 2
\end{array} \]

\[ C: \begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
1 & 1 & 3 & 4
\end{array} \]

\[ C: \begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
2 & 4
\end{array} \]

Figure 4: Counting Sort Execution

\[ C[3] = 3 \]
\[ A[4] = 4 \]
\[ C[4] = 5 \]