Complexity of Sorting

COMP 215 Lecture 14

Review of Sorts

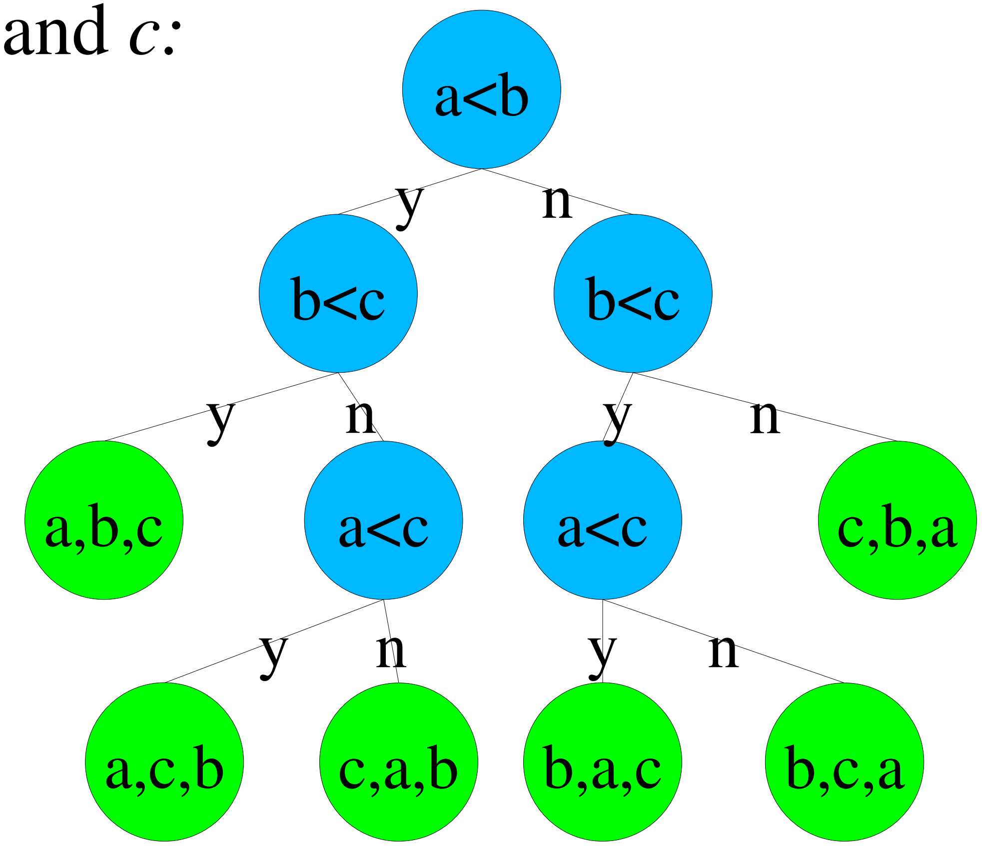
- Mergesort:
  - Comparisons: $W(n) = n \lg n$, $A(n) = n \lg n$.
  - Assignments: $T(n) = 2n \lg n$.
- Quicksort:
  - Comparisons: $W(n) = n^2/2$, $A(n) = 1.38\, n \lg n$.
  - Assignments: $T(n) = .69n \lg n$.
- Heapsort:
  - Comparisons: $W(n) = 2n \lg n$, $A(n) = 2n \lg n$.
  - Assignments: $W(n) = n \lg n$, $A(n) = n\, lg\, n$.
- (these are approximations.)

Complexity of the Sorting Problem

- All of these algorithms make fewer than $\Theta(n^2)$ comparisons.
  - They must be removing more than one inversion per comparison.
  - Our one-inversion-per-comparison bound does not apply in general.
- Can we get a tighter bound on the complexity of the sort problem?
- The key is decision trees...

Decision Trees

- For every sort algorithm there is a corresponding decision tree.
- Here is a decision tree for *some* sort applied to three items *a*, *b* and *c:*



Decision Trees

- For every deterministic algorithm that sorts *n* distinct keys there is a corresponding binary decision trees with exactly *n*! leaves.
  - There are *n*! different arrangements of *n* keys.
  - Any tree with fewer leaves would necessarily fail to sort some arrangement.
  - The tree is binary because our comparisons only tell us if one item is less than another. (Remember that we are thinking about *n distinct* items.)
- This allows us to find the lower bound on the complexity of the sort problem:
  - What is the minimum depth for a binary tree with *n*! leaves?

Binary Tree Depth

- Binary tree of depth d, can have no more than 2^d leaves.
  - (Go through the induction proof?)
- So we have $n! <= 2^d$, and we want to solve for d.
  - I.e. if a tree has $n!$ leaves how deep must it be?
- Take the lg of both sides:
  - $\lg(n!) \leq d$
  - Taking the log of $n!$ requires a little calculus:

$$\lg(n!) = \lg[n(n-1)(n-2)...(2)(1)] = \sum_{i=2}^{n} \lg i$$

$$\sum_{i=2}^{n} \lg i \geq \int_{1}^{n} \lg x \, dx = \frac{1}{\ln 2}((n \ln n) - n + 1) \geq n \lg n - 1.45\mathrm{n}$$

Lower Bound

- Any deterministic sort must make at least $\lceil n \lg n - 1.45\text{n} \rceil$ comparisons in the worst case.
  - The complexity of the sorting problem is in $\Omega(n \lg n)$.
- Recall that worst case performance of mergesort is $n \lg n - (n - 1)$.
- Additional DQs:
  - Can we imagine an algorithm guaranteed to match the lower bound?
  - What is the lower bound on the number of assignments?

Average Case

- We can also use a tree argument to find a lower bound for the average case.
- No longer interested in the minimum depth for a binary tree with n! leaves.
- Interested in the minimum average depth for such a tree.
  - Book defines external path length (EPL) to be the sum of the distances from the root to the leaves.
  - Average comparisons for a given decision tree is: $\frac{\mathrm{EPL}(n!)}{n!}$.
  - minEPL(m) is the minimum EPL for a binary tree with m leaves.
  - Lower bound on average case sorting is $\frac{\mathrm{minEPL}(n!)}{n!}$.

Average Case

- Computing $\frac{\text{minEPL}(n!)}{n!}$ is a bit of a slog. We won't do it in class.
- The final result is: $\lfloor n \lg n - 1.45\text{n} \rfloor$.
- At best one comparison fewer than the worst case.