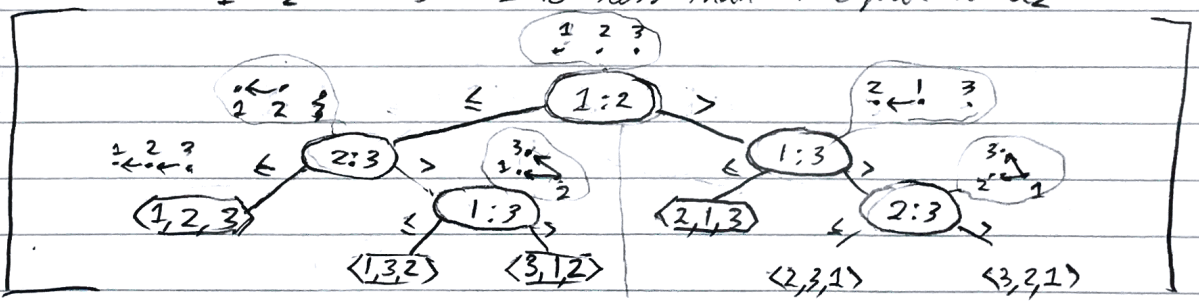
CS263 Wednesday Lecture: Problem Complexity — Mar 28, 2018

- Let P be some problem....
- ALGORITHM COMPLEXITY - given a specific algorithm A that solves P, what is the cost of running A?
- PROBLEM COMPLEXITY - what is the cost of solving P?
  - ↳ "using the best possible algorithm"

Ex
- P = sorting n integers
  - HEAPSORT, MERGESORT solve P in $O(n \log n)$ key comparisons
  - ⊛ Can we do better?... Any comparison-based algorithm that solves P takes $\Omega(n \log n)$ key comparisons
  - Then the above algorithms are optimal.

- Today, we will prove that ⊛ is true.
- Use decision-trees to model behaviour of algorithms

Ex. Insertion sort, 3 elements. // Recall: insertion sort algo.
- Notation: (1:2) means compare a_1 with a_2.
  - $1 \leftarrow 2$ means a_1 is less than or equal to a_2



- Any comparison-based algorithm can be modelled with a decision tree
- Max # comparisons = worst case run time = height of tree
- 6 leafs = $3C3 = 3!$; n permutations of input → n ways to unscramble
  - → Any comparison based algo on input of size n requires n! leafs
  - → to have n! leafs, tree must have height $\log_2 n!$
  - → ∴ worst-case # of comparisons any comp-based algo = $\log_2 n!$

Theorem: Every comparison-based algorithm A to sort n integers takes $\Omega(n \log n)$ key comparisons in the worst case.

Proof:

- Let A be any comparison-based algorithm to sort n integers.
- Let T_A be the decision tree that models A.

① for each input permutation of integers $1, 2, \ldots, n$, T_A must have at least one distinct leaf that sorts this permutation

↳ ∴ [A] $|\text{leaves}(T_A)| \geq n!$

② Let h be the height of T_A. Since T_A is a binary tree, we

↳ ∴ [B] $|\text{leaves}(T_A)| \leq 2^h$

- Together: $n! \leq |\text{leaves}(T_A)| \leq 2^h \rightarrow n! \leq 2^h \equiv 2^h \geq n!$

  $\therefore \log_2(2^h) \geq \log_2(n!) \therefore h \geq \log_2(n!)$
- Note: $\log_2(n!) \in \Theta(n \log n)$; see Eq 3.19 Pg. 58 CLRS
- Then we know the lower bound on the worst case of any comparison-based algo (height of tree) is at least $\in \Theta(n \log n)$.

↳ From this we know heapsort, mergesort are optimal