Decision Trees

- Read section 11.2 (pages 394-399)
- What is the **decision tree** for an algorithm?
- What can we say about the number of leafs for such a decision tree?
- What is the relation that must hold between the number of leaves in a tree and the height of the tree?
  - What is the importance of the relation in terms of a lower bound for the corresponding algorithm?
- For the problem of sorting an array of values:
  - In terms of thinking of a decision tree for such an algorithm, the possible outcomes are all permutations of the array elements. How many such permutations are there?
  - A formula due to Stirling (page 477) gives us an estimate for $n!$:

    $$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

    How does that relate to the lower estimate for the number of comparisons any sorting algorithm must perform?
  - Is the resulting lower bound a tight bound?
  - How can we find the average-case efficiency of a comparison-based algorithm using the decision tree?
- For the problem of searching for a key in a sorted array:
  - How does the decision tree for this problem differ from the previous one?
  - At least how many different leaves must such a decision tree have?
  - Can this tree ever be a full tree?
  - Why is it reasonable to incorporate the equals case into the node itself rather than a child-node? What do the leaves of this new tree represent?
- Practice problems: 11.2.2, 11.2.3, 11.2.4, 11.2.5