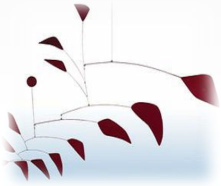
EECS 3101 - Design and Analysis of Algorithms



Shahin Kamali

Tutorial 5

York University

Picture is from the cover of the textbook CLRS.

Binary Search

Question

Recall that in a sorted array of n comparable items, we can use ***binary search*** *to search for a given item in $O(\log n)$. Prove that binary search is the optimal searching algorithm in a sorted array. You need to use a decision tree approach to show that no algorithm can search in a sorted array in time less than $O(\log n)$.*

Search in an Unsorted Array

Question

*We say an array of numbers is **almost-sorted** if at least half of elements appear in their right positions in the sorted array.*

- *For example, array $A = \{2, 1, 3, 4, 6, 5, 7, 8\}$ is almost-sorted because $3, 4, 7$, and 8 are in their correct position.*

*Provide a **tight lower bound** for sorting any almost sorted array of n numbers using a comparison-based sorting. A complete answer, includes an algorithm whose running time is asymptotically equal to your lower bound.*

Longest Palindromic Subsequence

Question

Given a string S, we want to find the longest subsequences of S that is also a palindrome.

- *For example, when $S = ABBDCAB$, the longest palindromic subsequence (LPS) of S is $ABBA$.*

Devise a dynamic programming algorithm to find LPS of S.

1. **Step 1:** define subproblems, and devise the value of the optimal solution for each subproblem using the value of the optimal solutions for smaller subproblems.
2. **Step 2:** write down a recursive formula for the value of optimal solutions.
3. **Step 3:** fill up the dynamic programming table recursively.
4. **Step 4:** retrieve the actual LPS by moving backwards in the table.