Design and Analysis of Algorithms

Decision Trees

Reetinder Sidhu

Department of Computer Science and Engineering

Design and Analysis of Algorithms

Decision Trees

Reetinder Sidhu

Department of Computer Science and Engineering

PES UNIVERSITY ONLINE

- Dynamic Programming
  - Computing a Binomial Coefficient
  - The Knapsack Problem
  - Memory Functions
  - Warshall's and Floyd's Algorithms
  - Optimal Binary Search Trees
- Limitations of Algorithmic Power
  - Lower-Bound Arguments
  - **Decision Trees**
  - P, NP, and NP-Complete, NP-Hard Problems
- Coping with the Limitations
  - Backtracking
  - Branch-and-Bound

Concepts covered

- Decision Trees
  - Smallest of three numbers
  - Sorting
  - Searching

Problem Types: Optimization and Decision

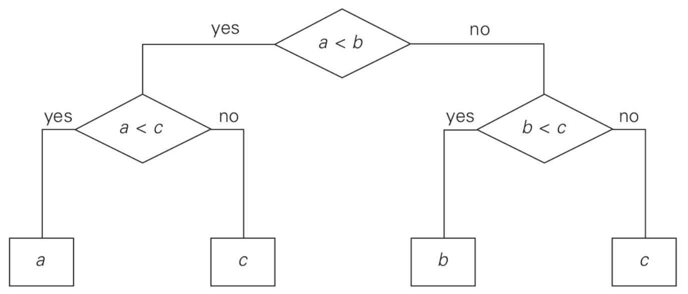
- Optimization problem: find a solution that maximizes or minimizes some objective function
- Decision problem: answer yes/no to a question
- Many problems have decision and optimization versions
  - Ex: traveling salesman problem
  - optimization: find Hamiltonian cycle of minimum length
  - decision: find Hamiltonian cycle of length m
- Decision problems are more convenient for formal investigation of their complexity

Introduction

- Many important algorithms, especially those for sorting and searching, work by comparing items of their inputs
- We can study the performance of such algorithms with a device called the decision tree

Example: Decision tree for minimum of three numbers

Decision tree for a determining the minimum of three numbers



Central Idea

- The central idea behind this model lies in the observation that a tree with a given number of leaves, which is dictated by the number of possible outcomes, has to be tall enough to have that many leaves
- Specifically, it is not difficult to prove that for any binary tree with leaves and height h

$$h \geq \lceil \log_2 l \rceil$$

- A binary tree of height h with the largest number of leaves has all its leaves on the last level
- Hence, the largest number of leaves in such a tree is 2^h
- In other words, $2^h \geq l$ which implies $h \geq \lceil \log_2 l \rceil$

Decision Trees for Sorting Algorithms

PES UNIVERSITY ONLINE

- Most sorting algorithms are comparison-based, i.e., they work by comparing elements in a list to be sorted
- By studying properties of binary decision trees, for comparison-based sorting algorithms, we can derive important lower bounds on time efficiencies of such algorithms
- We can interpret an outcome of a sorting algorithm as finding a permutation of the element indices of an input list that puts the list's elements in ascending order
- For example, for the outcome $a < c < b$ obtained by sorting a list a, b, c
- The number of possible outcomes for sorting an arbitrary n-element list is equal to $n!$

Decision Trees for Sorting Algorithms

- The height of a binary decision tree for any comparison-based sorting algorithm and hence the worst -case number of comparisons made by such an algorithm cannot be less than

$$C_{worst}(n) \geq \lceil \log_2 n! \rceil$$

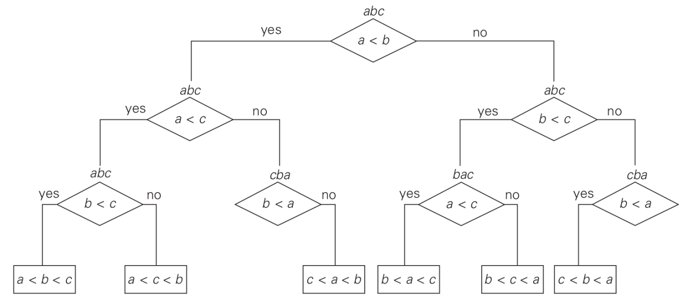
- Using Stirling's formula:

$$\lceil \log_2 n! \rceil \approx \log_2 \sqrt{2\pi n}\left(\frac{n}{e}\right)^n = n\log_2 n - n\log_2 e + \frac{\log_2 n}{2} + \frac{\log_2 \pi}{2} \approx n\log_2 n$$

- About $nlog_2 n$ comparisons are necessary to sort an arbitrary n-element list by any comparison-based sorting algorithm

Decision Trees for Sorting Algorithms

Decision Tree for Three Element Selection Sort



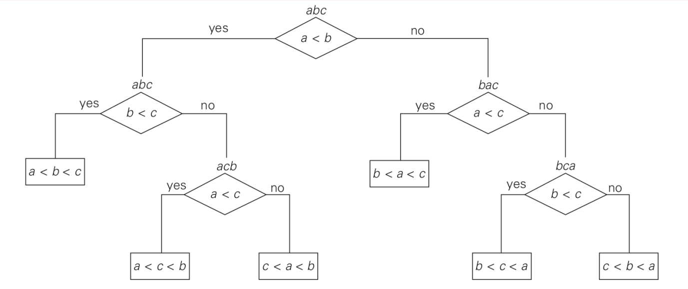
Decision Trees for Sorting Algorithms

- We can also use decision trees for analyzing the average-case behavior of a comparison based sorting algorithm
- We can compute the average number of comparisons for a particular algorithm as the average depth of its decision tree's leaves, i.e., as the average path length from the root to the leaves
- For example, for the three-element insertion sort this number is:

$$\frac{2+3+3+2+3+3}{6} = 2\frac{2}{3}$$

Decision Trees for Sorting Algorithms

Decision Tree for Three Element Insertion Sort



Decision Trees for Sorting Algorithms

- Under the standard assumption that all n! outcomes of sorting are equally likely, the following lower bound on the average number of comparisons Cavg made by any comparison-based algorithm in sorting an n-element list has been proved

$$C_{avg}(n) \geq \log_2 n!$$

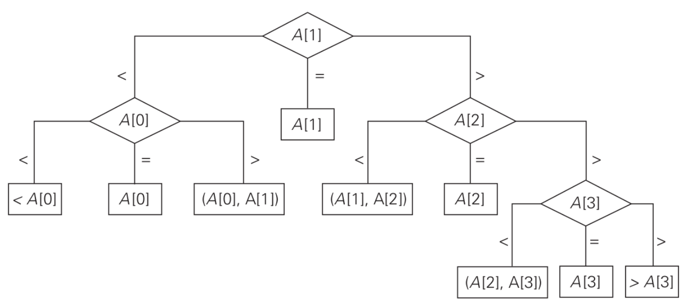
Decision Trees for Searching Algorithms

- Decision trees can be used for establishing lower bounds on the number of key comparisons in searching a sorted array of n keys: $A[O] < A[1] < \ldots < A[n-1]$
- The number of comparisons made by binary search in the worst case:

$$C_{worst}^{bs}(n) = \lfloor \log_2 n \rfloor + 1 = \lceil \log_2(n+1) \rceil$$

Decision Trees for Searching Algorithms

Ternary Decision Tree for Four Element Array Binary Search



Decision Trees for Searching Algorithms

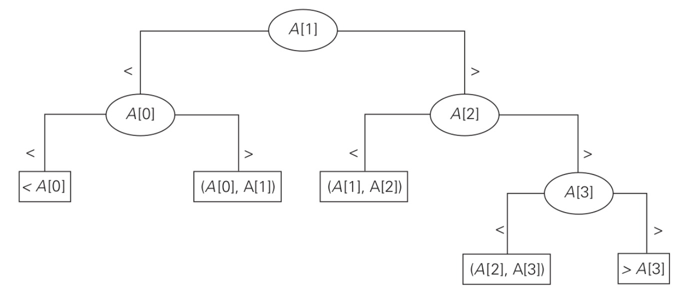
- For an array of n elements, all such decision trees will have 2n +1 leaves (n for successful searches and n + 1 for unsuccessful ones)
- Since the minimum height h of a ternary tree with l leaves is floor(log3l), we get the following lower bound on the number of worst-case comparisons:

$$C_{worst}(n) \geq \lceil \log_3(2n+1) \rceil$$

- This lower bound is smaller than $\lceil \log_2(n+1) \rceil$, the number of worst-case comparisons for binary search
- Can we prove a better lower bound, or is binary search far from being optimal?

Decision Trees for Searching Algorithms

Binary Decision Tree for Four Element Array Binary Search



- The binary decision tree is simply the ternary decision tree with all the middle subtrees eliminated

$$C_{worst}(n) \geq \lceil \log_2(n+1) \rceil$$