Lecture Summary: Tree Search Algorithms

Generated by LectureMate

February 22, 2025

1 Introduction

Today's lecture focused on tree search algorithms, using Haskell as a vehicle to discuss the fundamental concepts in computer science and artificial intelligence. The lecture began with a brief introduction to the C programming language, highlighting its low-level capabilities compared to Haskell.

2 Tree Structures

2.1 Binary Trees

The primary data structure discussed was binary trees, which are used to represent search spaces. Each node in a binary tree can have up to two children, and the trees discussed do not adhere to the typical binary search tree invariant. Instead, they are labeled with integers and used to represent various search scenarios, such as chess moves or flight connections.

2.2 Tree Representation in Haskell

Binary trees in Haskell can be represented using two constructors:

- `nil` for an empty tree
- `node` for a tree with a label and two subtrees

An example of a binary tree structure was provided, illustrating nodes with labels and their respective subtrees.

3 Search Algorithms

The lecture introduced two primary search algorithms: Depth First Search (DFS) and Breadth First Search (BFS).

3.1 Depth First Search (DFS)

DFS explores as far down a branch as possible before backtracking. The algorithm checks each node to see if it satisfies a given property, defined as a predicate function. The search proceeds as follows:

- Start at the root node.
- If the node satisfies the predicate, return it.
- If not, recursively search the left subtree.
- If the left subtree does not yield a result, search the right subtree.

DFS can be implemented using a simple recursive function in Haskell.

3.2 Breadth First Search (BFS)

BFS, in contrast, explores all nodes at the present depth before moving on to nodes at the next depth level. The algorithm proceeds in layers:

- Start at the root node.
- Explore all nodes at the current depth.
- Move to the next layer and repeat until the desired node is found.

BFS can also be implemented in Haskell, and it was noted that BFS is guaranteed to find a solution if one exists, while DFS may not terminate in certain cases, especially with infinite trees.

4 Comparison of Search Algorithms

The lecture highlighted the differences between DFS and BFS:

- DFS may run indefinitely on infinite trees, while BFS will always find a solution if it exists.
- DFS can be faster in cases where the solution is deep in the tree, while BFS may take longer in such scenarios.

5 Heuristic Search

The concept of heuristic search was introduced as a way to improve search efficiency by using additional information about the problem. This involves:

- Defining an evaluation function that assesses the potential of nodes.
- Using this function to prioritize which nodes to explore next, rather than following a strict left or layer-based approach.

The lecture discussed the implementation of a best-first search algorithm, which utilizes a priority queue to manage nodes based on their evaluation scores.

6 Conclusion

The lecture concluded with a discussion on the importance of search algorithms in computer science and artificial intelligence. The concepts of DFS, BFS, and heuristic search provide foundational knowledge for more advanced topics in AI and algorithm design.