Combinatorial Algorithms

Generated by LectureMate

February 22, 2025

1 Introduction

Good afternoon. Today's lecture focuses on combinatorial algorithms. Before diving into the content, a few announcements were made regarding the programming project released this morning, which is worth 20% of the course mark and is due in three weeks. The project involves implementing sequent calculus in Haskell, combining computation, logic, and functional programming.

2 Combinatorial Algorithms Overview

Combinatorial algorithms are used to solve problems that are often intractable or close to intractable. These problems may require enumerating possible solutions and checking them, a method known as *generate and test* algorithms.

2.1 Intractable Problems

Intractable problems often exhibit exponential complexity, such as 2^n or worse. Examples include:

- The satisfiability problem (SAT), which can be solved using the DPLL algorithm.
- Finding prime factors of large numbers, which underpins many cryptographic methods.
- The traveling salesman problem, where the goal is to find the shortest route visiting a set of cities.

3 Key Concepts

3.1 Combinatorial Explosion

The term *combinatorial explosion* refers to the rapid growth of the number of possible solutions as the problem size increases.

3.2 Subset Generation

To generate all sublists of a list, we note that the number of subsets of a set of size n is 2^n. The recursive approach to generate sublists involves:

- Generating all sublists of the tail of the list.
- Adding the head of the list to each of those sublists.

3.3 Distinct Elements

To check if a list contains distinct elements, we can use the function `nub`, which removes duplicates from a list. A list is distinct if it is equal to its `nub`.

4 Practical Applications

4.1 Generating Sublists

The function to generate sublists can be defined recursively. For a non-empty list, the function generates all sublists by combining the results of including or excluding the head of the list.

4.2 Permutations

The number of permutations of a list of length n is $n!$. To generate permutations, we can use:

- The Cartesian product of the list with itself.
- A recursive approach that splits the list and generates permutations of the remaining elements.

4.3 Choosing k Elements

Choosing k elements from a list of length n is represented by the binomial coefficient, calculated as:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

5 Conclusion

In conclusion, combinatorial algorithms are essential for solving complex problems in computer science. The lecture covered various aspects, including generating sublists, permutations, and the significance of distinct elements. Students are encouraged to review the programming project and prepare questions for the next lecture.