Algorithms and Data Structures - Assignment 2

Olav Witvliet & Berend Kerkvliet

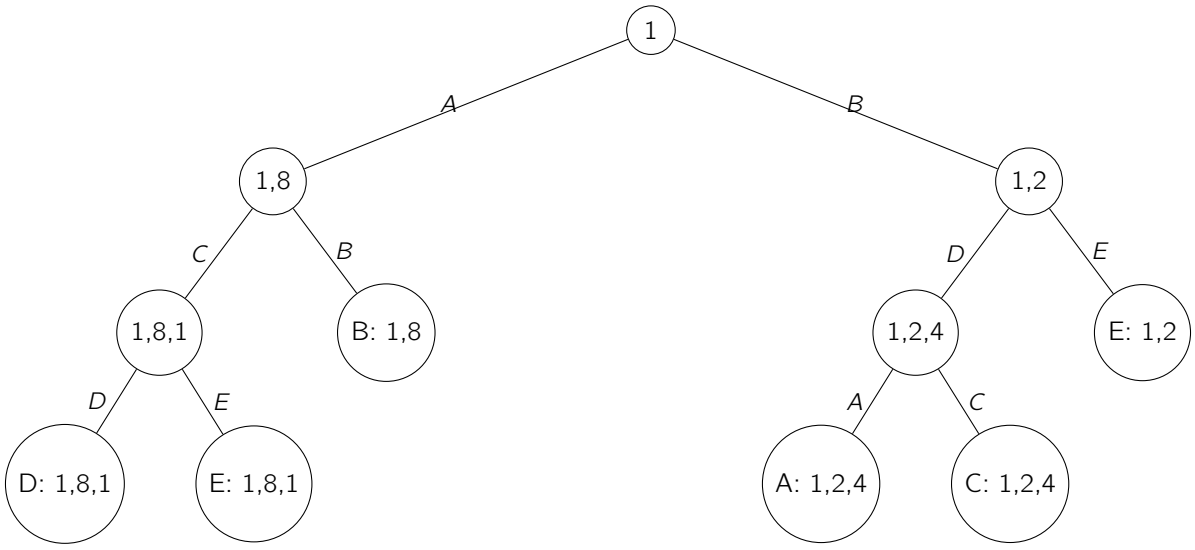
Introduction

The problem we need to solve involves finding the location label of a package based on its code in a 2D rectangular grid. In this grid, package codes increase from left to right in every row and from top to bottom in every column. LogTech ground personnel have visited several locations and left a package at each one. These packages have unique codes represented as integers. Our goal is to design an efficient divide and conquer algorithm to search for a given package code and return the corresponding location label.

To solve this problem, we've come up with a divide and conquer solution that takes advantage of the grid's properties. Our approach involves dividing the grid into smaller subrectangles, allowing us to focus on the correct area to search at each step. This significantly reduces the number of cells we need to examine. During each recursive step, we identify the middle cell of the current subrectangle and compare its value to the target package code. Depending on the comparison result, we can determine which quadrant (or quadrants) of the current subrectangle the target value might be in. We then recursively call the search function on the identified quadrant(s) until we find the target value or exhaust all possible search areas.

The divide and conquer approach is more efficient than a linear scan over all cells because it uses the grid's properties to eliminate large portions of the search space at each step. A linear scan, on the other hand, would require examining each cell in the grid one by one, resulting in a time complexity of O(n*m), where n and m are the dimensions of the grid. In contrast, the divide and conquer approach has a much lower time complexity, enabling us to find the desired package code more quickly, especially in larger grids.

Assignment specific question 1



In this example, we have a search tree where each node represents a location and the edges represent the packages found at each location. The labels on the edges indicate which package can be found by traversing that edge. To find the package, we can explore all possible paths (depth-first) in the search tree until we find the package.
This approach is guaranteed to find the package if it exists (thus, it is optimal), but it may not be the most efficient approach in terms of time and space complexity. There may be more optimized search algorithms that can find the package faster or with less memory usage.

Assignment specific question 2

This might work well in cases where the locations are evenly distributed and distances between them are similar, but not in cases where locations are clustered or there are obstacles. Greedy approaches are not always optimal since they may overlook the long-term consequences of decisions and become stuck in local minima.

An adversarial test case of this is when the closest location has a low probability of containing the package while another farther away has a higher probability. In this case, the greedy approach wastes time on low-probability locations.

Therefore, while a greedy approach may be straightforward, it may not always be optimal and may fail in certain situations.

Assignment specific question 3

We need to do some experiments to compare two different methods for searching packages on grids of different sizes. We will experiment with creating several grids of different sizes (like 5x5, 10x10, 15x15, etc.) and randomly place a package on each grid. Then we will run two methods - naive linear scan and divide and conquer - on each grid. We will record the number of cells searched by each method.

We will repeat this experiment multiple times for each grid size for more accurate results. After we finish the experiments, we will calculate the average number of cells searched by each method for each grid size. We can then plot the average number of cells searched for each method as a function of the grid size.

By analyzing the results, we can see that the number of cells searched by each method varies with grid size. The divide and conquer approach is more efficient for searching for packages on rectangular grids of different sizes, as it scales better with larger grid sizes. However, the efficiency difference between the divide and conquer and linear scan approaches can depend on and are different on the various grids and locations of the package.

Summary and Discussion

The goal of the assignment was to create a program that can efficiently search for a package on a rectangular grid. Two approaches were considered: a naive linear scan and a divide and conquer approach.

The divide and conquer approach was found to be more efficient than the linear scan approach for searching for packages on rectangular grids of different sizes. The running time of the divide and conquer approach increases much slower than the linear scan approach as the grid size increases. This indicates that the divide and conquer approach scales better with larger grid sizes and is more efficient overall.

The optimality of the search algorithm was also considered. While a search algorithm that explores all possible paths in the search tree is guaranteed to find the package if it exists, it may not be the most efficient approach in terms of time and space complexity.

A greedy approach to searching for a package was also discussed. While a greedy approach may be a simple and intuitive way to search for a package, it is not guaranteed to be optimal and may fail in certain situations.

Overall, the assignment helped me to develop my understanding of divide and conquer algorithms and their implementation in Python, as well as the importance of considering efficiency and optimality when developing algorithms for real-world problems

Contributions

Olav Witvliet (2642964): General report writing, Conclusion Report, code for functions fill_coordinate_to _loc and fill _loc _grid and divconq_search

Berend Kerkvliet (Student number): Introduction Report, general report writing, Code for functions encode_message and decode_message and help with the final functions