AAA Assignment 1 - Report

Shagan Plaatjies

April 2023

Abstract

This experiment is intended to develop and analyze a brute-force approach to counting the number of right-side vertical adjacencies between orthogonal rectangles. I used a subdivision method to divide a container rectangle into sub-triangles until a minimum rectangle size of 400 was reached. The approach I implemented was to use a nested for loop to go through the dataset and find the right side adjacencies and its complexity was determined to be O($n2$). The results showed a trend in the shape of O($n2$), which confirmed my theoretical analysis/predictions.

1 Introduction

The purpose of this experiment is to develop and evaluate and timing a brute force approach to count the number of right-hand side vertical adjacencies between different orthogonal rectangles within a set. The goal was to show the trend in the shape of the results and the complexity of my approach, detailing my decision-making throughout the process.

2 Methodology

The data for this experiment was generated using a subdivision method. I used a modified version of the code provided to us by Ian Sanders, to divide a container rectangle into sub-triangles until a minimum rectangle size of roughly 300 was reached. My algorithm's input was a list of rectangles, I used the data structure instructed in the experiment brief, the rectangles were described by the x and y coordinates of their bottom-left and top-right corner. The data was generated linearly increasing the size of the containing rectangle, 50000 pixels, in 25 iterations. It took unreasonable time to complete a 100 iterations test I initially set out for, so I adjusted the sample size to 25 samples. I recognize this is a middle ground as it's enough data to show a clear exponential trend, my intention was to be rigorous in the evaluation and demonstrate the exponential nature of this naive implementation.

I used an implementation of a nested for loop to go through the dataset to find the adjacencies. The output was a list of adjacent, orthogonal rectangles

on the right-hand side of each rectangle in the dataset, respectively. I provided files for my timings, the rectangles dataset that I used, as well as the adjacencies list and some figures to illustrate the timings and show an illustrative example of the rectangle generation method in action. The complexity of this approach was determined to be $O(n^2)$, based on the shape of the graph.

3 Observation

This experiment showed a trend in the shape of $O(n^2)$, which confirmed my theoretical analysis. The step sizes used in the experiment were sufficient to illustrate the trend of the algorithm. Matplotlib was used to generate the plots and along with Numpy arrays, so as to avoid any potential array optimizations that might skew the results from other libraries. I implemented my solution in Python and did not implement any parallelism in my approach. However, there is ample opportunity to parallelize this naive linear approach as each subsequent step is independent of the previous step. I have provided the timings, adjacency list (the output), as well as the rectangles used to generate the dataset within a zip file.

Below is a snippet of the algorithm I used:

```
def bruteForceApproach(dataset):
    resultsList = []

    for i in range(len(dataset)):
        resultsTemp = []
        count = 0
        for j in range(len(dataset)):
            if (i!=j) and (dataset[i][3]==dataset[j][1])
                      and (dataset[i][4] > dataset[j][2]
                      and dataset[i][1] < dataset[j][4]):

                resultsTemp.append(j)
                resultsTemp.append(dataset[i][1])
                resultsTemp.append(dataset[j][2])
                resultsTemp.append(dataset[j][4])
                count +=1

        resultsTemp.insert(0, count)
        resultsTemp.insert(0, i)
        resultsList.append(resultsTemp)

    return resultsList
```

4 Discussion

We were asked to create a brute force or naive approach. I do recognize that this is not the most efficient method for finding the adjacencies but focused on demonstrating rigorous testing and evaluation of timings and other factors. This experiment will therefore provide a good control sample to compare and evaluate more efficient methods. I believe that with more time, I can develop a more complex, diverse graph generation module, although the one used in this implementation is sufficient for the purposes of this experiment. Given the random nature of the data generated, there is often overlap between 2 consecutive iterations, I've sorted the output data visualization based on the number of rectangles in the dataset to account for this.

5 Conclusion

In conclusion, I implemented and evaluated a naive approach to counting the number of right-side vertical adjacencies between orthogonal rectangles in this experiment. The implementation used a nested for loop to linearly compare each rectangle in the dataset and find the adjacencies. I evaluated the Big-O complexity to be $O(n^2)$ as the results show a trend in the shape of $O(n^2)$, which confirms my theoretical analysis.

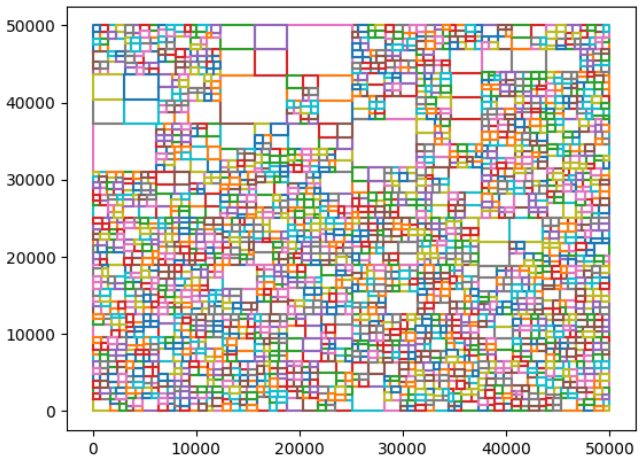

Figure 1: Visualization of subdivision approach

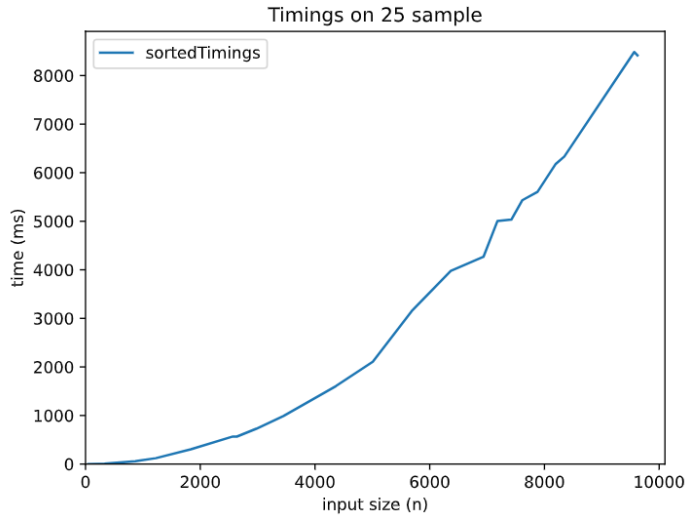

Figure 2: Timings for given input plotted against their sample sizes