Scientific Programming

Assignment: Quadtree.

Anand Kamble

amk23j@fsu.edu

9th November 2023

This project aims to implement a solution for grid evaluation using both the brute-force method and a QuadTree data structure. The specific grid evaluation functions are defined as

$$f_{ij} = f(x_i, y_j) = x_i^2 + y_j^2$$

$$f_{ij} = f(x_i, y_j) = x_i^2 - y_j^2$$

The objective is to efficiently check whether a given condition *C* satisfies the function for each cell in the grid.

Pseudo Code

In this program, we are first dividing the giving domain into N^2 cells. Using the recursive function divideNodes.

```
# Pseudo-code for divideNodes method


Method divideNodes(toDivide):
    # Check if any child node is null
    If arrayIncludesNullPointer(toDivide.children, 4):
        # If there is a null child, divide the current node
        Call toDivide.divide()
    Else:
        # If all children are non-null, recursively divide each child node
        For i from 0 to 3:
            Call this.divideNodes(toDivide.children[i])
```

After that, we are applying the conditions, using a class named ‘Condition’, and the evaluation functions are implemented using their own separate classes derived from the parent class ‘Condition’ using polymorphism.

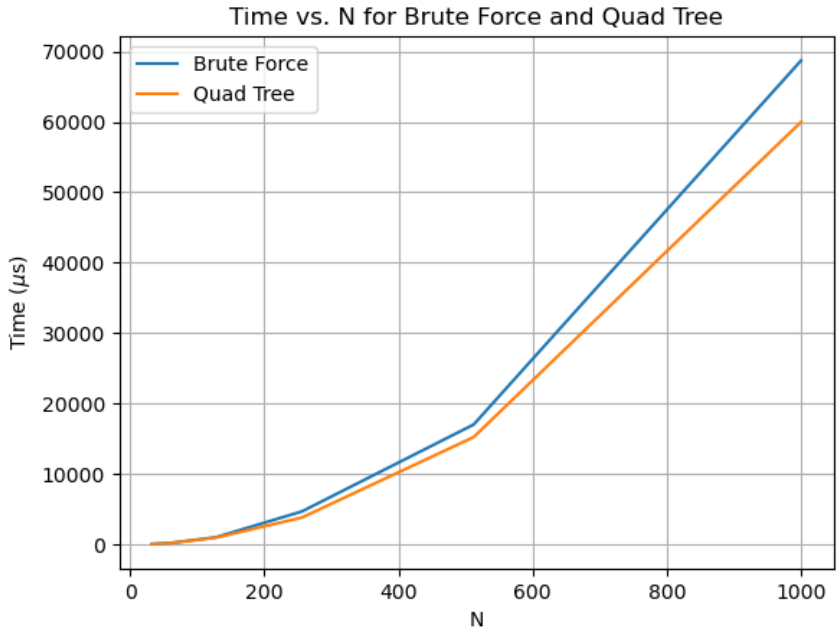
Pseudo code for the scan method is given below,

```
Method scan():

        If all elements in this.children are not null:

            For i from 0 to 3:

                Call this.children[i].scan()

        Else:

            Set condition_result to this.condition.check(this.c, this.points[0],
this.points[1], this.points[2], this.points[3])

            If condition_result is true:

                Set i to (this.points[0] - (-1)) / (this.points[1] -
this.points[0])

                Set j to (this.points[2] - (-1)) / (this.points[3] -
this.points[2])

                Append round(i) to this.i

                Append round(j) to this.j

            Return condition_result
```

Output

Time taken for different number of N for both the evaluation functions is printed to the console.

If we plot the time taken by each method to number of grid cells, we get something like following,



For the above graph, it can be seen that the time required for quad tree is less than the brute force method.

Execution

The Makefile is included with this code. You can run the command ‘make’ to compile the program. After successful compilation, you can find the executable named ‘a.out’ inside the bin folder. Run this executable by ‘./bin/a.out’. To clean the generated folders and files, use the command ‘make clean’.

Note

To print the output in a tabular format, please uncomment the functions ‘tree->printTree();’ and ‘bruteForce->print();’ in the main.cpp.

For example, after uncommenting ‘tree->printTree();’,the output of quad tree for function 2 and grid size 32 should be as follows,