UPPSALA
UNIVERSITET

THE BARNES-HUT METHOD

Oscar Hultmar]Johan Paulsson

1 Problem

The purpose of the program is to simulate a galaxy with a chosen amount of stars. To to this, Newtonian mechanics are used with a modified force, which is describes as,

$$\mathbf{F}_i = Gm_i \sum_{n=1, j\neq i}^{N-1} 2^{-n} = 1 \tag{1}$$

To make the calculations easier for a large amount of particles, the Barnes hut method is to be implemented for the calculations of the force. This is to make the program more efficient for larger amount of particles in the simulation.

2 Solution

The Barnes hut method inserts all particles into a quad tree by dividing all particles into quadrants. To do this, we created a node structure that contained four children nodes and information about the nodes position and the particles that in the node. Then, we implemented a recursive algorithm that created the four children and divided the particles from the node into the nodes four children based on position. If the children contains more than one particle, it continues to create children recursively until all nodes contain one or zero particles.

To calculate the force using this implementation, we used the Newtons law of gravity, but instead of calculating the force for the particles against each of the other particles, we can calculate the force for a particle against cluster of particles, by taking their combined mass and center of mass. These calculations will be done if the relation between the size of the node against the distance between the particle and the center of the node is larger than a specific number of our choosing, that we refer to as θ.

All code and optimization were distributed equally, to make sure that both participants understand all parts.

3 Performance and Optimizations

3.1 Reproduce

To measure the performance, the execution time for the complete code was measured using UNIX command time for N 5000 with input file *ellipse_N_05000.gal*. The code was executed on a single thread with the CPU model *intel core i7-8550u 1.8ghz 8M cache*. For Δt 10^{-5} was used and number of time steps was 100. When compiling the code, optimization O3 was used.

Using trail and error, we found that the optimal input theta for our code is approximately 0.252 and it resulted in an error of 0.000984 when running 200 time steps with input file *ellipse_N_02000.gal*.

3.2 Discussion

When comparing to the the code written with complexity $O(n^2)$, the Barnes hut method if in about the same slower time range for low n. When n increased, the Barnes-hut algorithm drastically improved performance. The complexity difference can be seen in figure 1.

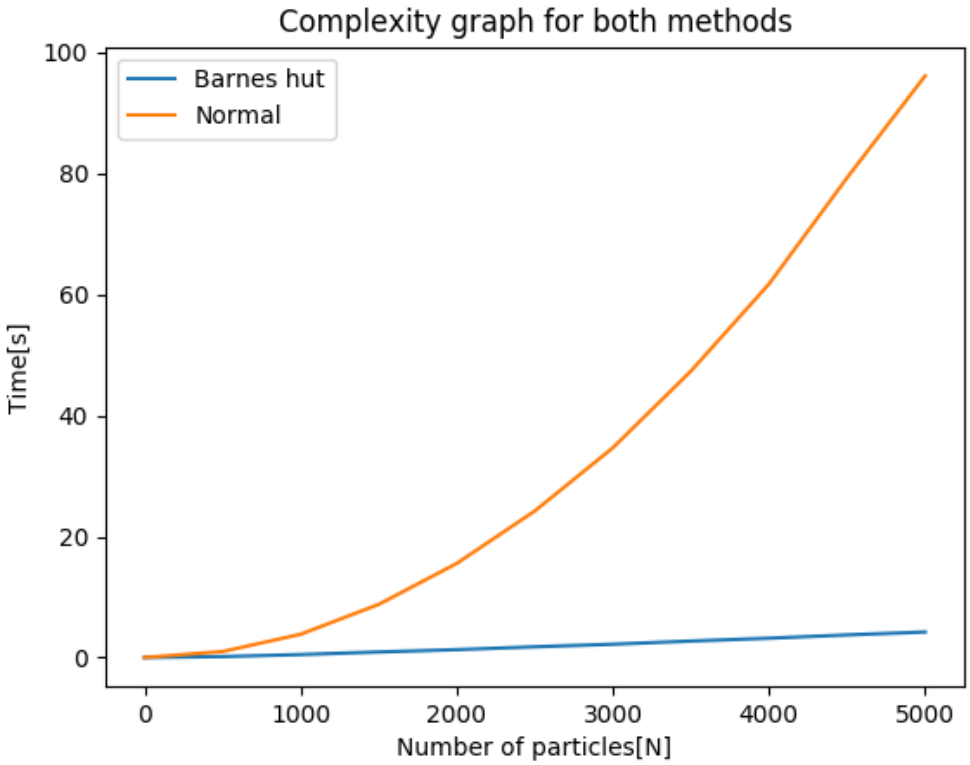

Figure 1: Plot showing the complexity vs time

We tried to implement other performance enhancers, but without any significance. When we tried to vectorize the problem, we could not finish the program due to the amount of time it took. We also tried to implement other optimizations such as unrolling for loops, changing constants to const, inlining functions and others, but if anything, most implementations made the code more time consuming. We believe that this is due to that the -O3 flag by itself take these optimizations to account. We also thought about changing our code so that it would only create nodes containing particles, so that the empty nodes would be neglected. This would lead to less comparisons in the force calculations and would possibly save a lot of time. Unfortunately, we did not find the time to finish this implementation.