Accelerating simulations by clustering bodies using the Barnes-Hut algorithm

Simulating forces such as gravity is a demanding task, because of the interactions every object has with all the other objects. With n objects, there are $n-1$ forces acting on each body, so all in all, there are $n \cdot (n-1)$ forces acting. The Barnes-Hut algorithm can be used to approximate the forces that need to be calculated by clustering the objects, sacrificing accuracy. In order to take those clusters into effect, the algorithm takes the size of the individual clusters and their distance to the respective object into account.

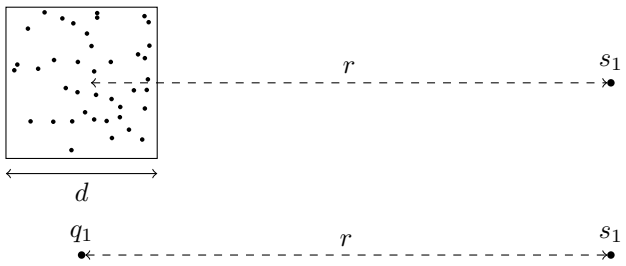


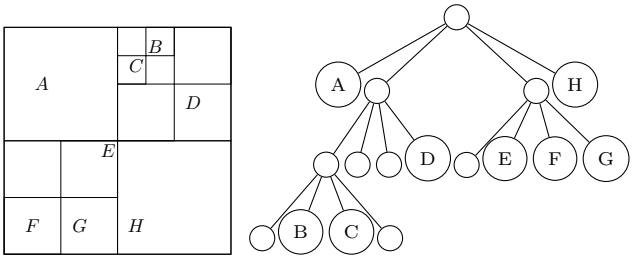
Figure 1: A cluster of stars that is far enough away from a single star can be abstracted as a single point in space.

$$\theta = \frac{d}{r} \tag{1}$$

The above equation describes how to cluster the objects. If a body (s_1) is far away from a small cluster ($r \gg d$), θ gets very small and the cluster in which the body is located can be abstracted to a single point. $0 \leq \theta \leq 1$ is provided by the user as a threshold impacting the accuracy and the speed of the simulation. Its value should be tuned in depending on the given data, as it decides which stars are approximated as a single cluster.

Everything is based on the stars being in a tree, so we need to subdivide the space into cells. Such a subdivision can be seen in Figure 2a and the process can be seen on the bottom of this page.

When calculating the forces affecting the object F in Figure 2a, the Barnes-Hut algorithm does not consider all objects indvidually, but only the ones that fall over the threshold θ. For the object F, this means that the Objects B and C are not calculated independently, but as a single object (a new abstract object is created in the center of gravity of B and C).



(a) Cell representation **(b)** Tree representation

Figure 2: Visual representations of the same Barnes-Hut tree. (`http://arborjs.org/docs/barnes-hut`)

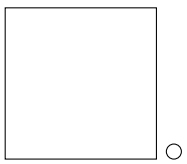
The tree in Figure 2b describes the cells from Figure 2a - top left, top right, bottom left and bottom right are depicted as a new layer in the tree accordingly. While building the tree, we are going to store the center of gravity and the total mass of each inner node. The complete process of simulating the force acting on a single star works in the following way:

We walk through the tree starting from the root in the direction of the leaves, using $\frac{d}{r} < \theta$ as the end condition. We use θ as a threshold for controlling how many forces to take into account ($0 \leq \theta \leq 1$). The force acting on a star is calculated when a leaf is reached or when an end-condition is met (thus resulting in no further recursion into the tree from that node on).

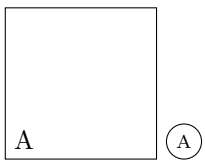
Experimenting with the value of θ on the dataset can optimize the runtime from $O(n^2)$ to as low as $O(n \cdot \log(n))$. This means that if we've got $2 \cdot 10^8$ bodies and can calculate the forces acting on 10^6 bodies per second, the total runtime is reduced from about 1200 Years down to 45 minutes optimally (the time to build the tree is an actual computational complexity ($\Theta(n \cdot \log(n))$), not a measured runtime and does not depend on θ).

This principle can also be applied to other types of problems such as simulating molecules. If you come to do something with it, don't mind writing to me!

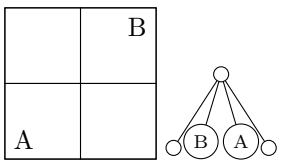
`@hanemile` on most platforms.



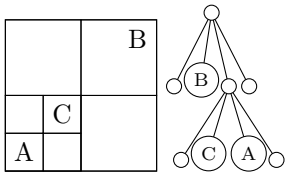
We start with an empty space



We insert the Star A



Inserting B: Subdivide, shift A, shift B from root



Inserting C: Subdivide, shift A, shift C from root