High-Performance Computing

12.2: Parallel Algorithms

Some Problems Are "Embarrassingly Parallel"

- Embarrassingly Parallel (C. Moler): A problem in which the same operation is applied to all elements (e.g., of a grid), with little or no communication among elements

- Example: in Matlab,

```
grid(rand(n) < probAnts)
```

  is inherently parallel in this way (and Matlab has begun to exploit this)

The Parallel Data Partition (Master/Slave) Approach

- Master process communicates with user and with slaves
  1. Partition data into *n* chunks and send each chunk to one of the *p* slave processes
  2. Receive partial answers from slaves
  3. Put partial answers together into single answer (grid, sum, etc.) and report it to user

The Parallel Data Partition (Master/Slave) Approach

- Slave processes communicate with master
  1. Receive one data chunk from master
  2. Run the algorithm (grid initialization, update, sum, etc.) on the chunk
  3. Send the result back to the master

Parallel Data Partition: Speedup

- Ignoring communication time, the theoretical speedup for *n* data items running on *p* slave processes is

$$\frac{n}{n/p + p}$$

- As *n* grows large, this value approaches *p* (homework problem): speedup is linear in # of slaves.

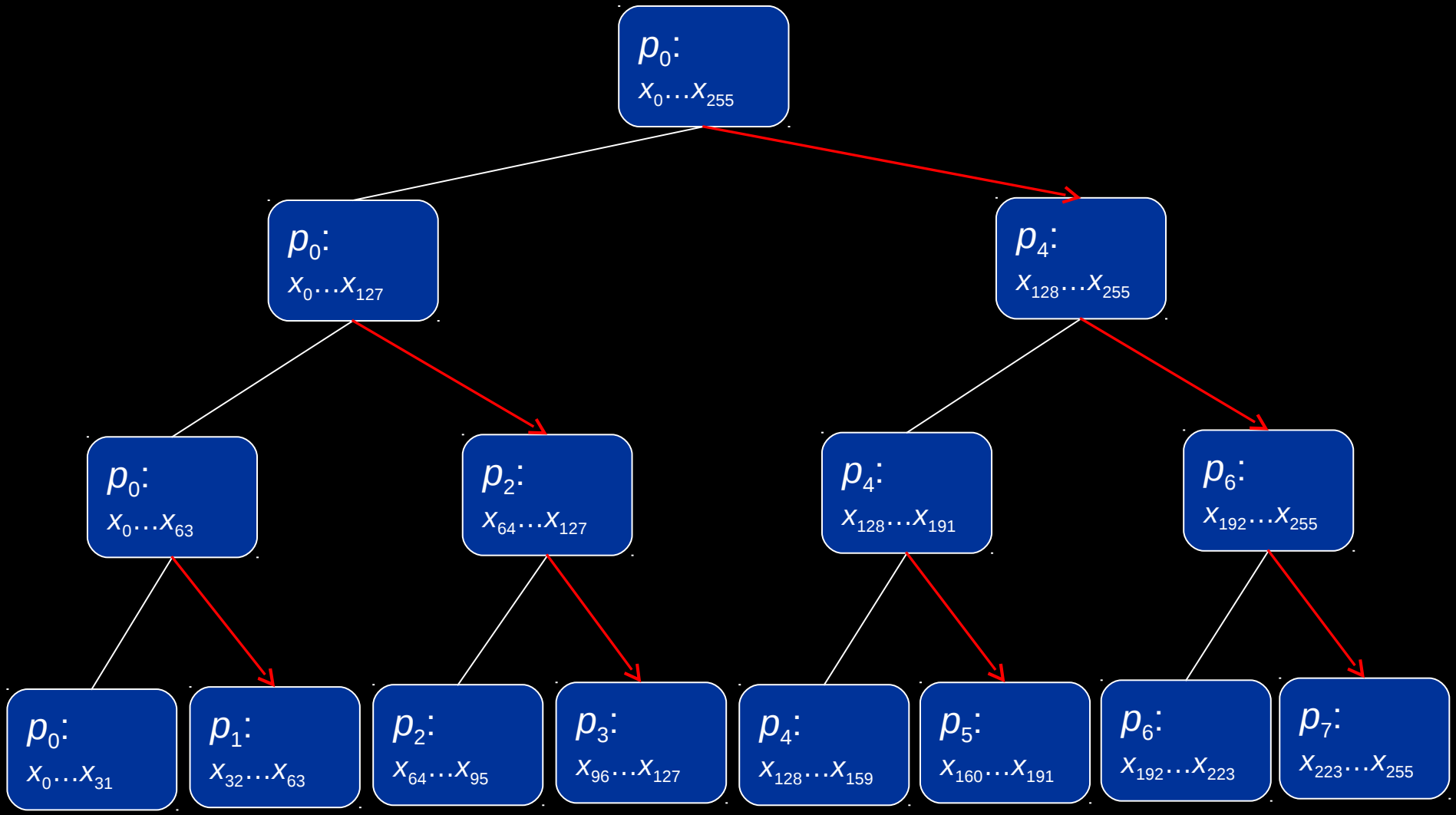
Simple Parallel Data Partition Is Inefficient

- **Communication time**: Master has to send and receive *p* messages out one at a time, like dealing a deck of cards.
- **Idle time**: Master and some slaves are sitting idle until all slaves have computed their results

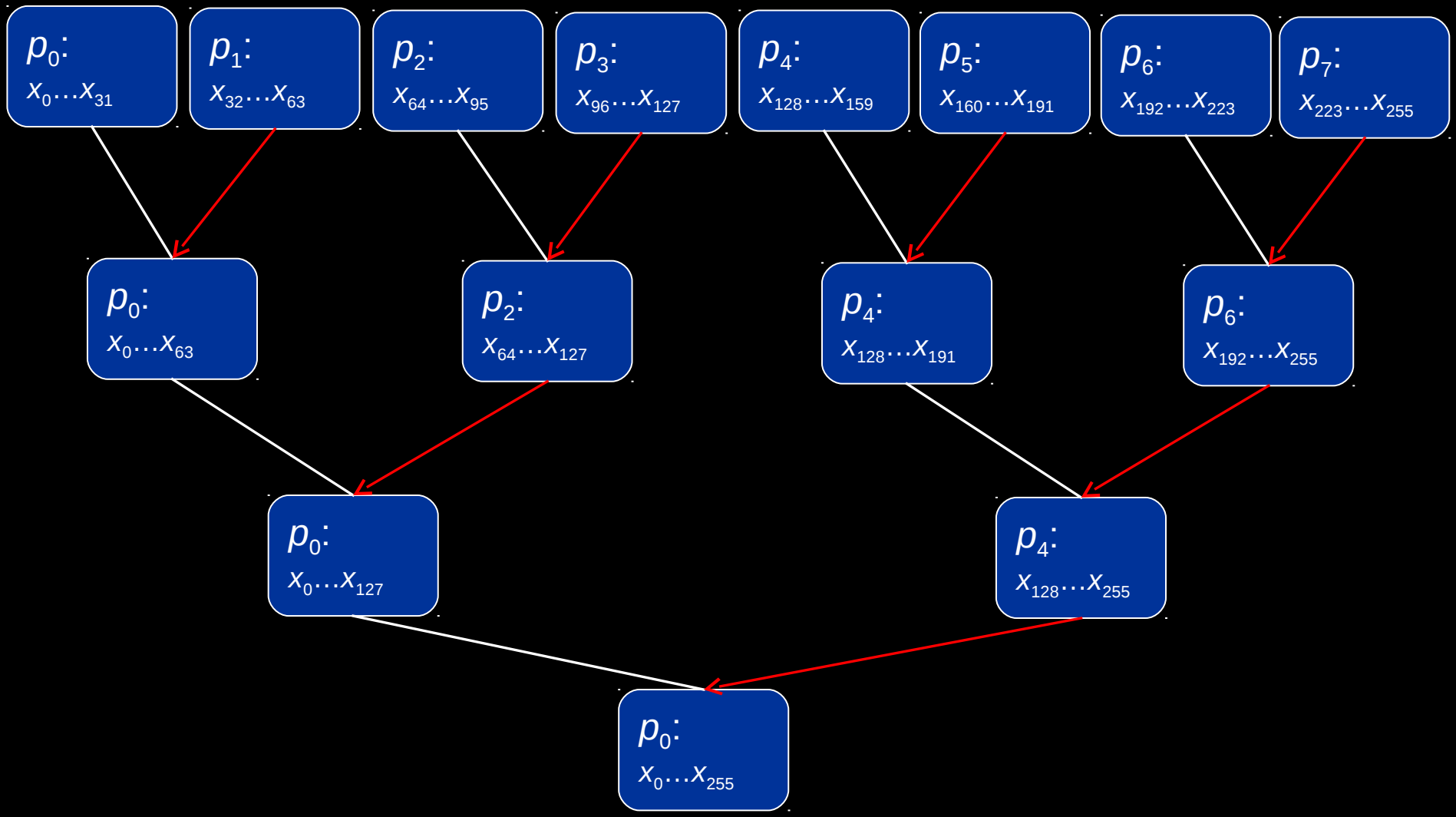
The Divide-and-Conquer Approach

- We can arrange things so that each process only sends/receives only two messages: less communication, less idling
- Think of dividing a pile of coins in half, then dividing each pile in half, till each pile has only one coin (or some fixed minimum # of coins)
- Example: summing 256 values with 8 processors ...

“Divide” Phase

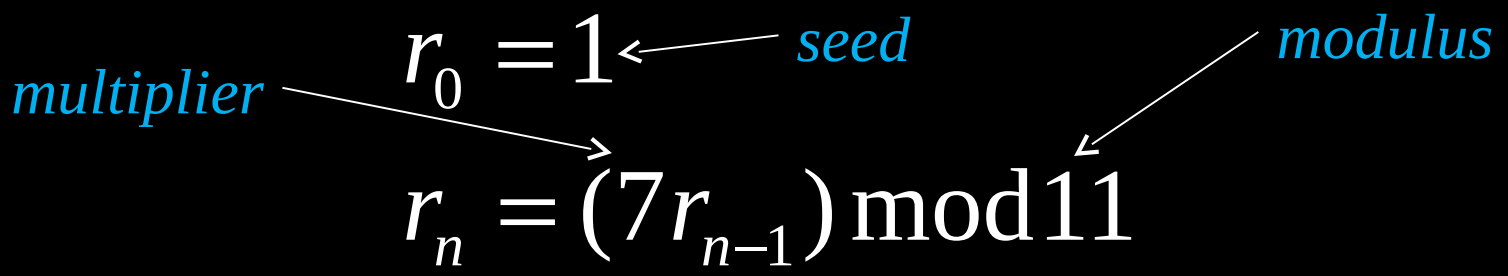


“Conquer” Phase



Parallel Random Number Generator

- Recall our linear congruential formula for generating random numbers; *e.g.*:



$r = 1, 7, 5, 2, 3, 10, 4, 6, 9, 8, \ldots$

- We'd like to have each of our *p* processors generate its share of numbers; then master can string them together

- Problem: each processor will produce the same sequence!

Parallel Random Number Generator

- "Interleaving" Trick: For *p* processors,

1. Generate the first *p* numbers in the sequence: *e.g.,* for *p* = 2, get 1, 7. These become the seeds for each processor.

2. To get the new multiplier, raise the old multiplier to *p* and mod the result with the modulus: e.g., for *p* = 2, $7^2 = 49$; 49 mod 11 = 5. This becomes the multiplier for all processors.

1. Master interleaves results from slaves.

Parallel Random Number Generator

p_0 :

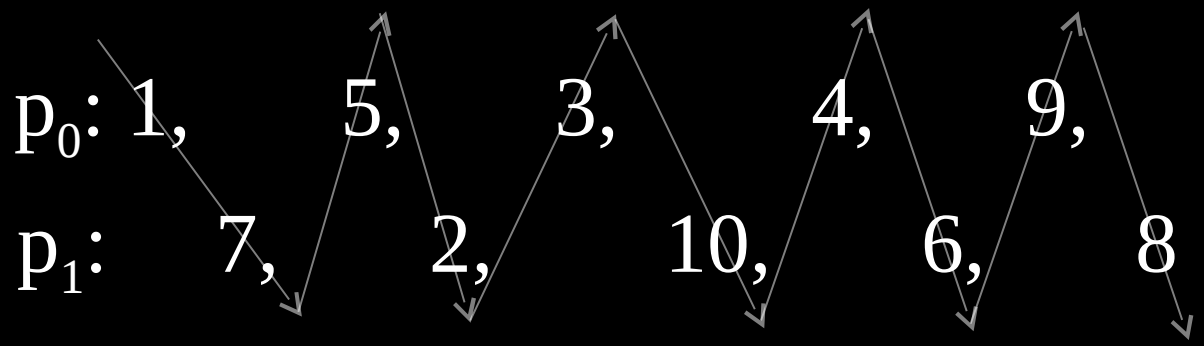
$r_0 = 1$

$r_n = (5r_{n-1}) \bmod 11$

p_1 :

$r_0 = 7$

$r_n = (5r_{n-1}) \bmod 11$



The *N*-Body Problem

- Consider a large number *N* of bodies interacting with each other through gravity (*e.g.* galaxy of stars)

- As time progresses, each body moves based on the gravitational forces acting on it from all the others:

$$F = \frac{G m_1 m_2}{r^2}$$

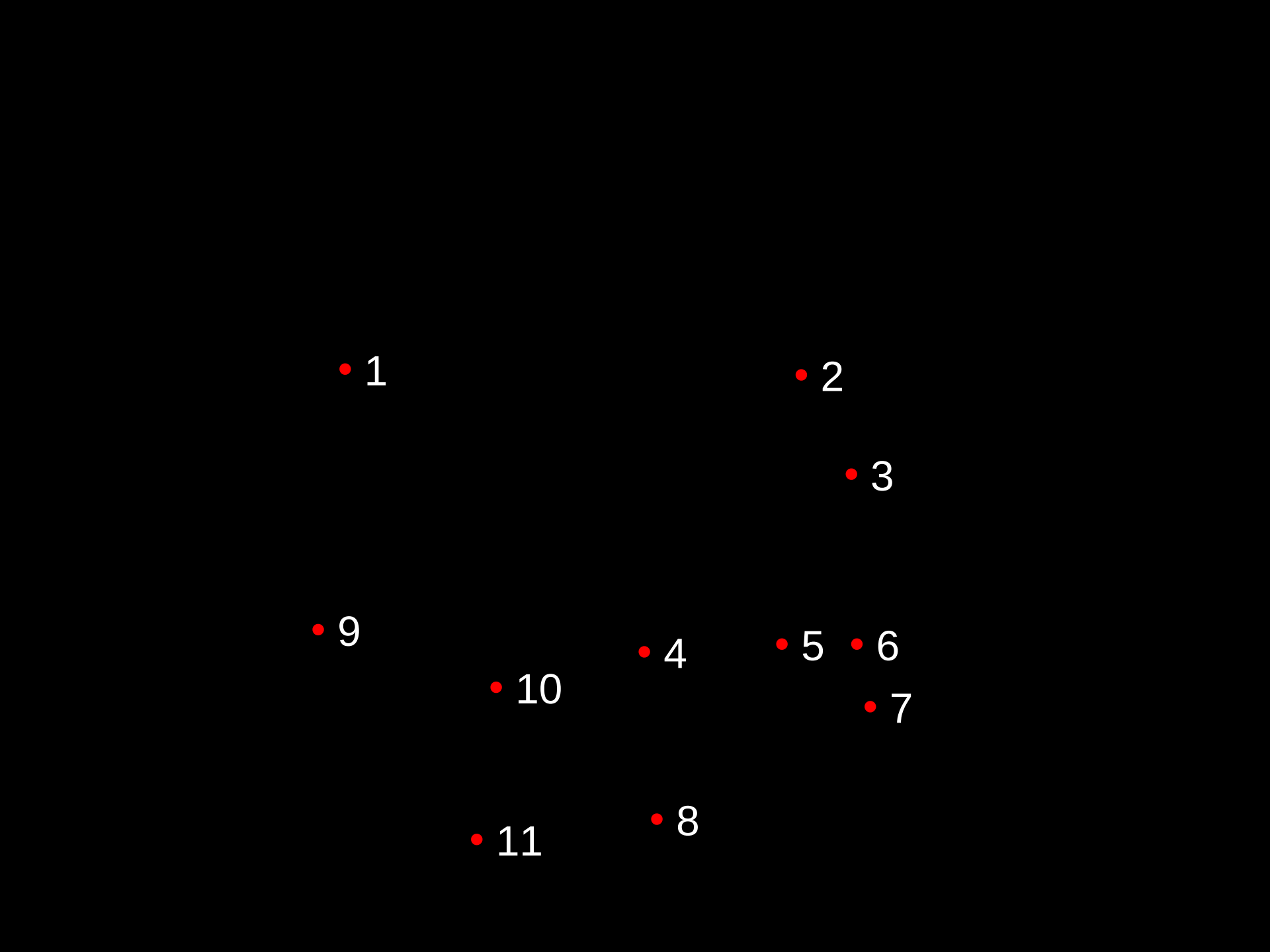
where m_1, m_2 are the masses of two objects, r is the distance between them, and G is Newton's Gravitational Constant.

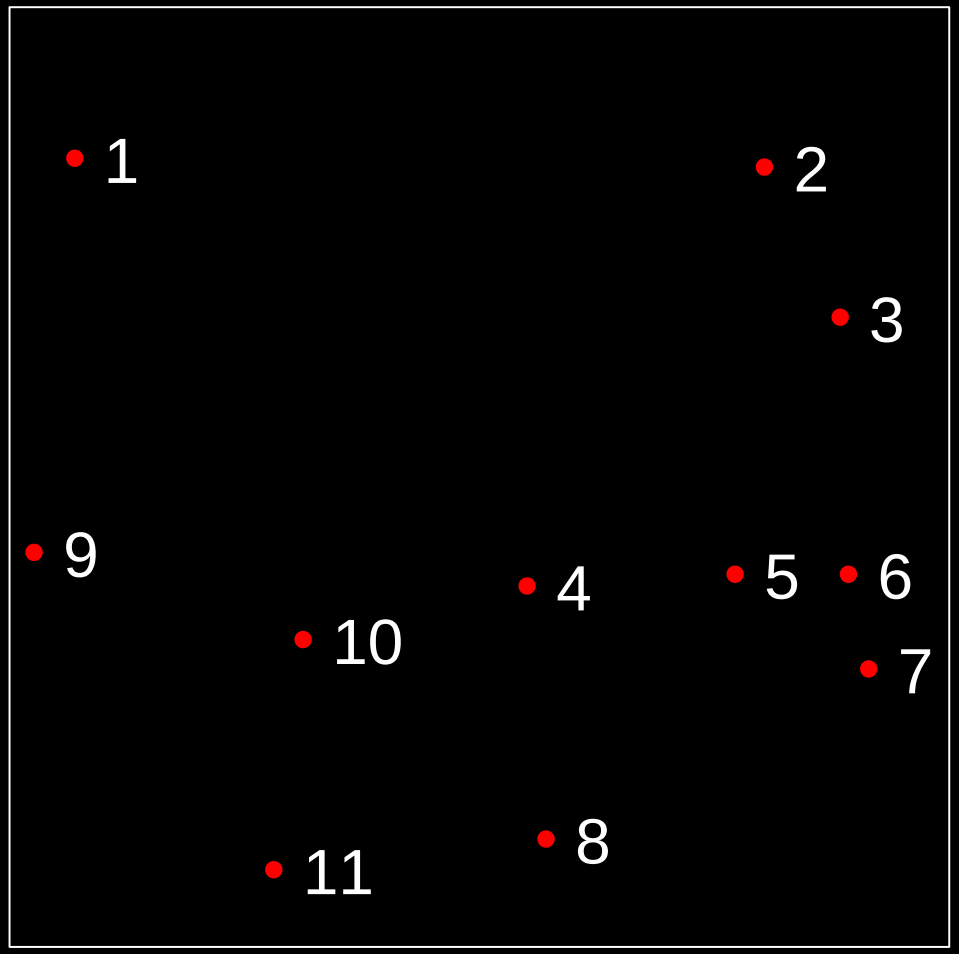
The *N*-Body Problem

- Problem: for each of the *N* bodies, we must compute the force between it and the other *N* -1 bodies.
- This is $N*(N-1)/2 = (N^2 - N)/2$ computations, which is proportional to N^2 as *N* grows large.
- Even with perfect parallelism, we still perform $1/p * N^2$ computations; *i.e.,* still proportional to N^2 .

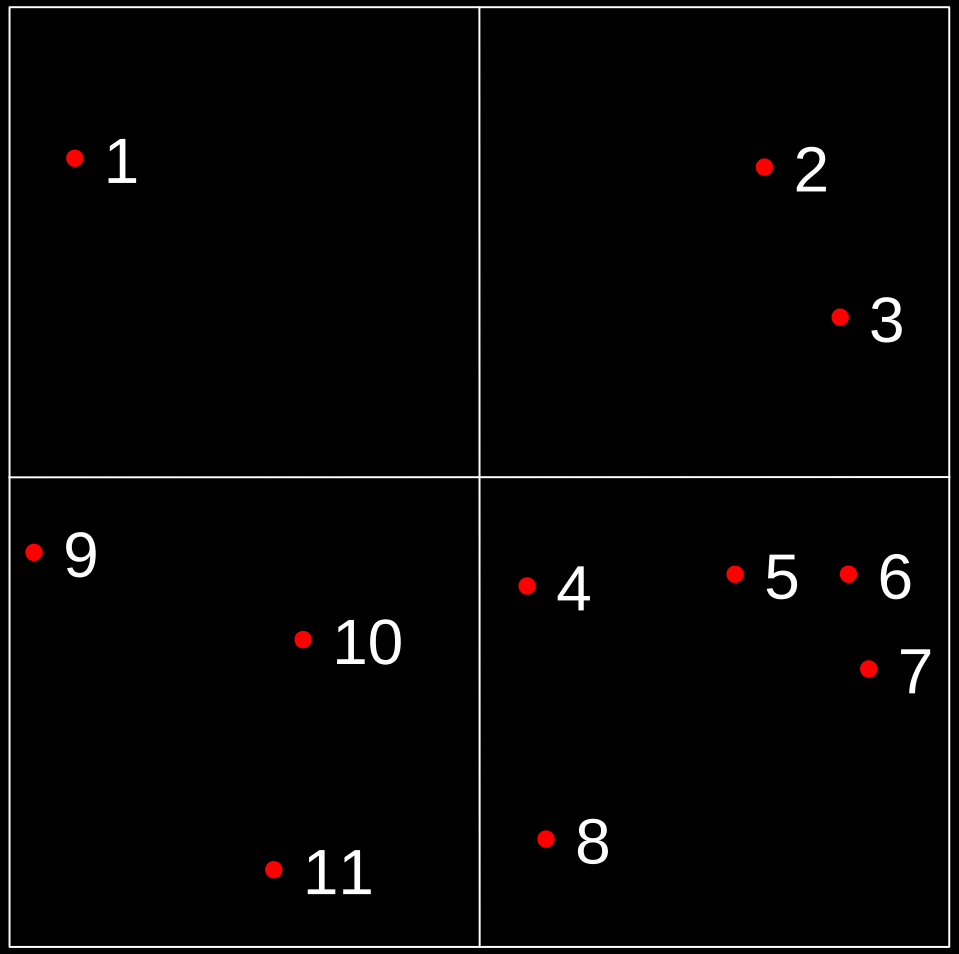
The *N*-Body Problem: Barnes-Hut Solution

- Division by r^2 means that bodies distant from each other have relatively low mutual force.
- So we can focus on small clusters of stars for the formula, and then treat each cluster as a single body acting on other clusters
- This is another instance of divide-and-conquer.
- We will treat space as 2D for illustration purposes.

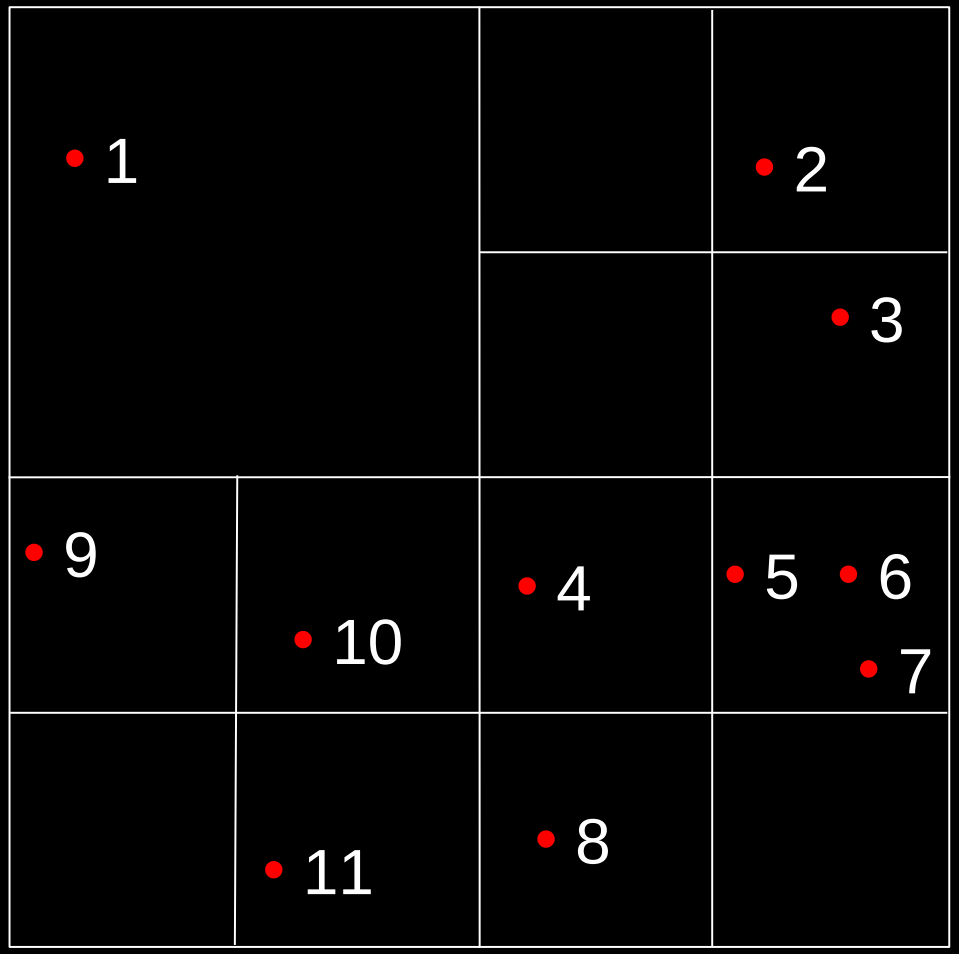

1
2
3
9
4
5
6
10
7
11
8


1
2
3
9
4
5
6
10
7
8
11

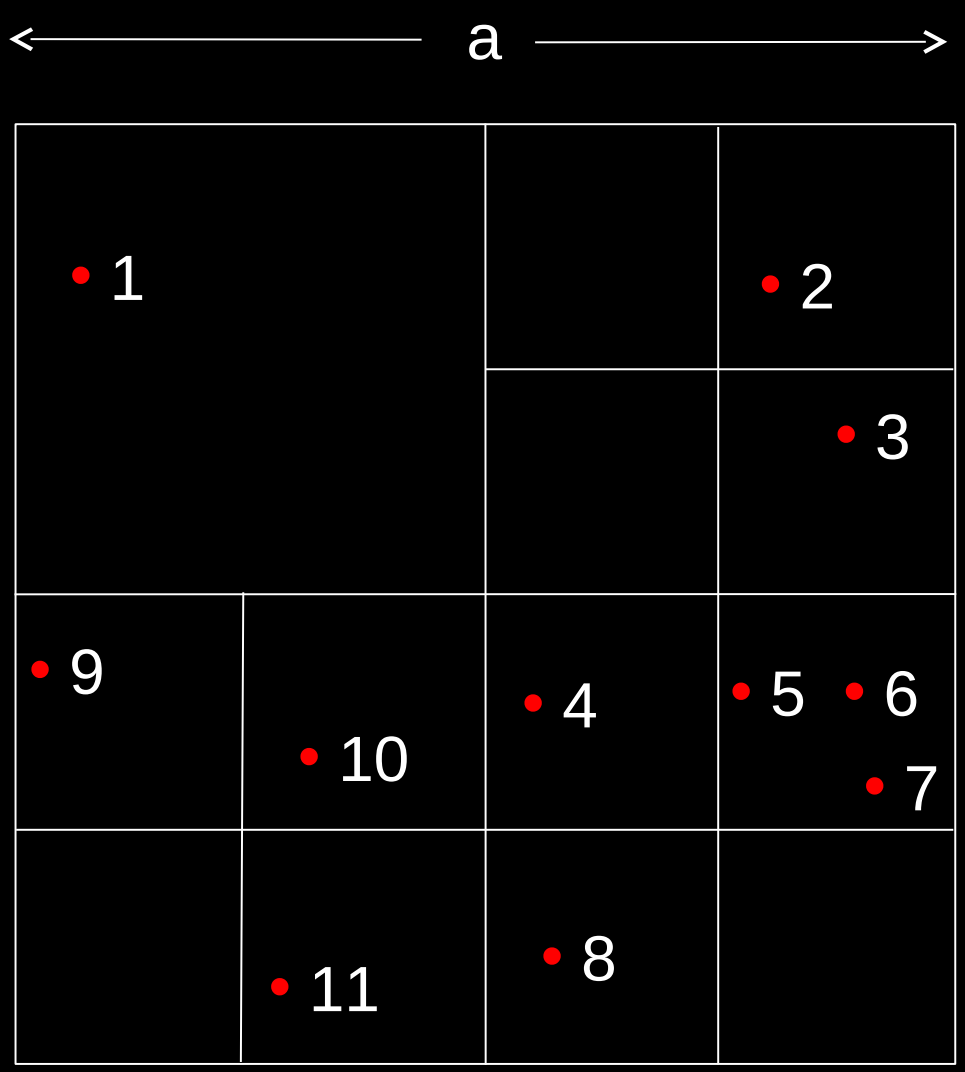
Divide



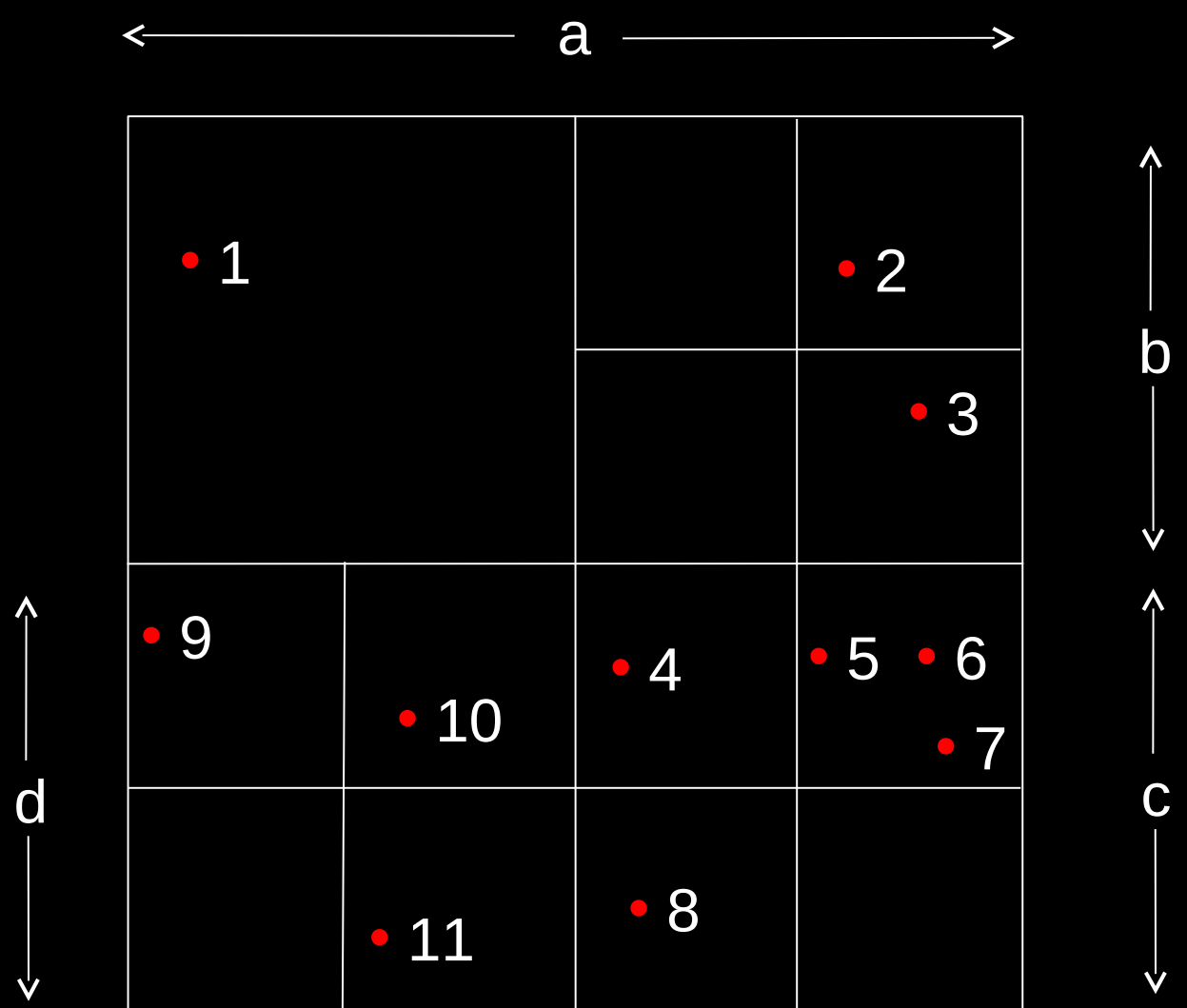
Divide



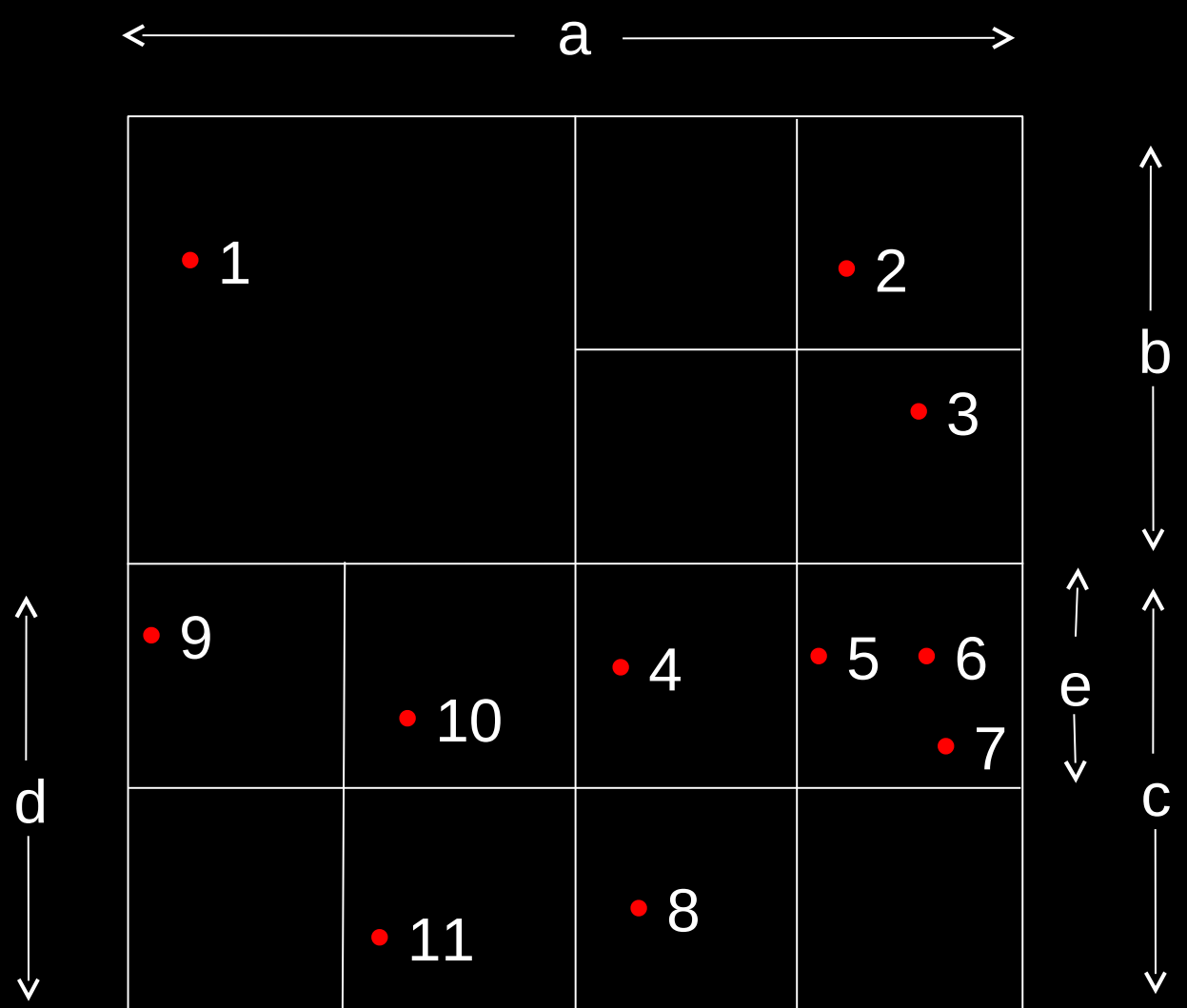
Build Labelled Hierarchy of Clusters



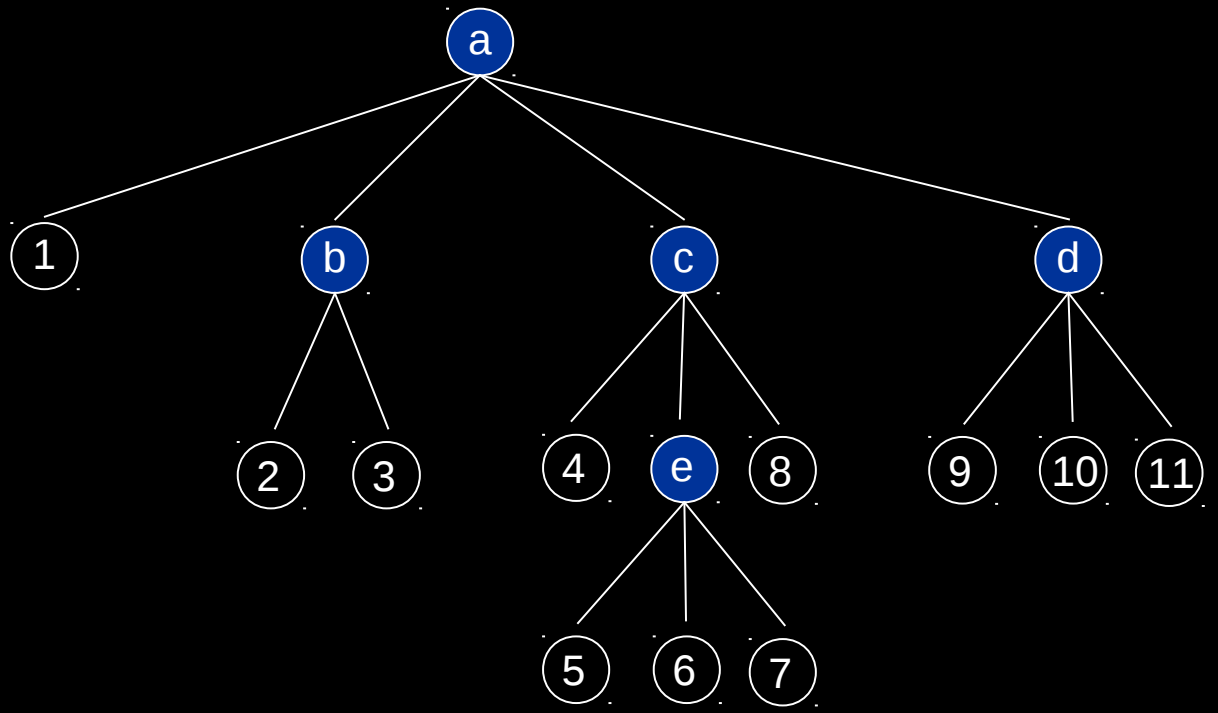
Build Labelled Hierarchy of Clusters



Build Labelled Hierarchy of Clusters



Produces a Quad-Tree



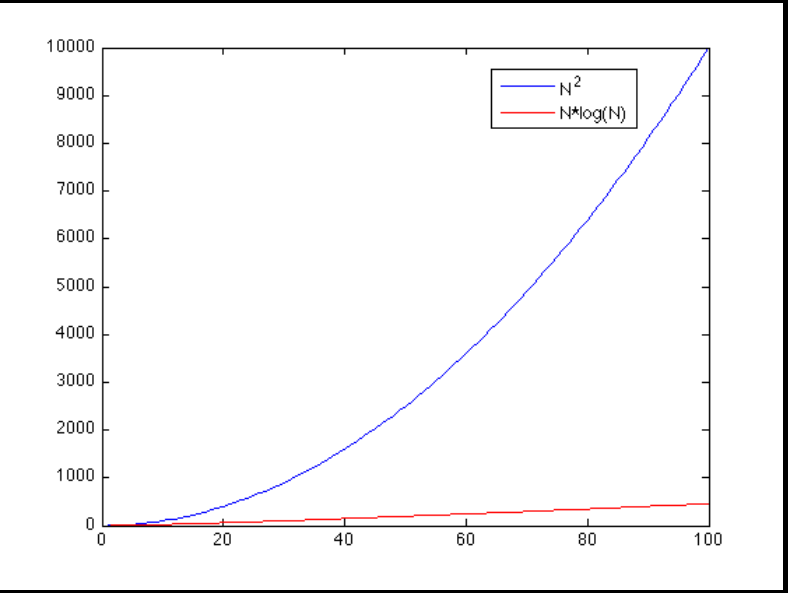
- Each node (circle) stores the total mass and center-of-mass coordinates for its members.

- If two nodes (*e.g.* 1, 5) are more than some predetermined distance apart, we use their clusters instead (1, e)

Barnes-Hut Solution: Speedup

- Each of the *N* bodies is on average compared with log *N* other bodies, so instead of N^2 we have $N * \log N$

- Can have each processor do the movement of its cluster in parallel with others (no communication)



http://www.shodor.org/galaxsee-simsurface/galaxsee.html