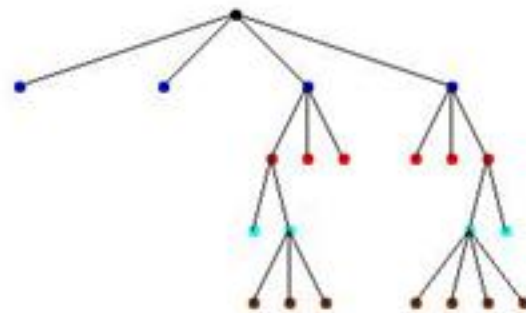
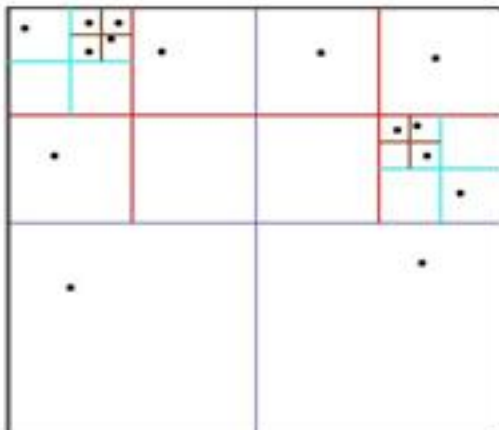


N-body Simulation

A Case Study



- There are N bodies with point mass m_1, m_2, \dots
- "Point mass" means the bodies are approximated as single points with the given mass
- Each *pair* of bodies are attracted by gravitational force of magnitude
 - $F = Gm_i m_j / d^2$, where
 - d is the distance between the points
 - $G = 6.67 \times 10^{-11} \text{m}^3 \text{kg}^{-1} \text{s}^{-2}$ is Newton's gravitational constant

- **Molecular dynamics, particles are atoms**
- **Charged particles, coulomb force**
- **Fluid dynamics, particles are droplets**

- **Force is mass x acceleration ($F=ma$)**
- **Acceleration is force/mass**
- **Acceleration is also change in velocity**
- **So net acceleration of the points can be obtained by**
 - **Computing for each point i , the vector *sum* over all *other* points j of Gm_j/d^2**
- **Euler's method of N-body calculations**
 - **Given the acceleration, we can compute the change in vector velocity for a small time-step**
 - **Given the vector velocity, we can compute the new position for a small time-step**

- The computation of acceleration for a single point is $O(n)$
- The computation for N points is $O(n^2)$
- This computation must be repeated each time step
- For large N , this can be significant

- **Comprises N bodies, each body characterized by**
 - **Position $p(x, y, z)$**
 - **Velocity $v(x, y, z)$**
 - **Mass m , and**
 - **Force $F(x, y, z)$**

- **Gravity causes the bodies to accelerate and to move**
 - **Each body attracted to each other body by gravitational forces**
 - **Movement of each body predicted by calculating total force on each body**

- **Motion of the bodies simulated by stepping through discrete instants of time**

```

initialize bodies;
for time step {
    calculate forces; // read p, m; compute F
    move bodies; // read F, m; compute new p and v
}
    
```

- **Force on a body is vector sum of the forces from all other bodies**
 - **Magnitude of gravitational force between bodies i and j**

$$F[i,j] = (G * m[i] * m[j]) / r[i,j]^2$$
 - **Magnitude is symmetric ("equal and opposite")**
 - **Direction represented by vector from one body to other**

- **Changes in velocity and position (moving a body) - leapfrog scheme**
 - **Acceleration: $a = F / m$**
 - **Change in velocity: $dv = a * DT$**
 - **Change in position**
 - **$dp = v * DT + (a/2)*DT^2 \sim (v + dv/2) * DT$**
 - **Here DT - the length of a time step**

- **Use of brute force**
 - n particles will require $O(n^2)$ calculations on a time step
 - Example, a galaxy containing 10^{11} stars will takes 10^9 years for one iteration
 - Assuming each calculation takes 1 ms

- **Alternative method is to use Barnes-Hut algorithm**
 - Uses approximations
 - Time can be improved to $O(n \log_2 n)$
 - Most scientists use it for N-Body simulation
 - It will take around one year for one iteration for above simulation
 - For live demo of Barnes-Hut N-body algorithm see: <http://www.cs.cmu.edu/~scandal/applets/bh.html>

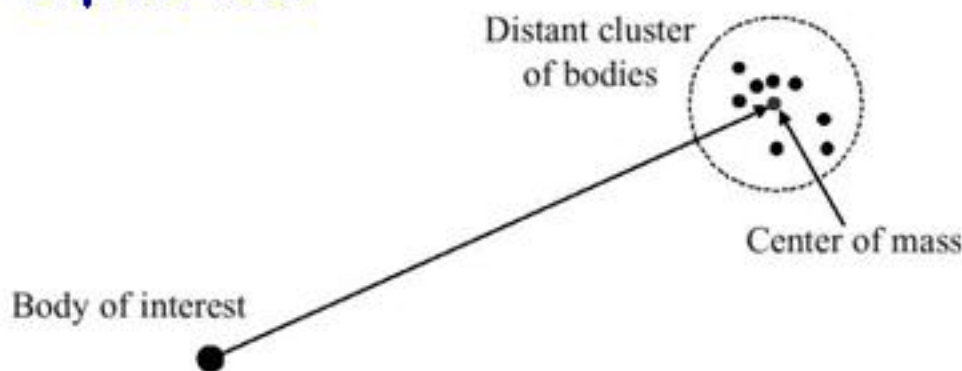
- Groups nearby bodies and approximates them as a single body
- Formally, if two bodies have positions (x_1, y_1) and (x_2, y_2) , and masses m_1 and m_2 , then their total mass and center of mass (x, y) are given by:

$$m = m_1 + m_2$$

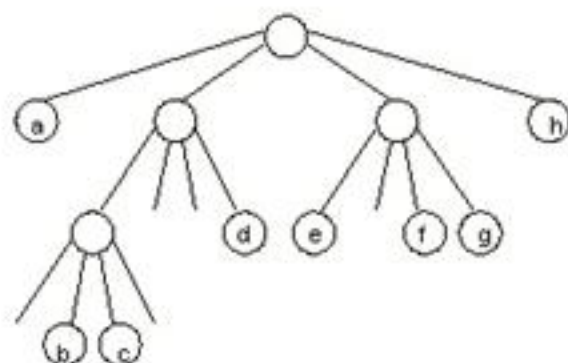
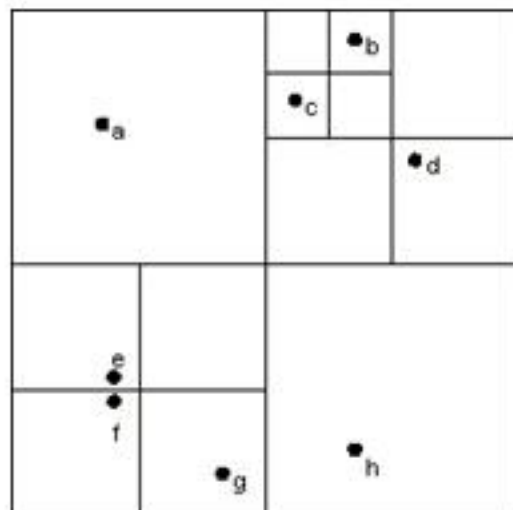
$$x = (x_1 m_1 + x_2 m_2) / m$$

$$y = (y_1 m_1 + y_2 m_2) / m$$

- It recursively divides the set of bodies into groups by storing them in a quad-tree



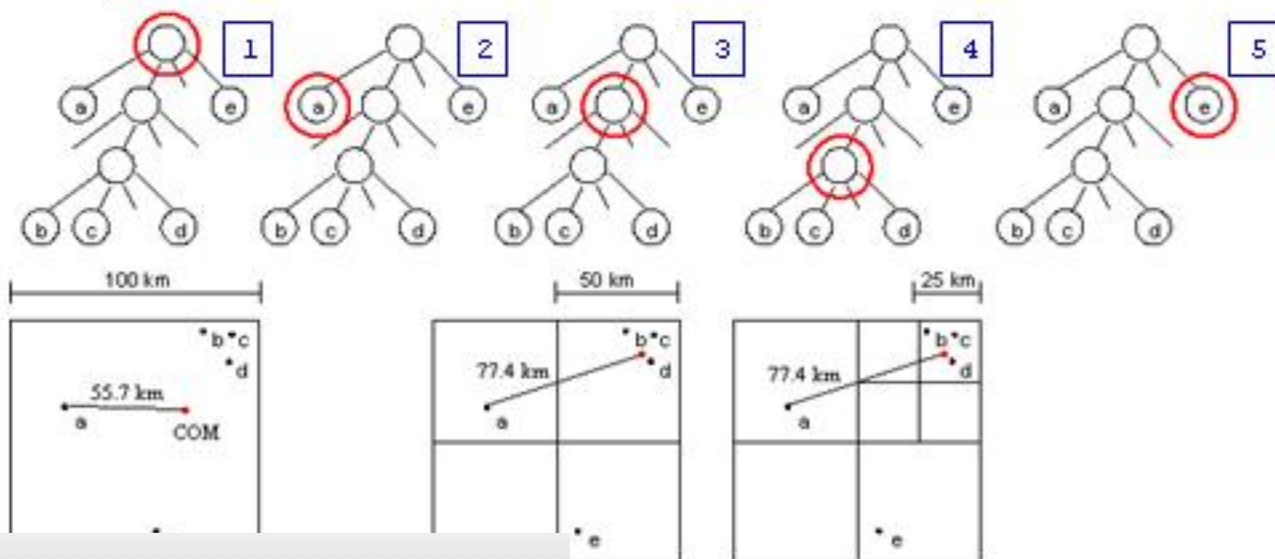
- **Each node represents region of the two dimensional space**
- **Topmost node represents the whole space**
 - Its four children represent the quadrants of space
 - Space recursively subdivided into quadrants until it contains 0 or 1 bodies
- **External node (leaf node) represents single body**
- **Internal node represents group of bodies beneath it**
 - And stores center-of-mass and total mass of all its children bodies

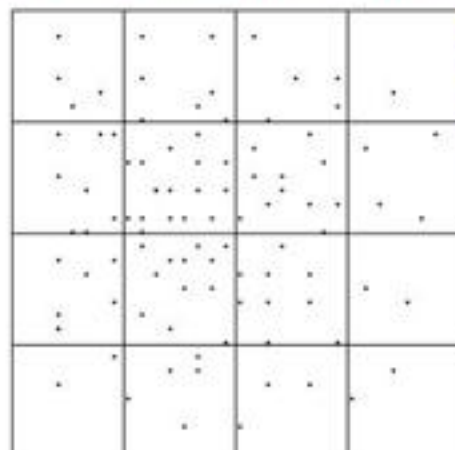


- **Traverse nodes of the quad tree, starting from the root**
 - **If current node is an external node (and is not body b)**
 - **Add force exerted by it on b to b 's net force**
 - **If current node is an internal node**
 - **Calculate the ratio s/d , where**
 - s is width of the region represented by internal node
 - d is distance between the body and the node's center-of-mass
 - **If $s/d < \theta$ (internal node is sufficiently far away)**
 - **Treat this internal node as single body**
 - **Having position and mass as group's center of mass and total mass, respectively**
 - **Add force it exerts on body b to b 's net force**
- Else, run the procedure recursively on each of the current node's children**

Case Study - N-body Simulation

1. **First node is a root node**
 - $s/d = 100/55.7 > \theta = 0.5$; So perform recursive process on each root's child
2. **First child is body a itself; So don't do anything**
3. **An internal node containing center-of-mass of bodies $b, c,$ and d**
 - $s/d = 50/77.4 > \theta$; So recursively calculate force exerted by node's only child
4. **Another internal node, containing center-of-mass of bodies $b, c,$ and d**
 - $s/d = 25/77.4 < \theta$
 - Treat node as single body, add force it exerts on body a to a 's net force
5. **Next child is an external node containing body e**
 - Add pairwise force between a and e to a 's net force





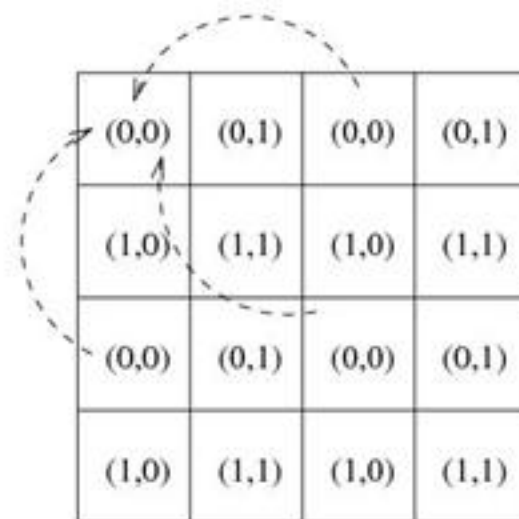
Partitioning the domain into $r = 16$ parts

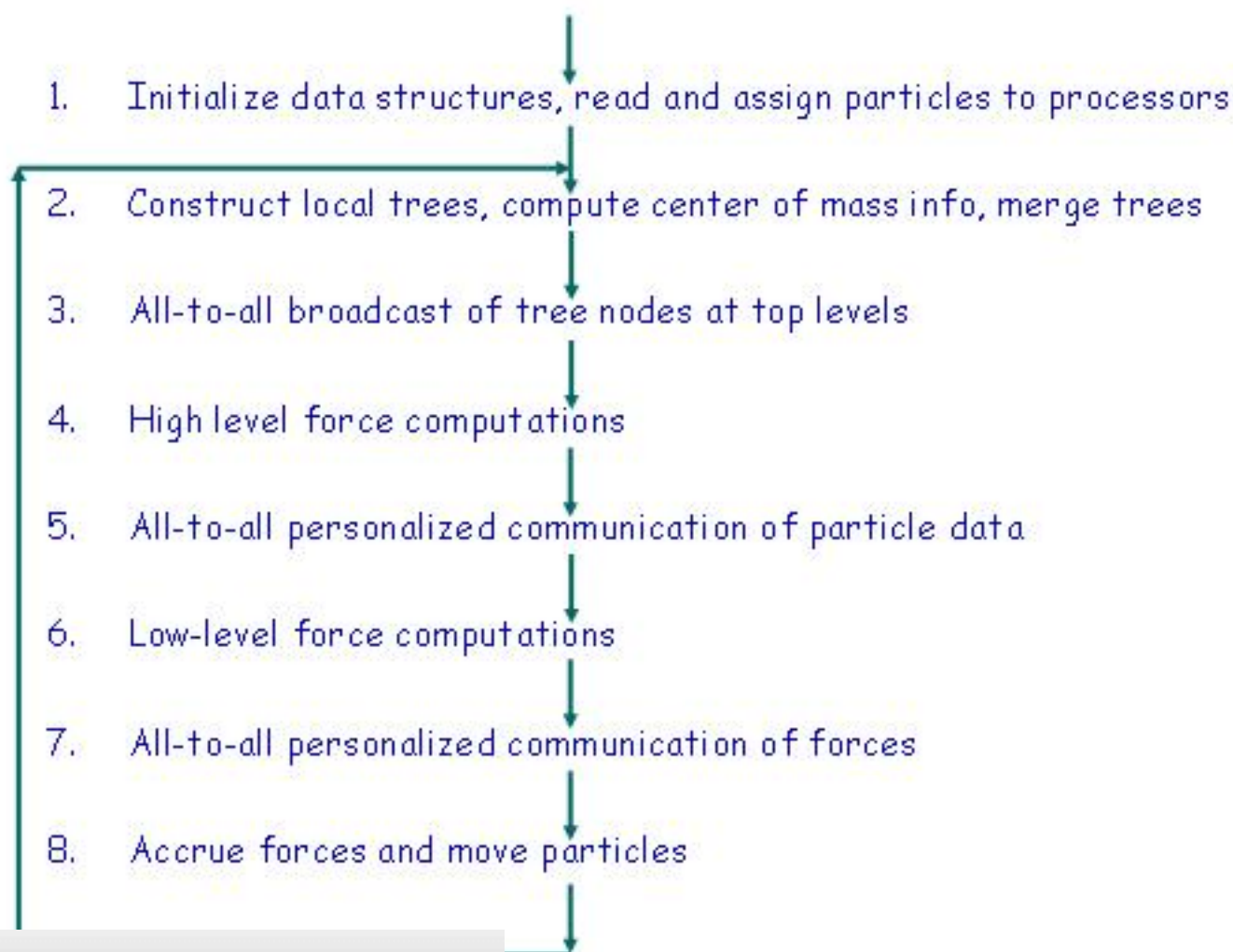
(0,0)	(0,1)	(0,0)	(0,1)
(1,0)	(1,1)	(1,0)	(1,1)
(0,0)	(0,1)	(0,0)	(0,1)
(1,0)	(1,1)	(1,0)	(1,1)

Mapping the sub-domains to processors

(0,0)	(0,1)	(0,0)	(0,1)
(1,0)	(1,1)	(1,0)	(1,1)
(0,0)	(0,1)	(0,0)	(0,1)
(1,0)	(1,1)	(1,0)	(1,1)

Communications for merging subtrees to form global tree





- Assume, shared memory programming model
- Parallelization - divide bodies among workers use a barrier after each phase

```

initialize bodies;
for time step {
    calculate forces;
    BARRIER; /* Check if it's really needed */
    move bodies;
    BARRIER;
}
    
```