CS601: Software Development for Scientific Computing

Autumn 2021

Week12:

N-Body problems and Hierarchical Methods

Course Progress..

- Particle (Simulation) Methods / N-Body Problems
  - PP, PM, P3M.
  - Hierarchical Methods
    - Tree-based codes

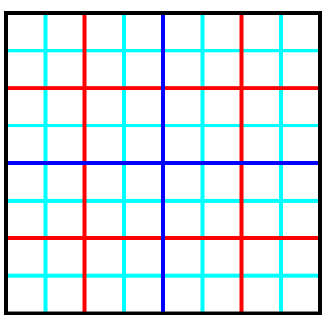
      - Preliminaries – Metric Trees

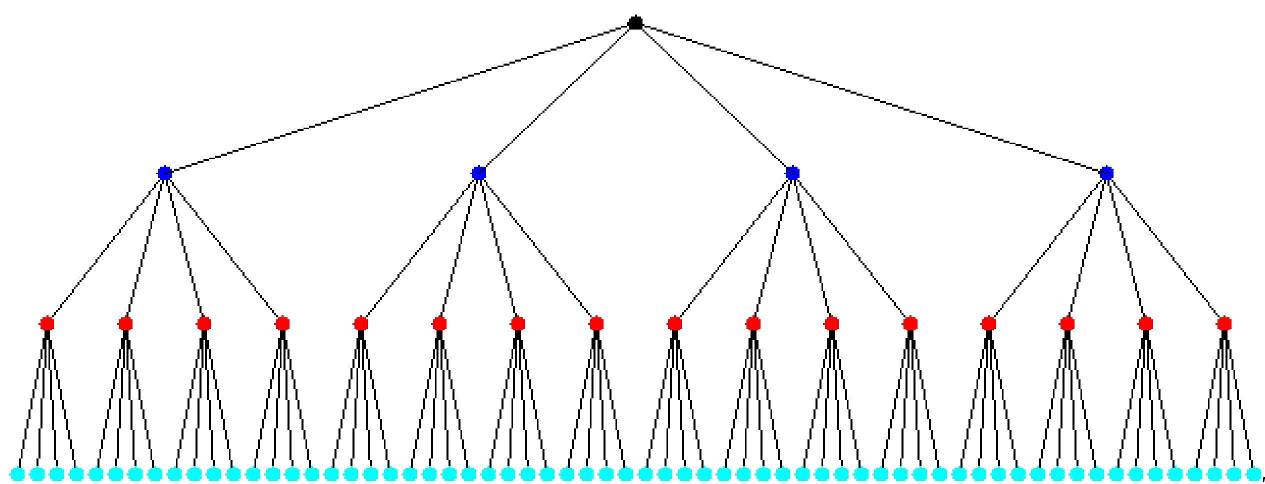
      - Quad Trees
- Applications:
  - Fluid Dynamics, Electromagnetics, Molecular Dynamics, Statistics, Astrophysics etc.

Quad Tree

- Data structure to subdivide the plane
  - Nodes can contain coordinates of center of box, side length.
  - Eventually also coordinates of CM, total mass, etc.
- In a complete quad tree, each non-leaf node has 4 children

A Complete Quadtree with 4 Levels



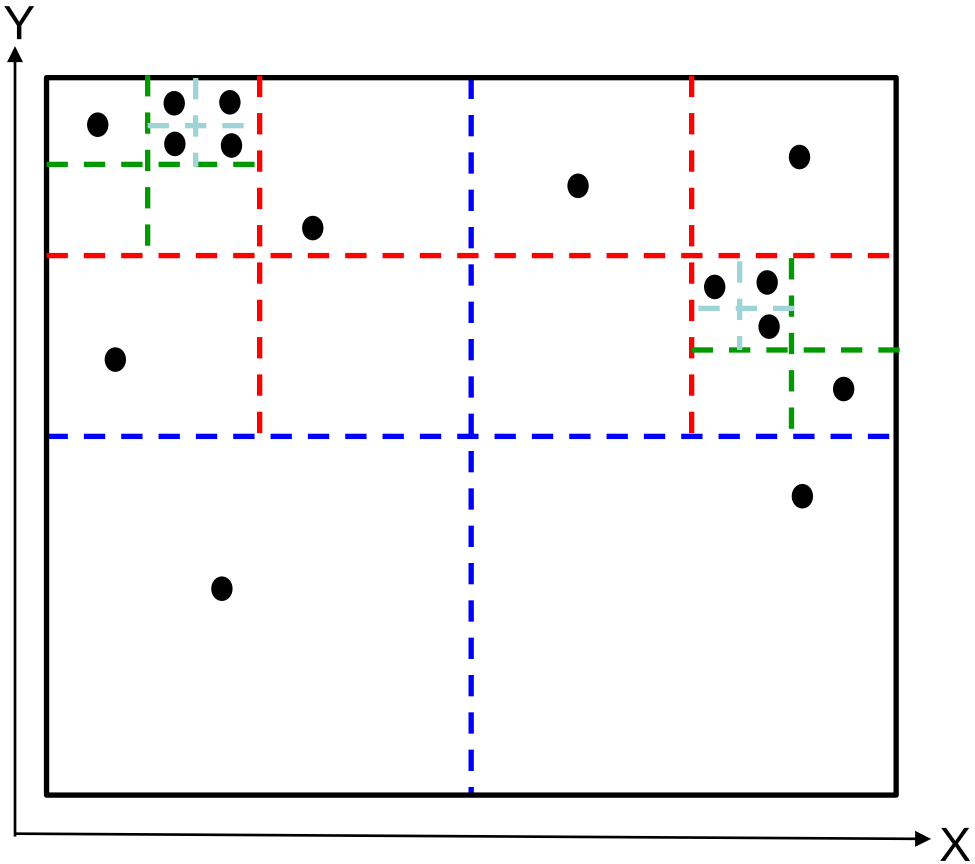


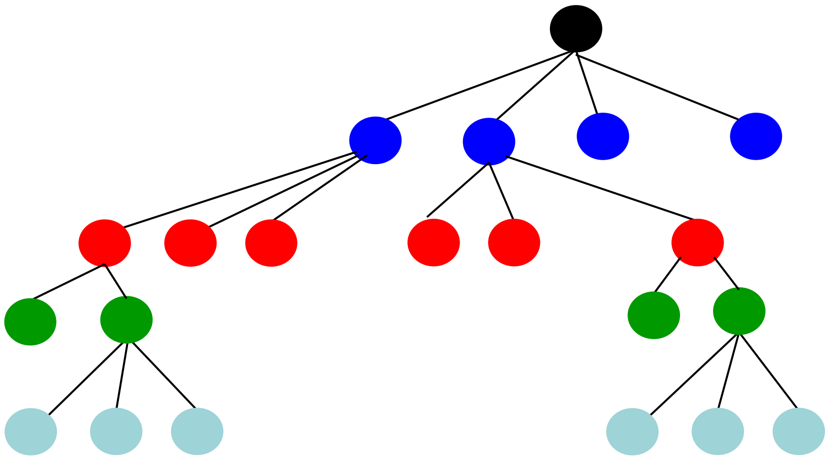
Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Using Quad Tree and Octree

1. Begin by constructing a tree to hold all the particles
   - Interesting cases have nonuniformly distributed particles
   - In a complete tree most nodes would be empty, a waste of space and time
   - Adaptive Quad (Oct) Tree only subdivides space where particles are located
2. For each particle, traverse the tree to compute force on it

Adaptive Quad Tree





- In practice, #particles/square > 1. tuning parameter
- Child nodes numbered as per *Z-order numbering*

Adaptive Quad Tree Construction

```
Procedure Quad_Tree_Build
  Quad_Tree = {emtpy}
  for j = 1 to N                          … loop over all N particles
     Quad_Tree_Insert(j, root)            … insert particle j in QuadTree
  endfor
  …  At this point, each leaf of Quad_Tree will have 0 or 1 particles
  …  There will be 0 particles when some sibling has 1
  Traverse the Quad_Tree eliminating empty leaves  … via, say Breadth First Search

Procedure Quad_Tree_Insert(j, n)   … Try to insert particle j at node n in Quad_Tree
  if n an internal node             … n has 4 children
     - determine which child c of node n contains particle j
     - Quad_Tree_Insert(j, c)
  else if n contains 1 particle   …  n is a leaf
     - add n's 4 children to the Quad_Tree
     - move the particle already in n into the child containing it
     - let c be the child of n containing j
     - Quad_Tree_Insert(j, c)
  else                              …  n empty
     - store particle j in node n
  end
```


Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Adaptive Quad Tree Construction – Cost?

```
Procedure Quad_Tree_Build
   Quad_Tree = {emtpy}                              ≤ N *max cost of Quad_Tree_Insert
   for j = 1 to N                       … loop over all N particles
       Quad_Tree_Insert(j, root)        … insert particle j in QuadTree
   endfor
   …   At this point, each leaf of Quad_Tree will have 0 or 1 particles
   …   There will be 0 particles when some sibling has 1
   Traverse the Quad_Tree eliminating empty leaves  … via, say Breadth First Search
```

```
Procedure Quad_Tree_Insert(j, n) … Try to insert particle j at node n in Quad_Tree
   if n an internal node              … n has 4 children
      - determine which child c of node n contains particle j
      - Quad_Tree_Insert(j, c)
   else if n contains 1 particle      … n is a leaf
      - add n's 4 children to the Quad_Tree
      - move the particle already in n into the child containing it
      - let c be the child of n containing j
      - Quad_Tree_Insert(j, c)                    ≤ max depth of Quad Tree
   else                               … n empty
      - store particle j in node n
   end
```


Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Adaptive Quad Tree Construction – Cost?

- Max Depth of Tree:
  - For uniformly distributed points?
  - For arbitrarily distributed points?

- Total Cost = ?

Adaptive Quad Tree Construction – Cost?

- Max Depth of Tree:
  - For uniformly distributed points? = O(log N)
  - For arbitrarily distributed points? = O(bN)
    - b is number bits used to represent the coordinates

- Total Cost = $O(\text{ b N})$ or $O(N * \log N)$

Barnes-Hut

- Simplest hierarchical method for N-Body simulation
  - "A Hierarchical O(n log n) force calculation algorithm" by J. Barnes and P. Hut, Nature, v. 324, December 1986
- Widely used in astrophysics
- Accuracy $\geq$ 1% (good when low accuracy is desired/acceptable. Often the case in astrophysics simulations.)

Barnes-Hut: Algorithm

(2D for simplicity)

1) Build the QuadTree using QuadTreeBuild
   … already described, cost = O(N log N) or O(b N)

2) For each node/subsquare in the QuadTree, compute the
   Center of Mass (CM) and total mass (TM) of all the particles it contains.

3) For each particle, traverse the QuadTree to compute the force on it,

Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Barnes-Hut: Algorithm (step 2)

```
Goal: Compute the Center of Mass (CM) and Total Mass (TM) of all the
particles in each node of the QuadTree. (TM, CM) = Compute_Mass( root )

(TM, CM)  = Compute_Mass( n )    //compute the CM and TM of node n
   if n contains 1 particle
        //TM and CM are identical to the particle’s mass and location
        store (TM, CM) at n
        return (TM, CM)
  else
     for each child c(j) of n  //j = 1,2,3,4
           ( TM(j), CM(j) ) = Compute_Mass( c(j) )
     endfor
     TM = TM(1) + TM(2) + TM(3) + TM(4)
     //the total mass is the sum of the children’s masses
     CM = ( TM(1)*CM(1) + TM(2)*CM(2) + TM(3)*CM(3) + TM(4)*CM(4) ) / TM
     //the CM is the mass-weighted sum of the children’s centers of mass
     store ( TM, CM ) at n
     return ( TM, CM )
  end if
```


Slide based on : CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Barnes-Hut: Algorithm (step 2 cost)

(2D for simplicity)

1) Build the QuadTree using QuadTreeBuild
   … already described, cost = O(N log N) or O(b N)

2) For each node/subsquare in the QuadTree, compute the
   Center of Mass (CM) and total mass (TM) of all the particles it contains.
   … cost = O(number of nodes in the tree) = O(N log N) or O(b N)

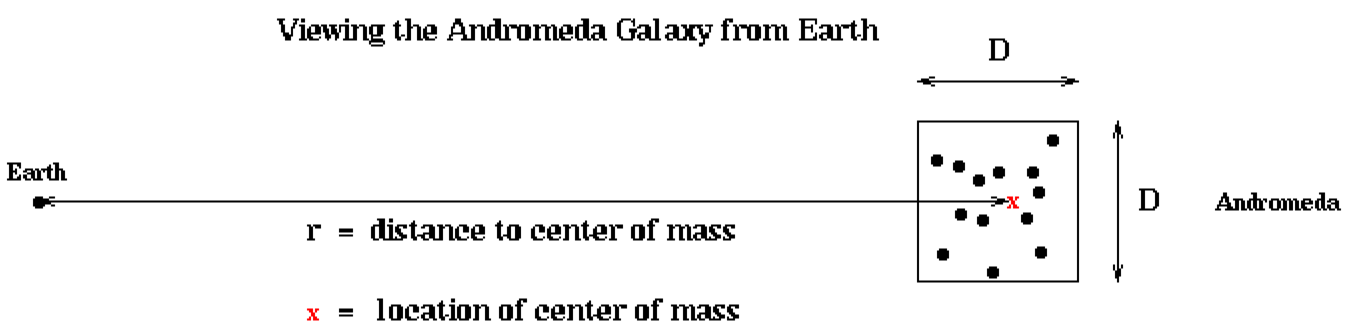
3) For each particle, traverse the QuadTree to compute the force on it,

Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Barnes-Hut: Algorithm (step 3)

Goal: Compute the force on each particle by traversing the tree. For each particle, use as few nodes as possible to compute force, subject to accuracy constraint.

- For each node = square, can approximate force on particles outside the node due to particles inside node by using the node's CM and TM
- This will be accurate enough if the node if "far away enough" from the particle
- Need criterion to decide if a node is far enough from a particle
  - **D = side length of node**
  - **r = distance from particle to CM of node**
  - **θ = user supplied error tolerance < 1**
  - **Use CM and TM to approximate force of node on box if D/r < θ**



Slide based on : CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Barnes-Hut: Algorithm (step 3)

```
//for each particle, traverse the QuadTree to compute the force on it
for k = 1 to N
    f(k) = TreeForce( k, root )
    //compute force on particle k due to all particles inside root (except k)
endfor
function f = TreeForce( k, n )
   //compute force on particle k due to all particles inside node n (except k)
   f = 0
   if n contains one particle (not k)  //evaluate directly
       return f = force computed using direct formula
   else
       r = distance from particle k to CM of particles in n
       D = size of n
       if  D/r  < q    //ok to approximate by CM and TM
            return f = computed approximately using CM and TM
       else                  //need to look inside node
            for each child c(j) of n //j=1,2,3,4
                  f = f + TreeForce ( k, c(j) )
            end for
            return f
       end if
   end if
```

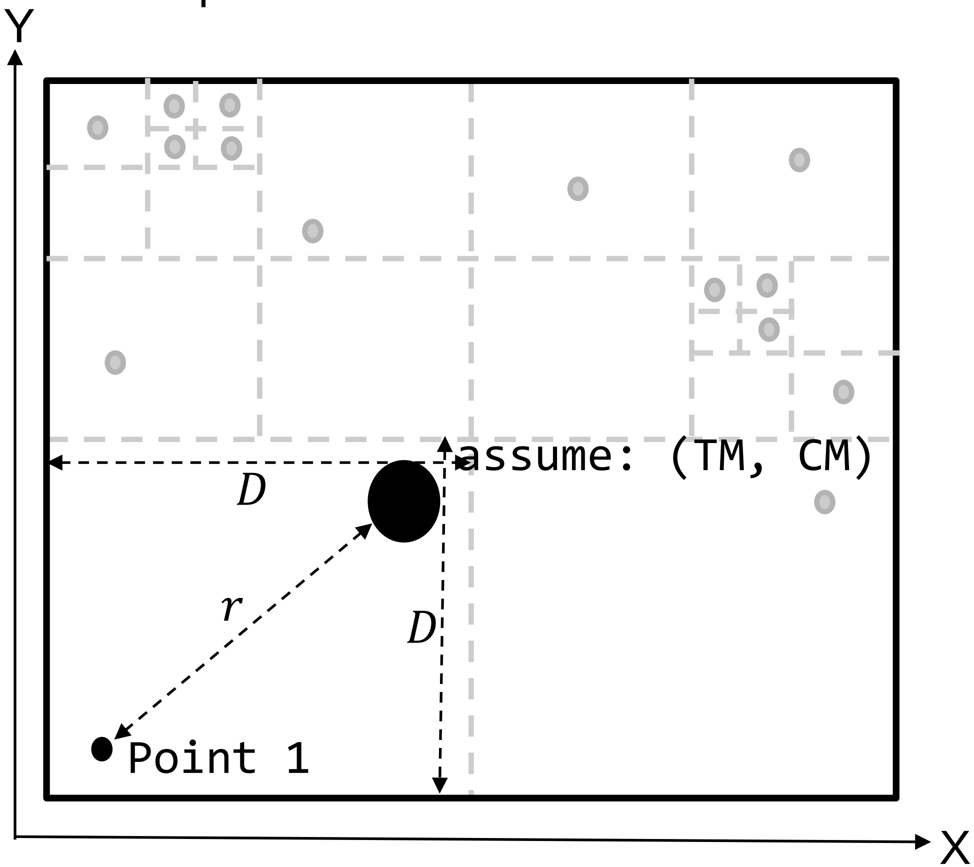
Slide based on : CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

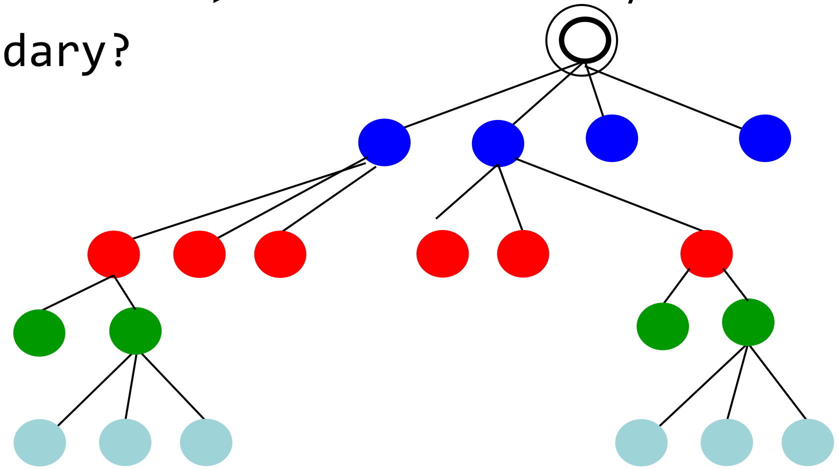
Barnes-Hut: step 3 example

- Example: Assume $\theta \geq 1$. In practice $\theta < 1$.

What is the force on Point 1 due to all other points in the box with black-boundary?

Point 1: is $D/r < \theta$?



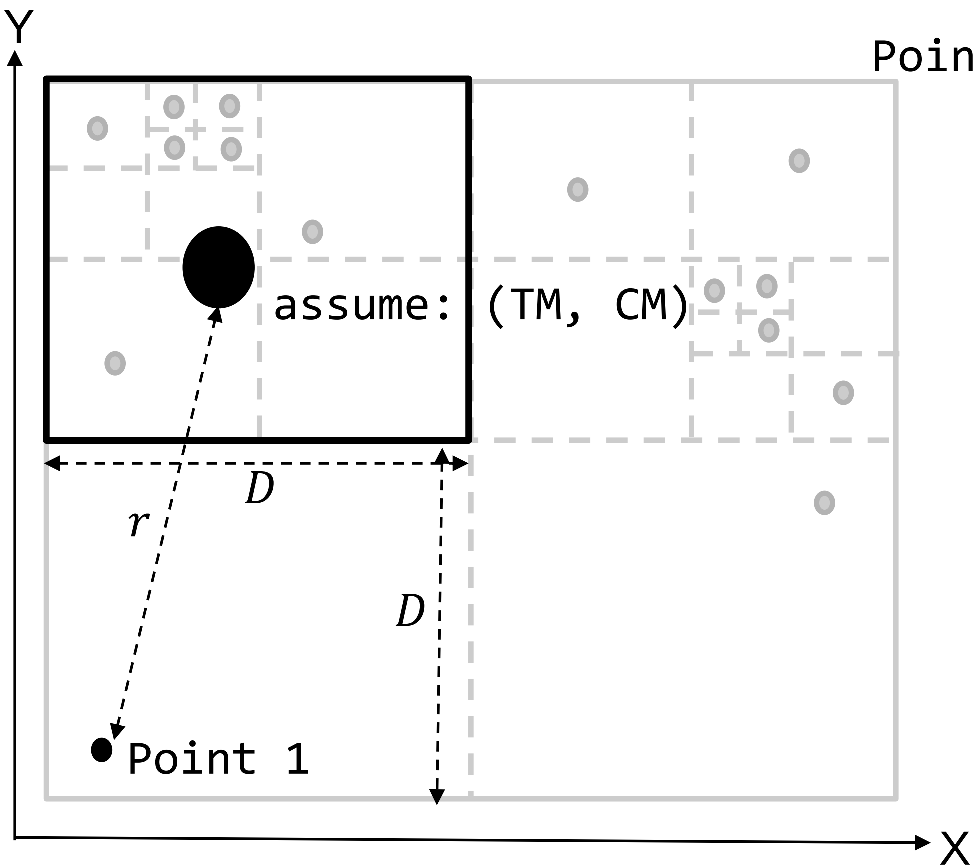


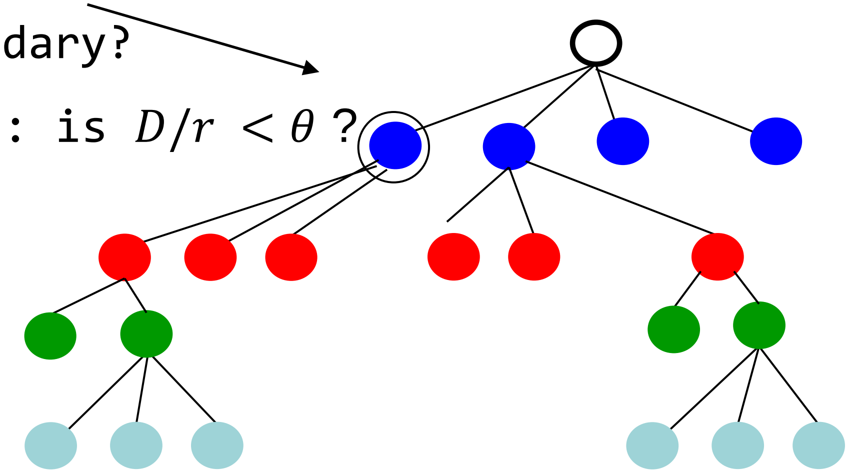
No. Compute force due to each child of the root node (i.e. particles in each quadrant of the square). Start with child 1: c(1).

Barnes-Hut: step 3 example

- Example: Assume $\theta \geq 1$. In practice $\theta < 1$.

`What is the force on Point 1 due to all other points in the box with black-boundary?`





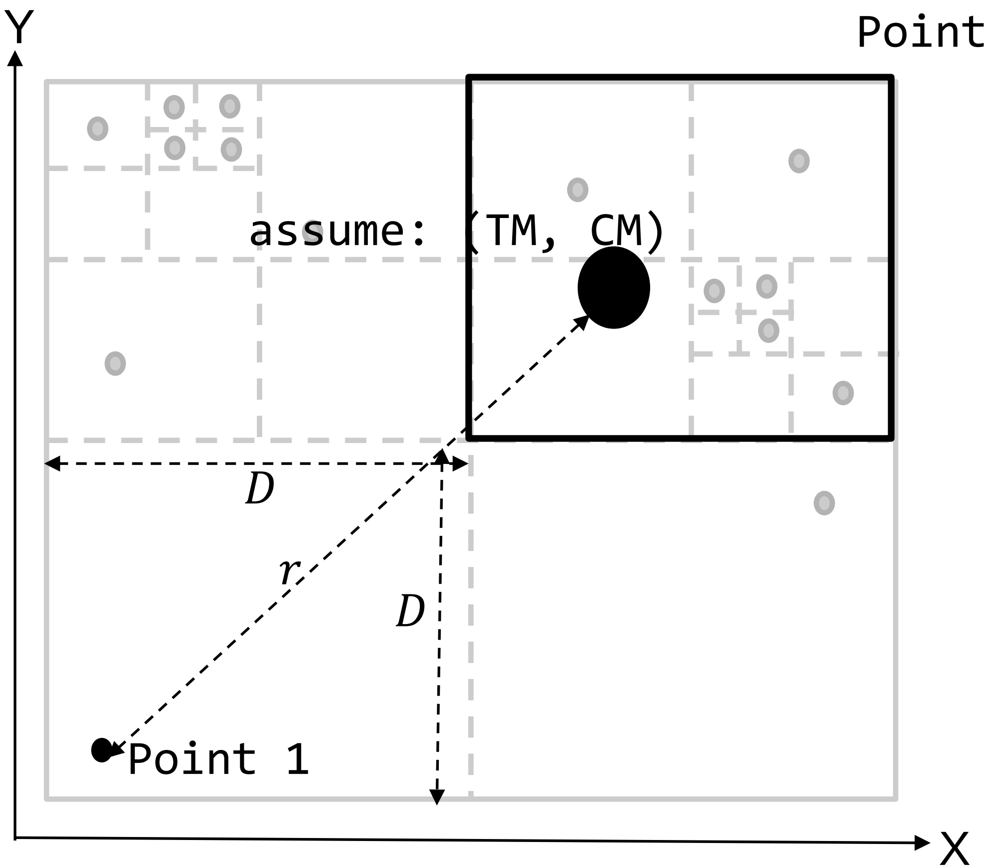
`Yes. Approximate force due to each particle contained in the black-boundary box by the TM and CM of the box.`

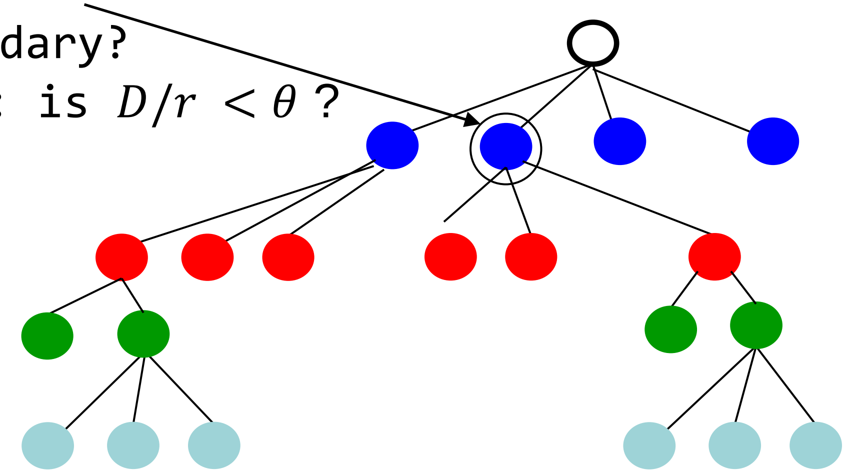
Barnes-Hut: step 3 example

- Example: Assume $\theta \geq 1$. In practice $\theta < 1$.

What is the force on Point 1 due to all other points in the box with black-boundary?

Point 1: is $D/r < \theta$?





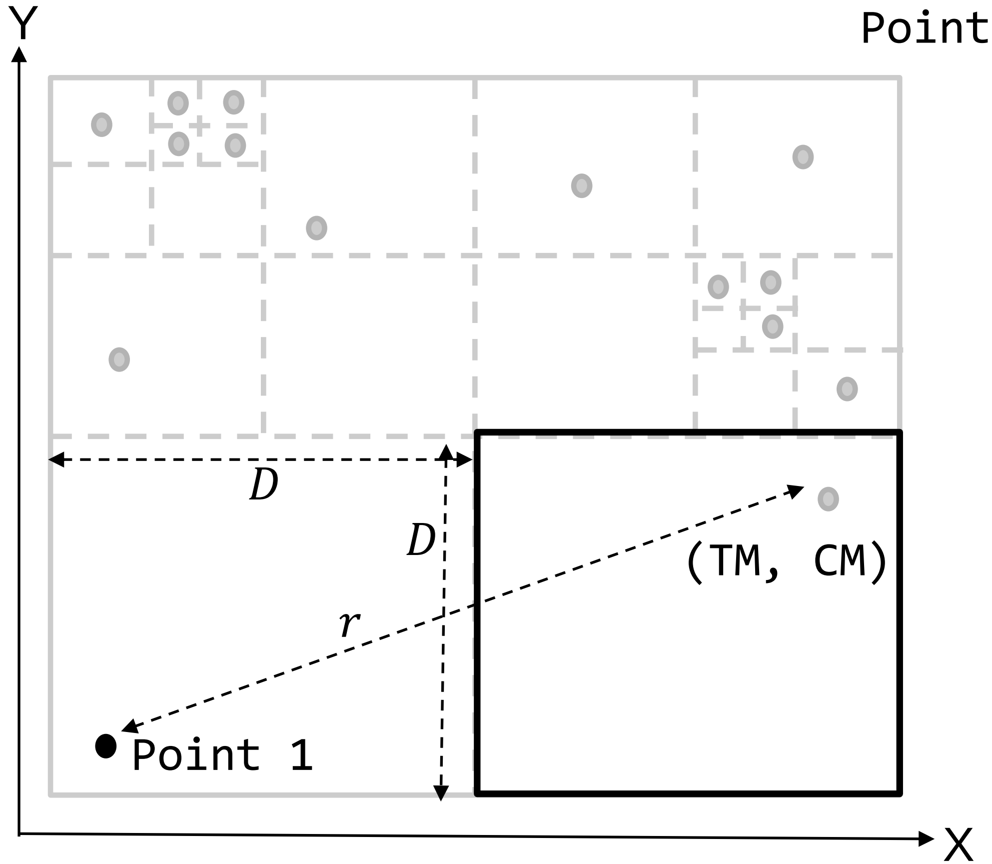
Yes. Approximate force due to each particle contained in the black-boundary box by the TM and CM of the box.

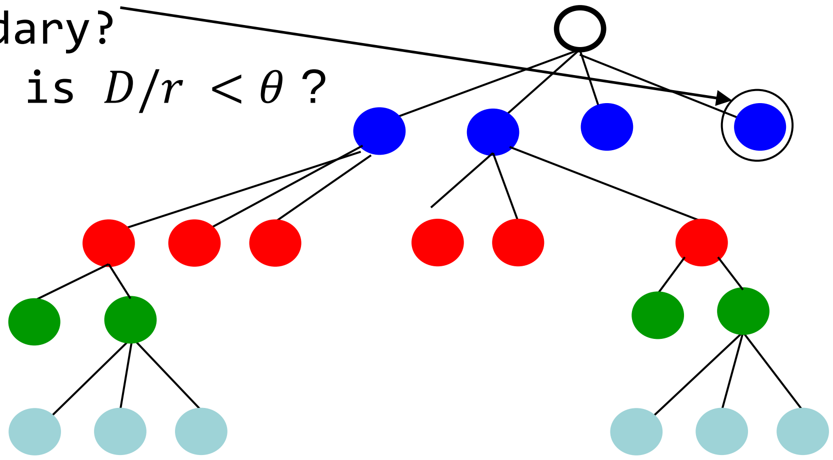
Barnes-Hut: step 3 example

- Example: Assume $\theta \geq 1$. In practice $\theta < 1$.

What is the force on Point 1 due to all other points in the box with black-boundary?

Point 1: is $D/r < \theta$?





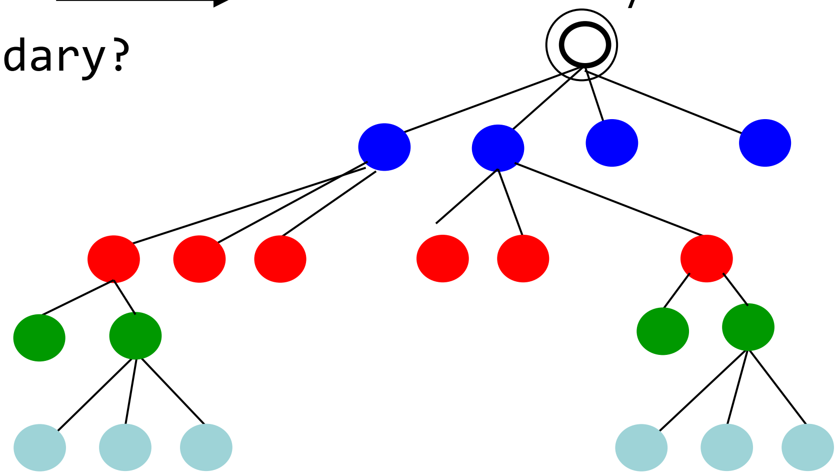
Contains 1 particle / leaf node. Compute force using direct formula.

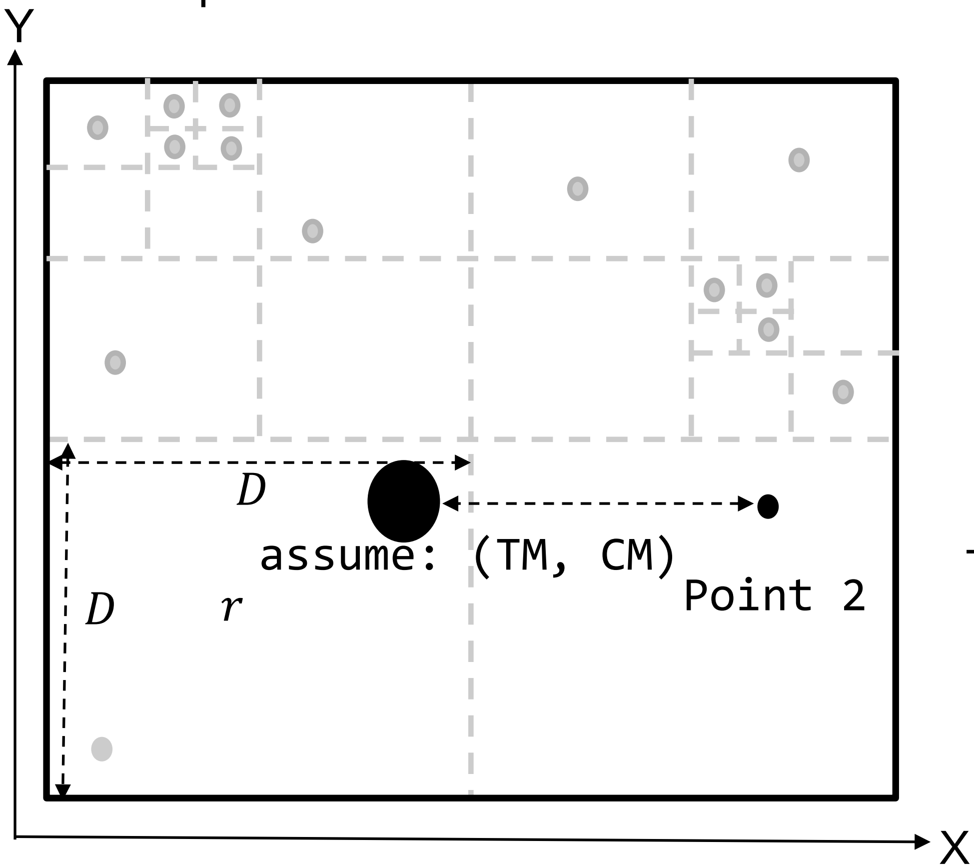
Barnes-Hut: step 3 example

- Example: Assume $\theta \geq 1$. In practice $\theta < 1$.

What is the force on **Point 2** due to all other points in the box with black-boundary?

Point 2: is $D/r < \theta$?





Traverse the tree for particle 2.

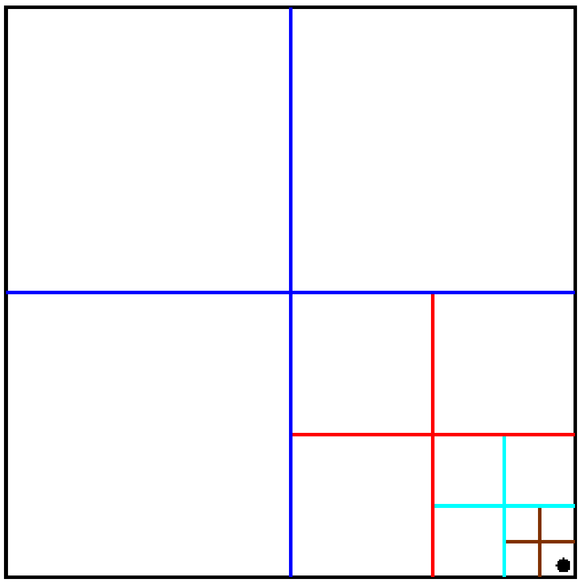
Barnes-Hut: Algorithm (step 3 cost)

- **Correctness** follows from recursive accumulation of force from each subtree
  - Each particle is accounted for exactly once, whether it is in a leaf or other node
- **Complexity** analysis
  - **Cost of TreeForce(k, root) = O(depth of leaf containing k in the QuadTree)**
  - Proof by Example (for $\theta > 1$):

- For each undivided node = square, (except one containing k), $D/r < 1 < \theta$
- There are at most 3 undivided nodes at each level of the QuadTree.
  - –There is O(1) work per node
  - –Cost = O(level of k)

Total cost = $O(\Sigma_k$ level of k) = O(N log N)

Strongly depends on θ



Slide based on : CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

Barnes-Hut: Algorithm (step 3 cost)

(2D for simplicity)

1) Build the QuadTree using QuadTreeBuild
   … already described, cost = O(N log N) or O(b N)

2) For each node/subsquare in the QuadTree, compute the Center of Mass (CM) and total mass (TM) of all the particles it contains.
   … cost = O(number of nodes in the tree) = O(N log N) or O(b N)

3) For each particle, traverse the QuadTree to compute the force on it,
   … cost depends on accuracy desired (θ) but still O(N log N) or O(bN)

Slide courtesy: CS267 Lecture 24, https://sites.google.com/lbl.gov/cs267-spr2019/

N-Body Simulation: Big Picture

- Recall:

```
t=0
while(t<t^final) {
//initialize forces

//Accumulate forces
      BH(steps 1 to 3)

//Integrate equations of motion

//Update time counter
      t = t + Δt
}
```

Fast Multipole Method (FMM)

- Can we make the complexity independent of the accuracy parameter (θ) ? FMM achieves this.
  - "Rapid Solution of Integral Equations of Classical Potential Theory", V. Rokhlin, J. Comp. Phys. v. 60, 1985 and
  - **"A Fast Algorithm for Particle Simulations",** L. Greengard and V. Rokhlin, J. Comp. Phys. v. 73, 1987.
- Similar to BH:
  - uses QuadTree and the divide-conquer paradigm
- Different from BH:
  - Uses more than TM and CM information in a box. So, computation is expensive and accurate than BH.
  - The number of boxes evaluated is fixed for a given accuracy parameter
  - Computes potential and not the Force as in BH

Background: Potential

- Force on a particle at (x, y, z) due to a particle at origin

$\propto -\frac{(x,y,z)}{r^3}$ (This is called inverse-square law. Gravitational and electrostatic forces obey this.) where, $r = \sqrt{x^2 + y^2 + z^2}$

- Force is a vector. Potential is a scalar. Hence, potential is simple to deal with.

Potential $\Phi(x, y, z) = -\frac{1}{r}$

- Negative of the gradient of potential = force

$$-\nabla\Phi(x, y, z) = -\left(\frac{d}{dx}\left(-\frac{1}{r}\right), \frac{d}{dy}\left(-\frac{1}{r}\right), \frac{d}{dz}\left(-\frac{1}{r}\right)\right)$$

Background: Potential

- In 2D, potential $\Phi(x, y) = \log r$
- Suppose we have N points (at $z_1, z_2, \dots, z_N$, where $z_i = (x_i, y_i)$) in a plane with masses $m_1, m_2, \dots, m_N$ resp.

then, their potential at $z = (x, y)$ is given by:

$$\Phi(x, y) = \sum_{i=1}^{N} m_i \log\left(\sqrt{(x - x_i)^2 + (y - y_i)^2}\right)$$

Goal: *evaluate $\Phi(x, y)$ and its derivatives at N points ($z_1, z_2, \dots, z_N$) in O(N) time.*

FMM Algorithm

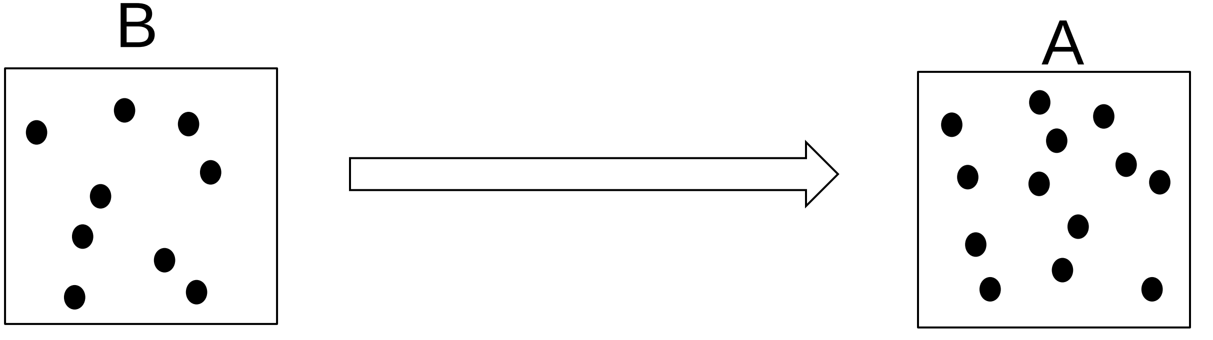
1. Build the quadtree containing all the points.
2. Traverse the quadtree from bottom to top, computing Outer(n) for each square n in the tree.
3. Traverse the quadtree from top to bottom, computing Inner(n) for each square in the tree.
4. For each leaf, add the contributions of nearest neighbors and particles in the leaf to Inner(n)

what is Outer(n) and Inner(n) ?

Well Separated Regions

- Compute the influence of all particles in source region (B) on every particle in target region (A)

  (assumption: A and B are well-separated)



- At each point p_i in A, compute potential:

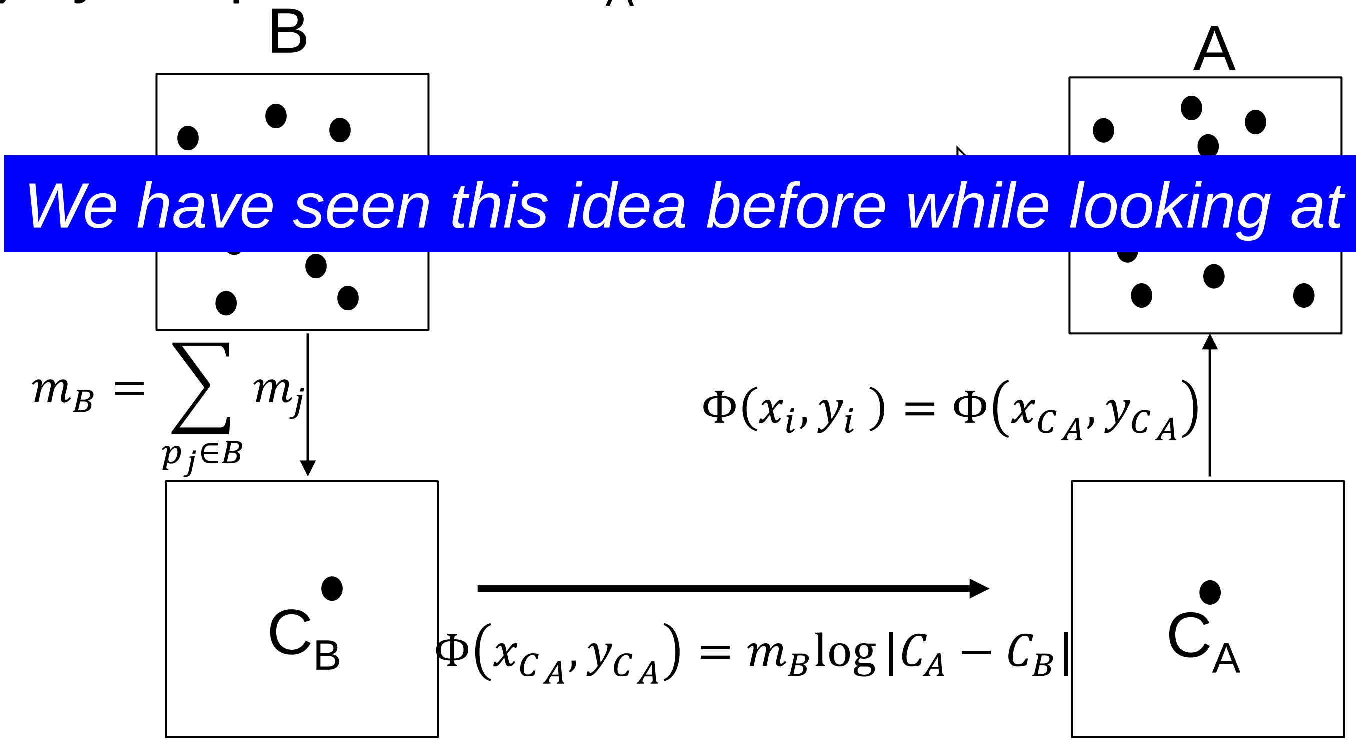
$$\Phi(x_i, y_i) = \sum_{p_j \in B} m_i \log |p_i - p_j|$$

$$i = 1\ to\ N_A, \qquad j = 1\ to\ N_B$$

- Cost: $O(N_A N_B)$

Well Separated Regions

- Approximate the potential at every particle in target region (A) by the potential at C_A



We have seen this idea before while looking at BH

- Cost: $O(N_A + N_B)$

Hierarchical Decomposition

- In N-body simulation, every point serves as source as well as target. *How to identify source, target, well-separated regions?*
  - Partition the space recursively till every leaf box contains O(1) number of points