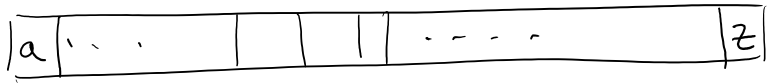
Binary Search Algorithm:
Find an element in a sorted array of elements.

Find: "Mary"

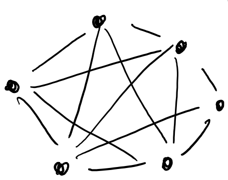
linear search $O(N)$



binary search $O(\log_2 N)$

But what about doing this in 2-3-D or N-dimensions in general.

N-points (5)



$\frac{N(N-1)}{2}$ Forces (10) $O(N^2)$

$N = 4 \times 10^{12}$ particles (2019)

$\frac{N^2}{2} = 8 \times 10^{24}$ Forces

20 Flops / Force

$\Rightarrow 1.6 \times 10^{26}$ Flop

Fast Computer

Petaflop Computer $10^{15}/s$

Exaflop Computer $10^{18}/s$

Time $= 1,6 \times 10^8$ s

1 year $= 10^{7.5}$ s

Several Years on Exaflop !

Used an $O(N)$ algorithm

$O(N \log N)$

Fast Fourier Transform $O(N \log N)$

$$\nabla^2 \phi = \rho(r)$$

FT?

FFT

$$-k^2 \Phi_k = \rho_K$$

$O(N \log N)$

$$-k^2 \phi_{\underline{k}} = \rho_{\underline{k}}$$

$$\phi_{\underline{k}} = -\frac{\rho_{\underline{k}}}{k^2}$$

IFFT

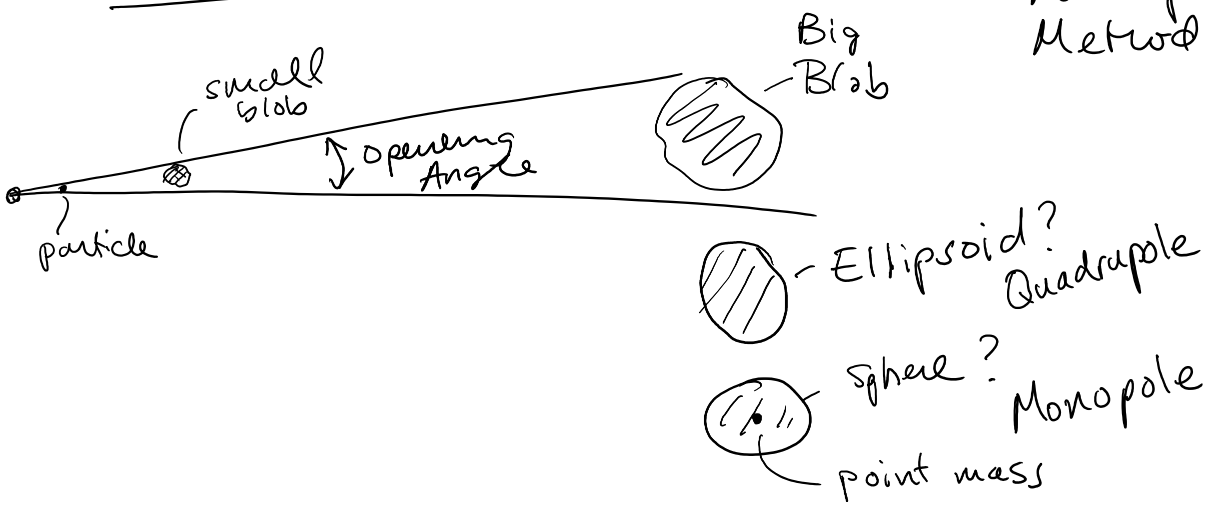
$$\phi(\underline{r})$$

$O(N \log N)$

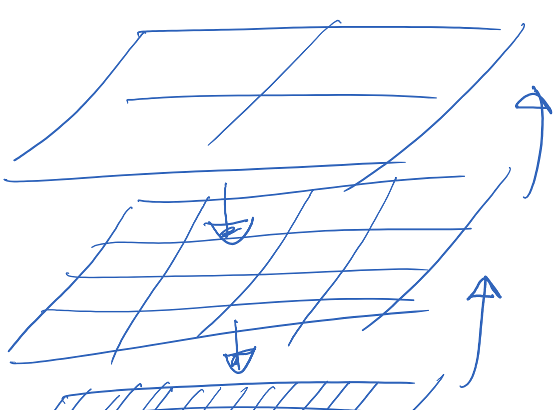
We accept a certain truncation error in the calculation, which frees us to invent new fast algorithms.

Multipole Methods $O(N \log N)$, $O(N)$ ← Fast Multipole Method

Tree structures: "Tree codes"

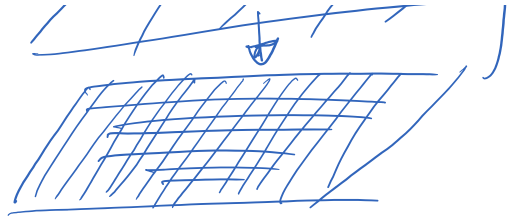


Multigrid ⇝ S.O.R.

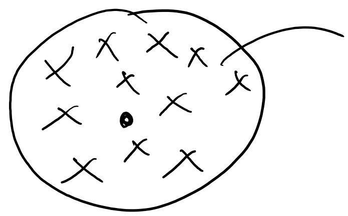


$O(N)$

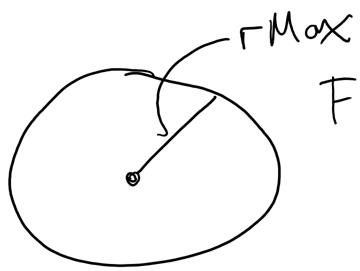
More general



Nearest Neighbor Searching

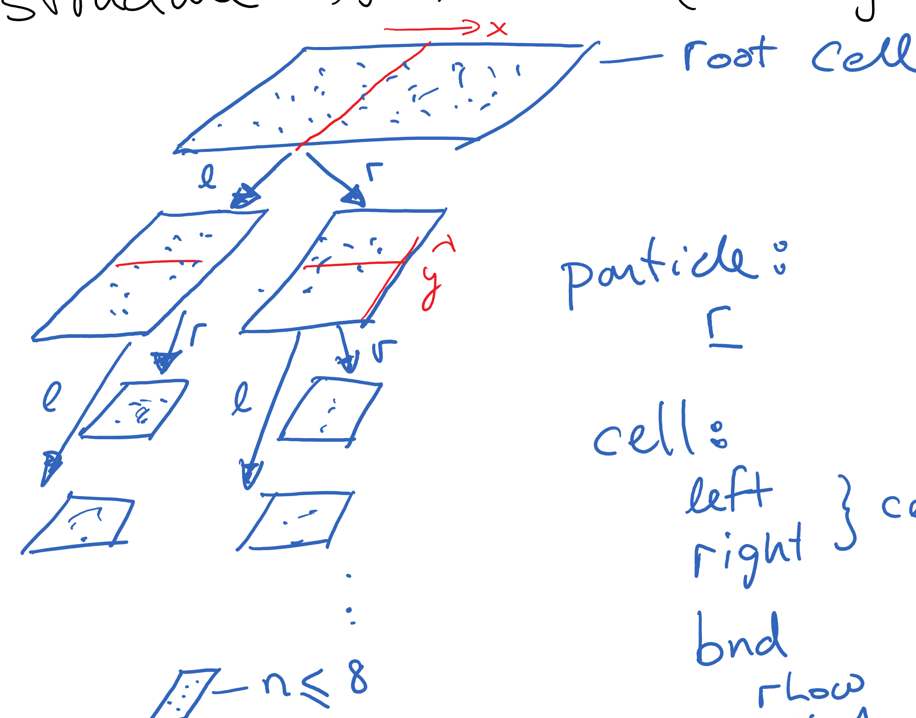


Find the 10 nearest neighbors.
(tiny bit harder)



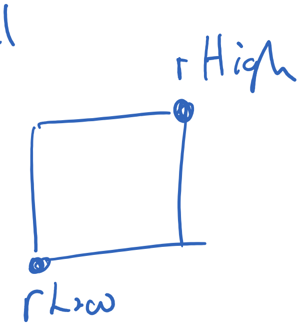
Find all points within some rMax of a given point.
(bit easier)

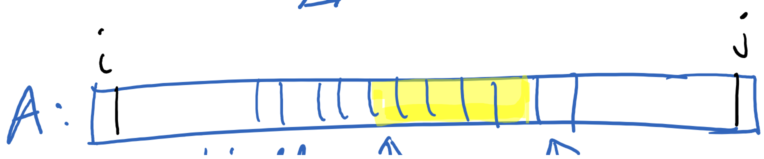
Tree structure is needed (Binary Tree)

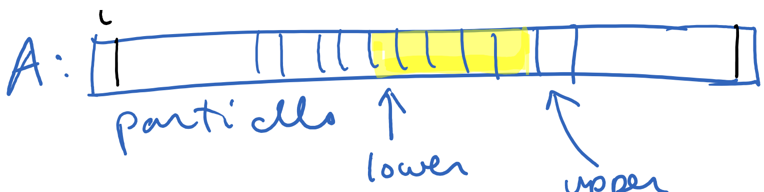


particle:
r

cell:
left } cell
right
bnd
rLow
rHigh
lower
upper







lower
upper

//split"

$S = \text{partition}(A, i, j, \overset{0.5}{v}, d)$ ← 4 lines! Write this