

Lecture 9 CMSC878R/AMSC698R

Outline

- Homework 3
- Review
- Homework 4
- Data Structures
- Quadrees, Octrees, Generalizations ...
- Operations on them

Algorithms

- Factorization was shown to be a fast method for achieving matrix vector products
- Approximate factorization was shown to be a fast method for achieving matrix vector products to arbitrary accuracy
 - Speed by adding coefficients of like terms in a series and evaluating consolidated series
- Translation allows consolidation of series
 - Last class “Middleman”
- Sometimes series may not converge fast or at all
 - Need multiple translation points
 - Last class SLFMM
- With data structures more generalized versions (FMM,)

The need for S expansions

- Near an analytic function expect quickly convergent power series (basis of interpolation)
 - However far away this series may converge slowly
 - Or not converge at all
- Far field expansions may converge faster
 - $1/(1+x^2)$ has small radius of convergence for power series
 - However, expansion in terms of $1/x$ is quickly convergent
- If the function is not analytic, we need an expansion that excludes the singularity

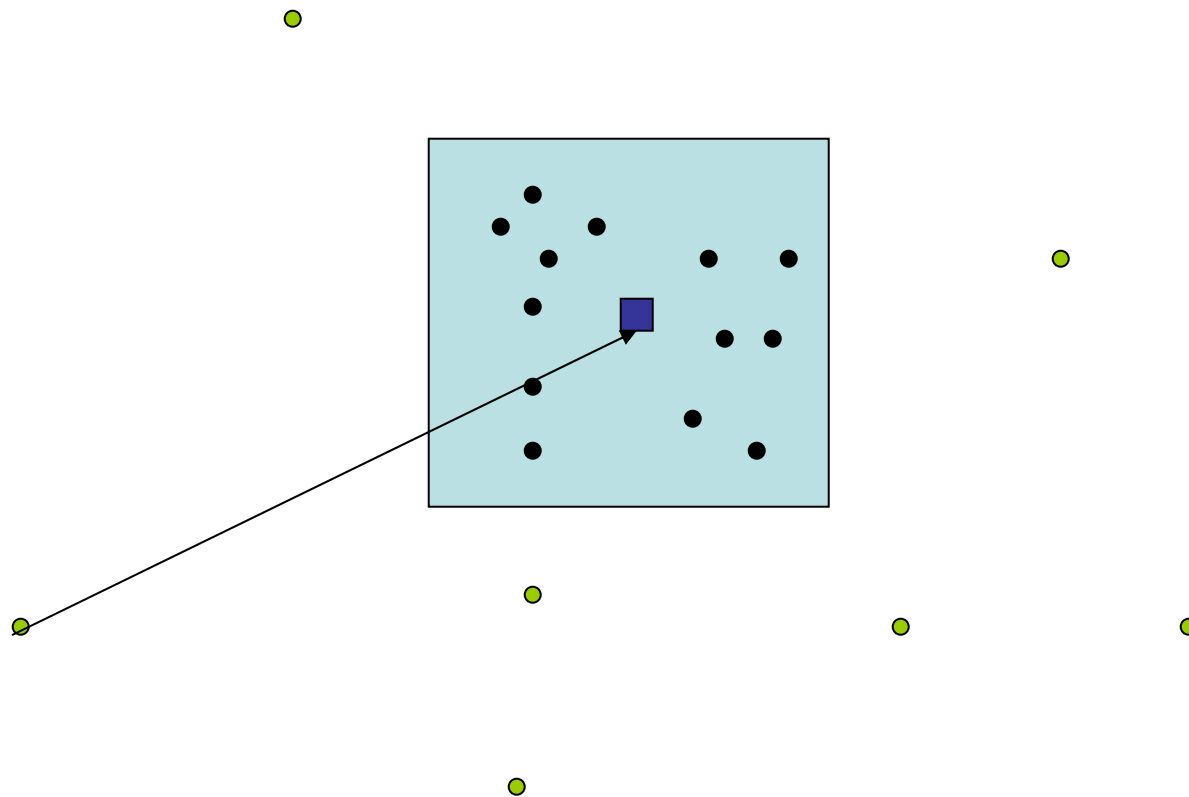
$$1/|\mathbf{x}_i - \mathbf{y}_j|$$

$$\exp(ik|\mathbf{x}_i - \mathbf{y}_j|)/|\mathbf{x}_i - \mathbf{y}_j|$$

- S expansions do this

The need for R expansions & S|R translation

- In a given neighborhood without singularities easier to consolidate multiple R and S expansions into one.



Notation and Glossary

- $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^d$ source points
- $Y = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$, $\mathbf{y}_i \in \mathbb{R}^d$ evaluation points
- i : index describing the source points
- j : index describing the evaluation points
- $\Phi(\mathbf{x}_i, \mathbf{y})$: scalar function $\mathbb{R}^d \rightarrow \mathbb{R}$
- Two types of expansions in series of this function
- “Local” or “Regular”called an “R expansion”
$$\Phi(\mathbf{x}_i, \mathbf{y}) = \sum_m^p a_m(\mathbf{x}_i, \mathbf{x}_*) R_m(\mathbf{y} - \mathbf{x}_*), \quad |\mathbf{y} - \mathbf{x}_*| < r < |\mathbf{x}_i - \mathbf{x}_*|$$
- “Far” or “Singular” Called an “S expansion”
$$\Phi(\mathbf{x}_i, \mathbf{y}) = \sum_m^p b_m(\mathbf{x}_i, \mathbf{x}_*) S_m(\mathbf{y} - \mathbf{x}_*), \quad |\mathbf{y} - \mathbf{x}_*| > r > |\mathbf{x}_i - \mathbf{x}_*|$$
 - \mathbf{x}_* : Point around which the function Φ is expanded in series

Notation

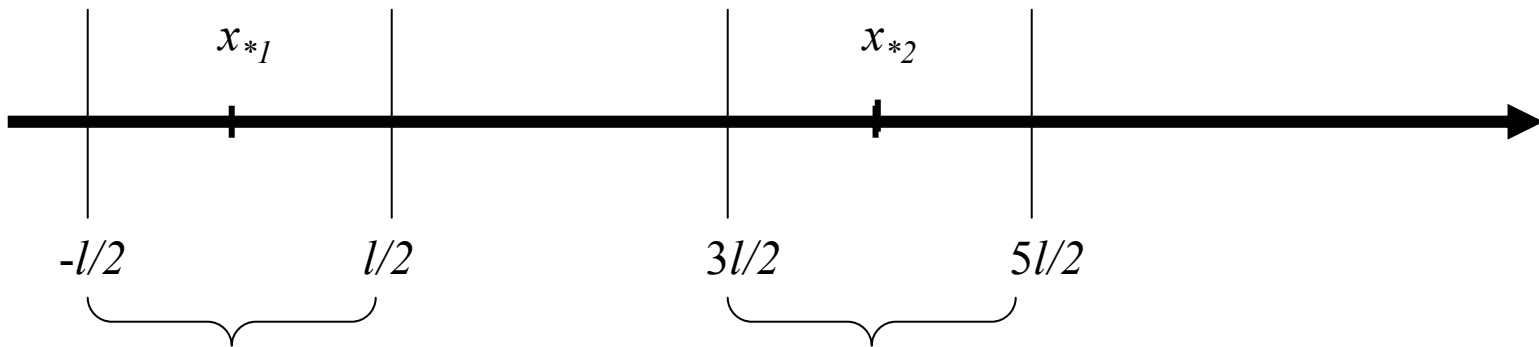
- Series are p truncated versions of infinite series
- Series have error bound $\varepsilon(p)$, and radius of convergence r
- For S expansion series is valid for all $|\mathbf{y}-\mathbf{x}_*| > r$
- If expansion is in terms of singular functions, then r must be large enough

Translation

- Take a function or equivalently a function expressed as a series expansion and express it in another coordinate system (reference frame)
- Function is the same on the common parts of domains of definition
- but is represented in different forms
 - $\Phi(\mathbf{x}_i, \mathbf{y})$
 - $\sum_m \mathbf{b}_m(\mathbf{x}_i, \mathbf{x}_{*1}) S_m(\mathbf{y} - \mathbf{x}_{*1})$
 - $\sum_m \mathbf{a}_m(\mathbf{x}_i, \mathbf{x}_{*2}) R_m(\mathbf{y} - \mathbf{x}_{*2})$ with $\mathbf{a}_m = (\mathbf{S}|\mathbf{R})_{mn} \mathbf{b}_m$

Homework 4

$$\sum_{n=1}^p b_n(x, x_{*1}) S_n(y, x_{*1})$$



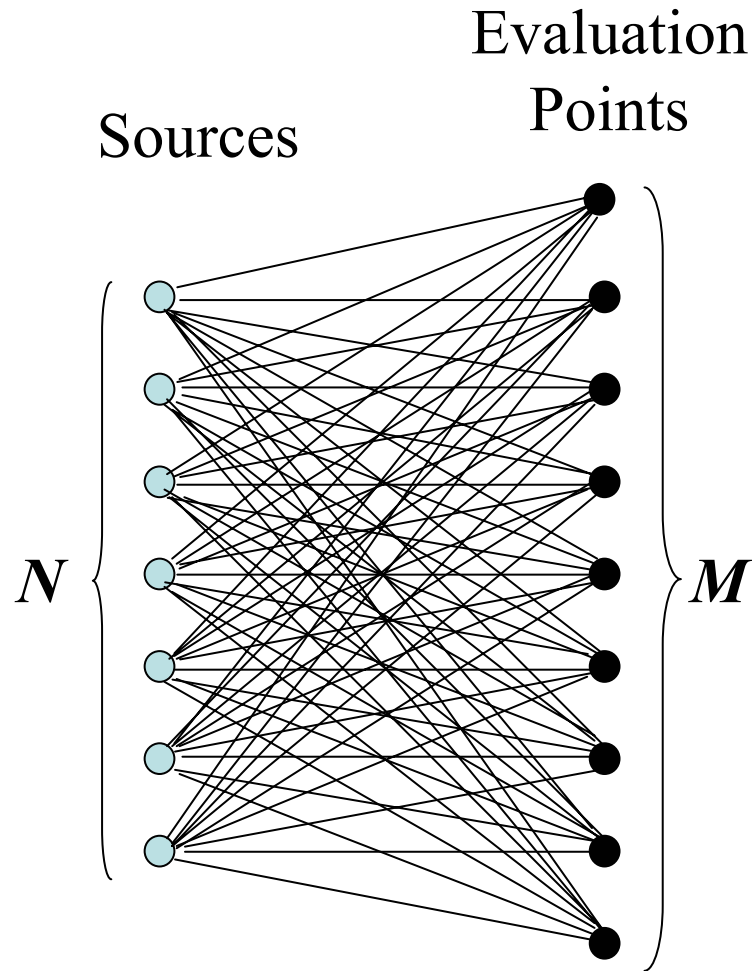
$$\sum_{m=1}^p a_m(x, x_{*2}) R_m(y - x_{*2})$$

$$\begin{aligned} & \sum_{n=1}^p b_n(x, x_{*1}) S_n(y, x_{*1}) \\ &= \sum_{n=1}^p b_n \left(\sum_{m=1}^p (S|R)_{mn}(t) R_m(y - x_{*2}) \right) \\ &= \sum_{m=1}^p a_m(x, x_{*2}) R_m(y - x_{*2}) \end{aligned}$$

Meaning of $(S|R)_{mn}$
Operator acting on S expansion coefficients with index n to produce R expansion coefficients

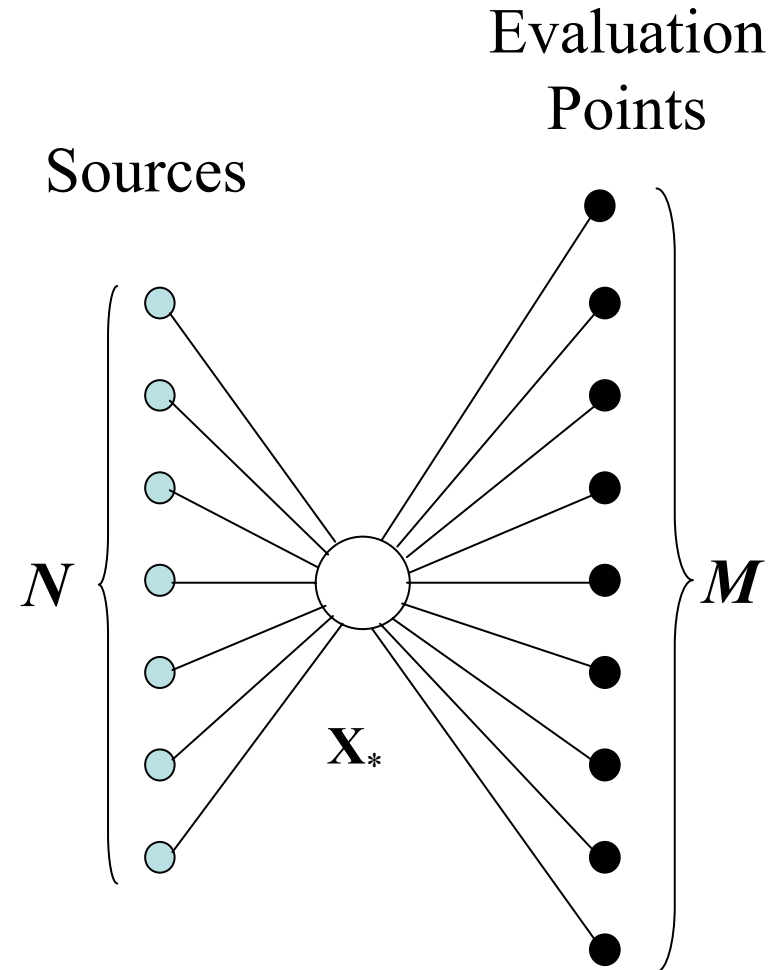
Middleman Algorithm

Standard algorithm



Total number of operations: $O(NM)$

Middleman algorithm



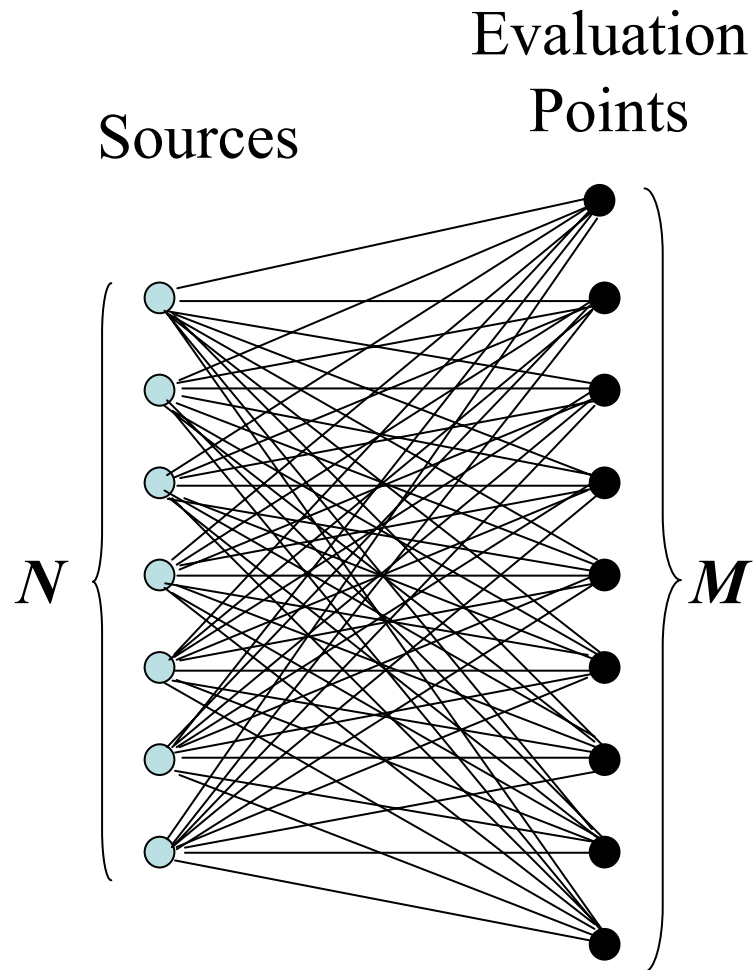
Total number of operations: $O(N+M)$

Four Key Stones of FMM

- Factorization
- Error
- Translation
- Grouping

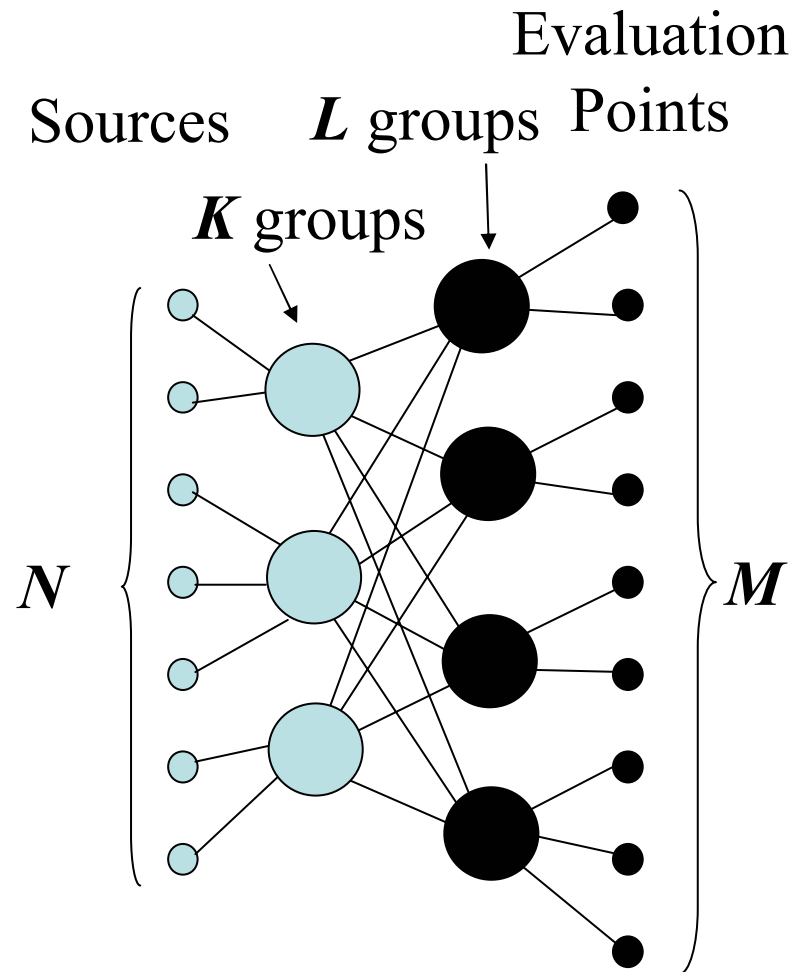
Idea of a Single Level FMM

Standard algorithm



Total number of operations: $O(NM)$

SLFMM



Total number of operations: $O(N+M+KL)$

SLFMM Algorithm

Step 1. Generate S-expansion coefficients
for each box

$$\Phi_1^{(n)}(\mathbf{x}) = \mathbf{C}^{(n)} \circ \mathbf{S}(\mathbf{x} - \mathbf{x}_c^{(n)}),$$
$$\mathbf{C}^{(n)} = \sum_{\mathbf{x}_i \in E_1(n,L)} u_i \mathbf{B}(\mathbf{x}_i, \mathbf{x}_c^{(n)}).$$

For $n \in \text{NonEmpty}$  loop over all non-empty boxes

Get $\mathbf{x}_c^{(n)}$, the center of the box;

$\mathbf{C}^{(n)} = \mathbf{0}$;

For $\mathbf{x}_i \in E_1(n)$  loop over all sources in the box

Get $\mathbf{B}(\mathbf{x}_i, \mathbf{x}_c^{(n)})$, the S-expansion coefficients
near the center of the box;

$\mathbf{C}^{(n)} = \mathbf{C}^{(n)} + u_i \mathbf{B}(\mathbf{x}_i, \mathbf{x}_c^{(n)})$;

End;

End;

Implementation can be different!
All we need is to get $\mathbf{C}^{(n)}$.

SLFMM Algorithm

Step 2. (S|R)-translate expansion coefficients

$$\Phi_3^{(n)}(\mathbf{y}) = \mathbf{D}^{(n)} \circ \mathbf{R}(\mathbf{x} - \mathbf{x}_c^{(n,l)}),$$
$$\mathbf{D}^{(n)} = \sum_{m \in I_3(n)} (\mathbf{S|R})(\mathbf{x}_c^{(n)} - \mathbf{x}_c^{(m)}) \mathbf{C}^{(m)}.$$

loop over all boxes
containing sources

For $n \in \text{NonEmptySource}$

Get $\mathbf{x}_c^{(n)}$, the center of the box;

$\mathbf{D}^{(n)} = \mathbf{0}$;

For $m \in I_3(n)$

loop over all non-empty boxes outside
The neighborhood of the n -th box

Get $\mathbf{x}_c^{(m)}$, the center of the box;

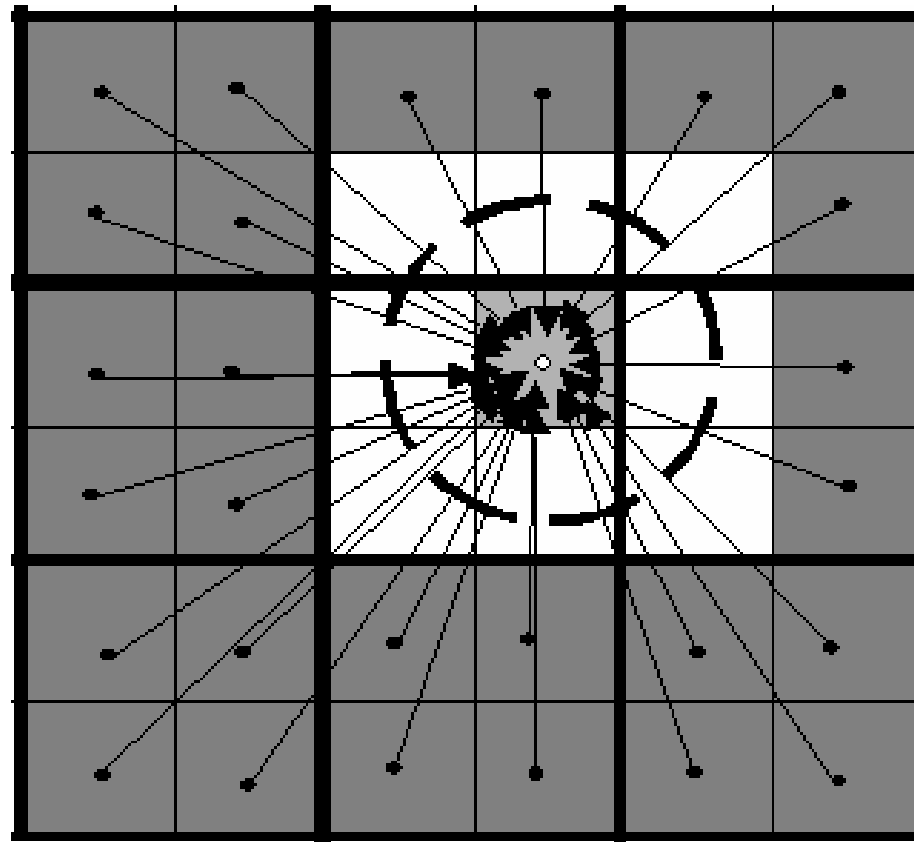
$\mathbf{D}^{(n)} = \mathbf{D}^{(n)} + (\mathbf{S|R})(\mathbf{x}_c^{(n)} - \mathbf{x}_c^{(m)}) \mathbf{C}^{(m)}$;

End;

End;

Implementation can be different!
All we need is to get $\mathbf{D}^{(n)}$.




S|R-translation



SLFMM Algorithm

Step 3. Final Summation

$$v_j = \Phi(\mathbf{y}_j) = \sum_{\mathbf{x}_i \in E_2(n)} \Phi(\mathbf{y}_j, \mathbf{x}_i) + \mathbf{D}^{(n)} \circ \mathbf{R}(\mathbf{y}_j - \mathbf{x}_c^{(n)}), \quad \mathbf{y}_j \in E_1(n).$$

For $n \in \text{NonEmptyEvaluation}$  loop over all boxes containing evaluation points
Get $\mathbf{x}_c^{(n)}$, the center of the box;
For $\mathbf{y}_j \in E_1(n)$  loop over all evaluation points in the box
 $v_j = \mathbf{D}^{(n)} \circ \mathbf{R}(\mathbf{y}_j - \mathbf{x}_c^{(n)});$
For $\mathbf{x}_i \in E_2(n)$  loop over all sources in the neighborhood of the n -th box
 $v_j = v_j + \Phi(\mathbf{y}_j, \mathbf{x}_i);$
End;
End;
End;

Implementation can be different!
All we need is to get v_j

Asymptotic Complexity of SLFMM

Assume that:

- By some magic we can easily find neighbors, and lists of points in each box.
- Translation is performed by straightforward $P \times P$ matrix-vector multiplication, where $P(p)$ is the total length of the translation vector. So the complexity of a single translation is $O(P^2)$.
- The source and evaluation points are distributed uniformly, and there are K boxes, with s source points in each box ($s=N/K$). We call s the *grouping* (or *clustering*) parameter.
- The number of neighbors for each box is $O(1)$.

Then Complexity is:

- For Step 1: $O(PN)$
- For Step 2: $O(P^2K^2)$
- For Step 3: $O(PM+Ms)$
- Total: $O(PN+ P^2K^2 +PM+Ms) =$
 $O(PN+ P^2K^2 +PM+MN/K)$

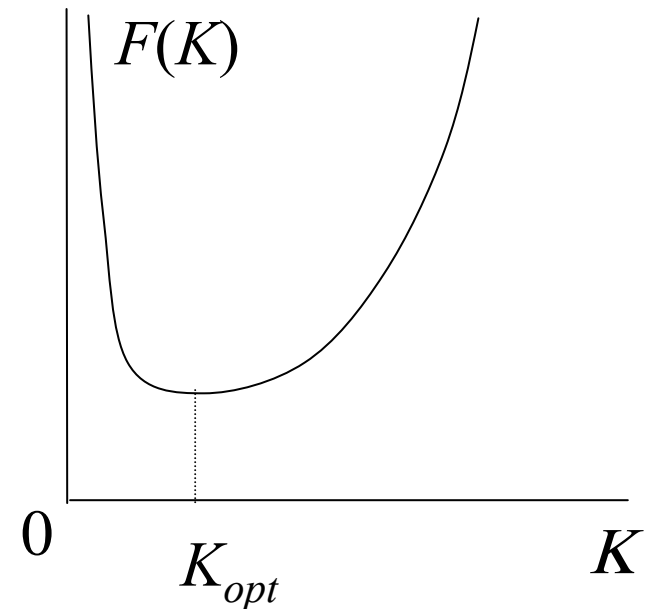
Selection of Optimal K (or s)

$$F(K) = PN + P^2K^2 + PM + PMN/K.$$

$$F'(K) = 2P^2K - PMN/K^2 = 0.$$

$$K_{opt} = \left(\frac{MN}{2P}\right)^{1/3} = O\left(\left(\frac{MN}{P}\right)^{1/3}\right).$$

$$s_{opt} = \frac{N}{K_{opt}} = \left(\frac{2PN^2}{M}\right)^{1/3} = O\left(\frac{PN^2}{M}\right)^{1/3}.$$



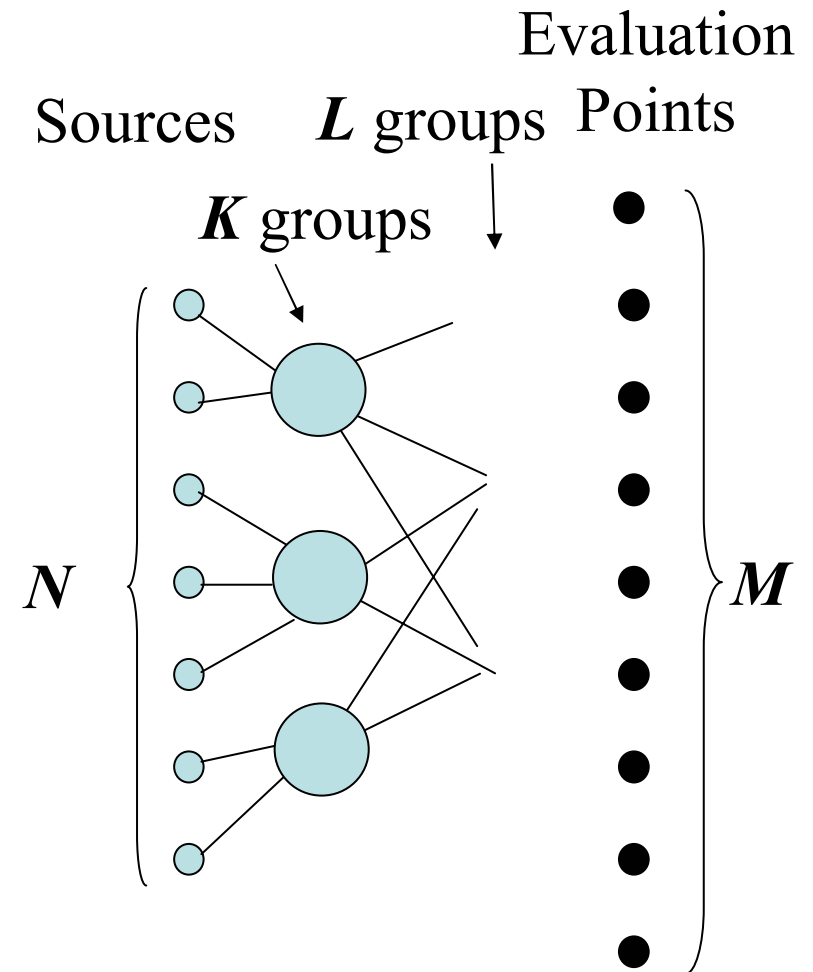
Complexity of Optimized SLFMM

$$\begin{aligned} F(K_{opt}) &= PN + P^2 \left(\frac{MN}{2P} \right)^{2/3} + PM + PMN \left(\frac{MN}{2P} \right)^{-1/3} \\ &= P(M + N) + (MN)^{2/3} O(P^{4/3}). \end{aligned}$$

At $K = K_{opt}$, and $M = O(N)$, the complexity of SLFMM is:

$$O(PN + P^{4/3} N^{4/3}) = O(P^{4/3} N^{4/3}).$$

SLFMM



Total number of operations: $O(N+M+KL)$

FMM CMSC 878R/AMSC 698R

Lecture 9

Outline

- Data Structures and FMM
- Hierarchical Space Subdivision
 - Binary Trees, Quad-trees, Oct-trees;
 - k-d trees;
 - 2^d -trees. Definitions.
- Hierarchical Numbering
- Spatial Ordering
 - Scaling.
 - Binary ordering.
 - Ordering in d-dimensions.
 - (to be continued on the next lecture)

Data Structures and FMM

- Spatial grouping techniques are employed in various versions of the FMM (SL_FMM, RML_FMM, AML_FMM, etc.)
- All major components of the FMM are interrelated: truncated factorization, error bounds, translation operations, and spatial grouping.

Data Structures and FMM (2)

- Since the complexity of FMM should not exceed $O(N^2)$ (at $M \sim N$), data organization should be provided for efficient numbering, search, and operations with these data.
- Some naive approaches can utilize search algorithms that result in $O(N^2)$ complexity of the FMM (and so they kill the idea of the FMM).
- In d -dimensions $O(M \log N)$ complexity for operations with data can be achieved.
- Some caution is needed while utilizing standard (library) routines on sets. If using such routines always check the complexity of the algorithm!
- Understanding of complexity of each FMM procedure is crucial. Normally it is worth to develop your own library for operations with FMM data (using standard routines as their complexity is satisfactory).

Data Structures and FMM (3)

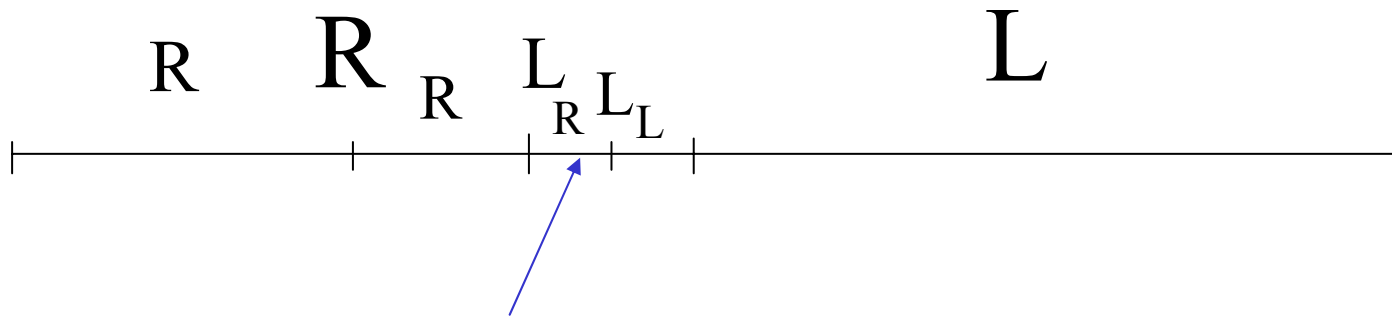
- Approaches include:
 - Data preprocessing
 - Sorting
 - Building lists (such as neighbor lists): requires memory, potentially can be avoided;
 - Building and storage of trees: requires memory, potentially can be avoided;
 - Operations with data during the FMM algorithm
 - Operations on data sets;
 - Search procedures.
- Preferable algorithms:
 - Avoid unnecessary memory usage;
 - Use fast (constant and logarithmic) search procedures;
 - Employ bitwise operations;
 - Can be parallelized.

Hierarchical Space Subdivision

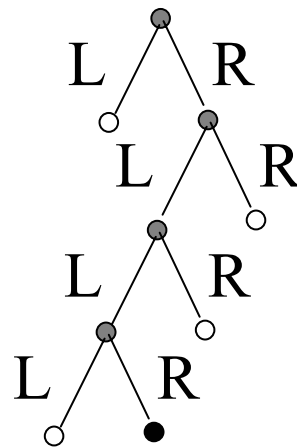
Historically:

- Binary trees (1D), Quad-trees (2D), Oct-trees (3D);
- For dimensions larger than 3 (sometimes for $d=2$ and 3 also) k-d trees.
- We will consider a concept of 2^d -tree:
 - $d=1$ – binary;
 - $d=2$ – quadtree;
 - $d=3$ – octtree;
 - $d=4$ – hexatree;
 - and so on..

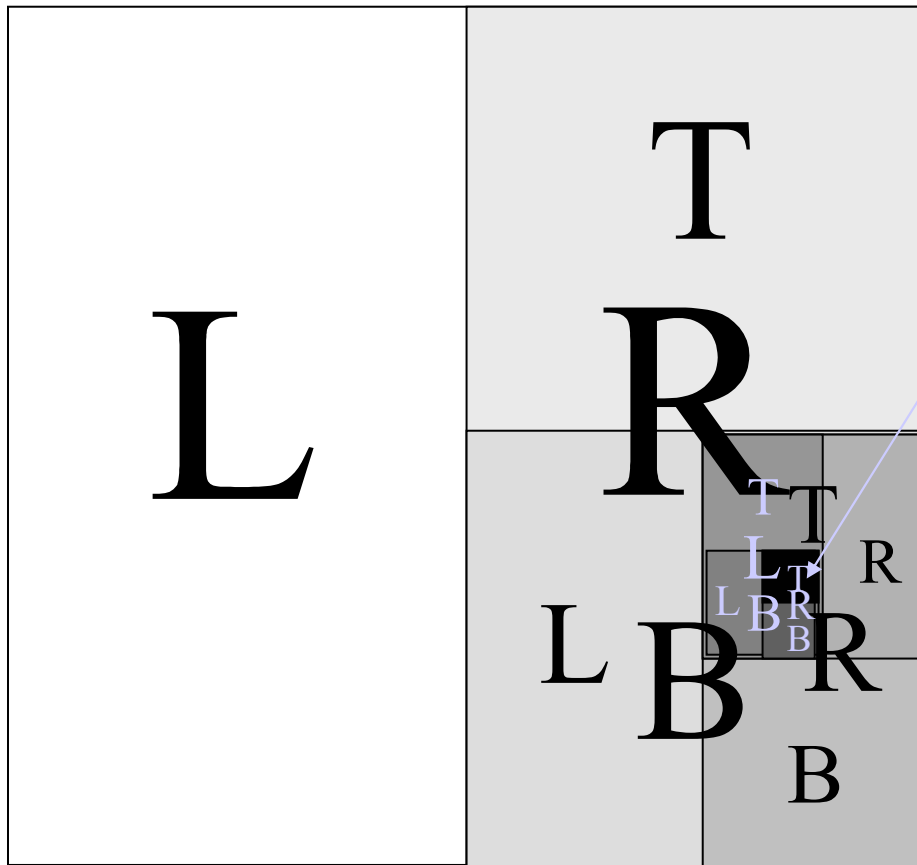
Binary Trees



Target segment coordinates:
RLLR

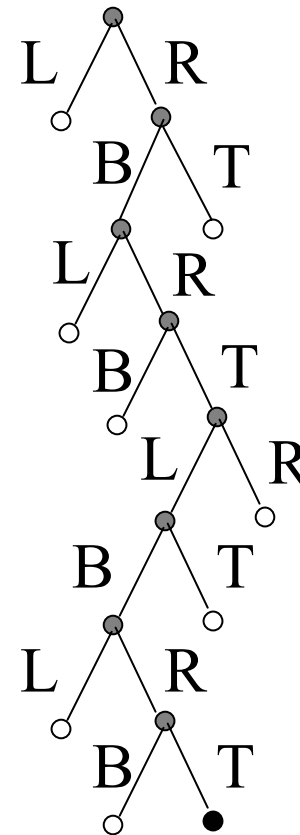


What are k-d trees?



Target box coordinates:

RBRTLBR

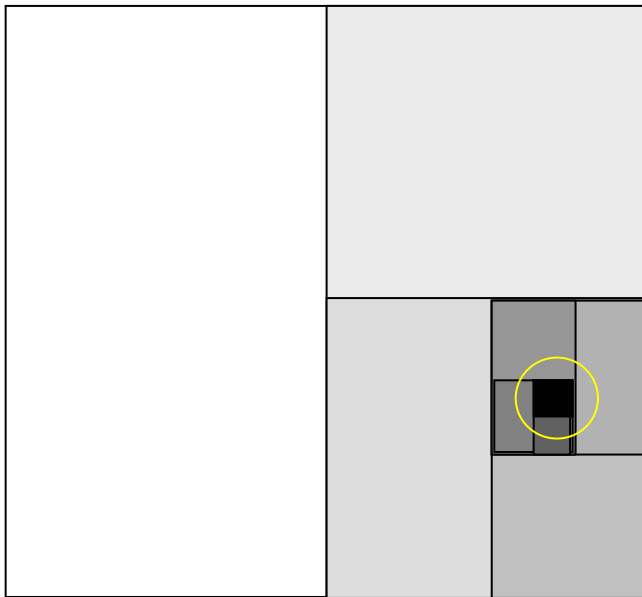


Like
a binary
tree!

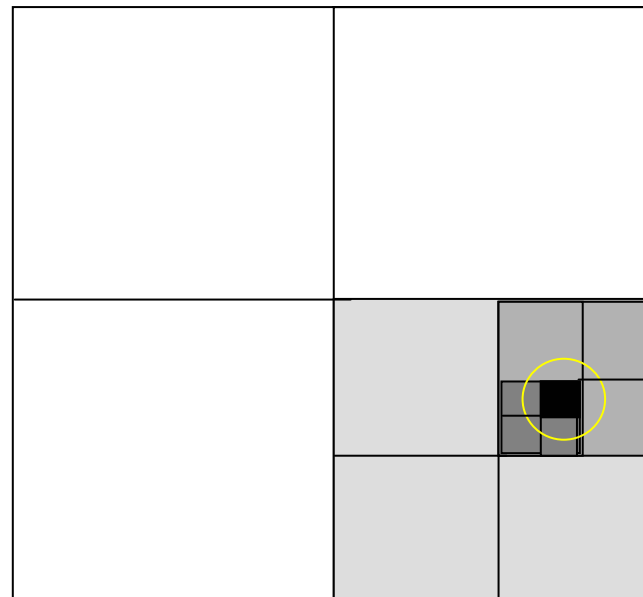
Why 2^d -trees?

It is possible to use k-d trees in the FMM
(while in this course we mainly focus on 2^d -trees)

k-d tree



2^d -tree



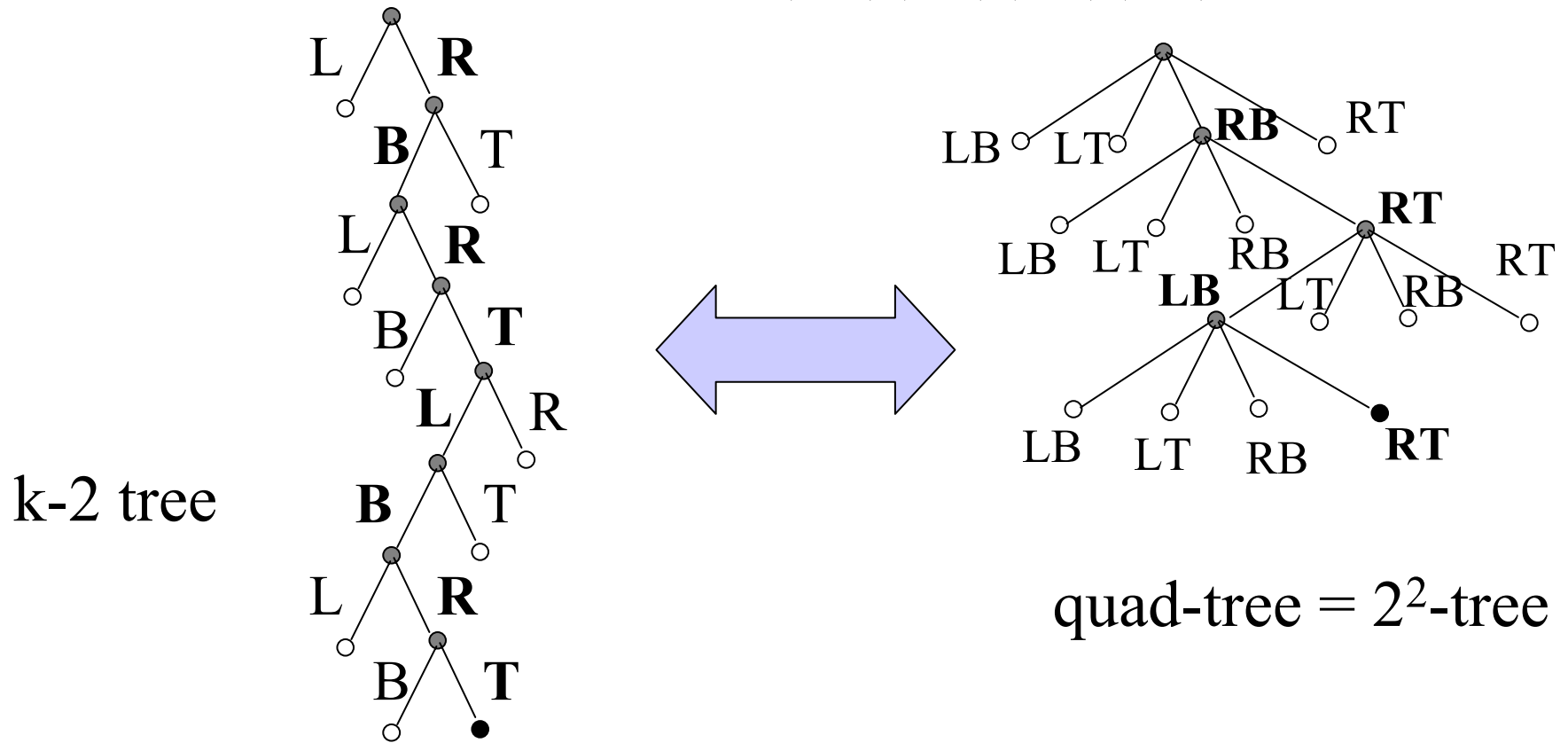
Simply, because

It is convenient in the FMM to enclose a cube into a sphere with easy determination of the neighborhood of similar boxes.

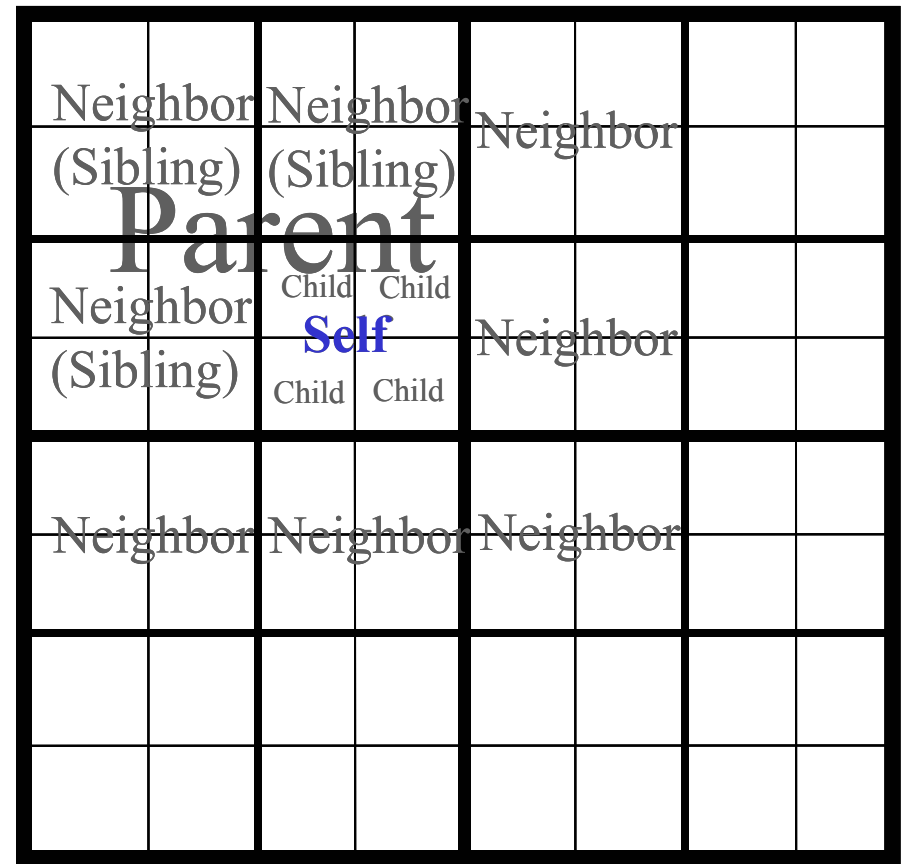
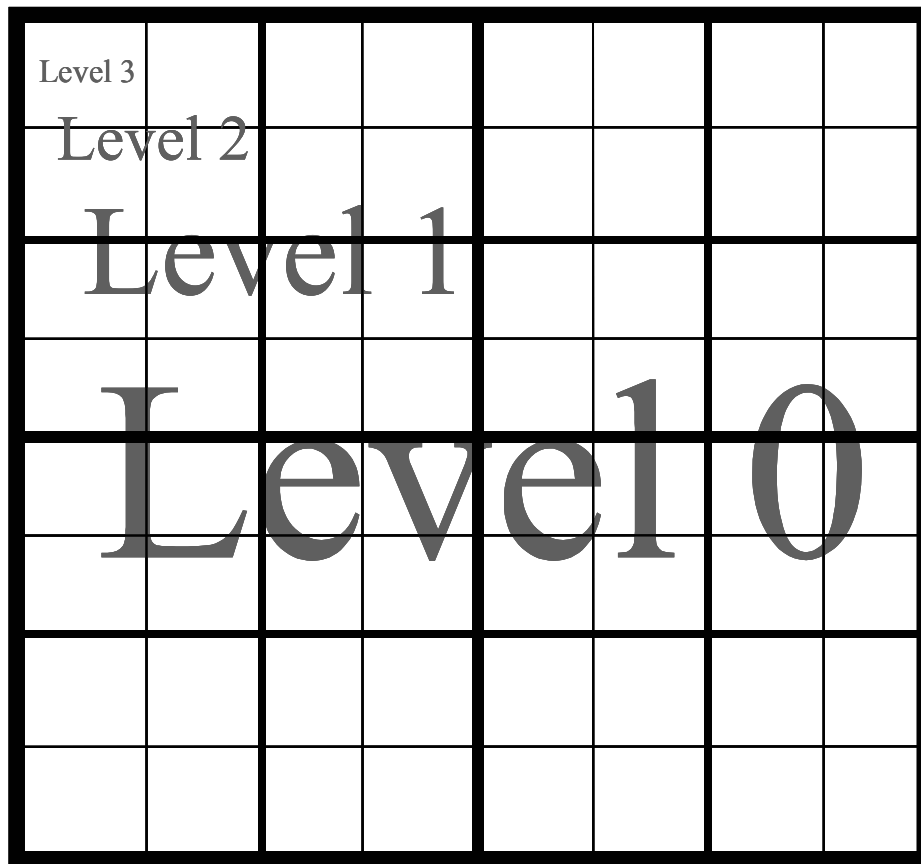
It is easy to convert k-d tree into 2^d -tree and back (for cubic boxes)

Target box coordinates:

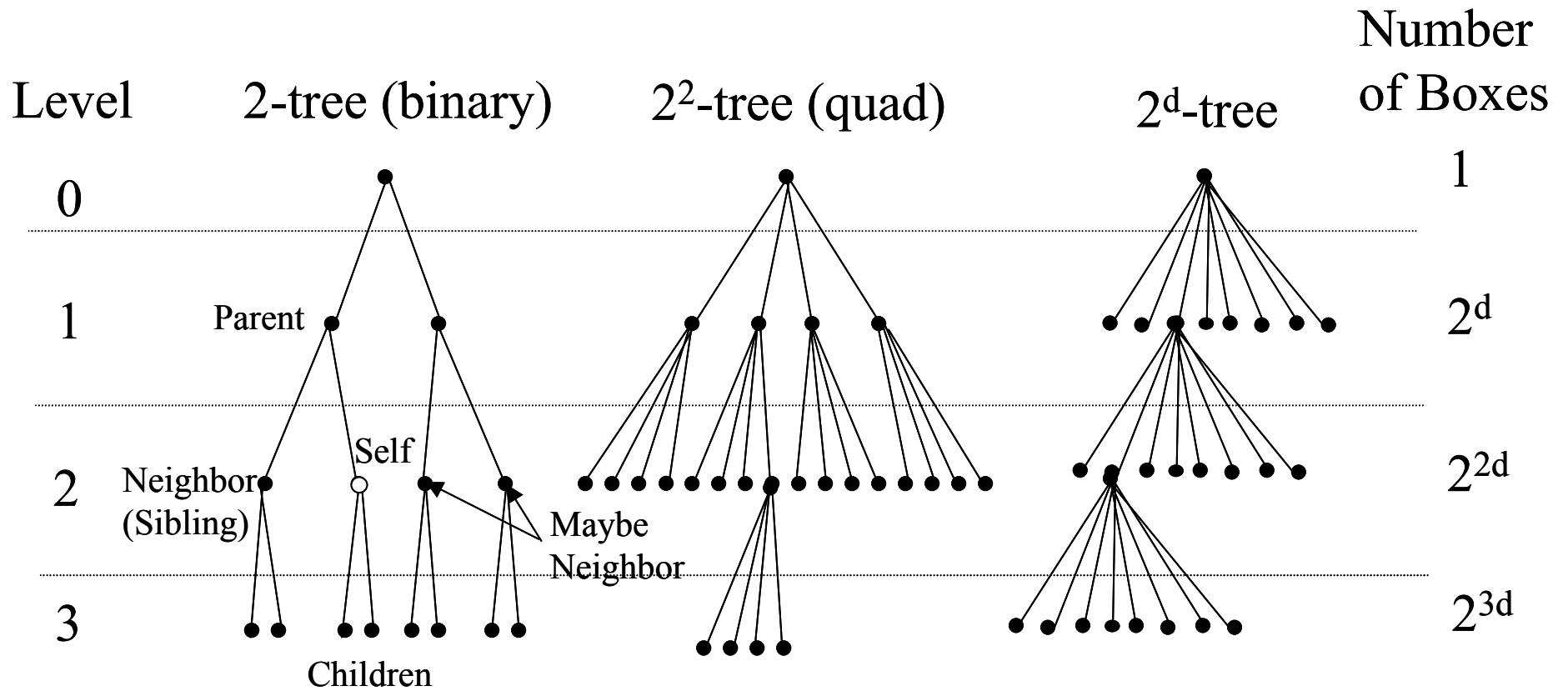
RBRTL BRT \rightarrow (RB)(RT)(LB)(RT)



Hierarchy in 2^d -tree



2^d-trees



Hierarchical Numbering in 2^d -trees.

Numbering String.

1	3	1	3	1	3	1	3
0	1	2	0	3	2	0	3
1	3	1	3	1	3	1	3
0	2	0	2	0	2	0	2
1	3	1	3	1	3	1	3
0	1	2	0	3	2	0	3
1	3	1	3	1	3	1	3
0	2	0	2	0	2	0	2

Numbering in quad-tree

The large black box has the numbering string (2,3);

The small black box has the numbering string (3,1,2);

$$\text{Number} = (N_1, N_2, \dots, N_l), \quad N_j = 0, \dots, 2^d - 1, \quad j = 1, \dots, l,$$

Numbering string

Hierarchical Numbering in 2^d -trees.

Number at the Level.

1	3	1	3	1	3	1	3
0	1	2	0	3	2	0	1
1	0	3	1	2	3	1	3
0	2	0	2	2	0	0	2
1	1	3	1	3	1	3	3
0	1	2	0	3	2	0	1
1	0	3	1	2	3	1	3
0	2	0	2	2	0	2	2

← Numbering in quad-tree

The large black box has the numbering string (2,3). So its number is $23_4 = 11_{10}$.

The small black box has the numbering string (3,1,2). So its number is $312_4 = 54_{10}$.

In general: Number at level l is:

$$\text{Number} = (2^d)^{l-1} \cdot N_1 + (2^d)^{l-2} \cdot N_2 + \dots + 2^d \cdot N_{l-1} + N_l.$$

Hierarchical Numbering in 2^d -trees. Universal Number.

1	3	1	3	1	3	1	3
0	1	2	0	3	2	0	3
1	3	1	3	1	3	1	3
0	2	0	2	0	2	0	2
1	3	1	3	1	3	1	3
0	1	2	0	3	2	0	3
1	3	1	3	1	3	1	3
0	2	0	2	0	2	0	2

← Numbering in quad-tree

The large black box has the numbering string (2,3). So its number is $23_4 = 11_{10}$ at level 2

The small gray box has the numbering string (0,2,3). So its number is $23_4 = 11_{10}$ at level 3.

In general: Universal number is a pair:

$$\text{UniversalNumber} = (\text{Number}, l)$$

← This number at this level

Parent Number

Parent numbering string:

$$\text{Parent}(N_1, N_2, \dots, N_{l-1}, N_l) = (N_1, N_2, \dots, N_{l-1}).$$

Parent number:

$$\text{Parent}(\text{Number}) = (2^d)^{l-2} \cdot N_1 + (2^d)^{l-3} \cdot N_2 + \dots + N_{l-1}.$$

1	3	1	3	1	3	1	3
0	2	0	3	0	2	0	3
1	3	1	3	1	3	1	3
0	2	0	2	0	0	0	2
1	3	1	3	1	3	1	3
0	2	0	3	0	1	2	0
1	3	1	3	1	3	1	3
0	2	0	2	0	0	0	2

Parent number does not depend on the level of the box! E.g. in the quad-tree at any level

$$\text{Parent}(11_{10}) = \text{Parent}(23_4) = 2_4 = 2_{10}.$$

Parent's universal number:

$$\text{Parent}((\text{Number}, l)) = (\text{Parent}(\text{Number}), l-1).$$

Algorithm to find the parent number:

$$\text{Parent}(\text{Number}) = \lfloor \text{Number} / 2^d \rfloor$$

For box #23₄ (gray or black) the parent box number is 2₄.

Children Numbers

Children numbering strings:

$$\text{Children}(N_1, N_2, \dots, N_{l-1}, N_l) = \{(N_1, N_2, \dots, N_{l-1}, N_l, N_{l+1})\}, \quad N_{l+1} = 0, \dots, 2^d - 1.$$

Children numbers:

$$\exists \text{Children}(\text{Number}) = \{(2^d)^l \cdot N_1 + (2^d)^{l-1} \cdot N_2 + \dots + (2^d) \cdot N_l + N_{l+1}\}, \quad N_{l+1} = 0, \dots, 2^d - 1.$$

**Children numbers do not depend on
the level of the box! E.g. in the quad-tree
at any level:**

$$\text{Children}(11_{10}) = \text{Children}(23_4) = \{230_4, 231_4, 232_4, 233_4\} = \{44_{10}, 45_{10}, 46_{10}, 47_{10}\}$$

Children universal numbers:

$$\text{Children}((\text{Number}, l)) = (\text{Children}(\text{Number}), l + 1).$$

Algorithm to find the children numbers:

$$\text{Children}(\text{Number}) = \{2^d \cdot \text{Number} + j\}, \quad j = 0, \dots, 2^d - 1,$$

A couple of examples:

Problem: Using the above numbering system and decimal numbers find parent box number for box #5981 in oct-tree.

Solution: Find the integer part of division of this number by 8. $[5981/8] = 747$.

Answer: #747.

Problem: Using the above numbering system and decimal numbers find children box numbers for box #100 in oct-tree.

Solution: Multiply this number by 8 and add numbers from 0 to 7.

Answer: ##800, 801, 802, 803, 804, 805, 806, 807.

Can it be even faster?

YES!

USE BITSHIFT PROCEDURES!

(HINT: Multiplication and division by 2^d
are equivalent to d -bit shift.)

Matlab Program for Parent Finding

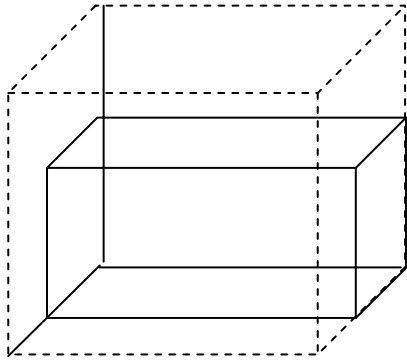
```
p = bitshift(n,-d);
```

Spatial Ordering

These algorithms of Parent and Children finding are beautiful ($O(1)$), but how about neighbor finding?

Also we need to find box center coordinates...

The answer is SPATIAL ORDERING.



Scaling

In physics-based problems ($d = 1, 2, 3$) we usually have symmetry of directions and can enclose that box to a cube of size $D \times \dots \times D$, where

$$D = \max_d D_d,$$

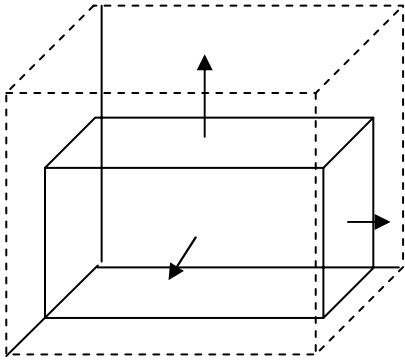
and the corner that has minimum values of Cartesian coordinates:

$$\mathbf{x}_{\min} = (x_{1,\min}, \dots, x_{d,\min}).$$

This cube then can be mapped to the unit cube $[0, 1] \times \dots \times [0, 1]$ by the shift of the origin and scaling:

$$\bar{\mathbf{x}} = \frac{\mathbf{x} - \mathbf{x}_{\min}}{D},$$

where \mathbf{x} are true Cartesian coordinates of any point in the cube, and $\bar{\mathbf{x}}$ are normalized coordinates of such a point.



Scaling (2)

In case if a 2^d -tree data structure is applied for parametric studies, where each parameter has its own scale D_j the mapping of the original box

$$[x_{1,\min}, x_{1,\max}] \times \dots \times [x_{d,\min}, x_{d,\max}], \quad x_{j,\max} - x_{j,\min} = D_j, \quad j = 1, \dots, d$$

to the unit cube $[0, 1] \times \dots \times [0, 1]$ can be also easily performed by scaling in each dimension as:

$$\bar{x}_j = \frac{x_j - x_{j,\min}}{D_j}, \quad j = 1, \dots, d, \quad \bar{\mathbf{x}} = (\bar{x}_1, \dots, \bar{x}_d).$$

Further we will work only with a unit cube, assuming that such scaling is performed and if necessary any point \mathbf{x} in the original d -dimensional space can be found from given $\bar{\mathbf{x}} \in [0, 1] \times \dots \times [0, 1]$ and back, for any \mathbf{x} its image $\bar{\mathbf{x}}$ can be found.

When scaling like this, don't forget about deformation of the domains for your R and S expansions!

Binary Ordering (1)

$d = 1$:

All $\bar{x} \in [0, 1]$ naturally ordered and can be represented in decimal system as

$$\bar{x} = (0.a_1a_2a_3\dots)_{10}, \quad a_j = 0, \dots, 9; \quad j = 1, 2, \dots$$

Note that the point $\bar{x} = 1$ can be written not only $\bar{x} = 1.0000\dots$, but also as

$$\bar{x} = 1 = (0.999999\dots)_{10}$$

We also can represent any point $\bar{x} \in [0, 1]$ in binary system as

$$\bar{x} = (0.b_1b_2b_3\dots)_2, \quad b_j = 0, 1; \quad j = 1, 2, \dots$$

By the same reasons as for decimal system the point $\bar{x} = 1$ can be written as

$$\bar{x} = 1 = (0.111111\dots)_2.$$

Binary Ordering (2)

Finding the number of the box containing a given point

Level	Box Size (dec)	Box Size (bin)
0	1	1
1	0.5	0.1
2	0.25	0.01
3	0.125	0.001
...

Level 1:

$$(0.0b_1b_2b_3\dots)_2 \in \text{Box}((0)), \quad (0.1b_1b_2b_3\dots)_2 \in \text{Box}((1)), \quad \forall b_j = 0, 1; \quad j = 1, 2, \dots,$$

Level 2:

$$(0.00b_1b_2b_3\dots)_2 \in \text{Box}((0,0)), \quad (0.01b_1b_2b_3\dots)_2 \in \text{Box}((0,1)),$$

$$(0.10b_1b_2b_3\dots)_2 \in \text{Box}((1,0)), \quad (0.11b_1b_2b_3\dots)_2 \in \text{Box}((1,1)),$$

$$\forall b_j = 0, 1; \quad j = 1, 2, \dots,$$

We use numbering strings !

Level l :

$$(0.N_1N_2\dots N_l b_1b_2b_3\dots)_2 \in \text{Box}((N_1, N_2, \dots, N_l)), \quad \forall b_j = 0, 1; \quad j = 1, 2, \dots$$

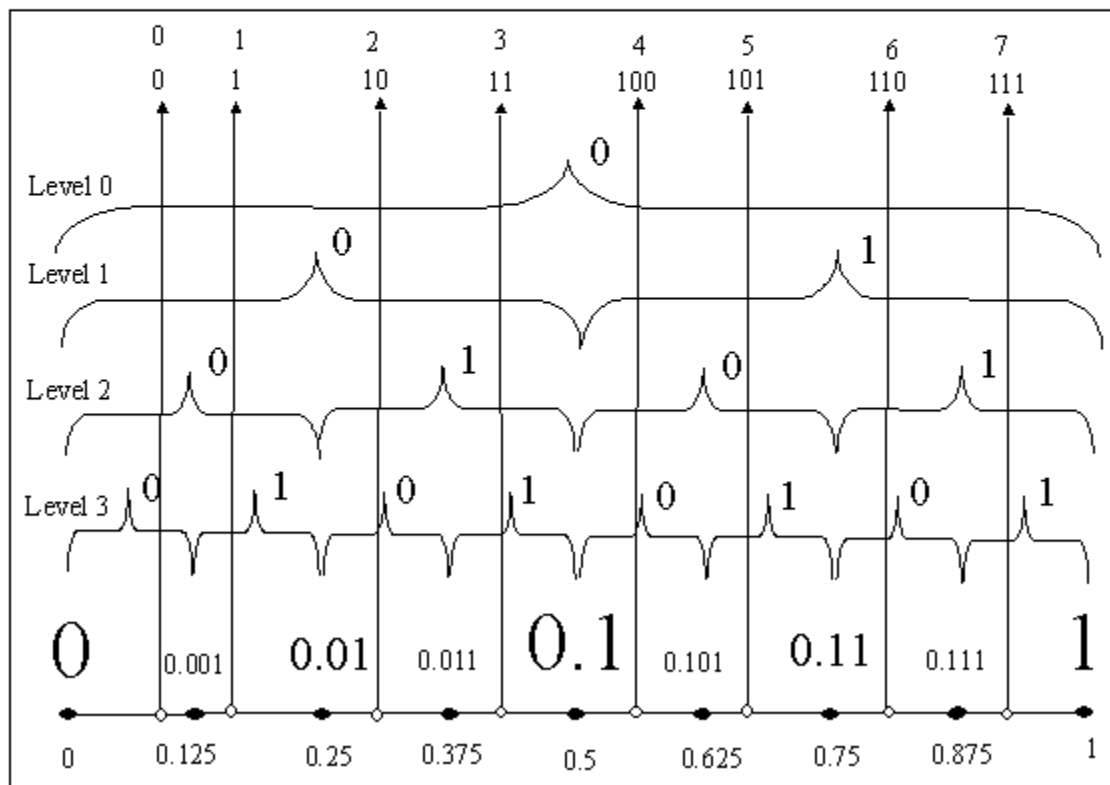
Binary Ordering (3)

Finding the number of the box containing a given point (2)

$$(0.N_1N_2\dots N_l b_1 b_2 b_3 \dots)_2 \rightarrow (N_1 N_2 \dots N_l . b_1 b_2 b_3 \dots)_2; \quad N_1 N_2 \dots N_l = [(N_1 N_2 \dots N_l . b_1 b_2 b_3 \dots)_2].$$

$$(Number, l) = [2^l \cdot \bar{x}].$$

This is an algorithm for finding of the box number at level l (!)



Binary Ordering (4)

Finding the center of a given box.

For box number $Number$ at level l the left boundary can be found by l -bit shift:

$$Number = (N_1N_2\dots N_l)_2 \rightarrow (0.N_1N_2\dots N_l)_2,$$

Add 1 as an extra digit (half of the box size), so we have for the center of the box at level l :

$$\bar{x}_c(Number, l) = (0.N_1N_2\dots N_l1)_2.$$

This procedure also can be written in the form that does not depend on the counting system:

$$\bar{x}_c(Number, l) = 2^{-l} \cdot Number + 2^{-l-1} = 2^{-l} \cdot (Number + 2^{-1}).$$

since addition of one at position $l+1$ after the point in the binary system is the same as addition of 2^{-l-1} .

Problem: Find the center of box #31 (decimal) at level 5 of the binary tree.

Solution: We have $\bar{x}_c(31, 5) = 2^{-5} \cdot (31 + 0.5) = 0.984375$.

Answer: 0.984375.

**This is the algorithm!**

Binary Ordering (5)

Neighbor finding

In the binary tree each box has 2 neighbors, except the boxes that have boundaries $\bar{x} = 0$ and $\bar{x} = 1$. The centers of the neighbor boxes:

$$\bar{x}_c(\text{Neighbor}((\text{Number}, l))) = \bar{x}_c(\text{Number}, l) \pm 2^{-l}.$$

In the binary form:

The size of the box at level l

$$\bar{x}_c(\text{Neighbor}((\text{Number}, l))) = (0.N_1N_2\dots N_l1)_2 \pm \left(\underbrace{0.0\dots01}_l \right)_2,$$

To find the number of the neighbor box we can then use the above algorithm for determining the number of the box for point \bar{x}_c :

$$\text{Neighbor}((\text{Number}, l)) = [N_1N_2\dots N_l.1 \pm 1] = N_1N_2\dots N_l \pm 1 = \text{Number} \pm 1.$$

If the neighbor number at level l equal 2^l or -1 we drop this box from the neighbor list.

Problem: Find all neighbors of box #31 (decimal) at level 5 of the binary tree.

Solution: The neighbors should have numbers $31 - 1 = 30$ and $31 + 1 = 32$. However, $32 = 2^5$, which exceeds the number allowed for this level. Thus, only box #30 is the neighbor.

Answer: #30.

This is the algorithm!

Ordering in d -dimensions (1). Bit Interleaving.

Coordinates of a point $\bar{\mathbf{x}} = (\bar{x}_1, \dots, \bar{x}_d)$ in the d -dimensional unit cube can be represented in binary form

$$\bar{x}_k = (0.b_{k1}b_{k2}b_{k3}\dots)_2, \quad b_{kj} = 0, 1; \quad j = 1, 2, \dots, \quad k = 1, \dots, d.$$

Instead of having d numbers characterizing each point we can form a single binary number that represent the same point by ordered mixing of the digits in the above binary representation (this is also called *bit interleaving*), so we can write:

$$\bar{\mathbf{x}} = (0.b_{11}b_{21}\dots b_{d1}b_{12}b_{22}\dots b_{d2}\dots b_{1j}b_{2j}\dots b_{dj}\dots)_2.$$

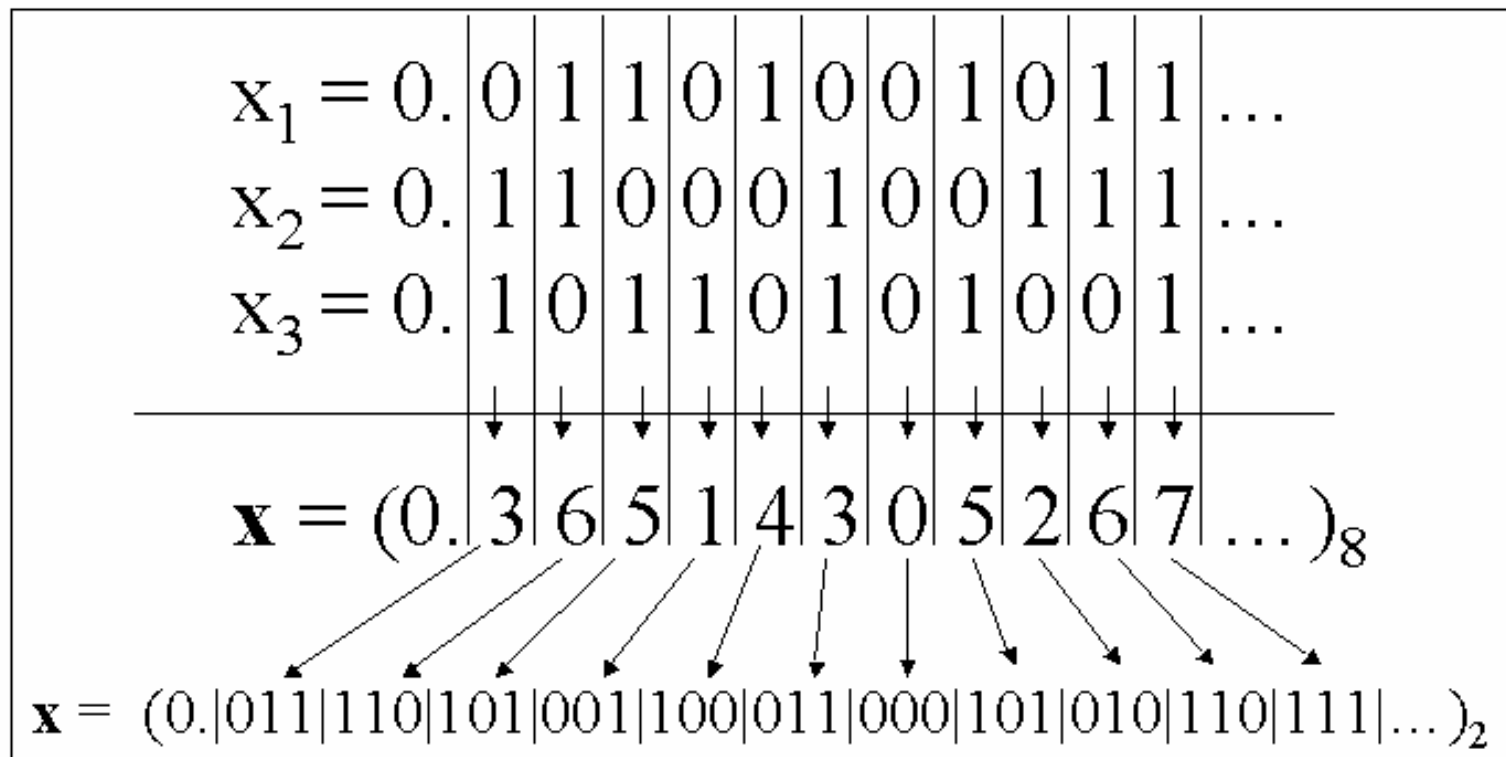
This number can be rewritten in the system with base 2^d :

$$\bar{\mathbf{x}} = (0.N_1N_2N_3\dots N_j\dots)_{2^d}, \quad N_j = (b_{1j}b_{2j}\dots b_{dj})_2, \quad j = 1, 2, \dots, \quad N_j = 0, \dots, 2^d - 1.$$

This maps $\mathbf{R}^d \rightarrow \mathbf{R}$, where coordinates are ordered naturally!

Ordering in d -dimensions (2). Bit Interleaving (2). Example.

Consider 3-dimensional space, and an oct-tree.

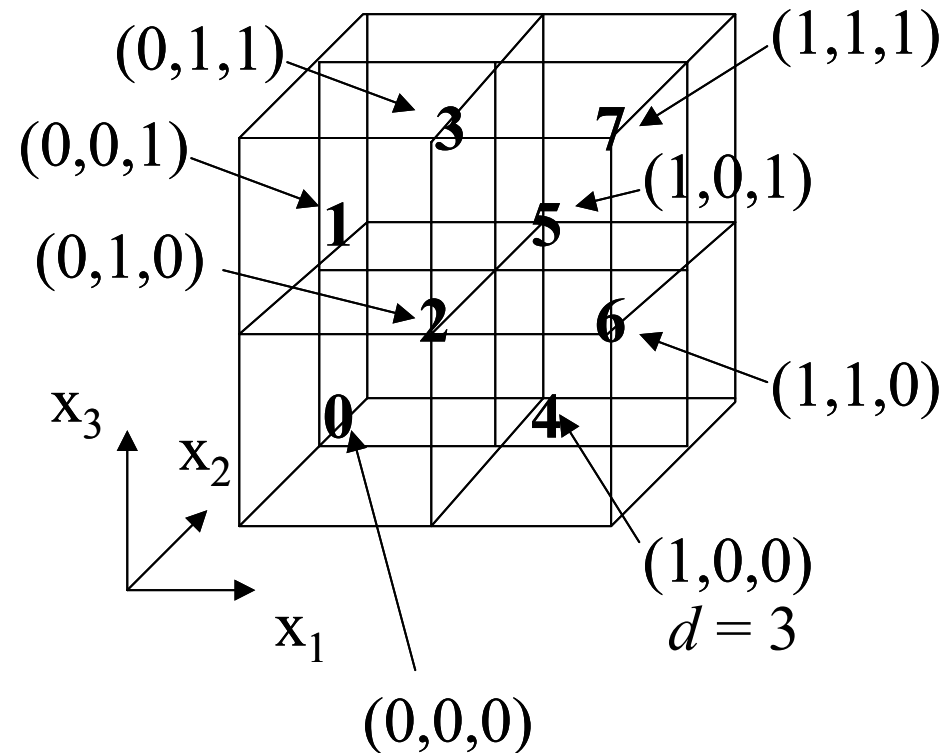
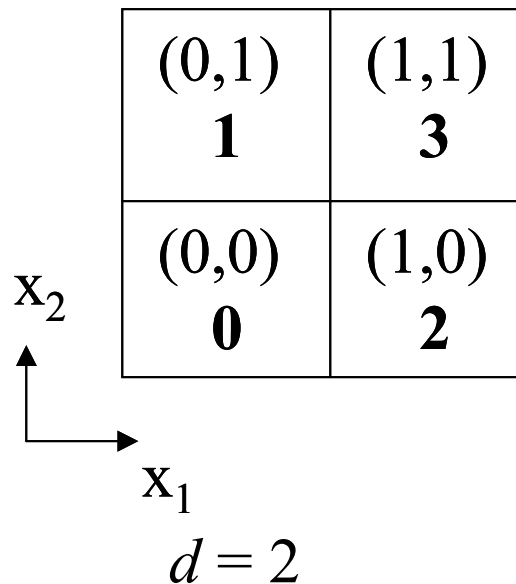


Ordering in d -dimensions (3). Convention for Children Ordering.

Any binary string of length d can be converted into a single number (binary or in some other counting system, e.g. with the base 2^d):

$$(b_1, b_2, \dots, b_d) \rightarrow (b_1 b_2 \dots b_d)_2 = N_{2^d}.$$

This provides natural numbering of 2^d children of the box.'



Ordering in d -dimensions (4). Finding the number of the box containing a given point.

Level 1:

$$\bar{x} = (0.b_{11}b_{21}\dots b_{d1}b_{12}b_{22}\dots b_{d2}\dots b_{1j}b_{2j}\dots b_{dj}\dots)_2 \in \text{Box}((b_{11}b_{21}\dots b_{d1})_2) = \text{Box}((N_1)_{2^d}),$$

Let us use 2^d -based counting system. Then we can find the box containing a given point at Level l :

$$(0.N_1N_2\dots N_l c_1c_2c_3\dots)_{2^d} \in \text{Box}((N_1, N_2, \dots, N_l)_{2^d}), \quad \forall c_j = 0, \dots, 2^d - 1; \quad j = 1, 2, \dots$$

Therefore to find the number of the box at level l to which the given point belongs we need simply shift the 2^d number representing this point by l positions and take the integer part of this number:

$$(0.N_1N_2\dots N_l c_1c_2c_3\dots)_{2^d} \rightarrow (N_1N_2\dots N_l.c_1c_2c_3\dots)_{2^d}; \quad N_1N_2\dots N_l = [(N_1N_2\dots N_l.b_1b_2b_3\dots)_{2^d}].$$

Ordering in d -dimensions (5). Finding the number of the box containing a given point (2). Algorithm and Example.

This procedure also can be performed in binary system by $d \cdot l$ bit shift:

$$(0.b_{11}b_{21}\dots b_{d1}b_{12}b_{22}\dots b_{d2}\dots b_{1l}b_{2l}\dots b_{dl}b_{\dots})_2 \rightarrow (b_{11}b_{21}\dots b_{d1}b_{12}b_{22}\dots b_{d2}\dots b_{1l}b_{2l}\dots b_{dl}.b_{\dots})_2;$$

$$Number = (b_{11}b_{21}\dots b_{d1}b_{12}b_{22}\dots b_{d2}\dots b_{1l}b_{2l}\dots b_{dl})_2.$$

In arbitrary counting system:

$$(Number, l) = \lceil 2^{dl} \cdot \bar{x} \rceil.$$

Problem: Find decimal numbers of boxes at levels 3 and 5 of the oct-tree containing point $\bar{x} = (0.7681, 0.0459, 0.3912)$.

Solution: First we convert the coordinates of the point to binary format, where we can keep only 5 digits after the point (maximum level is 5), so $\bar{x} = (0.11000, 0.00001, 0.01100)_2$. Second, we form a single mixed number $\bar{x} = 0.100101001000010_2$. Performing $3 \cdot 3 = 9$ bit shift and taking integer part we have $(Number, 3) = 100101001_2 = 297$. Performing $3 \cdot 5 = 15$ bit shift we obtain $(Number, 5) = 100101001000010_2 = 19010$.

Answer: #297 and #19010.

Ordering in d -dimensions (6).

Decomposition of the box number into coordinate numbers

Convert the box number at level l into binary form

$$Number = (b_{11}b_{21}\dots b_{d1}b_{12}b_{22}\dots b_{d2}\dots b_{1l}b_{2l}\dots b_{dl})_2.$$

Then we decompose this number to d numbers that will represent d coordinates:

$$Number_1 = (b_{11}b_{12}\dots b_{1l})_2.$$

$$Number_2 = (b_{21}b_{22}\dots b_{2l})_2.$$

...

$$Number_d = (b_{d1}b_{d2}\dots b_{dl})_2.$$

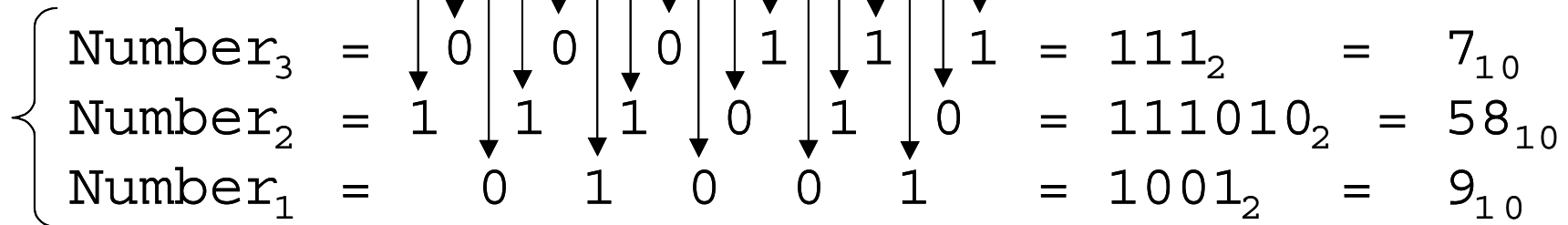
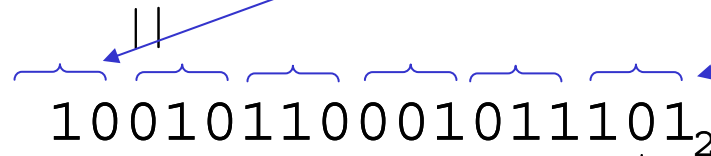
Ordering in d -dimensions (7).

Decomposition of the box number into coordinate numbers (2). Example.

Number = 76893_{10}

It is OK that the first group is incomplete

To break the number into groups of d bits start from the last digit!



Ordering in d -dimensions (8). Finding the center of a given box.

Coordinates of the box center in binary form are

$$\bar{x}_{k,c}(Number, l) = (0.b_{k1}b_{k2}\dots b_{kl}1)_2, \quad k = 1, \dots, d.$$

or in the form that does not depend on the counting system:

$$\bar{x}_{k,c}(Number, l) = 2^{-l} \cdot Number_k + 2^{-l-1} = 2^{-l} \cdot \left(Number_k + \frac{1}{2} \right), \quad k = 1, \dots, d.$$

Problem: Find the center of box #533 (decimal) at level 5 of the oct-tree.

Solution: Converting this number to the bit string we have $533_{10} = 1000010101_2$.

Retrieving the digits of three components from the last digit of this number we obtain:

$Number_3 = 1001_2 = 9_{10}$, $Number_2 = 10_2 = 2_{10}$, $Number_1 = 1_2 = 1_{10}$. We have then
 $\bar{x}_{1,c}(533, 5) = 2^{-5} \cdot (1 + 0.5) = 0.04875$, $\bar{x}_{2,c}(533, 5) = 2^{-5} \cdot (2 + 0.5) = 0.078125$,
 $\bar{x}_{3,c}(533, 5) = 2^{-5} \cdot (9 + 0.5) = 0.296875$.

Answer: $\bar{\mathbf{x}}_c = (0.04875, 0.078125, 0.296875)$.