Outline

• Ordering in d-dimensions
  – Bit deinterleaving
  – Neighbor and other search algorithms

• Spatial data structuring
  – Separability of space
  – Threshold level of subdivision
  – Data sorting
  – Binary search
  – About some operations on sets (union, difference, intersection)
Bit Deinterleaving

Convert the box number at level \( l \) into binary form

\[
Number = (b_{1_1}b_{2_1}...b_{d_1}b_{1_2}b_{2_2}...b_{d_2}...b_{1_l}b_{2_l}...b_{d_l})_2.
\]

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

\[
Number_1 = (b_{1_1}b_{1_2}...b_{1_l})_2.
\]

\[
Number_2 = (b_{2_1}b_{2_2}...b_{2_l})_2.
\]

\[
...
\]

\[
Number_d = (b_{d_1}b_{d_2}...b_{d_l})_2.
\]
Bit deinterleaving (2). Example.

Number = 76893_{10}

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

It is OK that the first group is incomplete

\[
\begin{aligned}
\text{Number}_3 &= 0 \ 0 \ 0 \ 1 \ 1 \ 1 = 111_2 = 7_{10} \\
\text{Number}_2 &= 1 \ 1 \ 1 \ 0 \ 1 \ 0 = 111010_2 = 58_{10} \\
\text{Number}_1 &= 0 \ 1 \ 0 \ 0 \ 1 = 1001_2 = 9_{10}
\end{aligned}
\]
Finding the center of a given box.

Coordinates of the box center in binary form are

\[ x_{k,c}(\text{Number}, l) = (0.b_{k1}b_{k2}...b_{kd}1)_2, \quad k = 1, ..., d. \]

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

\[ x_{k,c}(\text{Number}, l) = 2^{-l} \cdot \text{Number}_k + 2^{-l-1} = 2^{-l} \cdot \left( \text{Number}_k + \frac{1}{2} \right), \quad k = 1, ..., d. \]

**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:

\(\text{Number}_3 = 1001_2 = 9_{10}, \text{Number}_2 = 10_2 = 2_{10}, \text{Number}_1 = 1_2 = 1_{10}.\) We have then

\[ x_{1,c}(533, 5) = 2^{-5} \cdot (1 + 0.5) = 0.04875, \quad x_{2,c}(533, 5) = 2^{-5} \cdot (2 + 0.5) = 0.078125, \quad x_{3,c}(533, 5) = 2^{-5} \cdot (9 + 0.5) = 0.296875. \]

**Answer:** \(x_c = (0.04875, 0.078125, 0.296875).\)
Neighbor Finding

Step 1: Deinterleaving:

\[ Number \rightarrow \{ Number_1, \ldots, Number_d \} \]

Step 2: Shift of the coordinate numbers

\[ Number_k^+ = Number_k + 1, \quad Number_k^- = Number_k - 1, \quad k = 1, \ldots, d, \]

and formation of sets:

\[
s_k = \begin{cases} 
\{ Number_k^-, Number_k, Number_k^+ \}, & Number_k \neq 0, 2^l - 1 \\
\{ Number_k, Number_k^+ \}, & Number_k = 0. \\
\{ Number_k^-, Number_k \}, & Number_k = 2^l - 1. 
\end{cases} 
\]

The set of neighbor generating numbers is then

\[ n = (n_1, \ldots, n_d), \quad n_k \in s_k, \quad k = 1, \ldots, d. \]

where each \( n_k \) can be any element of \( s_k \), except of the case when all \( n_k = Number_k \) simultaneously for all \( k = 1, \ldots, d \), since this case corresponds to the box itself.
Neighbor Finding (2). Example.

\[ 26_{10} = 11010_2 \]

\[ (11,100)_2 = (3,4)_{10} \]

\[ (2,3), (2,4), (2,5), (3,3), (3,5), (4,3), (4,4), (4,5) = (10,11), (10,100), (10,101), (11,11), (11,101), (100,11), (100,100), (100,101) \]

\[ 1101, 11000, 11001, 1111, 11011, 100101, 110000, 110001 = 13, 24, 25, 15, 27, 37, 48, 49 \]
Spatial Data Structuring

Data Collection.

Scaling and mapping finite $d$-dimensional data into a unit $d$-dimensional cube yields in a collection $\mathcal{C}$ of $N$ points distributed inside such a cube:

$$\mathcal{C} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}, \quad \mathbf{x}_i \in [0, 1) \times [0, 1) \times \ldots \times [0, 1) \subseteq \mathbb{R}^d, \quad i = 1, \ldots, N.$$

Data Set.

We call a collection $\mathcal{C} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ “data set”, if $\forall i \neq j$, $\text{dist} (\mathbf{x}_i, \mathbf{x}_j) \neq 0$, where $\text{dist} (\mathbf{x}_i, \mathbf{x}_j)$ denotes distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Non-Separable (Multi-entry) Data Collection.

We call a collection $\mathcal{C} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ “non-separable data collection”, if $\exists i \neq j$, $\text{dist} (\mathbf{x}_i, \mathbf{x}_j) = 0$.

(By this definition a non-separable data collection cannot be uniquely ordered using distance function $\text{dist} (\mathbf{x}_i, \mathbf{x}_j)$).
Some properties of 2\textsuperscript{d}-tree hierarchy.

**Theorem:** Let \( dist(x_i, x_j) = |x_i - x_j| \). Then for any data collection \( C \) at level

\[
L > \frac{1}{d} \log_2 N
\]

of 2\textsuperscript{d}-tree hierarchical space subdivision there exist boxes that do not contain points from \( C \). We call such boxes “empty” or “zero” boxes.

**Proof.** The number of boxes at level \( L \) is \( 2^{Ld} \), which is larger than \( N \), at \( L > \frac{1}{d} \log_2 N \).

**Theorem:** Let \( dist(x_i, x_j) = |x_i - x_j| \). For any data set \( S \) of power \( N \geq 2 \) at level

\[
L > \log_2 \frac{d^{1/2}}{\bar{D}_{\text{min}}}, \quad \text{where}
\]

\[
\bar{D}_{\text{min}} = \min_{i \neq j} |x_i - x_j|, \quad i, j = 1, ..., N,
\]

of 2\textsuperscript{d}-tree hierarchical space subdivision each box contains not more than 1 data point.

**Proof.** The the main diagonal of the box at level \( L \) is \( d^{1/2} 2^{-L} \). This is smaller than \( \bar{D}_{\text{min}} \) if

\[
L > \log_2(d^{1/2}/\bar{D}_{\text{min}}).
\]
Threshold Level

We call level $L_{th}(C)$ “threshold level” of data collection $C$ if the maximum number of data points in a box for any level of subdivision $L > L_{th}(C)$ is the same as for $L_{th}(C)$ and differs from $L_{th}(C)$ for any $L < L_{th}(C)$.

Note: in case if $C$ is a data set of power $N \geq 2$, then at level $L_{th}(C)$ we will have maximum one data point per box, and at $L < L_{th}(C)$ there exists at least 1 box containing 2 or more data points.
Some Practical Issues Related to Spatial Ordering

If the type of data used allows to keep $\text{Bit}_{\text{max}}$ bits to represent each coordinate of a data point, then the maximum available level of space subdivision is $\text{Bit}_{\text{max}}$. If it happens that $\text{Bit}_{\text{max}} < L_h(C)$ then $C$ is non-separable in machine representation (by definition we always have a box containing at least 2 data points).

Indeed, limited (discrete) representation of numbers results in discrete distance between the points. If this distance is denoted as $dist$, we have

$$\exists \varepsilon > 0, dist(\bar{x}_i, \bar{x}_j) = 0 \text{ if } |\bar{x}_i - \bar{x}_j| < \varepsilon.$$ 

And a data collection, which is a data set in norm of $\mathbb{R}^d$, is a non-separable data collection in the norm of machine representation.

Box numbering in $d$-dimensions using interleaving shows that at level $L$ the number of bits required for box number is $Ld$. This may cause severe restrictions on use of standard types for representation of integers. E.g. if the max number of bits for integer is 31 and $d = 3$, $L$ cannot exceed 10. This means that if the difference between coordinates of data points is less than $3^{-10} \approx 1.7 \cdot 10^{-5}$ such points cannot be uniquely ordered using the oct-tree. For larger dimensions extended types for integers utilized in hierarchical numbering should be defined.
Spatial Data Sorting

Consider data collection C. Each point can be then indexed (or numbered):

\[ v = (v_1, v_2, \ldots, v_N), \quad v_i = Number(x_i, L), \quad i = 1, \ldots, N, \]

where \textit{Number} can be determined using the algorithm described in the previous sections.

The array \( v \) then can be sorted for \( O(N\log N) \) operations:

\[ (v_1, v_2, \ldots, v_N) \rightarrow (v_{i_1}, v_{i_2}, \ldots, v_{i_N}), \quad v_{i_1} \leq v_{i_2} \leq \ldots \leq v_{i_N}. \]

using standart sorting algorithms. These algorithms also return the permutation index (other terminology can be permutation vector or pointer vector) of length \( N \):

\[ \text{ind} = (i_1, i_2, \ldots, i_N), \]

that can be stored in the memory. In terms of memory usage the array \( v \) should not be rewritten and stored again, since \( \text{ind} \) is a pointer and

\[ v(i) = v_{i_i}, \quad \text{ind}(j) = i_j, \quad v(\text{ind}(j)) = v(i_j) = v_{i_j}, \quad i, j = 1, \ldots, N, \]

so

\[ v(\text{ind}) = (v_{i_1}, v_{i_2}, \ldots, v_{i_N}). \]
Spatial Data Sorting (2)

• Before sorting represent your data with maximum number of bits available (or intended to use). This corresponds to maximum level $L_{\text{available}}$ available (say $[L_{\text{available}} = \text{BitMax}/d]$).

• In the hierarchical $2^d$-tree space subdivision the sorted list will remain sorted at any level $L < L_{\text{available}}$. So the data ordering is required only one time.
After data sorting we need to find the maximum level of space subdivision that will be employed.

In Multilevel FMM two following conditions can be mainly considered:

- At level $L_{\text{max}}$ each box contains not more than $s$ points ($s$ is called clustering or grouping parameter).
- At level $L_{\text{max}}$ the neighborhood of each box contains not more than $q$ points.
The threshold level determination algorithm in $O(N)$ time

\[
\begin{align*}
&i = 0, \ m = s, \\
&\text{while } \ m < N \\
&\quad \ i = i + 1, \ m = m + 1; \\
&\quad \ a = \text{Interleaved}(v(\text{ind}(i))); \\
&\quad \ b = \text{Interleaved}(v(\text{ind}(m))); \\
&\quad \ j = \text{Bit}_{\text{max}} + 1 \\
&\text{while } a \neq b \\
&\quad \ j = j - 1; \\
&\quad \ a = \text{Parent}(a); \\
&\quad \ b = \text{Parent}(b); \\
&\quad \ l_{\text{max}} = \max(l_{\text{max}}, j); \\
&\text{end}; \\
&\text{end};
\end{align*}
\]

$s$ is the clustering parameter.
Binary Search in Sorted List

• Operation of getting non-empty boxes at any level $L$ (say neighbors) can be performed with $O(\log N)$ complexity for any fixed $d$.
  • It consists of obtaining a small list of all neighbor boxes with $O(1)$ complexity and
  • Binary search of each neighbor in the sorted list at level $L$ is an $O(Ld)$ operation.
  • For small $L$ and $d$ this is almost $O(1)$ procedure.
Operations on Sets

*Difference:* $C = A \setminus B$

*Intersection:* $C = A \cap B$

*Union:* $C = A \cup B$

Let $\text{Pow}(A) = N$, $\text{Pow}(B) = M$, $N \geq M$,

Then the complexity for sorted input/output:

$A \setminus B : N$

$A \cap B : \min(N, M \log N)$

$A \cup B : N$

Operations

$\text{Neighbors}(W; n, l) = \text{NeighborsAll}(n, l) \cap W$, $W = X, Y$,

$\text{Children}(W; n, l) = \text{ChildrenAll}(n, l) \cap W$, $W = X, Y$.

are $O(\log N)$ operations for minimum memory requirements and $O(1)$ for sufficiently large memory.