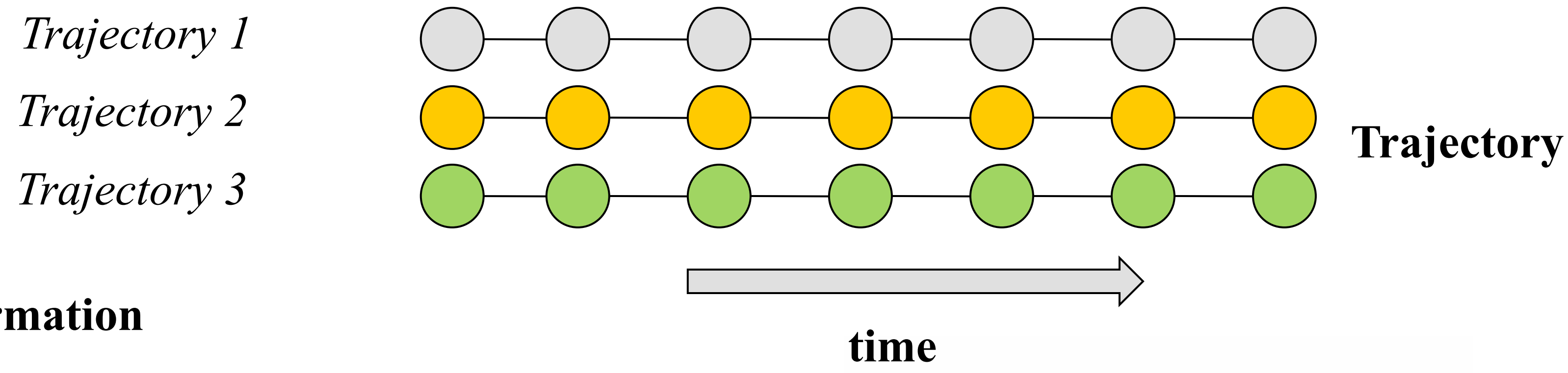
聚类分析与生物分子动力学
四 Cover Tree

陈　赢
姚　远

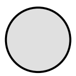
2011.3.15

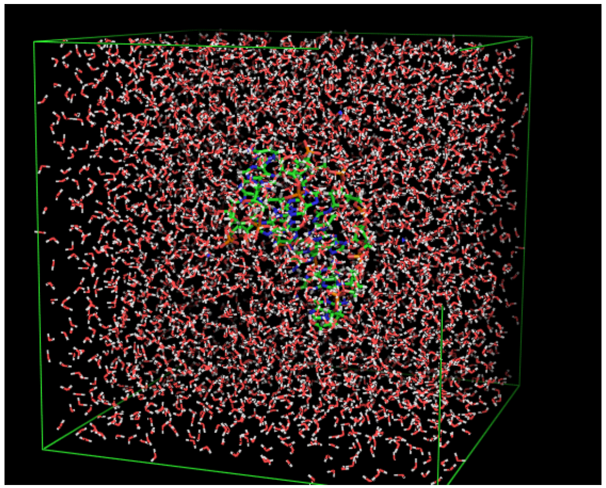
Time Series Analysis in Molecular Dynamics

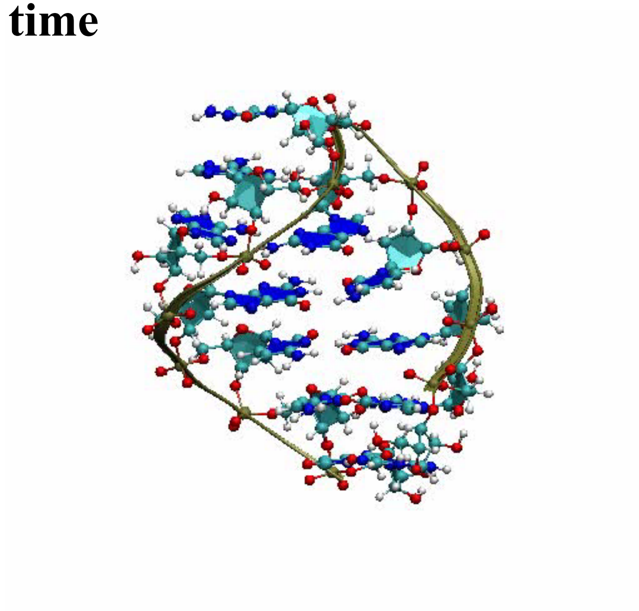
Dataset: Multiple trajectories with a lot of conformations.



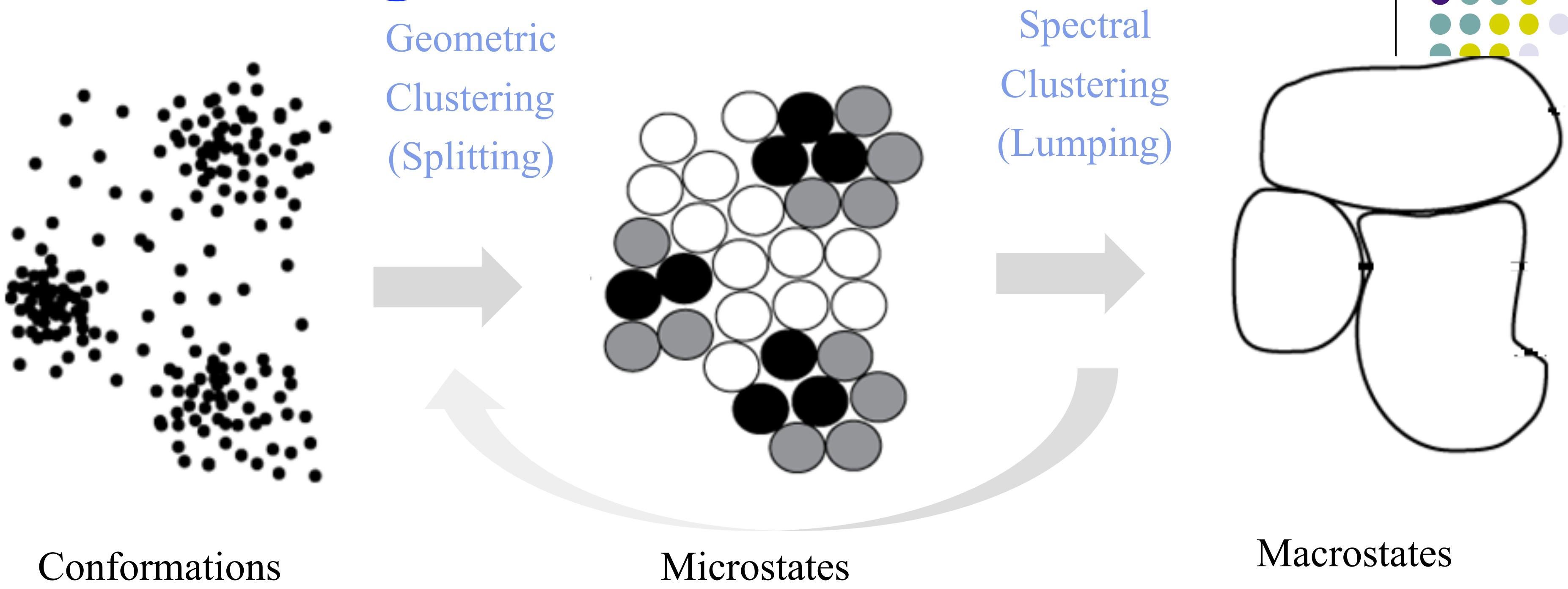
Conformation







Coarse-grain Markov Models



Bayesian Inference of MSM

$$T(\tau) = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{15} \\ p_{21} & p_{22} & & \\ \vdots & & \ddots & \\ p_{51} & & & p_{55} \end{bmatrix}$$

Chodera. et. al. *J. Chem. Phys.* 2007

Noé. et.al. *J. Chem. Phys.* 2007

Deuflhard and Weber, *ZIB-report,* 2003

Weber, *ZIB-report,* 2004

Bowman, Huang, and Pande. *Methods* 2009.

Barcalado, et al. *J. Chem. Phys.* 2009

How?

- Build up Microstates:
  - k-center
  - cover-tree (CHEN, Ying: this lecture)
- Build up Macrostates:
  - Lumpability of Markov chains
  - Spectral clustering for lumping
  - Nystrom method for denoising
- Bayesian Inference for MSM (next lecture)

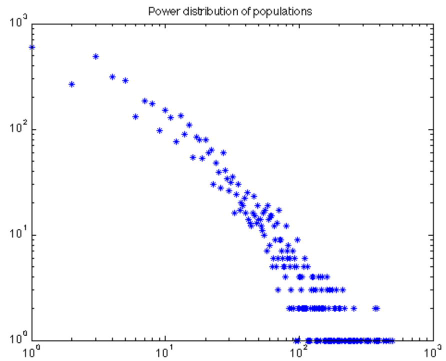
Geometric Clustering in Splitting

- Target: decompose the sampled region in high dimensional state space (3N) into small cells by geometric affinity (RMSD distance)

- Why:
  - Small RMSD distance implies that two structures are similar and thus kinetically close
  - But large RMSD distance tells us nothing about kinetics

Splitting Algorithms

- Geometric Clustering:
  - k-center: fast O(kn), geometric r_k-net
  - Cover-tree: online, hierarchical (CHEN, Ying)
- Pros:
  - Fast (compared to K-means)
  - Geometric uniform partition
  - Hiearchical, online
- Cons:
  - Sensitivity to outliers
  - Large amount of low populated microstates



Cover Tree For Nearest Neighbour Search

ChenYing
陈赢

Reference

- Acknowledgement: part of the slides are from Victoria Choi's slides with some modification.
- A. Beygelzimer, S. Kakade, and J. Langford.

Cover trees for nearest neighbor, ICML2006

Outline

- 1 Introduction: What’s cover tree?
- 2 Tree Construct and Search Nearest Neighbor
- 3 Search Approximating Nearest Neighbor
- 4 Complexity Analysis
- 5 Application

Introduction

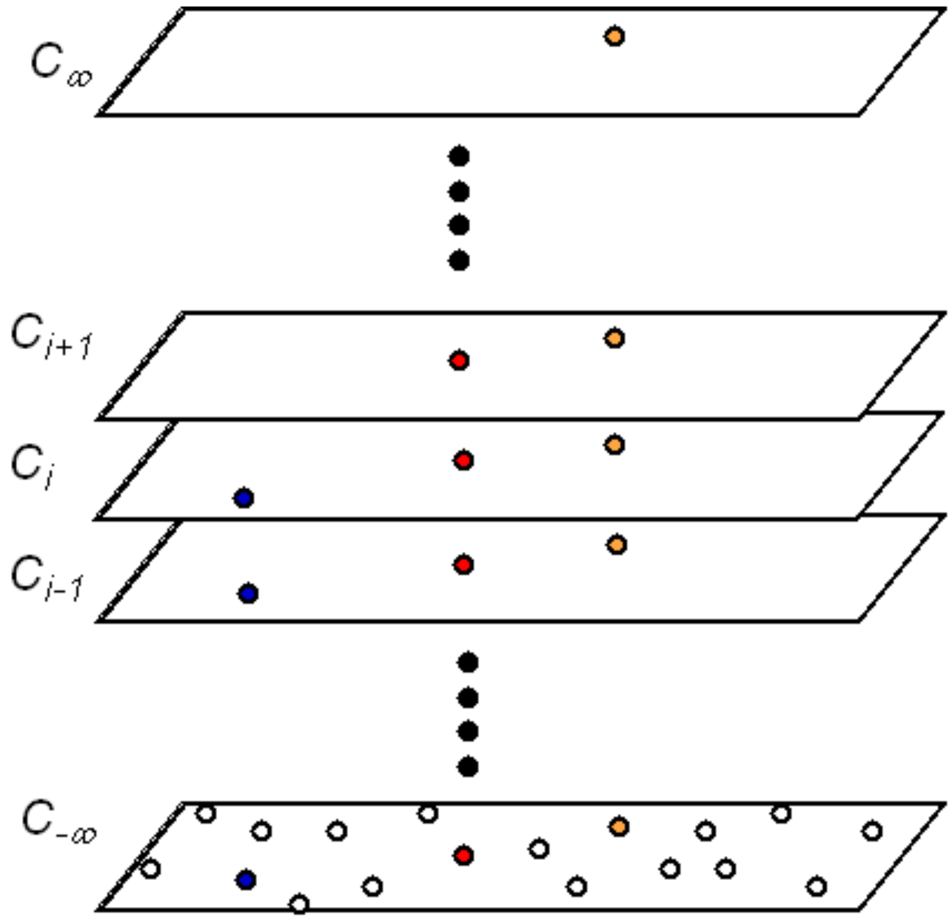
- Goal
  - Nearest-neighbour search
  - Preprocess a dataset *S* of *n* points in some metric space *X* so that given a query point $p \in X$, a point $q \in S$ which minimises $d(p,q)$ can be efficiently found
- Solution: Cover Tree
  - Leveled tree
  - Each level is a “cover” for the level beneath it
  - O(n) space bound

Cover Tree Data Structure

- A cover tree *T* on a dataset *S* is a leveled tree where each level is indexed by an integer scale *i* which decreases as the tree is descended
- C_i denotes the set of nodes at level *I*
- *d(p,q)* denotes the distance between poitns *p* and *q*
- A valid tree satisfies the following properties
  - *Nesting*: $C_i \subset C_{i-1}$
  - *Covering tree*: For every node $p \in C_{i-1}$, there exists a node $q \in C_i$ satisfying $d(p,q) \leq 2^i$ and exactly one such *q* is a parent of *p*
  - *Separation:* For all nodes $p,q \in C_i$, $d(p,q) > 2^i$

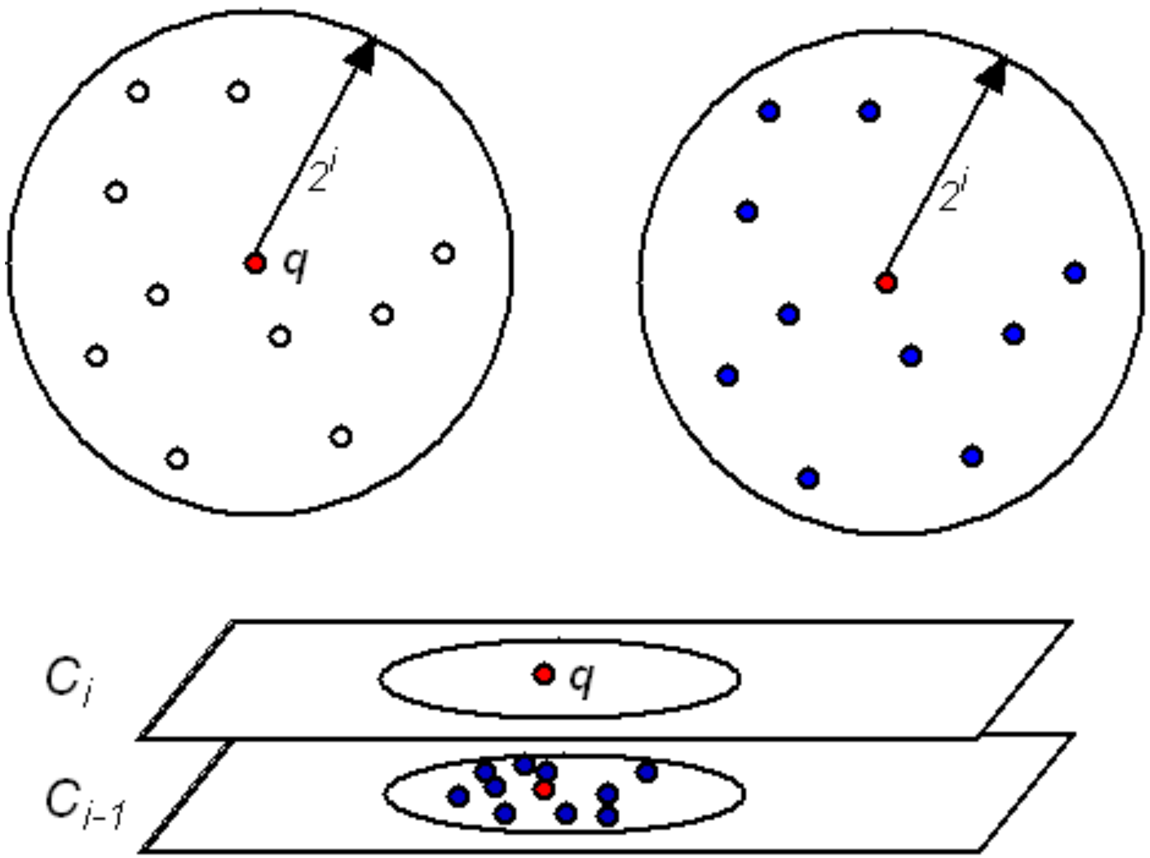
Nesting

- $C_i \subset C_{i-1}$
  - Each node in set C_i has a self-child
  - All nodes in set C_i are also nodes in sets C_j where j<i
  - Set $C_{-\infty}$ contains all the nodes in dataset S



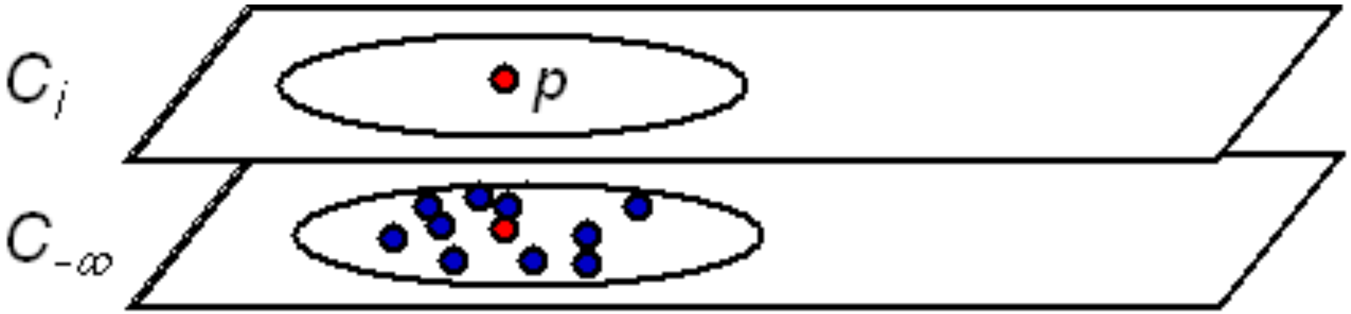
Covering Tree

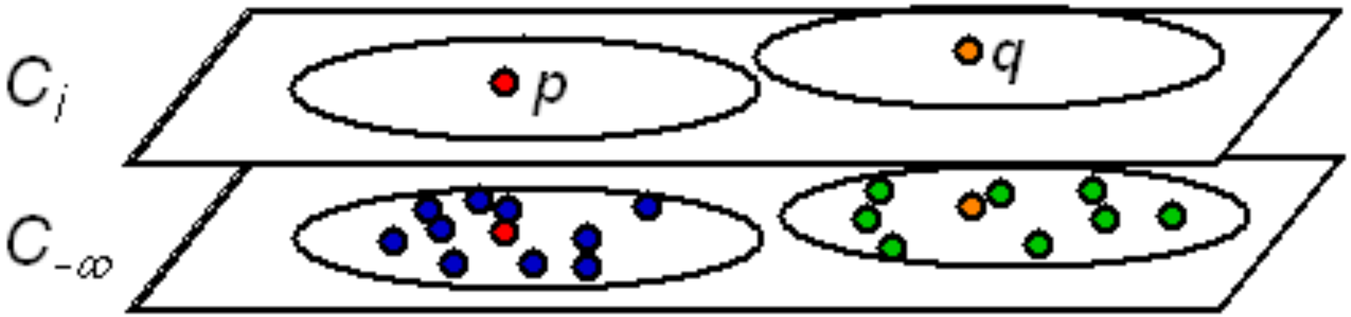
- For every node $p \in C_{i-1}$, there exists a node $q \in C_i$ satisfying $d(p,q) \le 2^i$ and exactly one such q is a parent of p



Separation

- For all nodes $p, q \in C_i$, $d(p,q) > 2^i$





Tree Construction

- Single Node Insertion (recursive call)

$$
\begin{array}{l}
\text{Insert(point } p, \text{cover set } Q_i, \text{level } i) \\
\quad \text{set } Q = \{Children(q) : q \in Q_i\} \\
\quad \text{if } d(p,Q) > 2^i \text{ then return "no parent found"} \\
\quad \text{else} \\
\qquad \text{set } Q_{i-1} = \{q \in Q : d(p,q) \le 2^i\} \\
\qquad \text{if Insert}(p, Q_{i-1}, i-1) = \text{"no parent found" and } d(p, Q_i) \le 2^i \\
\qquad\quad \text{pick } q \in Q_i \text{ satisfying } d(p,q) \le 2^i \\
\qquad\quad \text{insert } q \text{ into Children}(q) \\
\qquad\quad \text{return "parent found"} \\
\qquad \text{else return "no parent found"} \\
\text{Insert}(p, \text{root}, \text{begin\_level});
\end{array}
$$

- Batch insertion algorithm also available

Searching the nearest neighbor

- Iterative method

$$\text{set } Q_\infty = C_\infty$$

$$\text{for } i \text{ from } \infty \text{ down to } -\infty$$

$$\text{consider the set of children of } Q_i :$$

$$set = \{Children(q) : q \in Q_i\}$$

$$\text{form next cover set} :$$

$$Q_{i-1} = \{q \in set : d(p,q) \le d(p,set) + 2^i\}$$

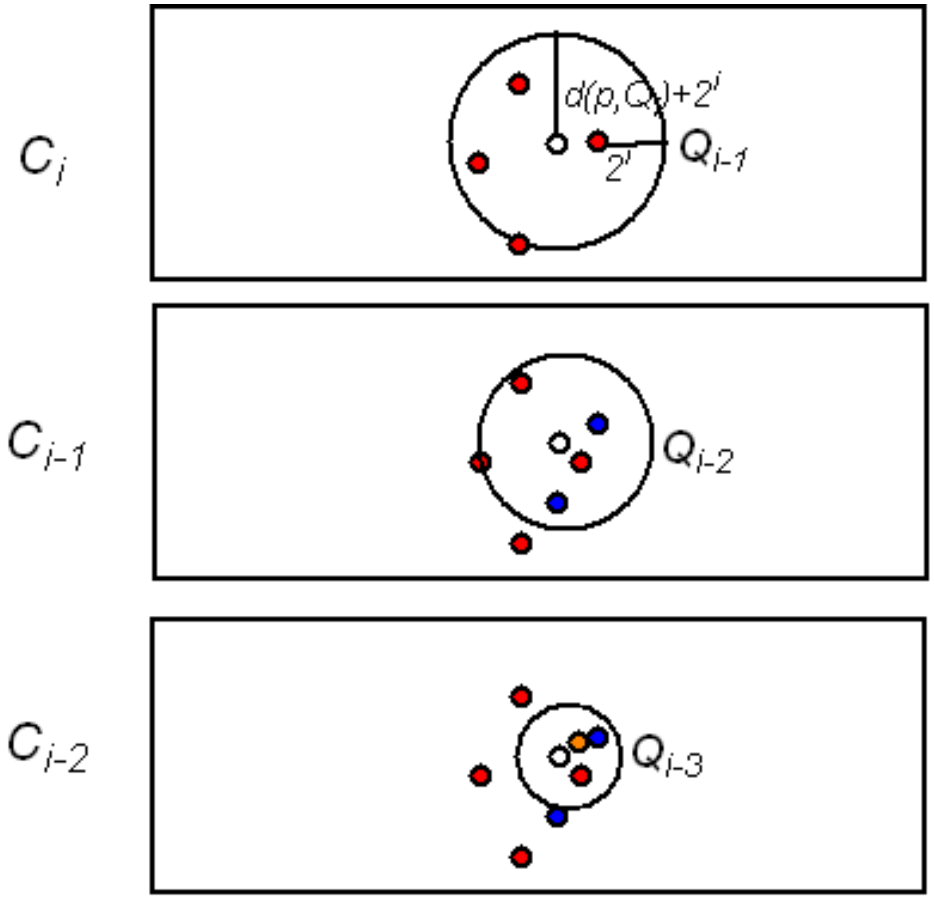
$$\text{return arg min}_{q \in Q_{-\infty}} d(p,q$$

Prove the correctness

- Theorem: $Insert(p, C_\infty, \infty)$ and Find-Nearest(p) are correct.

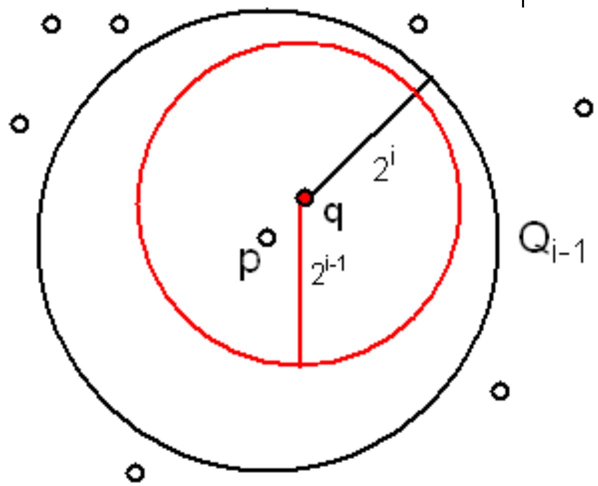
Why can you always find the nearest neighbour?

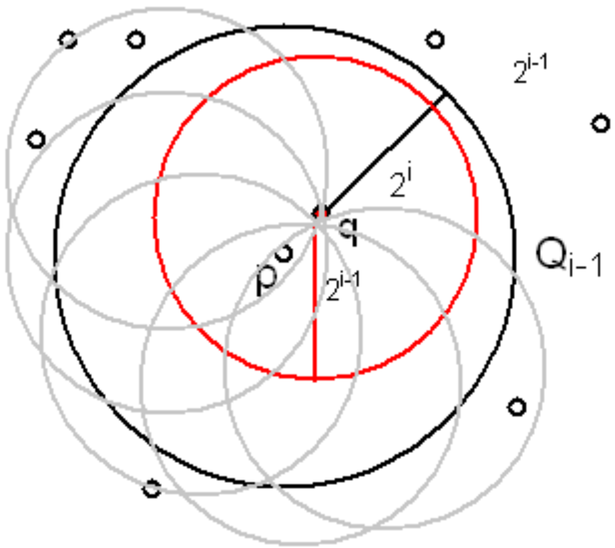
- When searching for the nearest node at each level *i*, the bound for the nodes to be included in the next cover set Q_{i-1} is set to be $d(p,Q)+2^i$ where $d(p,Q)$ is the minimum distance from nodes in Q_i
- Q will always center around the query node and will contain at least one of its nearest neighbours



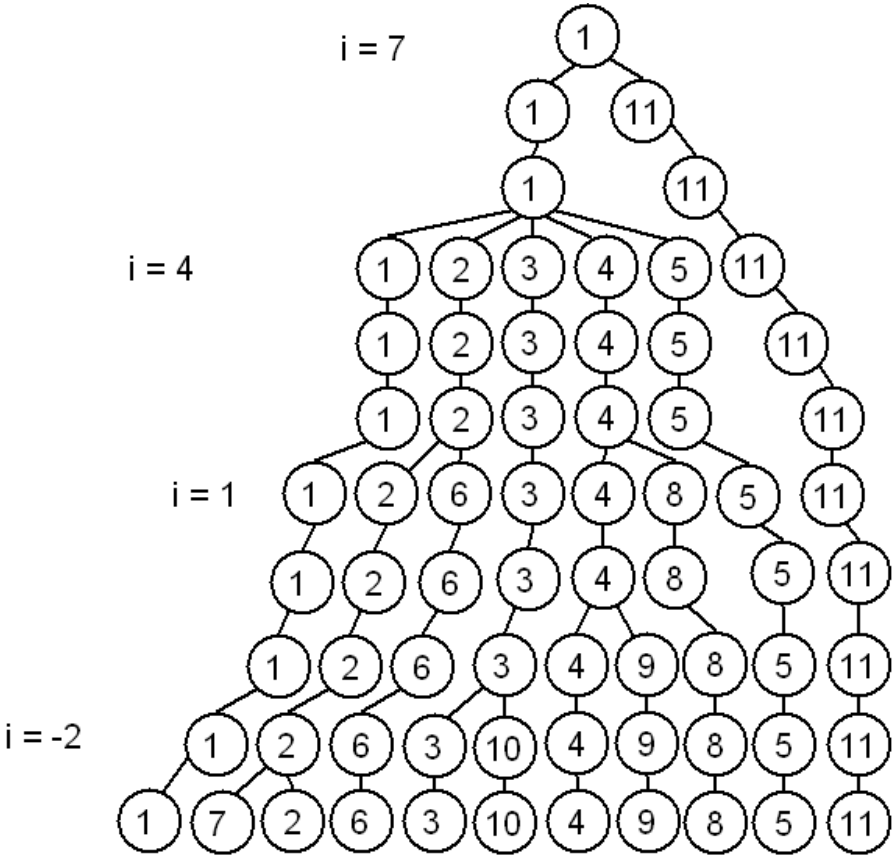
How?

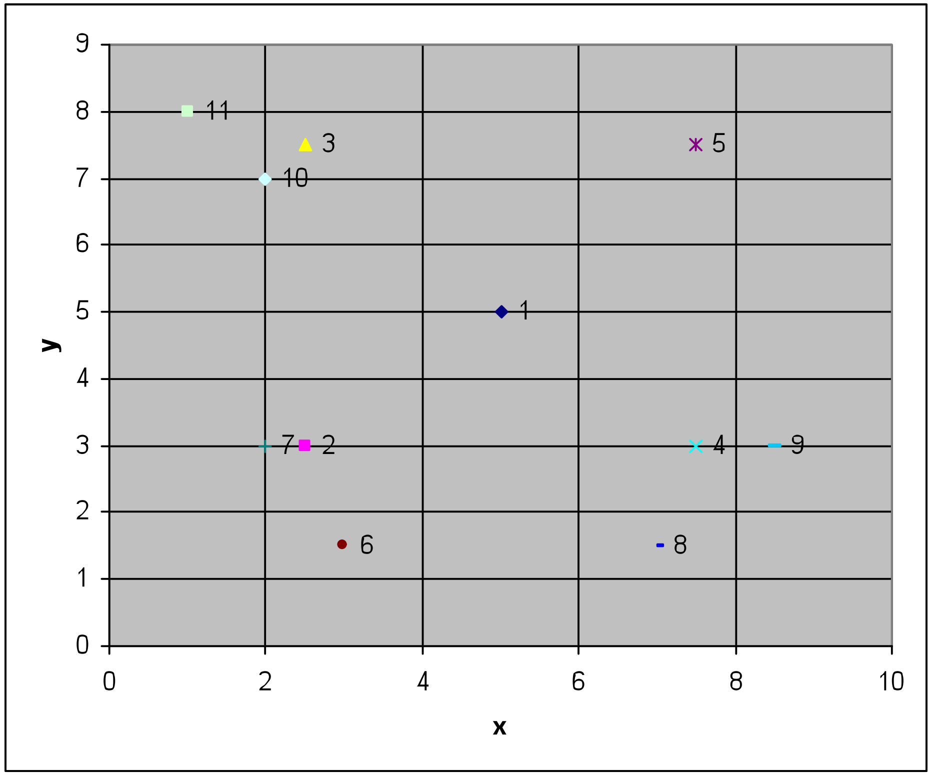
- All the descendents of a node *q* in C_i is less than or exactly 2^i away (2^{i-1} in C_{i-1})
- By setting the bound to be $d(p,Q)+2^i$, we have included all the nodes with descendents which might do better than node *p* in Q_{i-1} and eliminated everything else



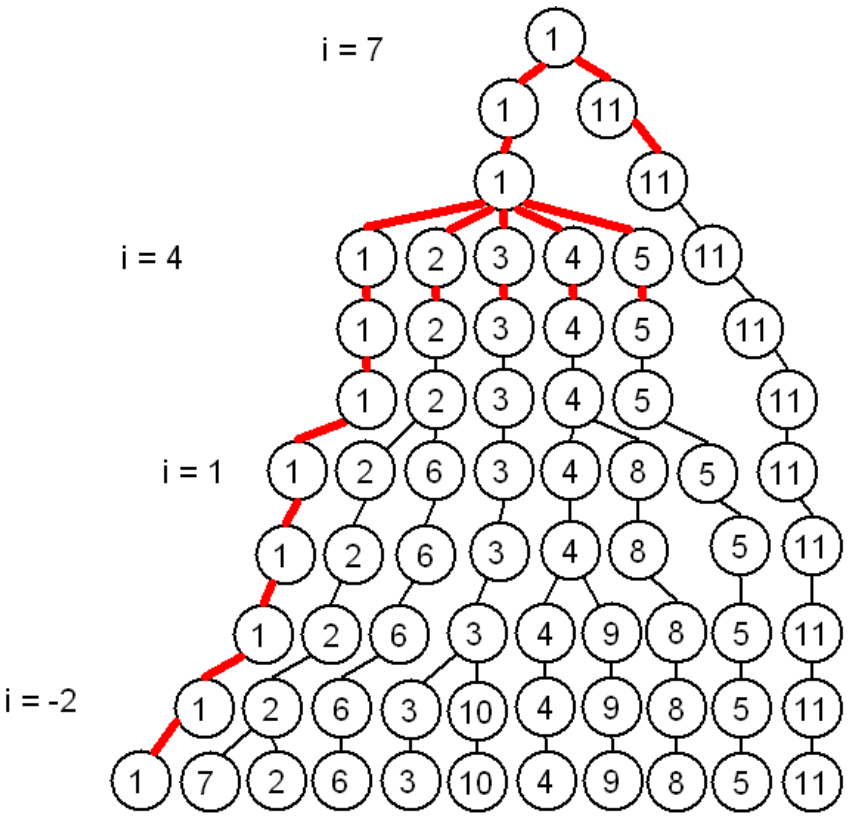


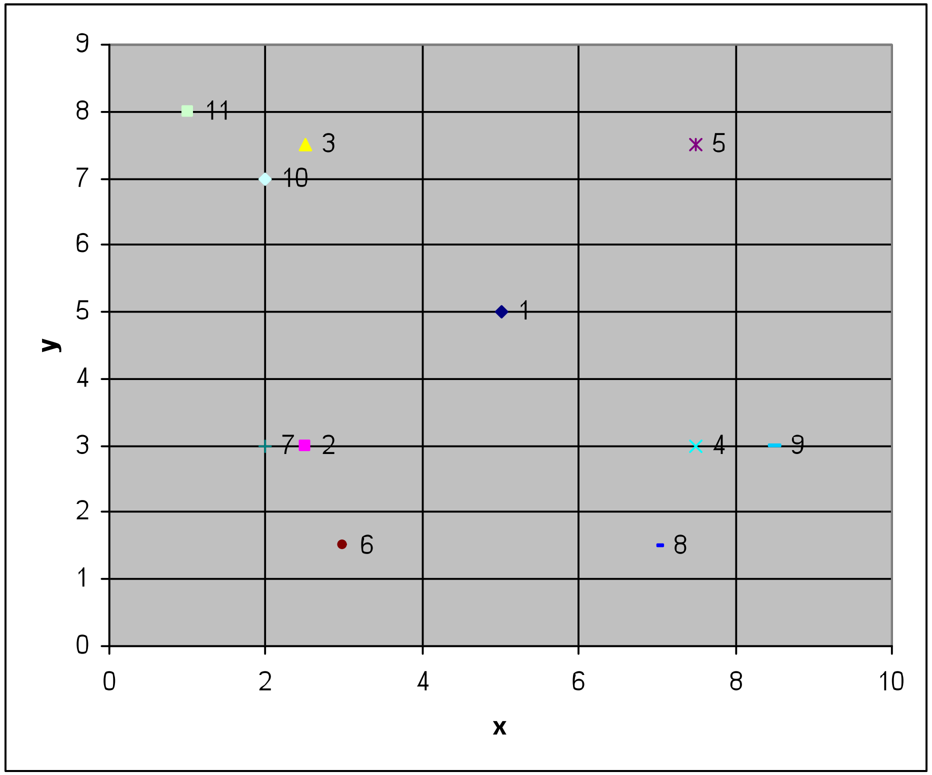
Search Examples



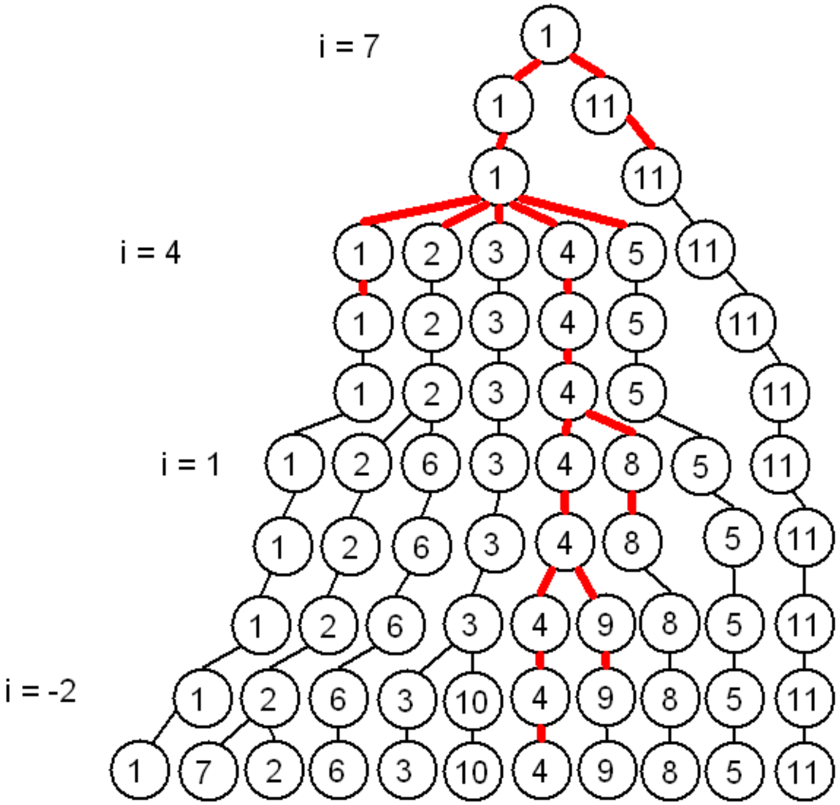


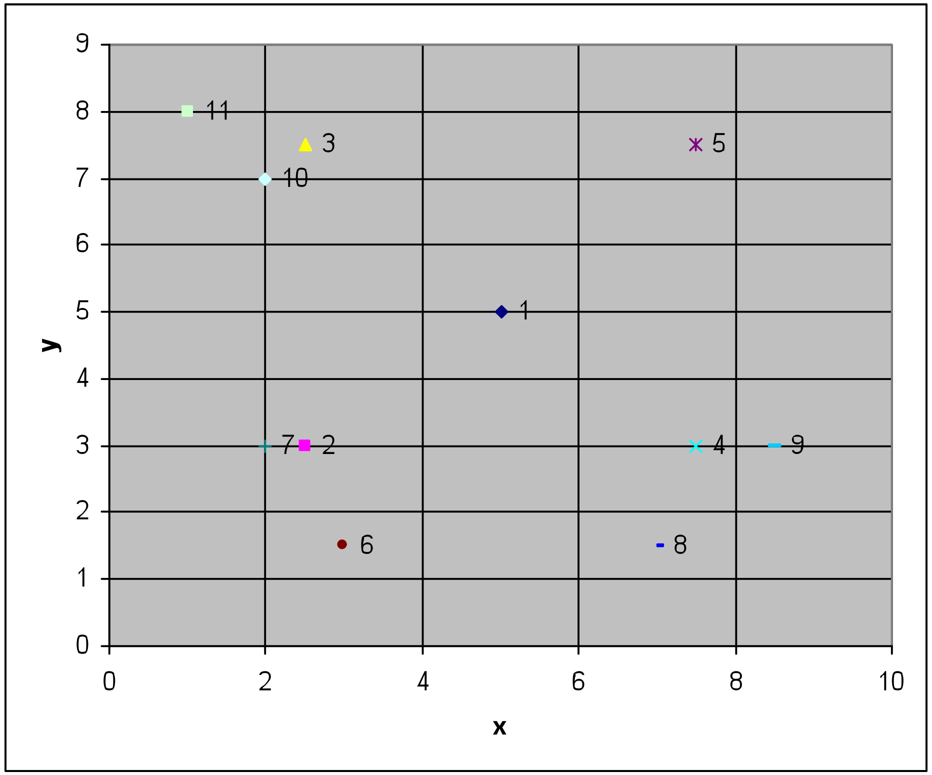
Search for Node 1



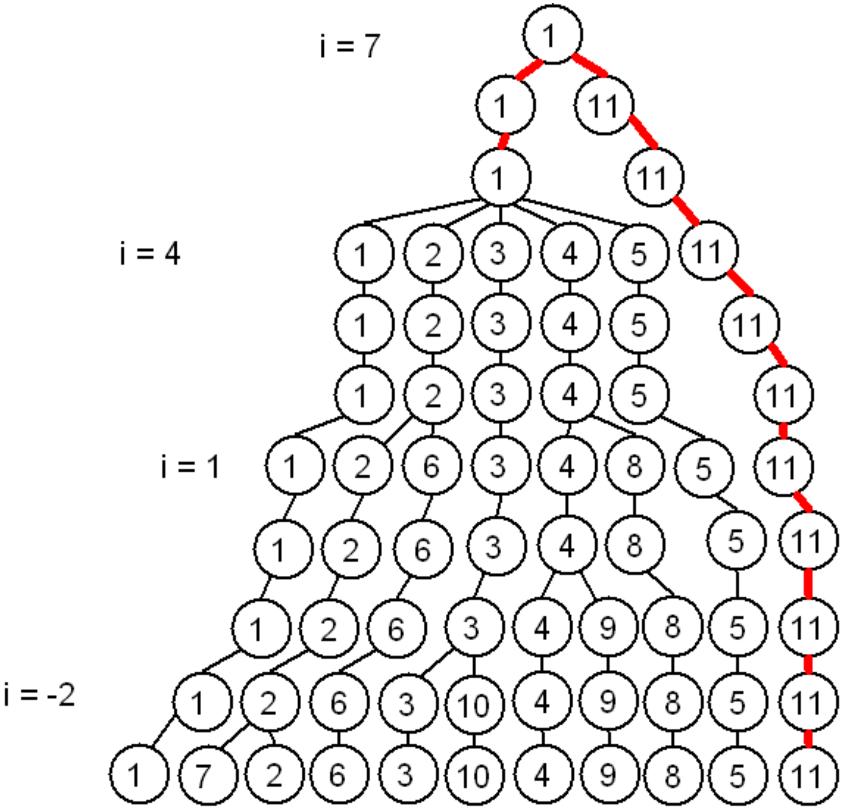


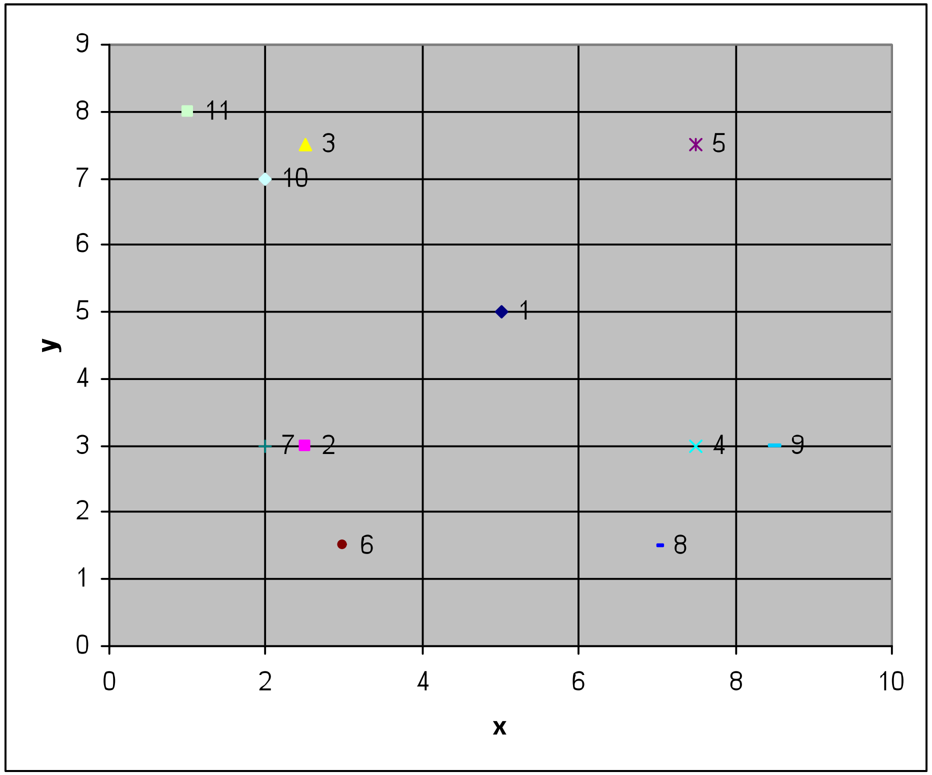
Search for Node 4





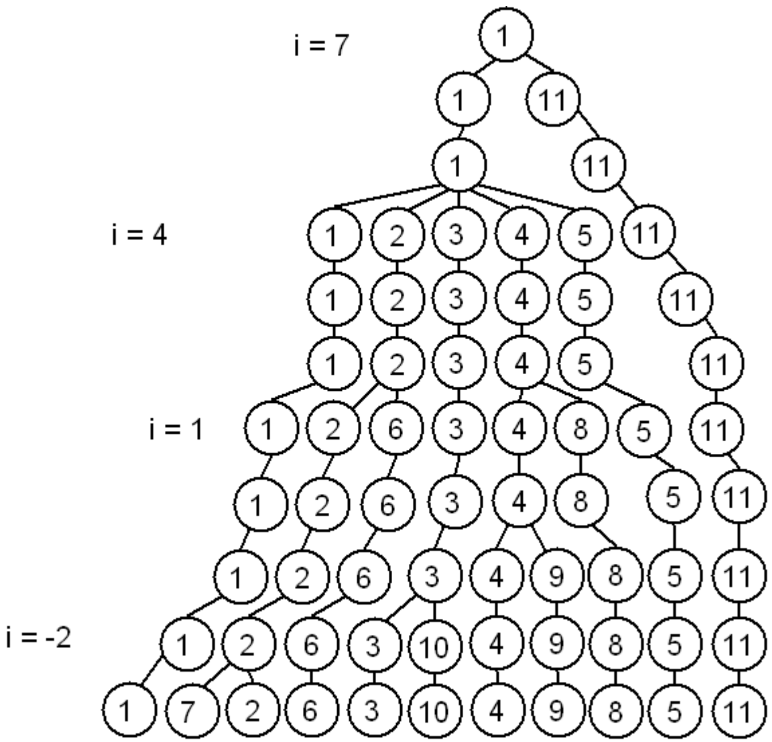
Search for Node 11

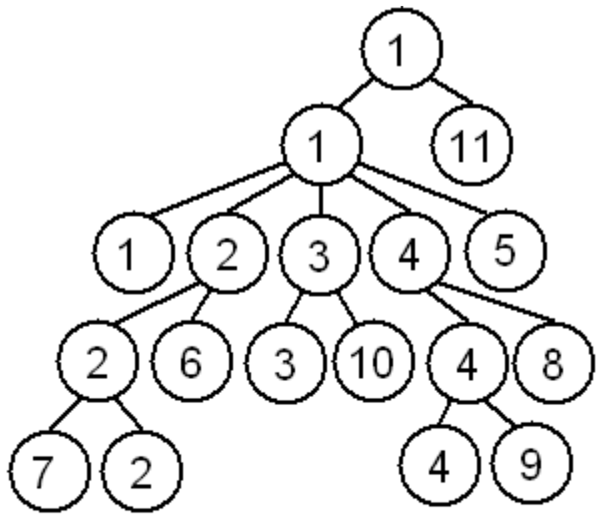




Implicit v. Explicit

- Theory is based on an implicit implementation, but tree is built with a condensed explicit implementation to preserve O(n) space bound





Search Approximating Nearest Neighbor

- Goal:Given a point $p \in X$,find a point $q \in X$
- $d(p,q) \leq (1+\varepsilon)d(p,S)$.

Expansion Constant

- Expansion constant *c* of dataset *S* is defined as the smallest value $c \geq 2$ such that $|B_S(p,2r)| \leq c\,|B_S(p,r)|$ for every $p \in X$ and $r > 0$
- The number of children of any node is bounded by c^4 (width bound)
- The maximum depth of any point is $O(c^2 \log n)$ (depth bound)
- A balanced tree would have a smaller expansion constant *c* than a tree that is not balanced
- *c* obtained from our current matrices are inaccurate since they are too small

Complexity Analysis

	Cover Tree	Nav. Net	[KR02]
Constr. Space	$O(n)$	$c^{O(1)}n$	$c^{O(1)}n \ln n$
Constr. Time	$O(c^6 n \ln n)$	$c^{O(1)}n \ln n$	$c^{O(1)}n \ln n$
Insert/Remove	$O(c^6 \ln n)$	$c^{O(1)} \ln n$	$c^{O(1)} \ln n$
Query	$O(c^{12} \ln n)$	$c^{O(1)} \ln n$	$c^{O(1)} \ln n$

Conclusion

- Since *c* cannot be accurately determined from the size of our matrices, we estimate the balance of the tree from the number of levels and the number of children
- For all three matrices tested, the trees constructed are well-balanced and speedup times are excellent
- Strong evidence that the cover-tree algorithm will be suitable for curve-matching distances

Example on the Biological Data

- Data description:
  - load ../data/alanine_dipeptide_phi-psi.mat
  - [x,y,z]=embedTorus(3,1,phi,psi);
  - X=[x;y;z]';
  - save confs_3D.txt X -ascii
- Data format
  - confs_3D.txt is a 3x195000 matrix M, every column is a conformation. M[0,i], M[1,i],M[2,i] are the three-dimensional coordinates
  - This file is saved under subdirectory ./data/

How to run covertree

- BaseDirectory: Math.pku.edu.cn/yaoy/teachers/Spring2011/
  - CoverTree (use Euclid distance) for Linux: [BaseDirectory]/covertree/newVersion/linux/Euclid/
  - CoverTree (use RMSD distance) for Linux: [BaseDirectory]/covertree/newVersion/linux/Rmsd/
  - CoverTree for Windows (XP and 7.0): [BaseDirectory]/covertree/newVersion/linux/windows/
- There are several parameter that need to be set in advance.
- Main.cpp is the main file with those parameters
- Readme.txt gives a short introduction

How to run covertree

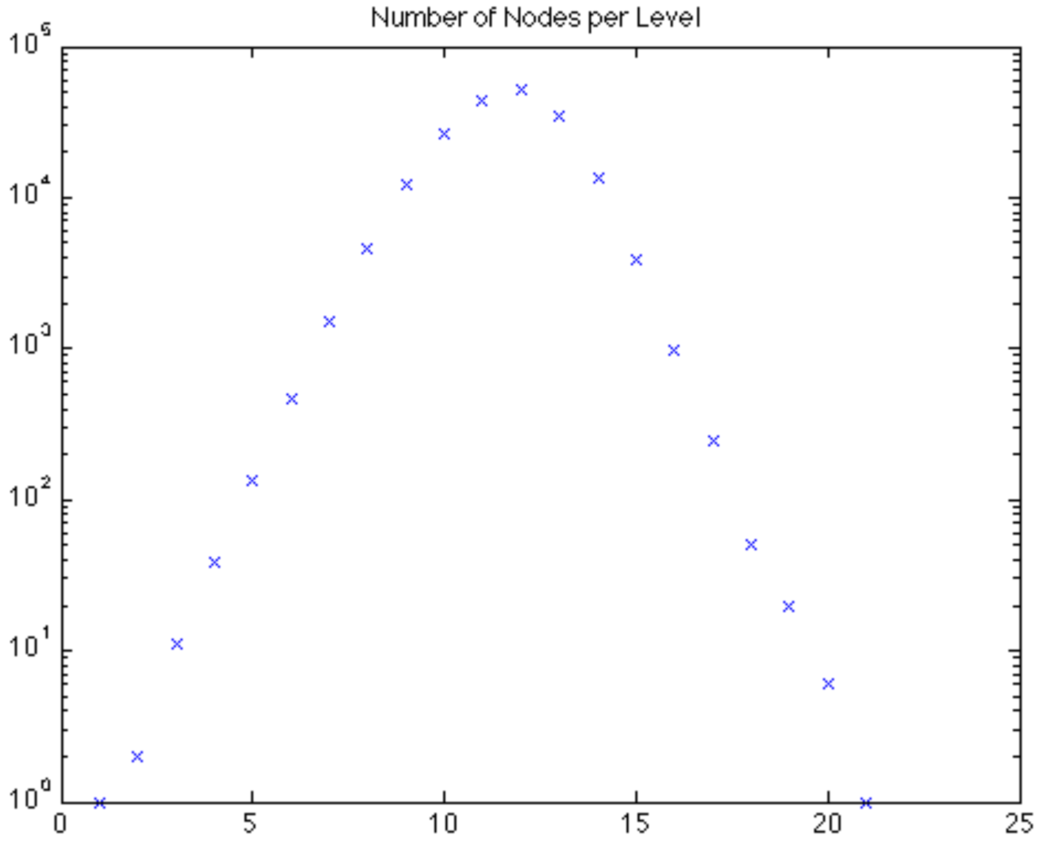
- Parameters (in Main.cpp):
  - level_begin: the level of root in covertree [Default: 5, i.e. cover radius =2^5]
  - natom: the number of atoms in a conformation [Default: 1]
  - nconformation: the number of conformations [Default: 195000]
  - IsCheck: whether to check the covertree is correct (It will cost much time) [Default: false]
  - char *filename = "…/data/confs_3D.txt";
- Compile the codes, you will get insert[.exe] as executable
  - In Linux, compile and run:
    - make
    - ./insert

OutPut

- 1 ./result/levelNumber.txt: Record the number of nodes in every level
- 2 ./result/level$i.txt: Record the node id in the level i
- 3 ./result/covertree.dot: Record the covertree structure in Graphviz dot format
- 4 Checking: whether the properties ‘separation’ and ‘covering’ are satisfied.
  - ./result/covertreefail.txt
  - ./result/separationfail.txt

-

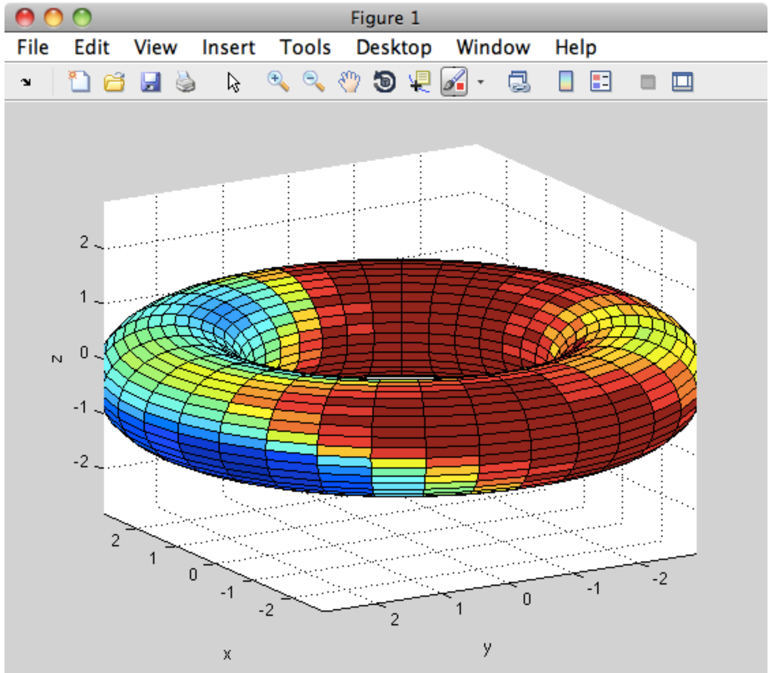
Number of Nodes per Levels

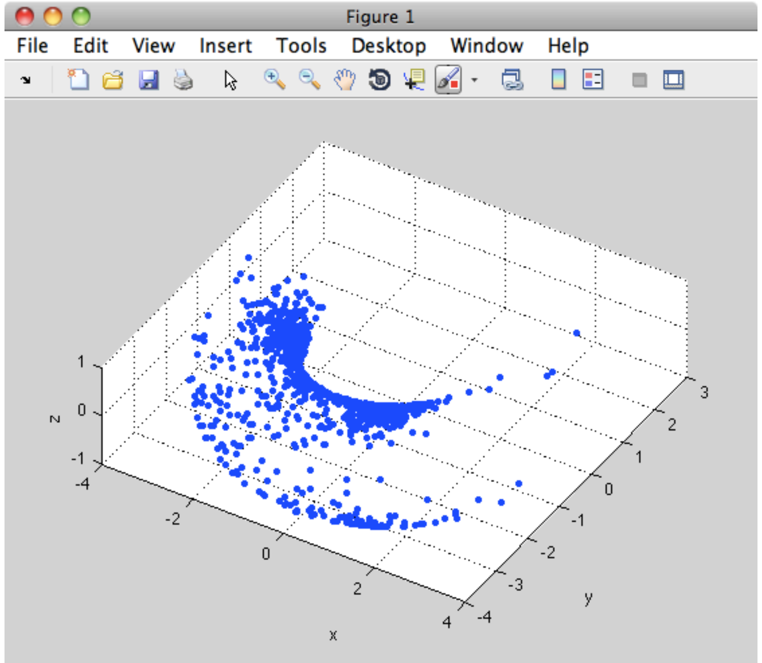


From levelNumber.txt

Recall: Torus Embedding

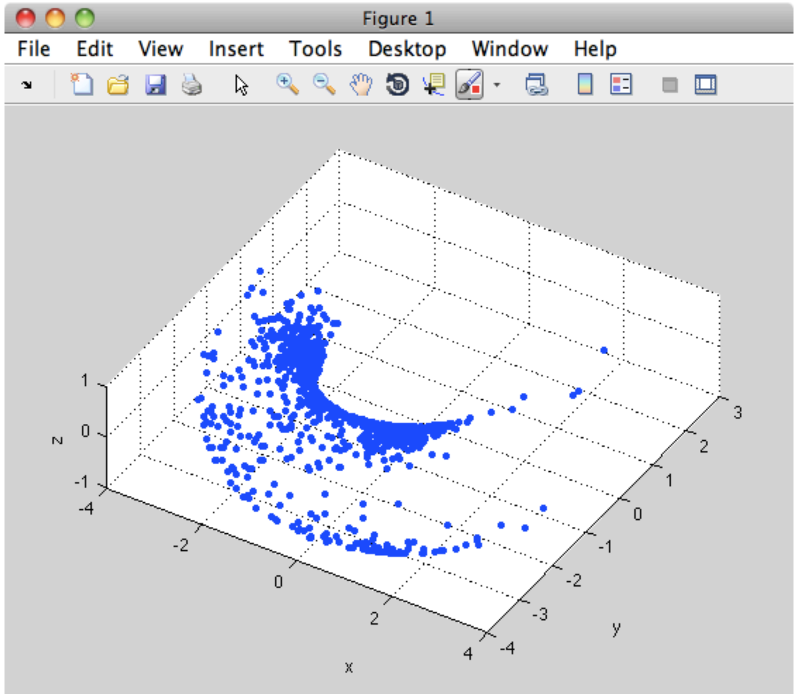
```
>> [x,y,z]=embedTorus(3,1,phi,psi);
>> freeEnergyTorus;
>> idx=randperm(length(phi));
>> scatter3(x(idx(1:1000)),y(idx(1:1000)),z(idx(1:1000)),'.')
```

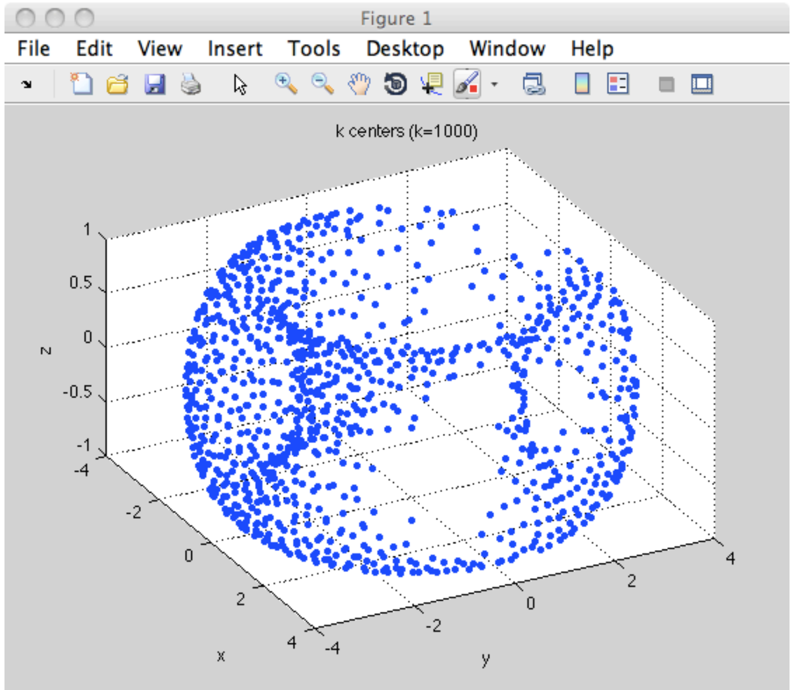




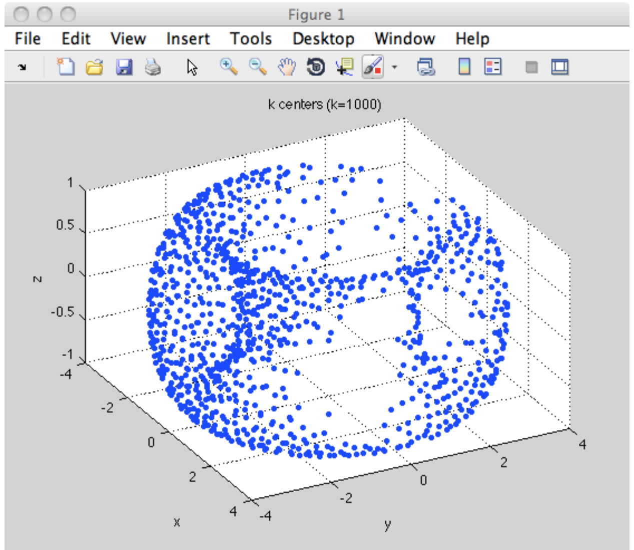
Random vs. Kcenter

```
>> idx=randperm(length(phi)); % 随机采样
>> scatter3(x(idx(1:1000)),y(idx(1:1000)),z(idx(1:1000),'.')
>> L=kcenter([x,y,z],1000); % 笔记本上需要几分钟…
>> scatter3(x(L),y(L),z(L),'.')
```

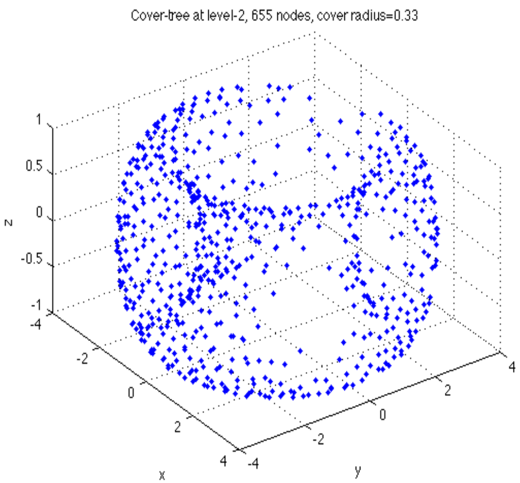




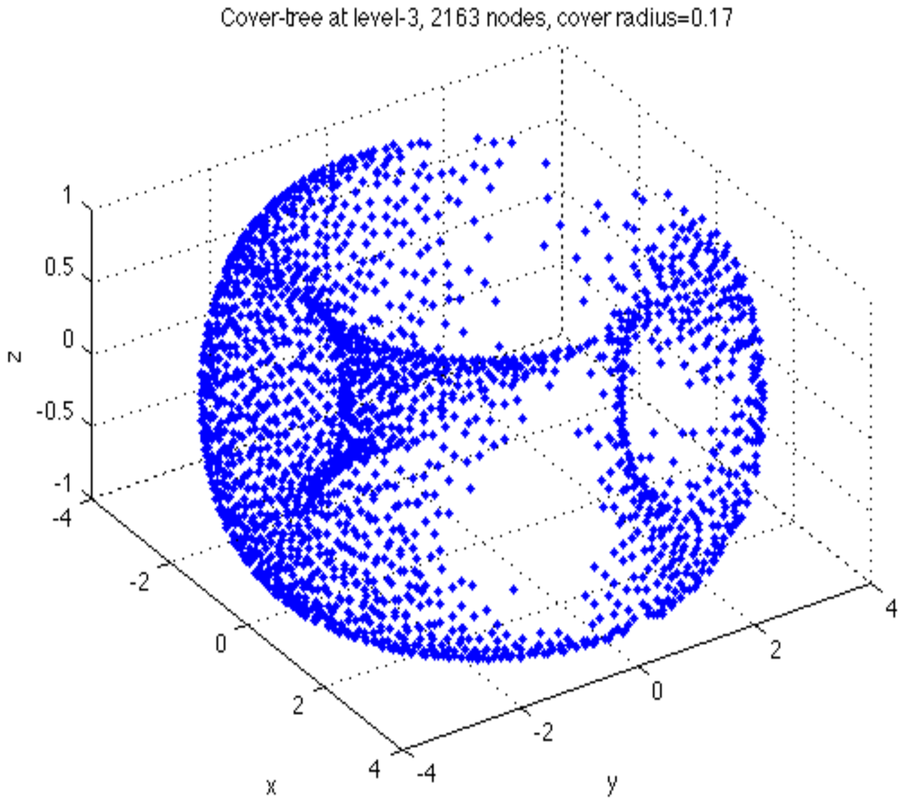
Kcenter vs. Covertree



Kcenter k=1000



Cover Tree Level=-2, 655 nodes



Cover Tree Level=-3, 2163 nodes

CoverTree is thus hierarchical online kcenter!