Self-Adjusting Geometric Structures for Latent-Space Embedding

Eunhui Park and David M. Mount

Problem Statement:

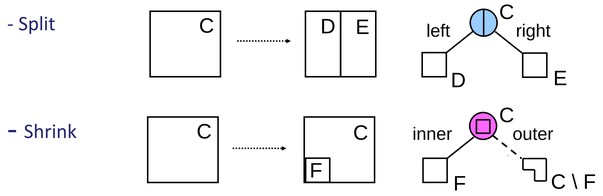
- Analysis of human social networks involves storing and retrieving large dynamic point sets.
- Latent Space Embedding:
  - Given a social network, map the nodes into a geometric space in accordance with a logistic regression model, where the likelihood of an edge increases as the distance between points decreases.
  - Solved by application of MCMC methods, such as Metropolis-Hastings.
  - Efficiency depends on the ability to quickly answer queries regarding point relationships in a dynamic setting.
- Statistical Analysis of Moving Entities:
  - Given the motion sequence for a set of agents, perform statistical analyses of their pattern of motion and their spatial relationships.
  - This involves storing dynamic point sets and performing queries over these sets.

Our Approach:

- Given the unpredictable nature of MCMC algorithms, it is important that data structures adapt to the algorithm's access pattern. This leads to the concept of self-adjusting data structures.
- Sleator & Tarjan (1985) introduced the splay tree, a self-adjusting data structure for 1-dimensional data.
- We developed a splay quadtree, a new self-adjusting data structure for multi-dimensional data.

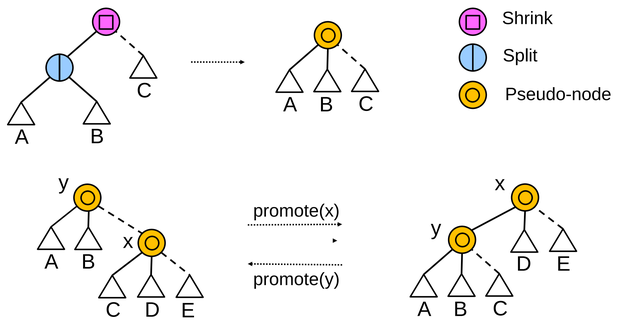
BD-tree:

- Spatial decomposition based on:



Rotation:

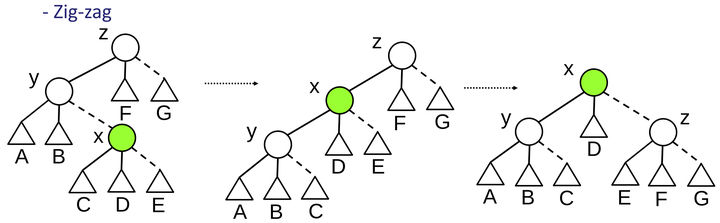
Balanced tree structure is maintained by rotating alternating pairs of shrink-split nodes:



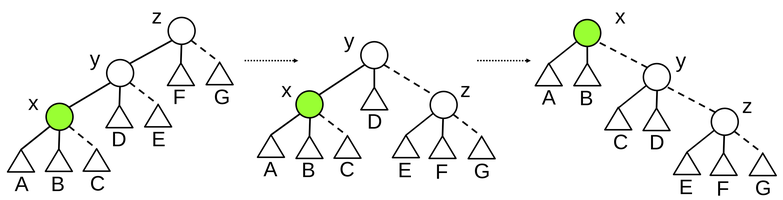
Basic splaying:

Primitive rotation operations:

- Zig-zag

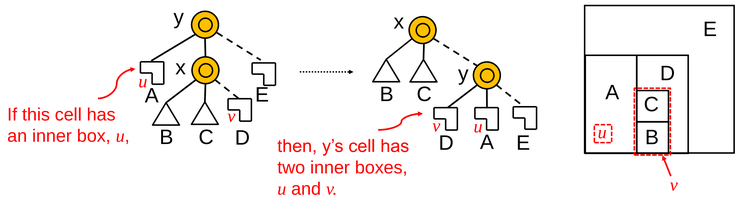


- Zig-zig



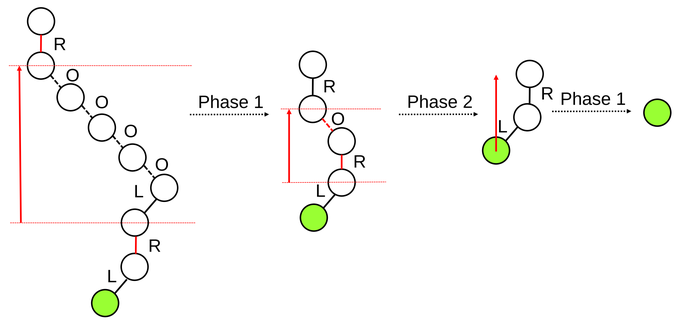
Splaying:

- May be applied to any internal node.
- Brings this node to the root through basic splaying operations.
- Thus, each access makes future accesses to the same node more efficient.
- Problem with a right promotion:



→ Promotions must be structured so that for each right promotion, left sibling has no inner box.

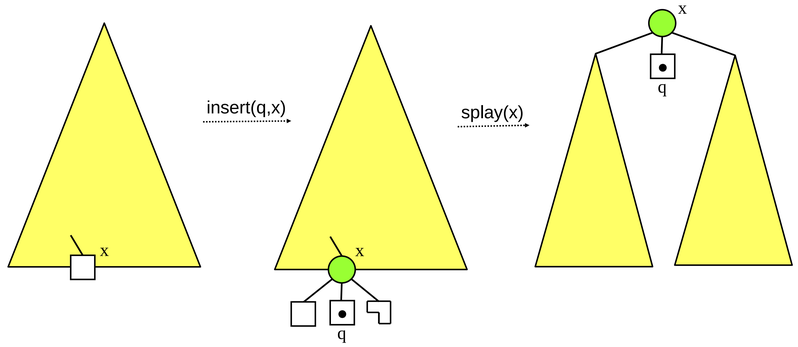
→ 3-phase approach (3 passes from bottom to top)



- Efficiency is established through an amortized analysis based on a potential function as in Sleator & Tarjan (1985) .

Main Results:

Updates:



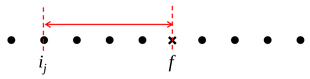
Theorem:

- Insertion can be performed in $O(\log n)$ amortized time.
- Deletion can be performed in $O(\log n)$ amortized time.

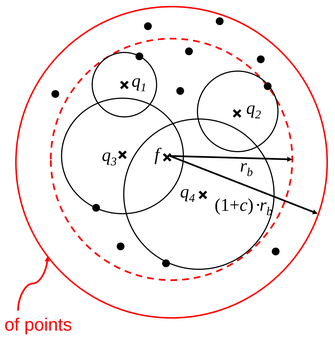
Static Finger Theorem:

Splay trees adapt to structure of queries.

- 1-dim (Sleator & Tarjan 1985)
  total access for $i_1, i_2, \ldots, i_m$ takes
  $O(m + \sum_{i=0}^{m} \log(|i_j - f| + 1))$.



- d-dim (new):
  Define the working set:
  W: set of points in a ball of
  radius $(1+c)\cdot r_b$ centered at f, where
  - for box queries, $Q_1, Q_2, \ldots, Q_m$,
    $r_b : \max_j dist(f, Q_j)$
  - for approx. nearest neighbor queries,
    $q_1, q_2, \ldots, q_m$,
    $r_b : \max_j dist(f, q_j) + dist(q_j, NN(q_j))$
  - for approx. range queries, $Q_1, Q_2, \ldots, Q_m$,
    $r_b : \max_j \max_{q \in Q_j} dist(f, q)$



Theorem:

Given a set of n points in R^d, a splay quadtree storing these points can answer:

- m box queries, in $O(m \log(|W| + (1/c)^{d-1}) + n \log n)$ time,
- m approximate nearest neighbor queries, or m approximate range queries, in $O(m (1/\varepsilon)^{d-1} \log(|W| + (1/c)^{d-1}) + n \log n)$ time.

Scalable Methods for the Analysis of Network-Based Data