SURVEY OF THE QUERY TIME OF RANDOM KD-TREES

Christopher Bireley
University of Utah
CS 6160
Spring 2010
Abstract

- Determine the running time of KD-Trees
- Does the average case really lead to $O(n^{1/2})$?
- Hypothesis: The running time is much better
The KD-Tree

- Binary Tree composed of multidimensional points / nodes
- 1…k dimensions leads to the name KD-Tree
- Divide the space with hyper-planes
- Alternate dimensions to split at each level
Generic KD-Tree Query

Query \((v, R)\)

if \(v\) is a leaf

report the points stored at \(v\) that lie in \(R\)

else if node rectangle(\(v_{\text{left}}\)) is inside \(R\)

Report_Subtree(\(v_{\text{left}}\))

else if node rectangle(\(v_{\text{left}}\)) intersects \(R\)

Query(\(v_{\text{left}}, R\))

if node rectangle(\(v_{\text{right}}\)) is inside \(R\)

Report_Subtree(\(v_{\text{right}}\))

else if node rectangle(\(v_{\text{right}}\)) intersects \(R\)

Query(\(v_{\text{right}}, R\))
Variation of the KD-Tree

More dynamic and scalable

Multi-way tree with all the leaves on the same level

Goal is to optimize I/O bounds: $O\left( \frac{n}{B}^{\frac{1}{2}} + \frac{n}{B} \right)$

No-overlapping of partitions and has dimension-independent fan-out
SH-Tree

- Extension of the hybrid-tree
- Enables fan-out of each node to be independent of the number of dimensions
- Faster insertions, deletions, and searches on average
- Guarantees untilization with dimension independent fan-out
Sampling

- Data sizes ranging from 500 to 16,000
- Iterations ranging from 5,000 down to 1,000
- Goal: total run times should be the upper end of the same order of magnitude in running time
Random Number Generation

- Originally to use three approaches: 
  rand() 
  jittered rand() 
  my random number generator 
- Abandoned due to uninteresting results
Three C implementations on one computer
Difficulty in finding such computer due to deprecated libraries
Solution: remote log in to old Linux box
Analysis

- Break down total running time
- Subtract overhead
- Solve for constant based on $O( n^{1/2} )$
- Plot against $O( n^{1/2} )$ and $O( \log(n) )$
Traditional KD-Tree

Graph showing the performance of Traditional KD-Tree, log(n), and sqrt(n) as functions of input size (n). The Traditional KD-Tree performance is higher than log(n) and sqrt(n) for larger input sizes.