CS 615 Skyline Queries in Databases

Assignment 2

By Swapnil Mhamane (15111044)

Objective:

To understand the basics of indexed skyline algorithms though implementation of the BBS algorithm using an R-tree simulating of disk behaviour.

Overview:

BBS, acronym for *Branch and Bound Skyline* algorithm basically works on datasets indexed by R-tree data structure. It is an extension to nearest neighbour search algorithm. In R-tree the data is stored in nodes, each representing minimum bounding rectangle(MBR) for tuples within that node. So dataset is basically divided in different node or to be more precise say MBR in hierarchical structure. In disk based system, each node is stored in different data file. So it's clear that reading tuples from each node requires disk IO operation which is time consuming. BBS avoids the disk access by pruning nodes not containing any skyline objects. Also it just access node containing skyline object only once. So it optimizes the time of disk access. For this, it uses pruning strategy to avoid accessing nodes dominated by skylines points found till moment. It is also responsive enough to return skyline objects found while search for other skyline objects is still in progress.

Steps to Run attached program:

On command prompt run the following command

- java –jar BBSSkyline.jar "dataSetFilePath" "QueryFilePath" Y

Where third argument is Y/N Boolean indicating whether to print skyline tuple to output file. Output will be written to file "dataSetFilePath.out"

Analysis:

Algorithm	Parameters	Smaller Data Set	Correlated Data Set	Anti-Correlated Data Set	Random Data Set	Larger Data set
	No. of Objects	**4**	**1000**	**1000**	**1000**	**3000**
	No. of Skyline Points	**2**	**13**	**705**	**166**	**207**
BNL	**No. of Comparisons**	4	1603	315909	34442	115177
	Running Time	62 ms	2016 ms	4890 ms	2438 ms	5520ms
SFS	**No. of Comparisons**	3	1101	265134	17201	31174
	Running Time	516 ms	1950 ms	3032 ms	2000 ms	4484 ms
BBS	**No. of Comparisons**	4	334	510841	33273	58625
	RTree Building Time	18ms	586ms	540ms	593ms	1095ms
	In-memory BBS	1ms	1ms	111ms	20ms	27ms
	File based BBS	15ms	146ms	527ms	428ms	618ms

Observations:

- **Kind of Data:**
  - As we can observe from the result table that as the correlation between data set goes on decreasing no. of skyline points, for same data set length, goes on increasing, so is the no. of object to object comparisons and execution time.
  - Since for correlated data set probability of objects getting dominated by other correlated object increases, it results in lesser no. of skyline objects.
- **No. of Comparisons:**
  - With BBS algorithm no. of object to object comparisons decreases overall. But as the anti-correlation among data increases comparisons in BNL and SFS algorithms dominates it.
  - Pre-sorting in SFS algorithm helps in *reduction of number of object to object comparison.*
- **Running Time:**
  - Total execution time for finding all skyline points in data set using BBS algorithm is very much optimized provided data set is stored in R-tree structure. But it even with consideration of R-tree build up time BBS takes comparatively less time than BNL or SFS algorithm.
- **Disk Based simulation using file :**
  - Although this is implementation concern, With enough availability of memory if we work on data rather r-tree nodes completely in memory takes less time.
  - Storing nodes in file and loading them in memory as needed includes IO operation overhead

Conclusion:

So we can say in that BBS algorithm is more optimized than BNL and SFS algorithm takes less time in finding skyline objects.