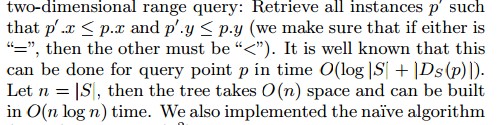
PODS09：comparing with Priority search tree algorithm and naïve algorithm. Both have the worst case O(n2). PST is build upon O(n) space, O(nlogn) time.

Naïve O(mn2): m is the object num. and n is the whole instance number.

PODS09: O(n5/3(log n)2/3) for two-dimensional

O(n2-1/(d+1)) for high dimensional

Priority search tree algorithm: which queries the priority search tree built upon the data set S to ﬁnd all instances that dominate a given instance p, i.e. the dominance set DS(p). The skyline probability of p then can be computed from Equation 1. The reason that we can leverage PST for ﬁnding DS(p) is that this problem can be converted to a two-dimensional range query:



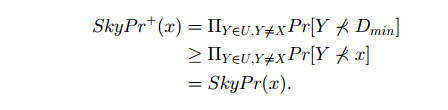
VLDB07: comparing BU, TD with EX(exhaustive algorithm)

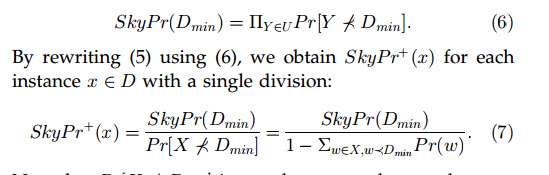
Dataset: 10,000 objects, [1,400] instances per object.

In Anti, 40% of objects are pruned. In Corre, 90% of objects are pruned.

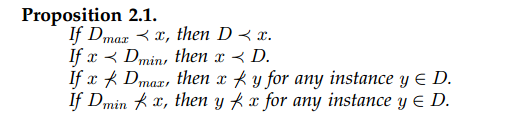
TKDE(2012), comparing with naïve and PODS09. The both compute all skyline points.

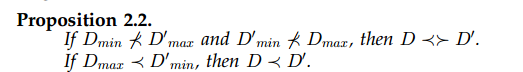
This paper, using three propositions, prunes all objects which don’t need to be considered.

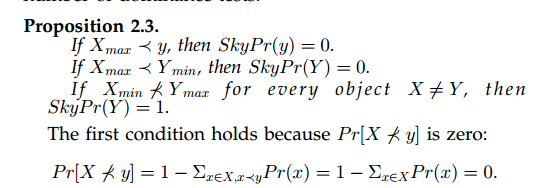












When z-tree is being built, it only considers the instance ID. When the comparison is held after building, it always hold an object array for every instance in a node box, compare all other nodes it traverse in the tree, ignore instance whose object is the same as the target instance.

ICDT 11: It introduced two approaches to this problem.

One, O(m3/2), m is the number of the whole instances. It set up a KD-tree for all instances. It compute skyline of every node from the root to leaves. Intermediate results helps the computation of the following nodes.

Two, O(min{n,k}mlogm), n is the object number and k is the instance number of one obj. The general idea is almost the same as in PODS09. It create grid for following computation.