PRINCIPLES OF DATA ORGANISATION

Spatial join for Internal Memory

MOTIVATION

- Key, pointer pairs ~ index
- Non-spatial join
- Spatial join in main memory
- We focus only on intersection joins

NESTED LOOP JOIN

- The algorithm is identical to the standard relational one
  - It works with arbitrary object type and join condition
- Given datasets A and B the join takes $O(|A| \times |B|)$ time
  - Suitable for small datasets

NESTED_LOOP_JOIN(setA, setB, joinCondition)
INPUT: Sets to join and condition based on which the join happens
OUTPUT: Pairs of objects satisfying the join condition

```
FOREACH a ∈ setA
   FOREACH b ∈ setB
      IF Satisfied(a, b, joinCondition) THEN
         REPORT(a, b);
```

INDEX NESTED LOOP JOIN

- Variant of nested loop join where first a spatial index is created over one of the sets
- The indexed set is queried by each of the objects from the second set for intersection (window query)
- Given datasets A and B, the join takes $O(\log(|A|) \times |B|)$ time
  - Not including time needed for building the index

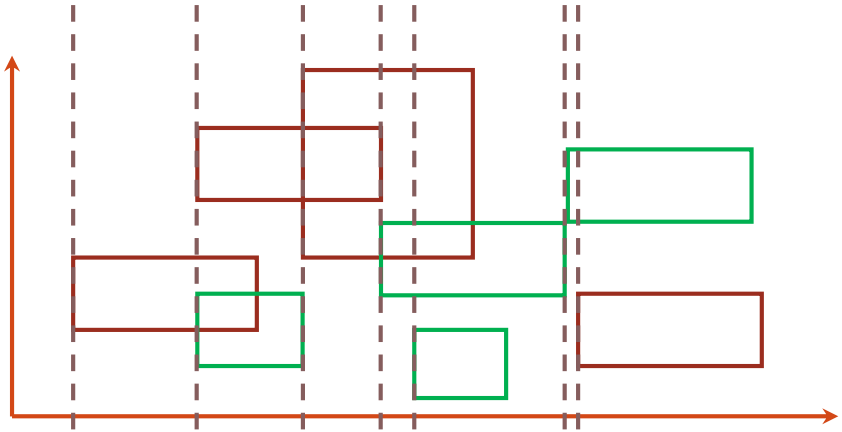
```
INDEX_NESTED_LOOP_JOIN(setA, setB)
INPUT: Sets to join
OUTPUT: Pairs of intersecting objects

ix := CreateSpatialIndex(setA)
FOREACH b ∈ setB
   REPORT(ix.Search(b));
```

PLANE SWEEP

- Similar to non-spatial sort-merge join
- Two-phase algorithm for identification of intersecting rectangles:

1. Sorting the rectangles in ascending order based on their left sides (x-axis)
2. Sweeping a vertical scan line through the sorted list from left to right, halting at each of the rectangle's lower **x** coordinates and identification of rectangles intersecting (vertical line) with the current rectangle and checking for intersection based on the y-axis



PLANE SWEEP

- Sweep structure tracks active rectangles and has to support three operations
  - **Insert**
  - **RemoveInactive**
    Removes from the active set all rectangles that do not overlap a given rectangle (line)
  - **Search**
    Searches for all active rectangles that intersect a given rectangle and outputs them
- If the data are sorted, only the data in the sweep structures need to be kept in internal memory
- Given datasets A and B the algorithm takes $O(\log(n)\times n)$ where $n=|A+B|$
  - Including the list sort time

PLANE SWEEP: ALGORITHM

```
PLANE_SWEEP(setA, setB)
INPUT: Sets of rectangles to join
OUTPUT: Pairs of intersecting rectangles

listA ← SortByLeftSide(setA);
listB ← SortByLeftSide(setB);
sweepStructA ← CreateSweepStructure();
sweepStructB ← CreateSweepStructure();
WHILE NOT(listA.End()) || NOT (listB.End()) DO
   IF listA.First() < listB.First() THEN
      sweepStructA.Insert(listA.First());
      sweepStructB.RemoveInactive(listA.First());
      sweepStructB.Search(listA.First());
      listA.Next();
   ELSE
      sweepStructB.Insert(listB.First ());
      sweepStructA.RemoveInactive(listB.First());
      sweepStructA.Search(listB.First());
      listB.Next();
```