+

INFS4205/7205 Advanced Techniques for High Dimensional Data

Spatial Query Processing 2

Semester 1, 2021

University of Queensland

+ Advanced Techniques for High Dimensional Data

+ Advanced Spatial Queries

- Last week
  - The Filter-and-Refine approach
  - Some basic computational geometry algorithms
  - Intersection join algorithms using various spatial indexes
- Nearest Neighbor and Skyline queries
  - Very different query processing strategies
  - With applications beyond spatial databases
  - Lots of variations

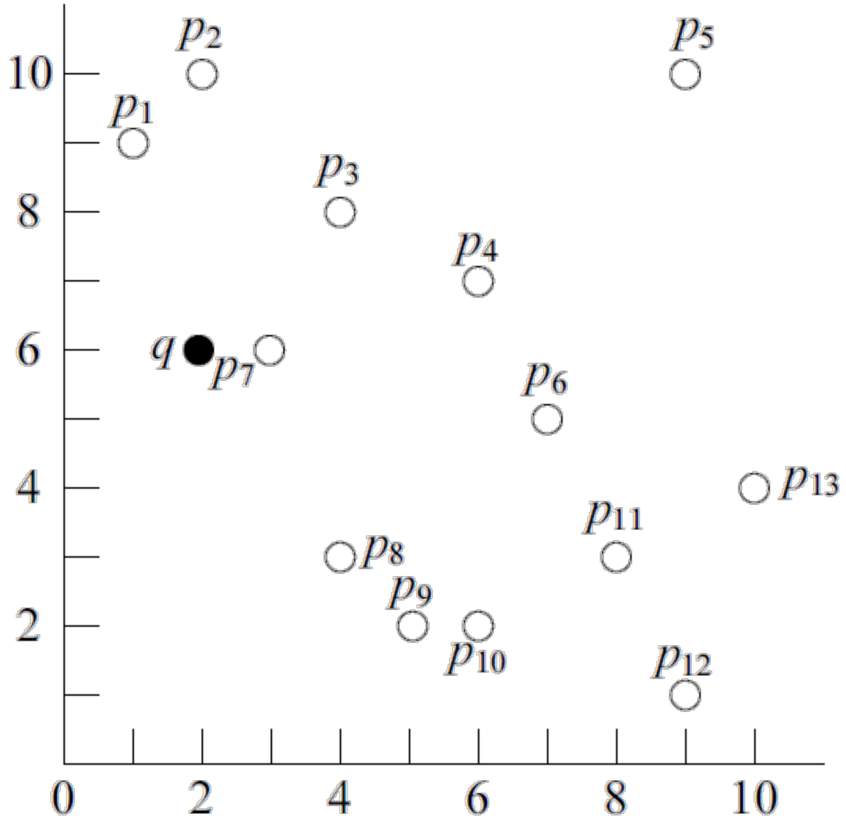
  - Examples of how to utilize / combine the indexes to solve new problems

+ Recommended Readings

- Nick Roussopoulos, Stephen Kelley and Frederic Vincent, Nearest Neighbour Queries, SIGMOD'95
- Stephan Börzsönyi, Donald Kossmann, Konrad Stocker, The Skyline Operator, ICDE'01
- Dimitris Papadias, Yuefei Tao, Greg Fu and Bernhard Seeger, An Optimal and Progressive Algorithm for Skyline Queries, SIGMOD'03

+ K-NN Query Processing

- Nearest neighbour query
  - Given a point q, a set of points P and distance function d, find p in D such that for any point p' in D, $d(q,p) \leq d(q,p')$
    - The NN of q is p_7
    - Applications
      - Find the McDonald that is nearest to me
      - Find the customer profile in the database that is most similar to the profile of the new customer
      - Retrieve the image from the database that is most similar to the one given by the user

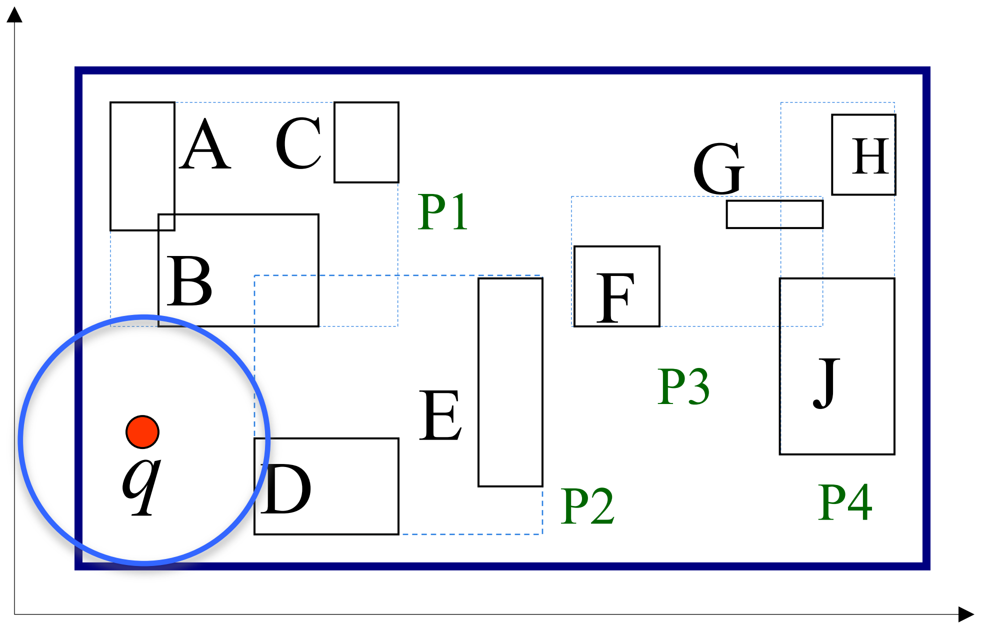


+ K-NN Methods

- Exhaustive search – the naïve method
  - To compute the distance from q to every point in P
  - What are the problems?
- Using spatial indexes – minimize the data accessed
  - How to decide if an index node (i.e., MBR) needs to be searched or not, and when to search it?
    - Goal: visit a node only when necessary – how?
  - Search with expanding ranges
    - Start with a circle centred at q, and increase the radius when necessary - can we do better?

+ R-Tree - NN search

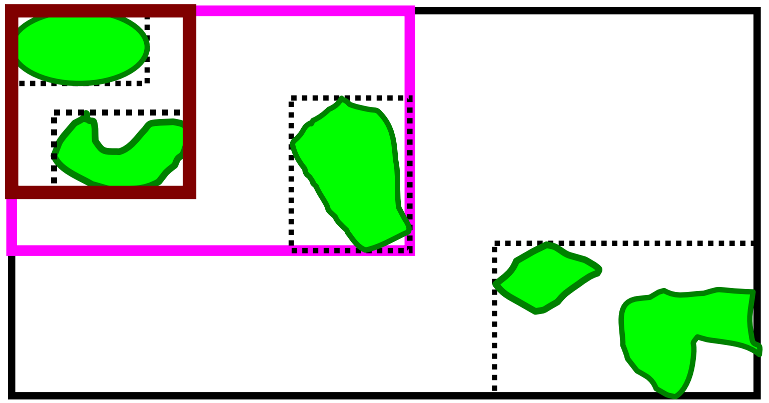
- Using a circle centered at q
  - Question: how to decide the radius?
  - Different strategies to set the initial size and to increase
- Or, selecting those MBRs which may contain the results
  - Question: which MBRs to check?
  - Estimate lower bound and upper bound of the distance from q to an MBR, such that within this range there must be one point

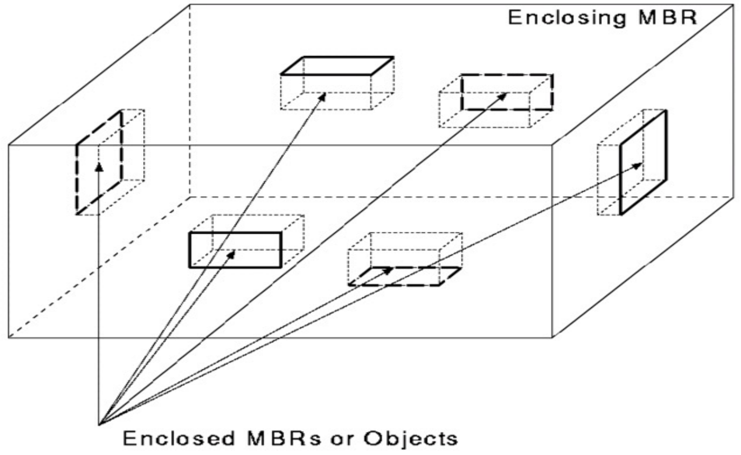


...a tight range is preferred, that is, we prefer the largest posisble lower bound and smallerest upperbound

+ MBR Face Property

- MBR is a multi-dimensional rectangle, which is the minimal rectangle that fully encloses an object or a set of objects
- MBR Face Property: Every face of the MBR contains at least one point of an object in the database

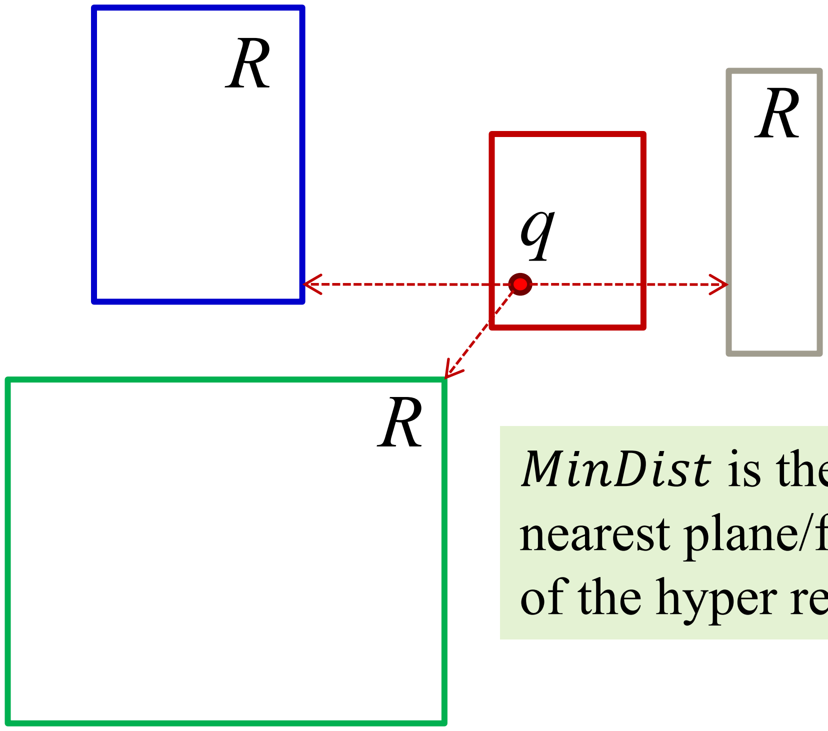


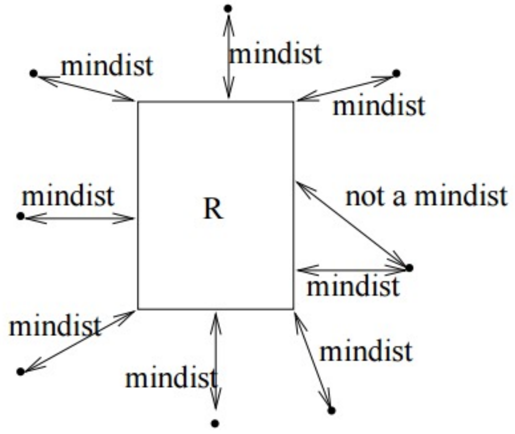


+ MinDist

A lower bound distance for any point in R to q

- If q is inside R, then $MinDist = 0$
- If q is outside of R, $MinDist$ is the distance of q to the closest point of R (one point of the perimeter)





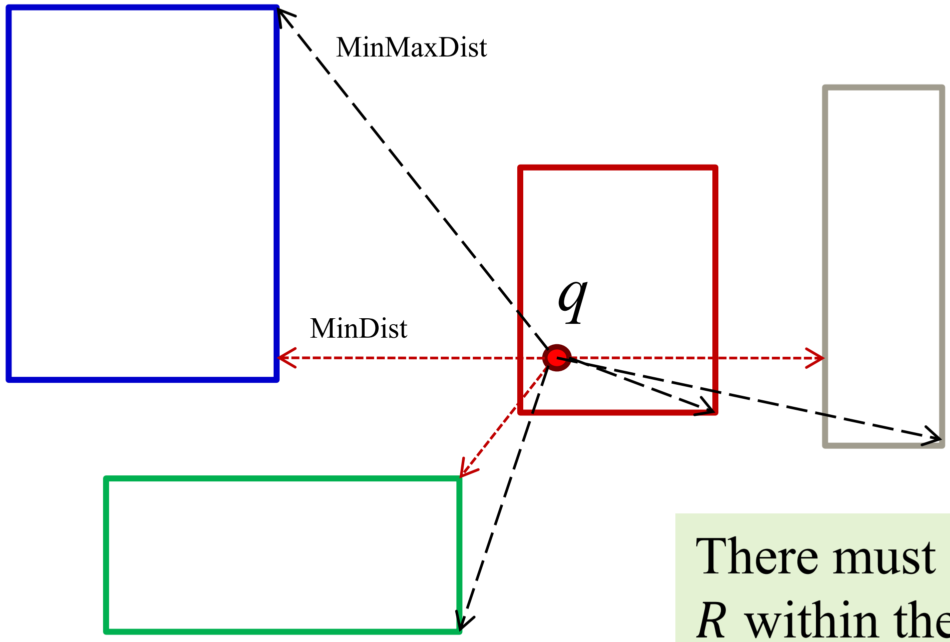
$MinDist$ is the distance of the point q to the nearest plane/face (nearest side in 2 dimension) of the hyper rectangle R.

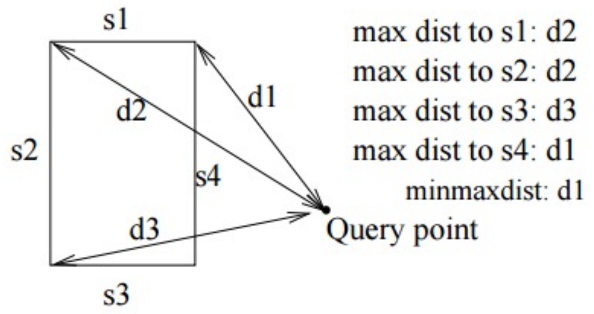
...to order and prune the search in an R-tree

+ MinMaxDist

An upper bound of distance from q to R

- For *each* dimension, find the closest face, compute the distance to the furthest point on this face and take the minimum of all these distances
- It is the smallest possible upper bound of distances from q to R
- There exists at least one point p in $R, d(p, q) \leq MinMaxDist$





There must exist a point of some spatial object in R within the distance $MinMax$ from the point q

+ Pruning Strategies

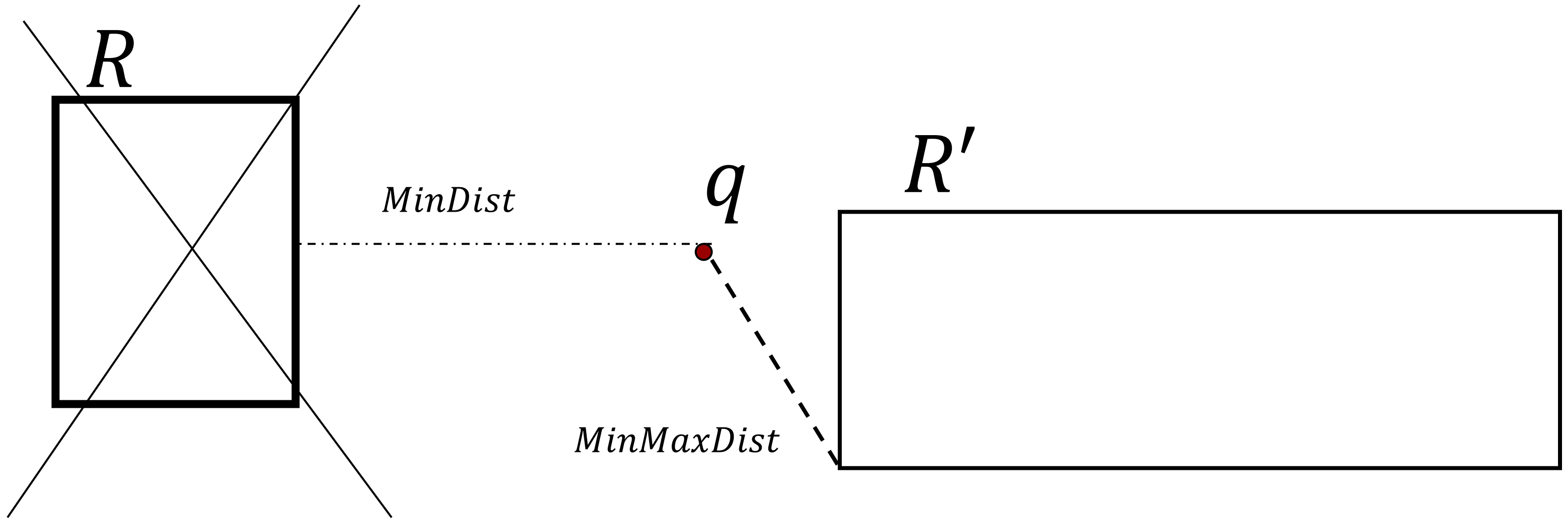
- Key observation

$$MinDist(q,R) \leq NN(q) \leq MinMaxDist(q,R)$$

- Strategy:
  1. An MBR R is discarded if there exists another R' such that $MinDist(q,R) > MinMaxDist(q,R')$
  2. An object q is discarded if there exists an R such that $d(q,p) > MinMaxDist(q,R)$
  3. An MBR R is discarded if a point p is found such that $MinDist(q,R) > d(q,p)$

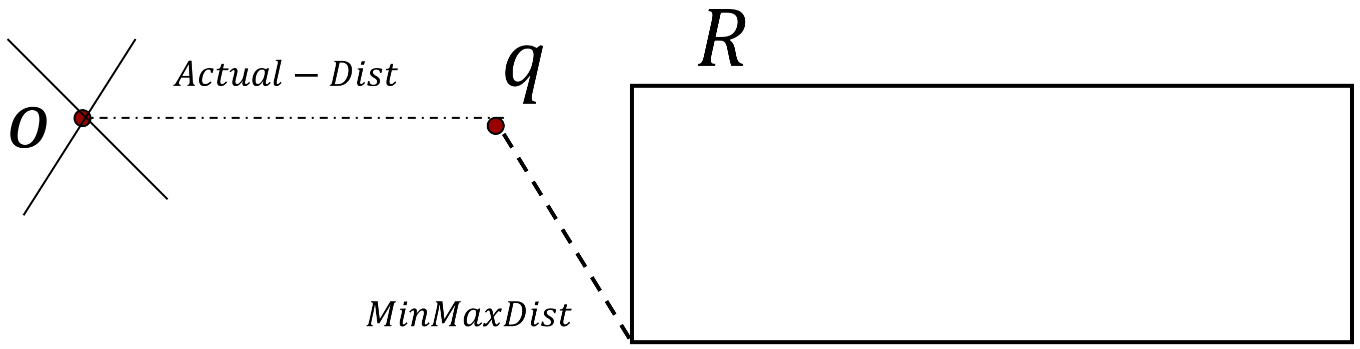
+ Pruning 1 Example

- An MBR R is discarded if there exists another R' s.t. $MinDist(q, R) > MinMaxDist(q, R')$



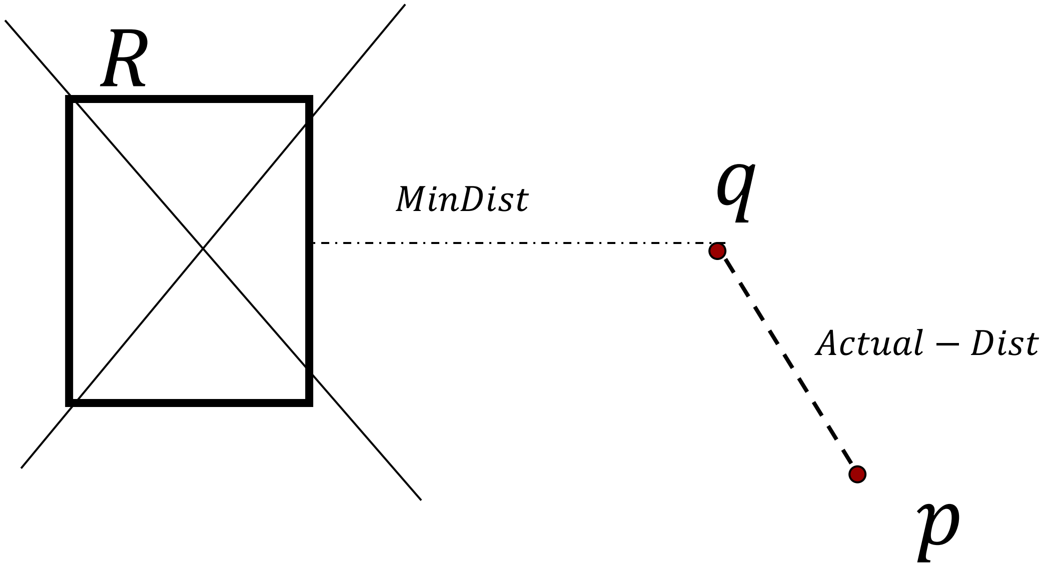
+ Pruning 2 Example

- An object o is discarded if there exists an R s.t. the $Actual - Dist(q, o)\ >\ MinMaxDist(q, R)$



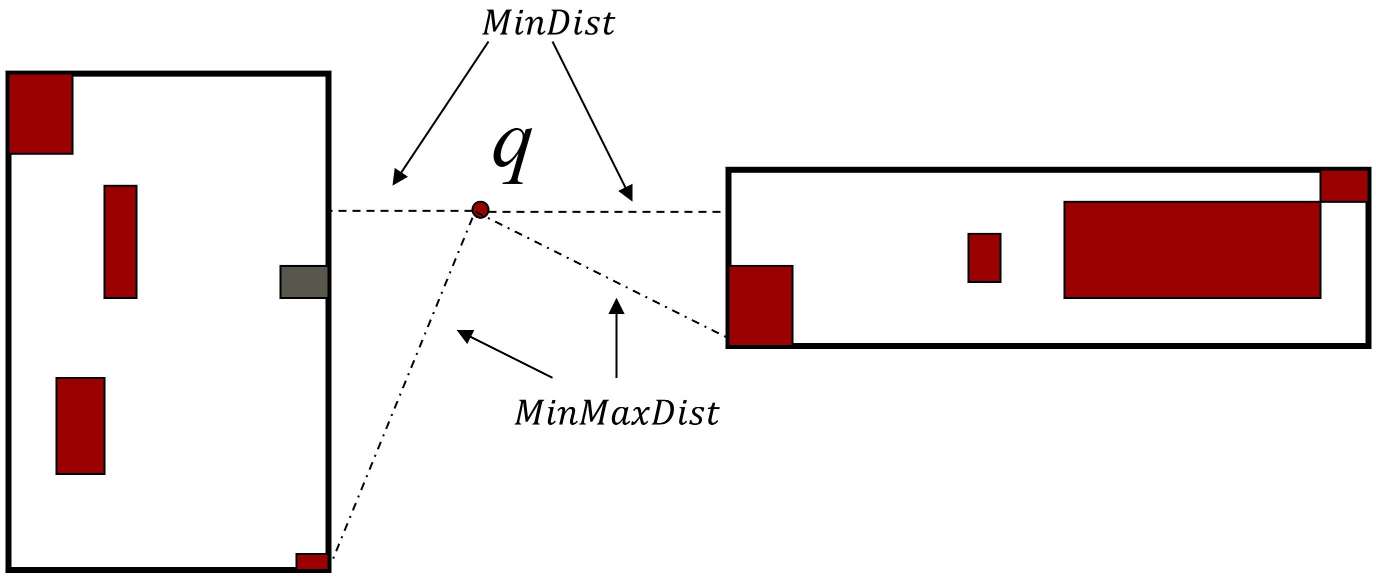
+ Pruning 3 Example

- An MBR R is discarded if an object p is found s.t. the $MinDist(q, R) > Actual - Dist(q, p)$



+ Ordering Distance

- $MinDist$ is an optimistic distance
- $MinMaxDist$ is a pessimistic one.



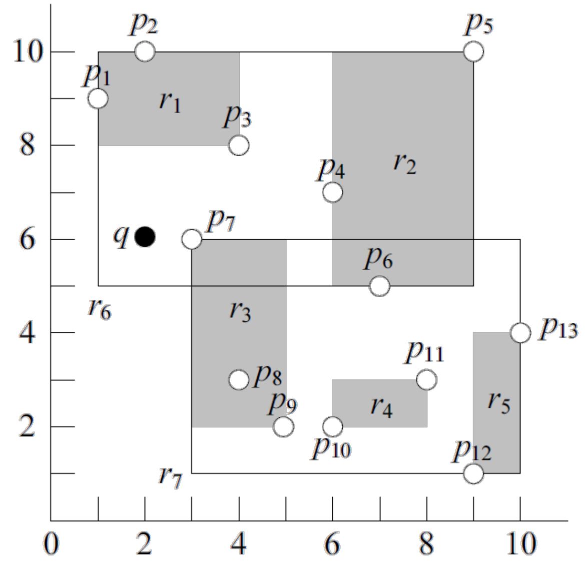
Which one is better? When to use them? Tutorial~

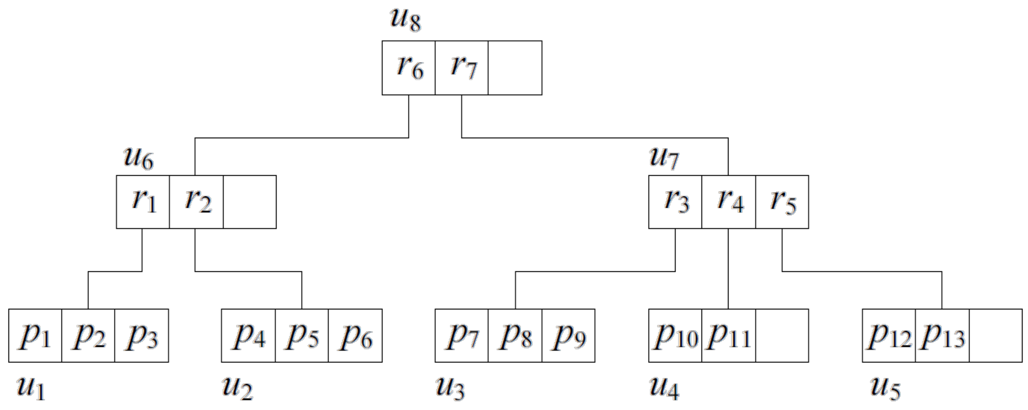
+ kNN Branch-and-Bound (BaB)

- Depth-First Traversal of the R-tree
  - Use $MinDist$ to prioritize the nodes for accessing
  - Prune the nodes that cannot contain the final answer

- Visit root u_8
  - r_6 and r_7
  - $MinDist(q, r_6) = 0 < MinDist(q, r_7) = 1$

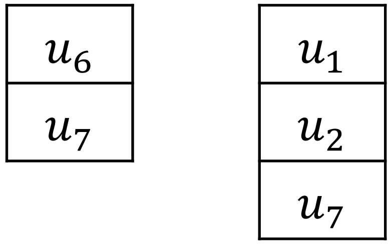
u_6
u_7

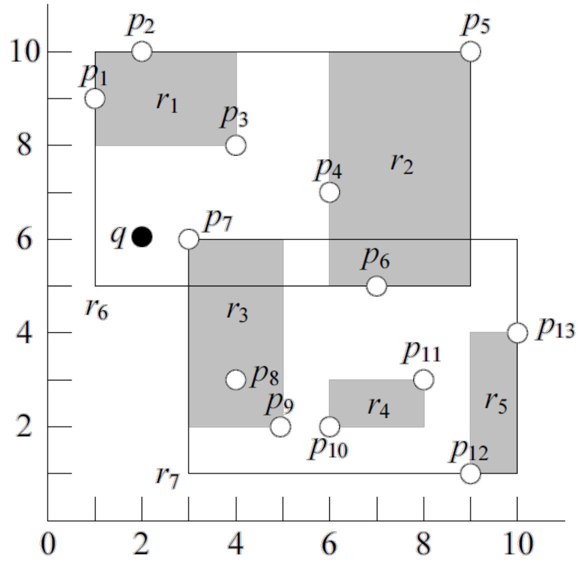


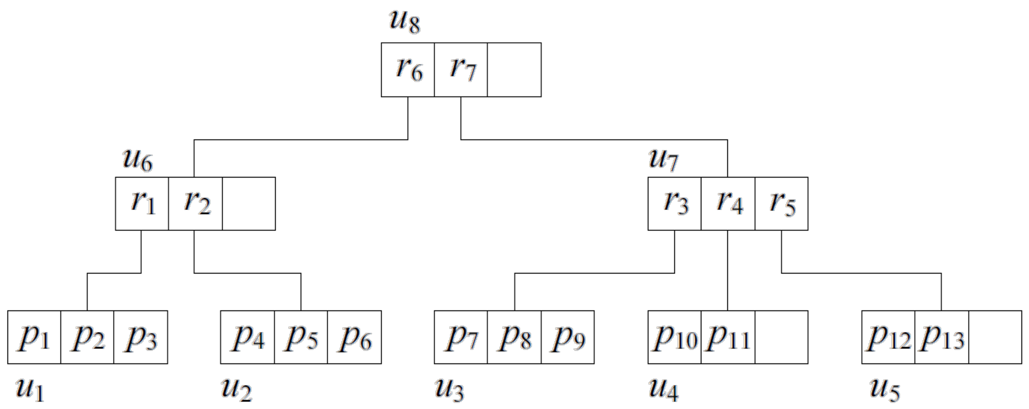


+ kNN Branch-and-Bound (BaB)

- At u_6
  - r_1, r_2
  - $MinDist(q, r_1) = 2 < MinDist(q, r_2) = 4$

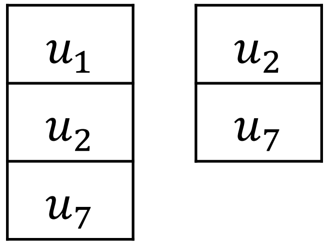


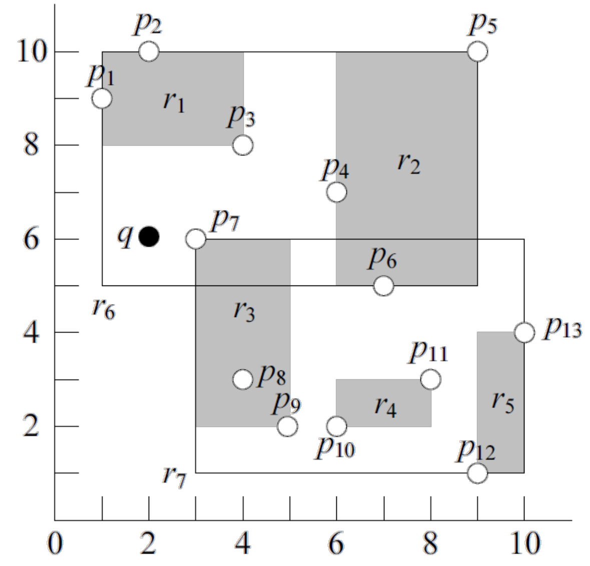


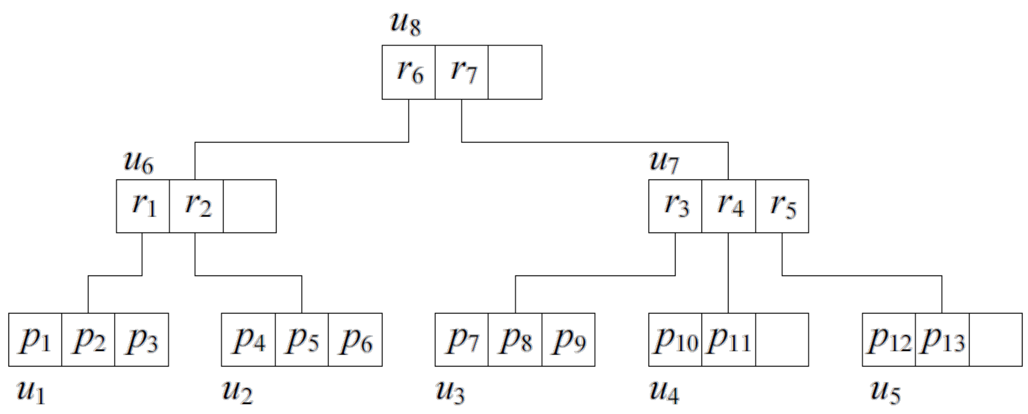


+ kNN Branch-and-Bound (BaB)

- At u_1
  - Compute the distance from q to each data points p_1, p_2, p_3
    - $d(q, p_1) = \sqrt{10}$
    - $d(q, p_2) = 4$
    - $d(q, p_3) = \sqrt{8}$
  - p_3 is the nearest so far

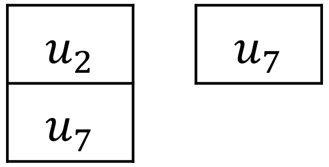


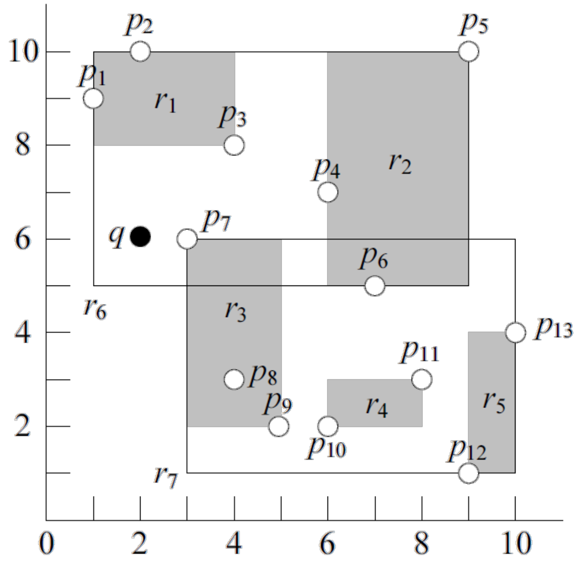


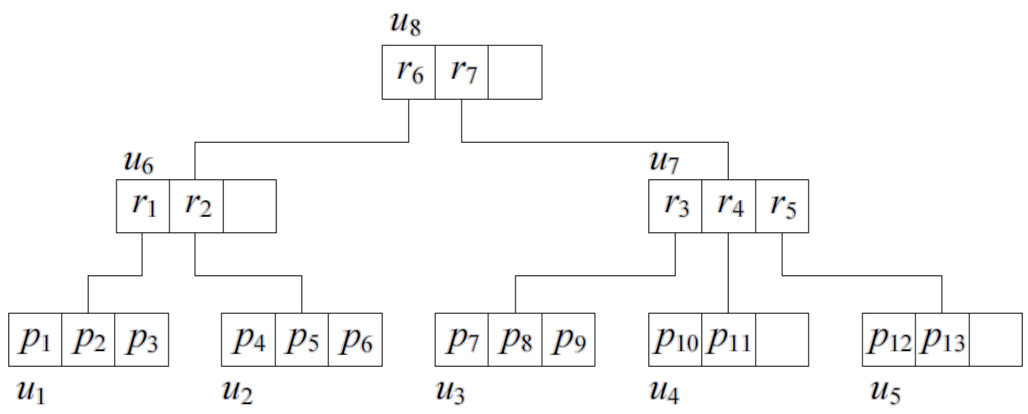


+ kNN Branch-and-Bound (BaB)

- At u_2
  - $d(q, r_2) = 4 > d(q, p_3) = \sqrt{8}$
    - Pruned r_2

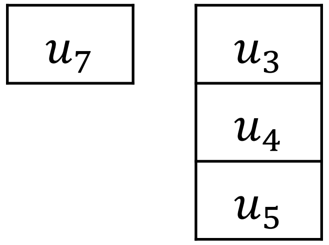


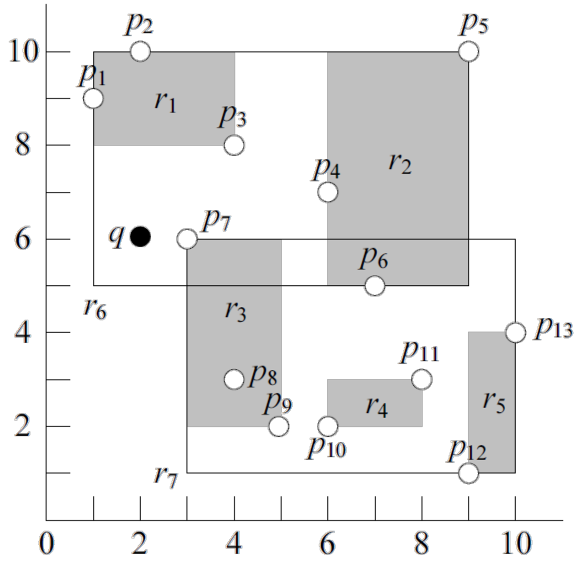


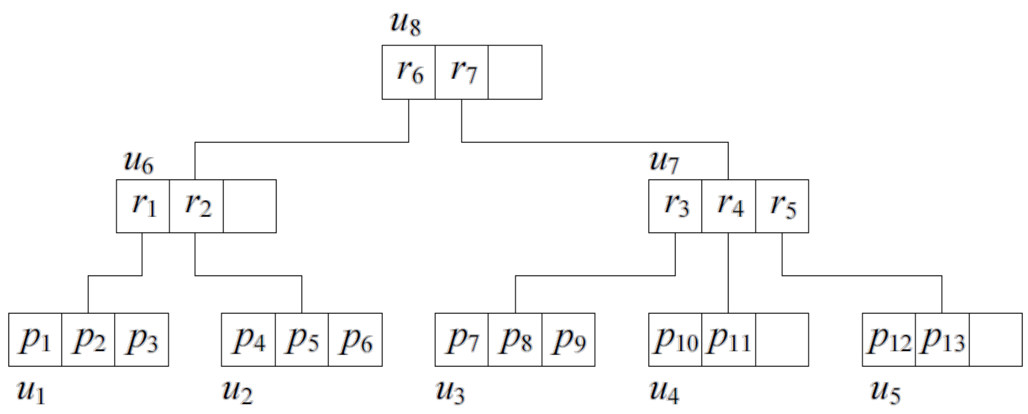


+ kNN Branch-and-Bound (BaB)

- At u_7
  - $d(q, r_7) = 1 < d(q, p_3) = \sqrt{8}$
    - Visit children of u_7: u_3, u_4, u_5
    - $d(q, r_3) = 1$



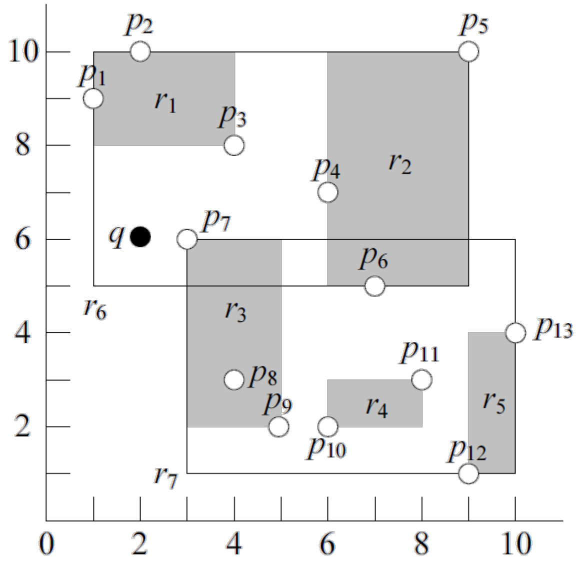


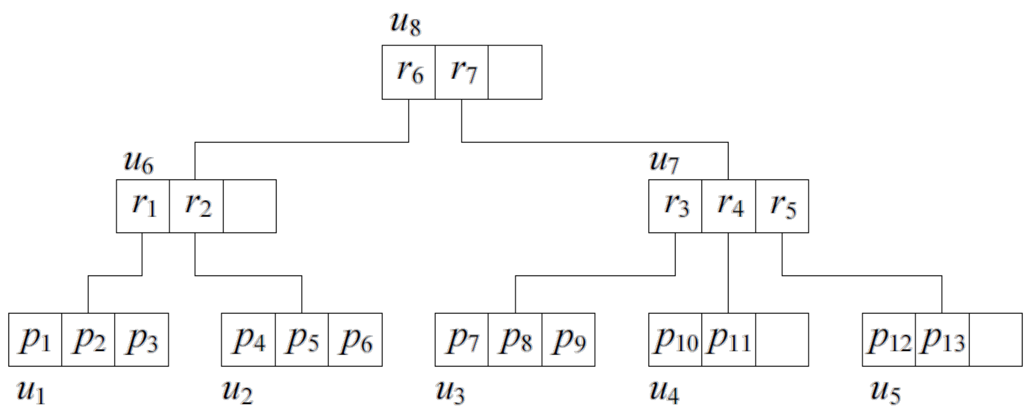


+ kNN Branch-and-Bound (BaB)

- At u_3
  - $d(q, p_7) = 1 < d(q, p_3) = \sqrt{8}$, NN is p_7 for now
    - $d(q, p_7) = 1 < d(q, r_4) = 5$, prune u_4
    - $d(q, p_7) = 1 < d(q, r_5) = \sqrt{53}$, prune u_5
  - p_7 is the final result

u_4
u_5



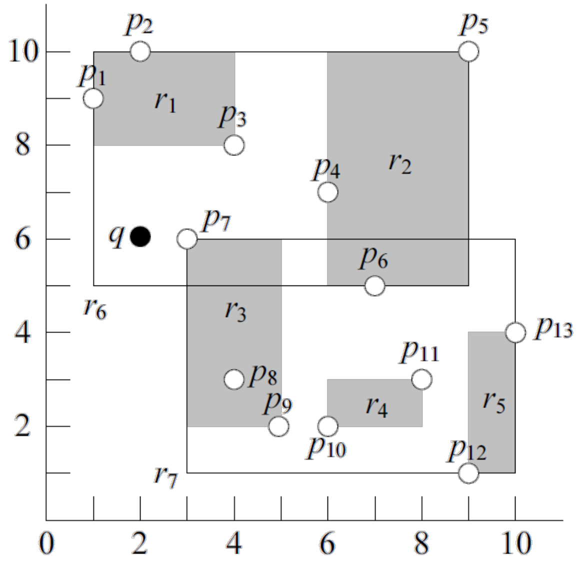


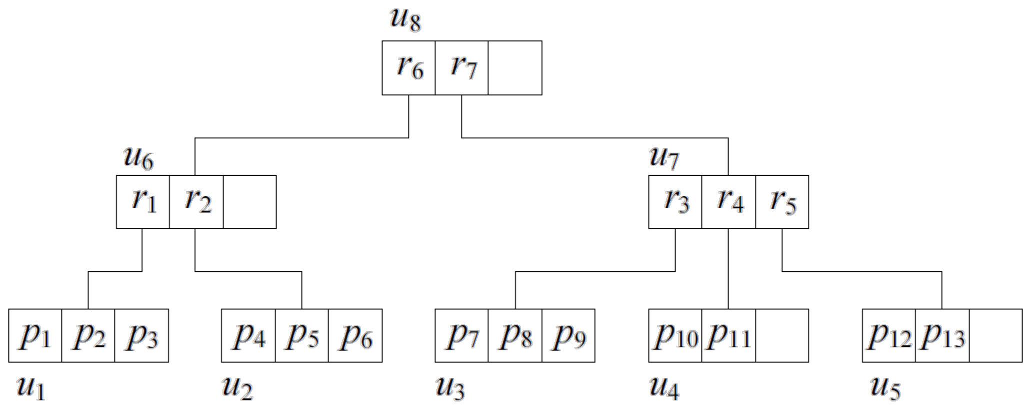
+ kNN Best First (BF)

- H: A sorted list with $MinDist$ as key
  - Use a heap

  - r_6 and r_7
  - $MinDist(q, r_6) = 0 < MinDist(q, r_7) = 1$

$r_6, 0$
$r_7, 1$



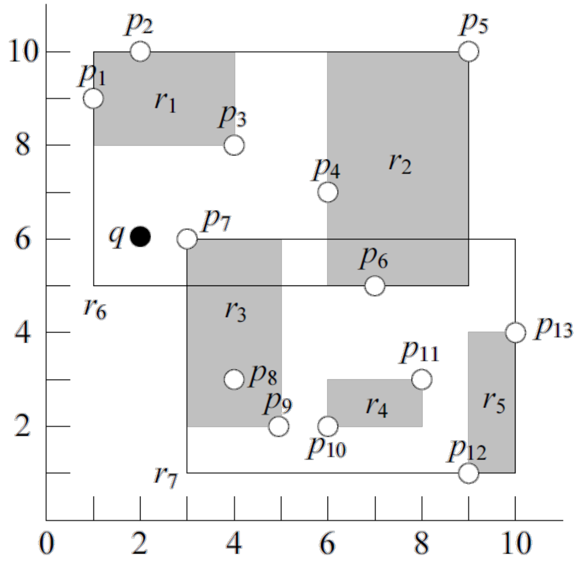


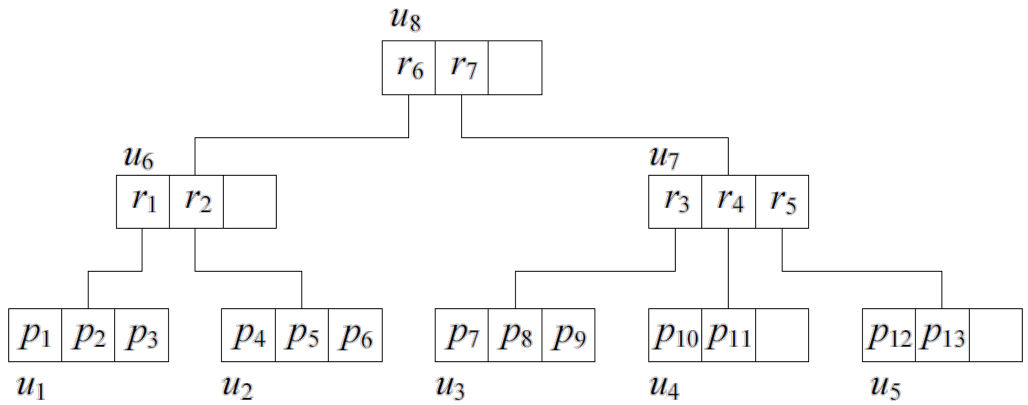
+ kNN Best First (BF)

- Visit u_6
  - r_1 and r_2
    - $MinDist(q, r_1) = 2$
    - $MinDist(q, r_2) = 4$

$r_6, 0$
$r_7, 1$

$r_7, 1$
$r_1, 2$
$r_2, 4$



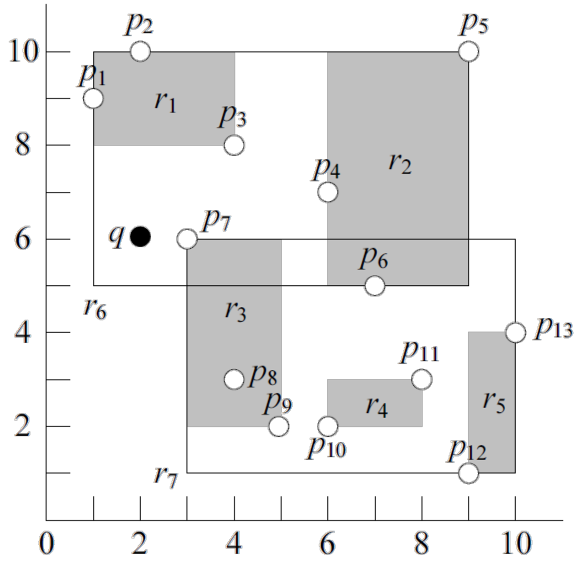


+ kNN Best First (BF)

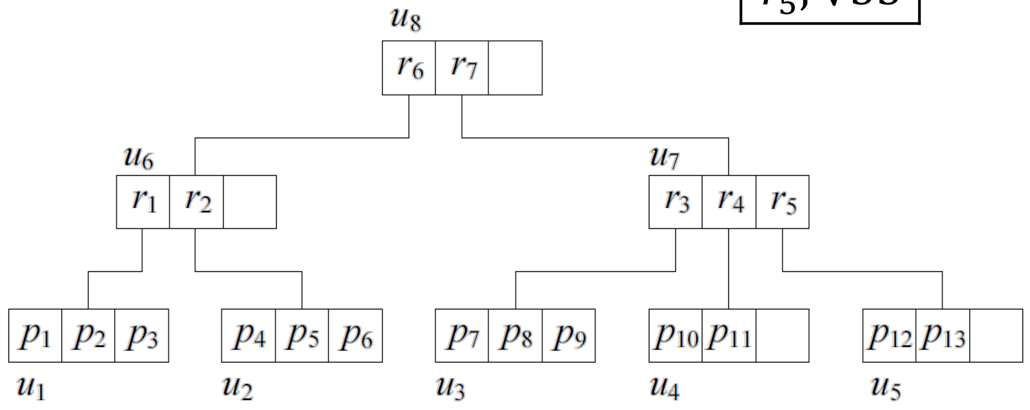
- Visit u_7
  - r_3, r_4 and r_5
    - $d(q, r_3) = 1$
    - $d(q, r_4) = 5$
    - $d(q, r_5) = \sqrt{53}$

$r_7, 1$
$r_1, 2$
$r_2, 4$

$r_3, 1$
$r_1, 2$
$r_2, 4$
$r_4, 5$
$r_5, \sqrt{53}$



p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12} p_{13}
q
r_1 r_2 r_3 r_4 r_5 r_6 r_7
0 2 4 6 8 10

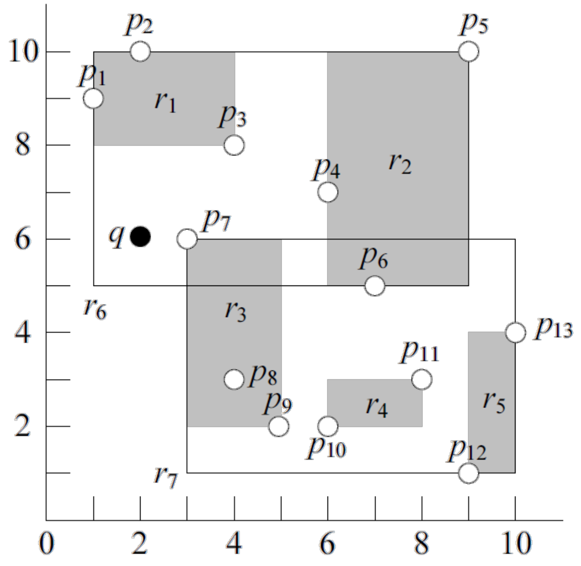


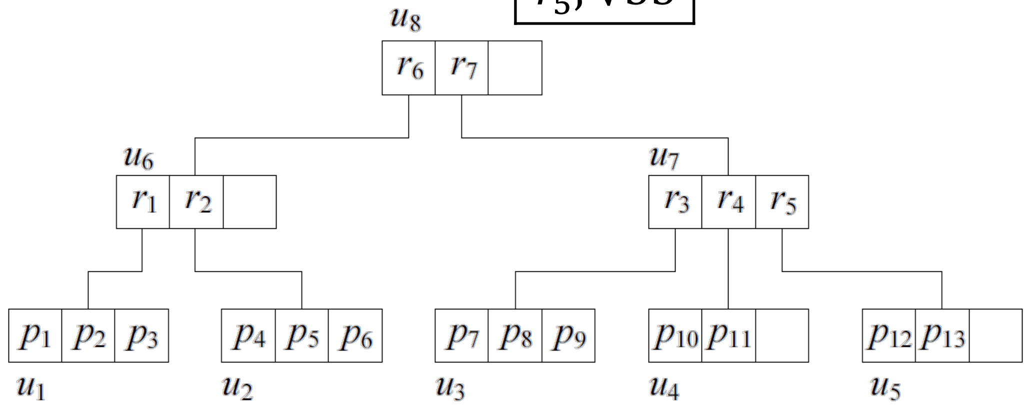
+ kNN Best First (BF)

- Visit u_3
  - p_7, p_8 and p_9
    - $d(q, p_7\ ) = 1$
    - $d(q, p_8) = \sqrt{13}$
    - $d(q, p_9) = 5$
  - 1<2, p_7 is the nearest neighbour

$r_3, 1$
$r_1, 2$
$r_2, 4$
$r_4, 5$
$r_5, \sqrt{53}$

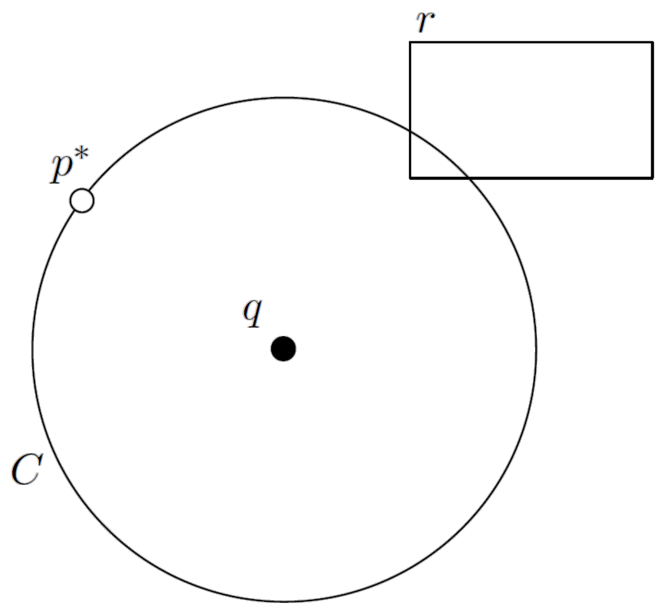
$r_1, 2$
$r_2, 4$
$r_4, 5$
$r_5, \sqrt{53}$





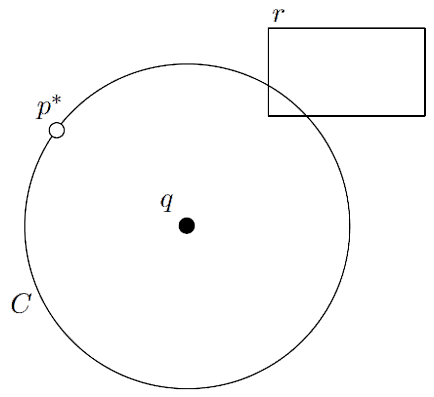
+ Optimality of BF

- C: A circle centers at q, with radius $d(q, p^*)$
  - p^* is an arbitrary NN of q
  - S^*: All the nodes whose MBRs intersect C
    - All the algorithm must access all the nodes in S^*
    - Assume, for example, that the node with MBR r in the figure below was not accessed. How could the algorithm assert that no point in r is closer to q than p^* ?



+ Optimality of BF

- BF accesses only those nodes whose MBRs intersect C
  1. BF accesses MBRs in non-decreasing order of their $MinDist$ to q
     - Let r_1 and r_2 be two MBRs accessed consecutively
     - r_2 either already existed in H when r_1 was visited, or r_2 is an MBR inside r_1
       - In either case, it must hold $MinDist(q, r_2) \geq MinDist(q, r_1)$
  2. Let r be the MBR of a leaf node containing an arbitrary NN of q, r' be an MBR that does not intersect C
     - r is visited before r'
     - When r is found, BF must necessarily discover p^*
       - Won't access r'



+ K-NN search

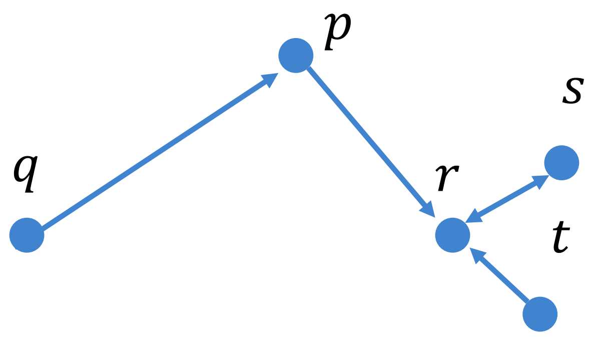
- Just a simple extension
  - Keep the sorted buffer of at most k current nearest neighbors
  - Pruning is done using the k-th distance

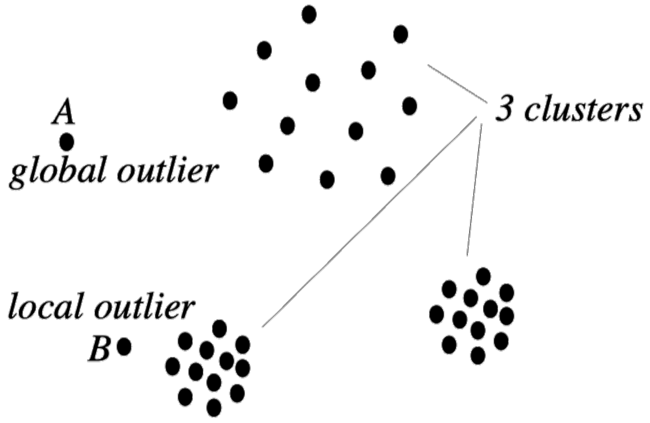
+ K-NN Query Processing

- Many variations
  - *k*NN
    - Find k nearest neighbours
  - Distance ranking
    - Browse points by their distances to the query point
  - Constrained space
    - Distance is calculated using environment data, such as terrain surface, road networks
  - Constraint keyword
    - With labels
  - k furthest neighbours
    - Away from rubbish dump
  - Reverse *k*NN
    - For asymmetric relations
  - Aggregate *k*NN
    - Distance sum to a group of points
  - Continuous *k*NN
    - When query point is moving
  - …

+ Reverse-NN search

- Nearest neighbors need not be symmetric
  - Nearest neighbor is indicated by the arrow
    - NN of q is p
    - NN of p is r
    - NN of r is s
    - NN of t is r
    - NN of s is r

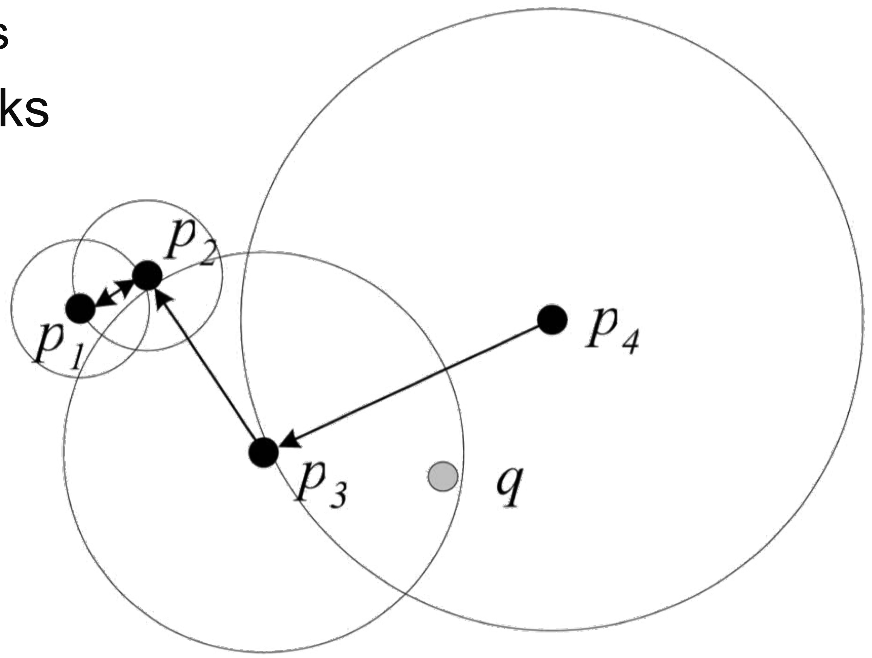




- Applications
  - Business Location Planning
    - How many customers are nearer to me than to my competitors?
  - Profile-based Marketing
    - How many new users will regard the new plan as the best match?
  - Outlier Detection
    - Point without an RNN might be an outlier

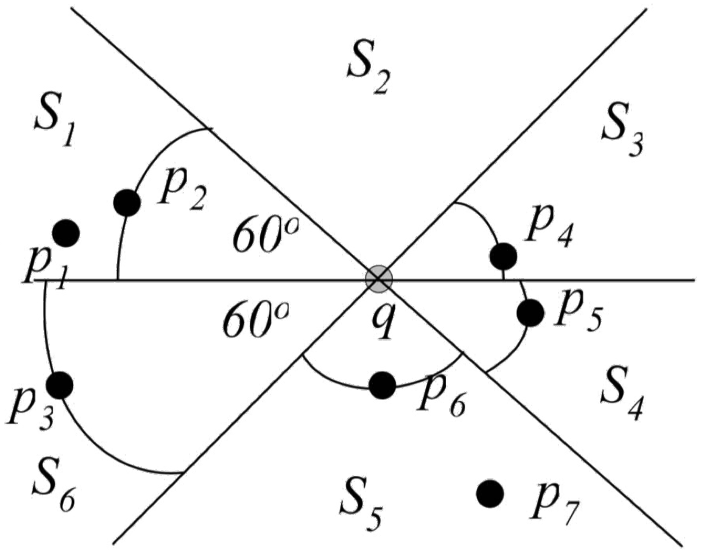
+ Reverse-NN search

- RNN-Tree
  - Pre-computed all NN results of all the points
  - One disk of each point
    - Center at p, radius of p to its NN
    - All points falling into my disk is my nearest neighbor
  - RNN query becomes a point query
    - All the disks cover q are the results
  - Use R-Tree to organize these disks



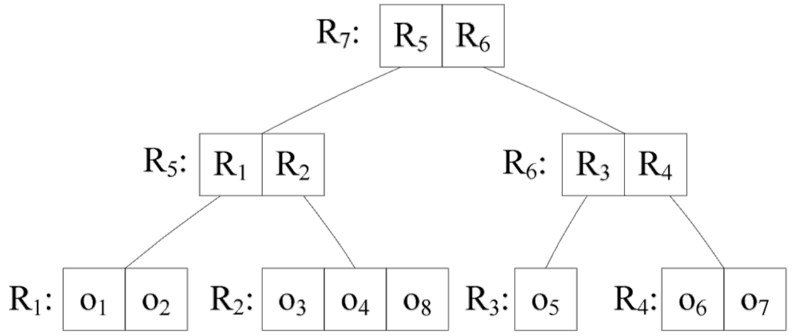
+ Reverse-NN search

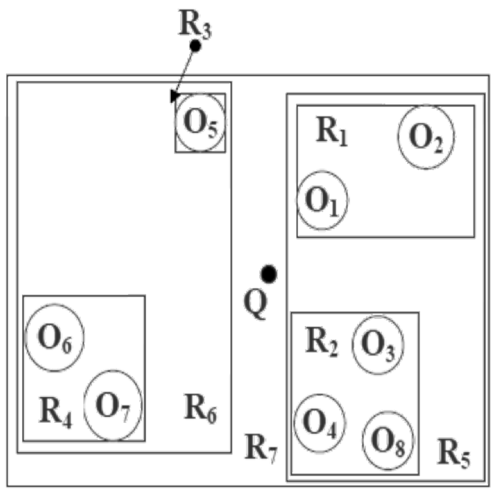
- Space Dividing
  - Divide the space into six equal-degree regions S_1 to S_6
  - p is the NN of q in some region S_i
    - Either $p \in RNN(q)$ or there is no RNN of q in S_i
      - S_1: NN of q is p_2, but p_2's NN is p_1, so prune search in S_1
      - Also prune S_2, S_3, S_4, and S_6
      - S_5: NN of q is p_6, and p_6's NN is q, so $RNN(q) = \{p_6\}$
  - Two Steps
    1. Find the candidates in six regions
    2. Refine



+ Keyword-based KNN

- Text Description + Location
  - Boolean Range Query
    - Retrieve objects containing all the query keywords and in the query region
  - Boolean kNN Query
    - Ranked according to their distance
  - Top-k kNN Query
    - $\alpha \cdot Dist(q, o) + (1-\alpha) \cdot TRel(q, o)$

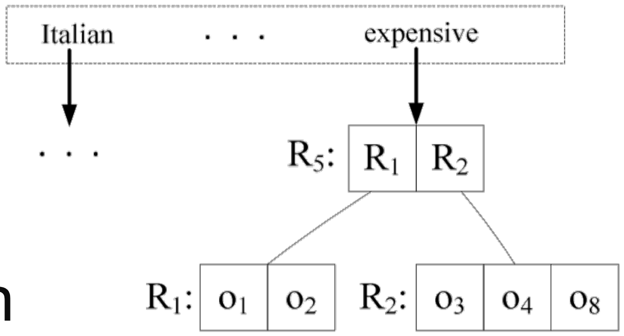


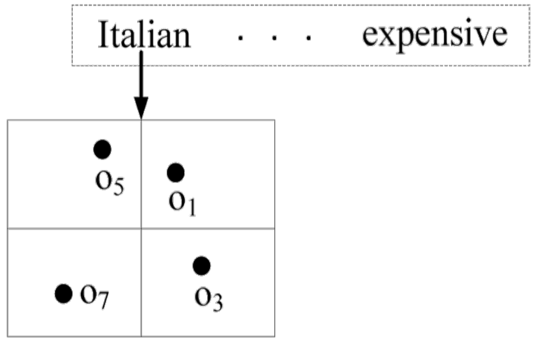


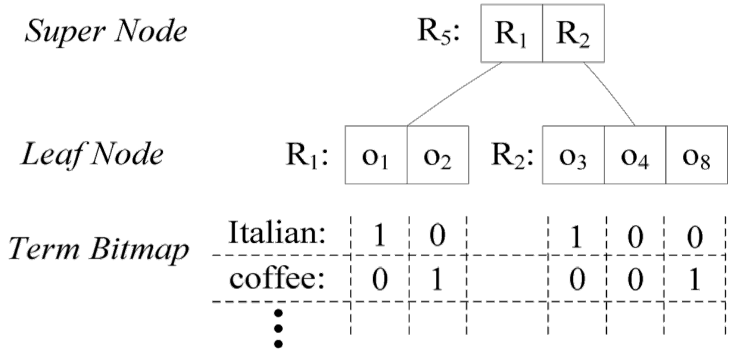
Object	Terms and term frequencies
o_1	(Italian, 5)(restaurant, 5)(expensive, 2)
o_2	(coffee, 5)(restaurant, 5)(expensive, 1)
o_3	(Italian, 7)(pizza, 1)(expensive, 1)
o_4	(restaurant, 7)(pizza, 1)(expensive, 1)
o_5	(Italian, 4)(restaurant, 4)
o_6	(coffee, 4)(restaurant, 3)
o_7	(Italian, 1)(coffee, 1)(restaurant, 4)(pizza, 1)
o_8	(coffee, 3)(restaurant, 3)(expensive, 1)

+ Keyword-based KNN

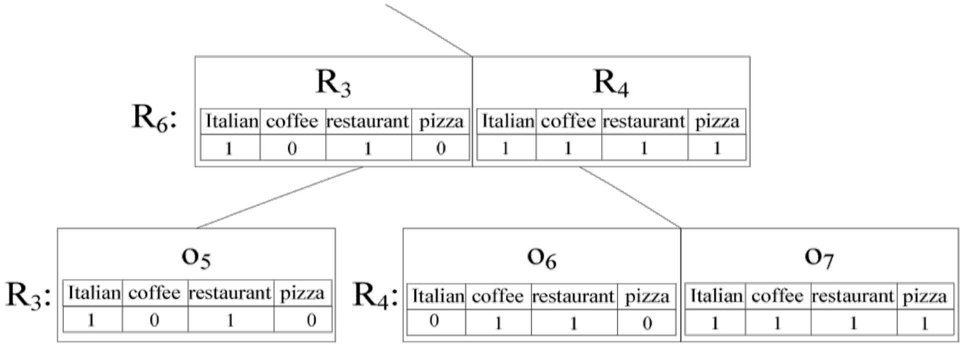
- Text First
- Spatial First
- Combination







	o1	o2	o3	o4	o8
Italian:	1	0	1	0	0
coffee:	0	1	0	0	1
⋮					



R6:

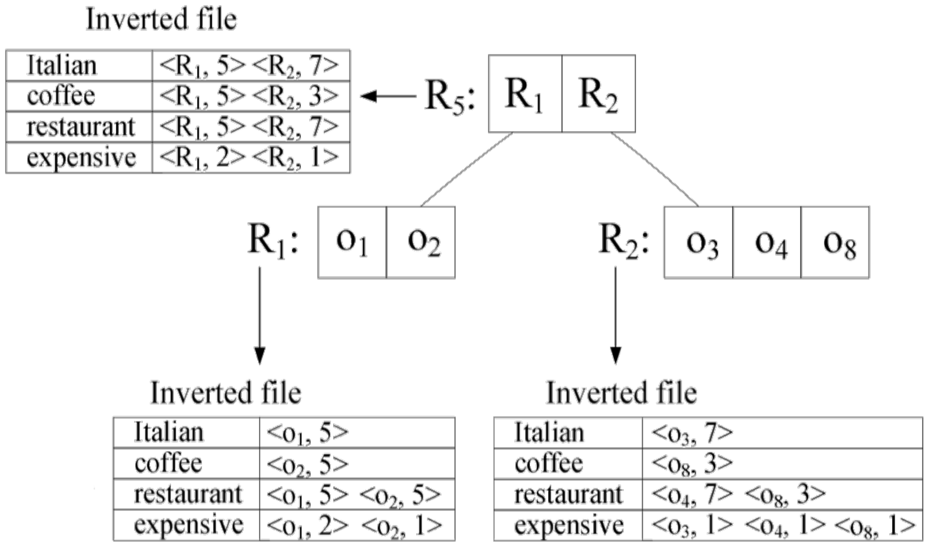
	R3				R4			
	Italian	coffee	restaurant	pizza	Italian	coffee	restaurant	pizza
	1	0	1	0	1	1	1	1

R3:

o5			
Italian	coffee	restaurant	pizza
1	0	1	0

R4:

o6				o7			
Italian	coffee	restaurant	pizza	Italian	coffee	restaurant	pizza
0	1	1	0	1	1	1	1



Inverted file

Italian	<R1, 5> <R2, 7>
coffee	<R1, 5> <R2, 3>
restaurant	<R1, 5> <R2, 7>
expensive	<R1, 2> <R2, 1>

Inverted file

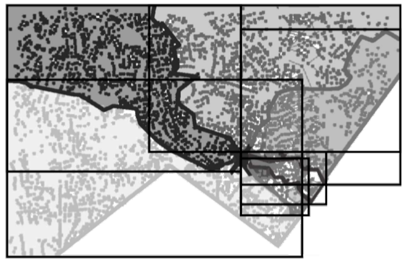
Italian	<o1, 5>
coffee	<o2, 5>
restaurant	<o1, 5> <o2, 5>
expensive	<o1, 2> <o2, 1>

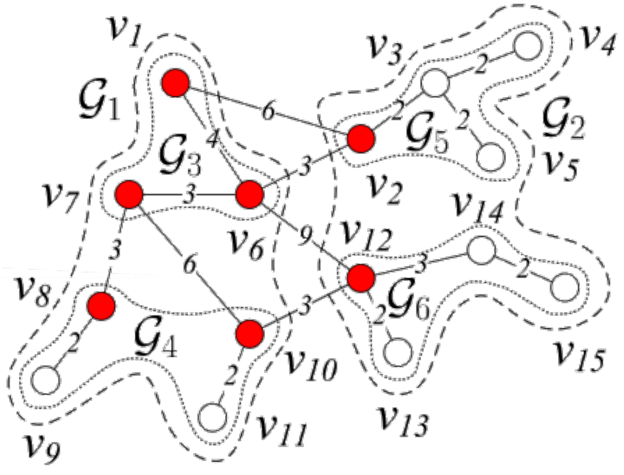
Inverted file

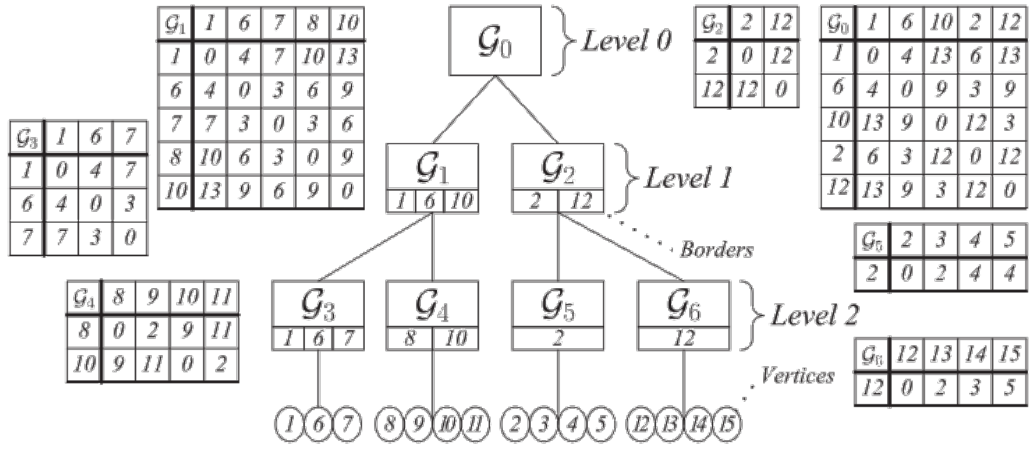
Italian	<o3, 7>
coffee	<o8, 3>
restaurant	<o4, 7> <o8, 3>
expensive	<o3, 1> <o4, 1> <o8, 1>

+
KNN in Road Network

- Graph $G = (V, E)$
  - Distance computing in road networks is costly
  - Shortest path: *Dijkstra's* algorithm and A^* algorithm
    - Euclidean distance for approximation
  - Optimization focus is different
    - Not only the number of points to be accessed
    - But also the network data to be accessed







$\mathcal{G}_3$	1	6	7
1	0	4	7
6	4	0	3
7	7	3	0

$\mathcal{G}_1$	1	6	7	8	10
1	0	4	7	10	13
6	4	0	3	6	9
7	7	3	0	3	6
8	10	6	3	0	9
10	13	9	6	9	0

$\mathcal{G}_4$	8	9	10	11
8	0	2	9	11
10	9	11	0	2

$\mathcal{G}_2$	2	12
2	0	12
12	12	0

$\mathcal{G}_0$	1	6	10	2	12
1	0	4	13	6	13
6	4	0	9	3	9
10	13	9	0	12	3
2	6	3	12	0	12
12	13	9	3	12	0

$\mathcal{G}_5$	2	3	4	5
2	0	2	4	4

$\mathcal{G}_6$	12	13	14	15
12	0	2	3	5

+ Skyline Queries

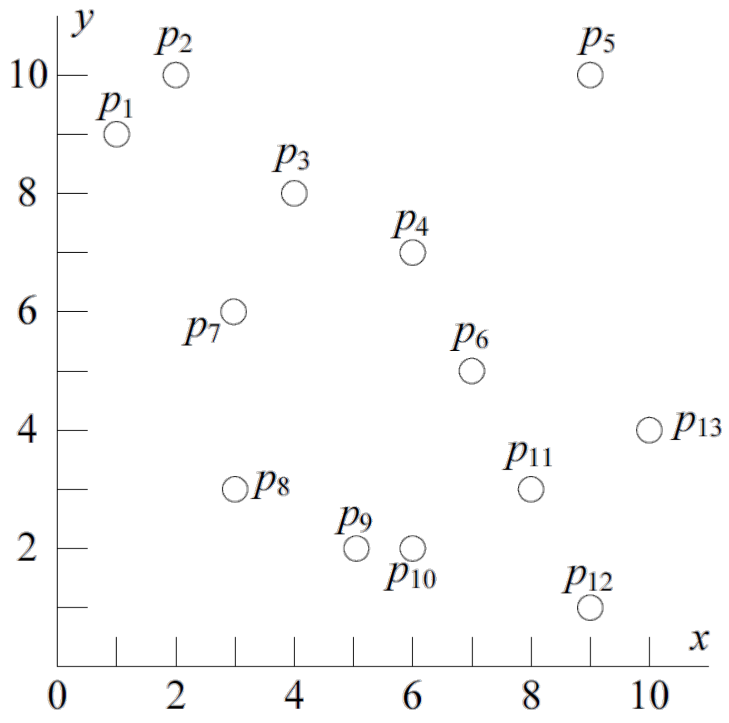
Which buildings can you see?

+ Monotonically Increasing Functions

- Monotonically Increasing Functions
  - Let p be a d-dimensional point in $\mathbb{R}^d$
  - Let $f : \mathbb{R}^d \rightarrow R$ a function that calculates a score $f(p)$ for p
  - We say that f is monotonically increasing if the score increases when any coordinate of p increases.

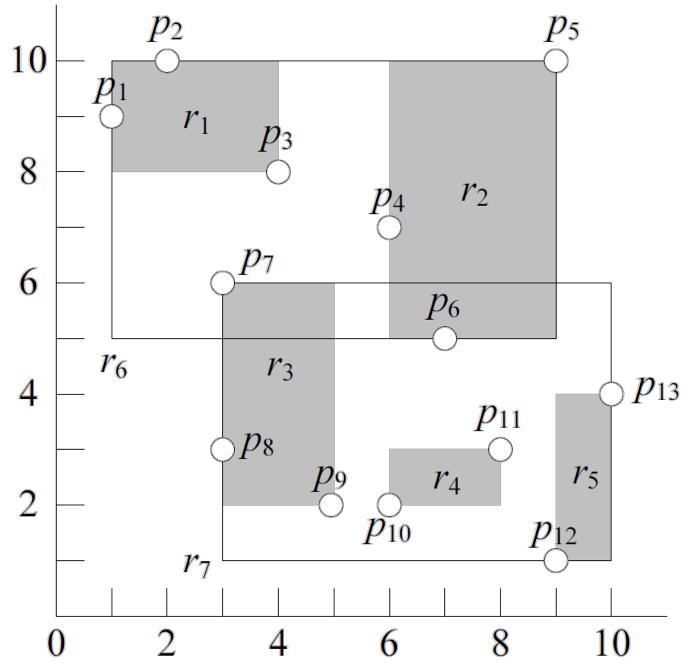
+ Multi-Criteria Optimization

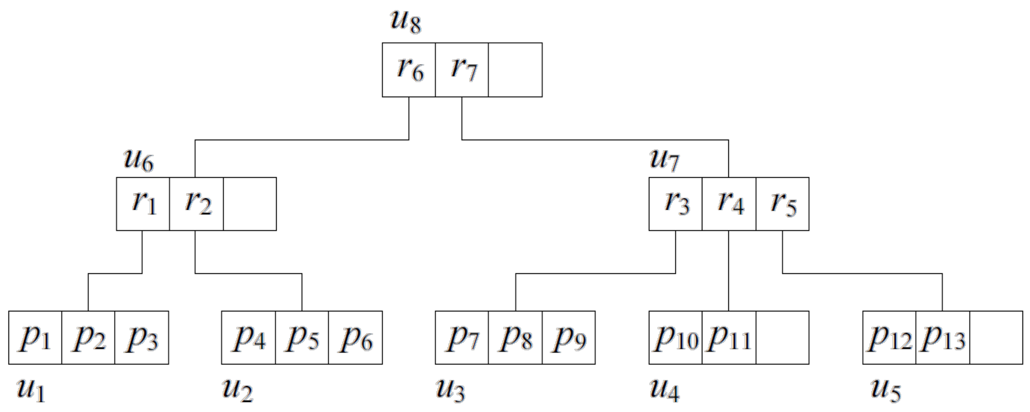
- Top-1 Search
  - Let P be a set of d-dimensional points in $\mathbb{R}^d$. Given a monotonically increasing function f , a top-1 query finds the point in P that has the smallest score
  - f is often referred to as the preference function of the query
  - If $f(x, y) = x + y$, the top-1 point is p_8
  - Applications
    - Find the hotel that minimizes price + distance (to the beach)
    - Find the second-handed car that minimizes price + 0.1× mileage
    - Find the interviewee that maximizes 0.4×education + 0.4×experience + 0.2×personality
    - ...



+ Top-1 Search

- Top-1 Search in NN
  - The NN of the origin of the data space according to the distance function f
  - What is the $MinDist$ of an MBR to the query point?

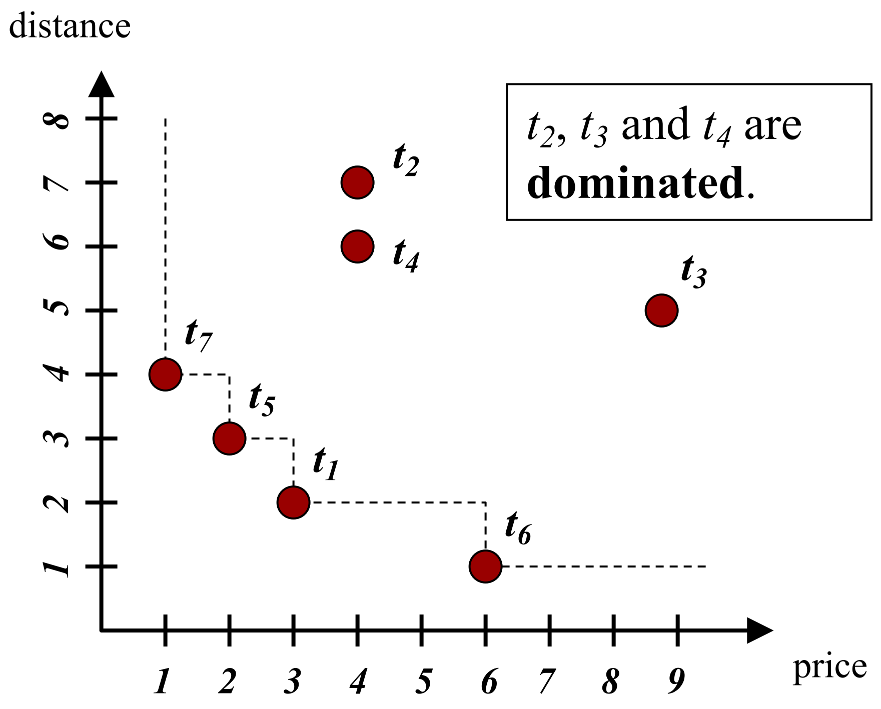




+ Top-1 Search

- Drawback
  - It is difficult to decide which preference function f to use
    - Why is $f(x, y) = x + y$ a good function?
    - Why not $2x + y$?
    - Why not $\sqrt{x} + y^2$?
    - Why not drop the preference function?
      - How do we return the top-1 point?
- Skyline Operator
  - Return a small set of objects that is guaranteed to cover the result of any top-1 query
    - Regardless of the preference function

+ Finding a Hotel…



t_2, t_3 and t_4 are **dominated**.

Skyline query: retrieve all points that are not dominated

+ Skyline Query Applications

- Find best NBA players
  - (#points, #rebounds), or any other subset of the 17 dimensions.
- Find best laptops
  - (price, screen size, weight, battery, memory size, disk size, CPU speed, warranty…)
- Any table in a RDBMS has a list of records with multiple attributes, so ……

+ A Formal Definition

- Assume the preference is smaller (e.g., closer to beach, cheaper hotel etc) – other semantics can be supported too
- For two d-dimensional points p and q, p dominates q iff
  - (1) $\forall 1 \leq i \leq d, p_i \leq q_i$ and
  - (2) $\exists 1 \leq i \leq d, p_i < q_i$
- Informally, a point dominates another point if it is as good or better in all dimensions and better in at least one dimension
  - Dominance is transitive
- A **skyline query** is to find all points from a dataset which are not dominated by any other points in the dataset

…Pareto dominance

+ Skyline For Top-1

- For any monotonically increasing function f, a top-1 point is definitely in the skyline
  - Proof
    - Suppose it is not true, and there exists a function f for which a top-1 point p is not in the skyline
    - p is dominated by at least another point p'
    - f is monotonic, so $f(p') < f(p)$
    - Contradicts that p is the top-1 point
- Every point in the skyline is definitely a top-1 point of some monotonically increasing function

+ Skyline Block Nested Loop

- Basic Idea
  - Compare each point p with every other point
  - If p is not dominated, then it is part of the skyline
    - $O(n^2)$

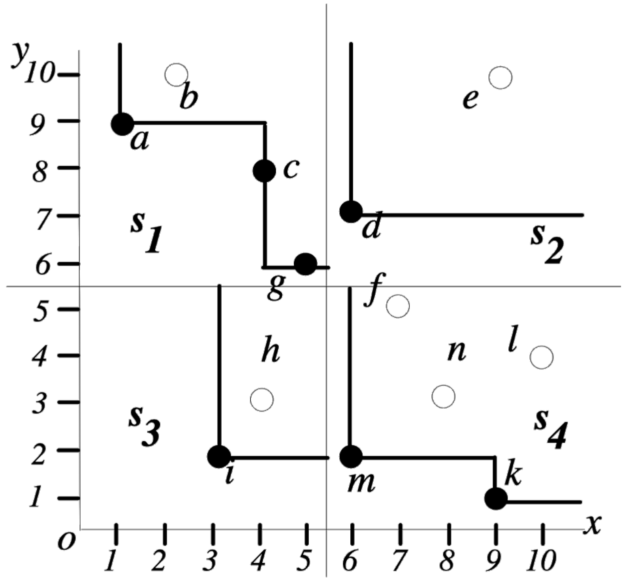
- BNL (Block Nested Loop)
  - Keeps a list of candidate skyline points in memory

  1. If p is dominated by any point in the list, discard it
  2. If p dominates any point in the list, insert p in the list and discard all the points dominated by p in the list
  3. If p is neither dominated, nor dominates, any point point in the list, insert p to the list

  - Heuristic
    - When a point in list dominates p, move this point to the head of the candidate list

+ Skyline Divide and Conquer

- Basic Idea
  1. Divide the dataset into several partitions so that each partition fits in memory
  2. Compute partial skyline of the points each partition with the previous BNL
  3. Merge the partial results
     - All points in the skyline of s_3 must appear in the final skyline
     - If s_3 is not empty, discard all result of s_2 directly
     - s_1 and s_4 only needs to compare with s_3
- Disadvantages
  - Only efficient for small dataset
  - Large IO for big dataset
  - Not suitable for online processing
    - Only report result when finish

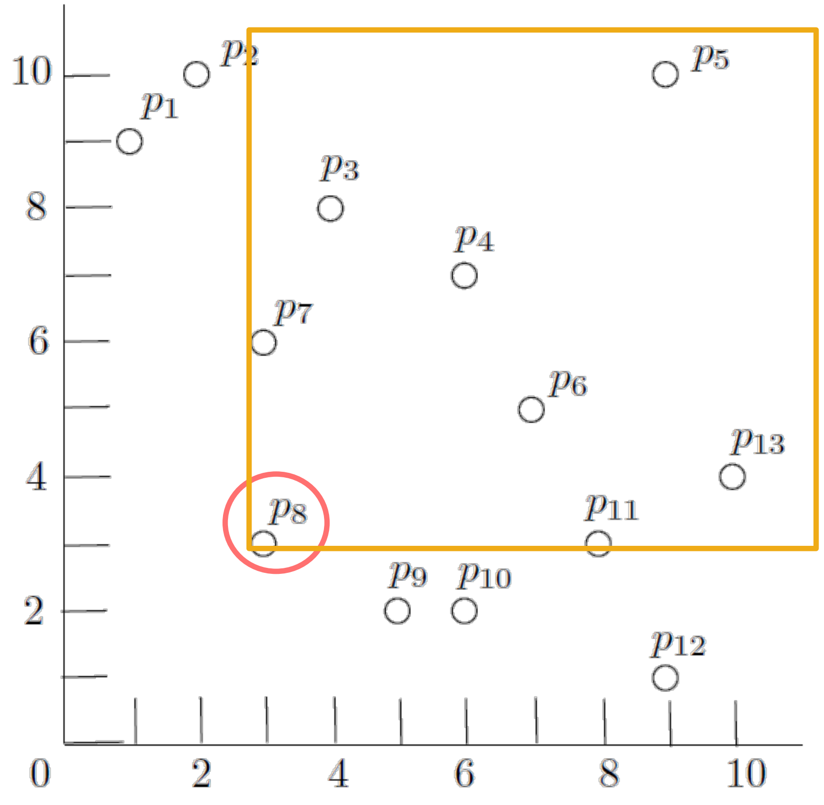


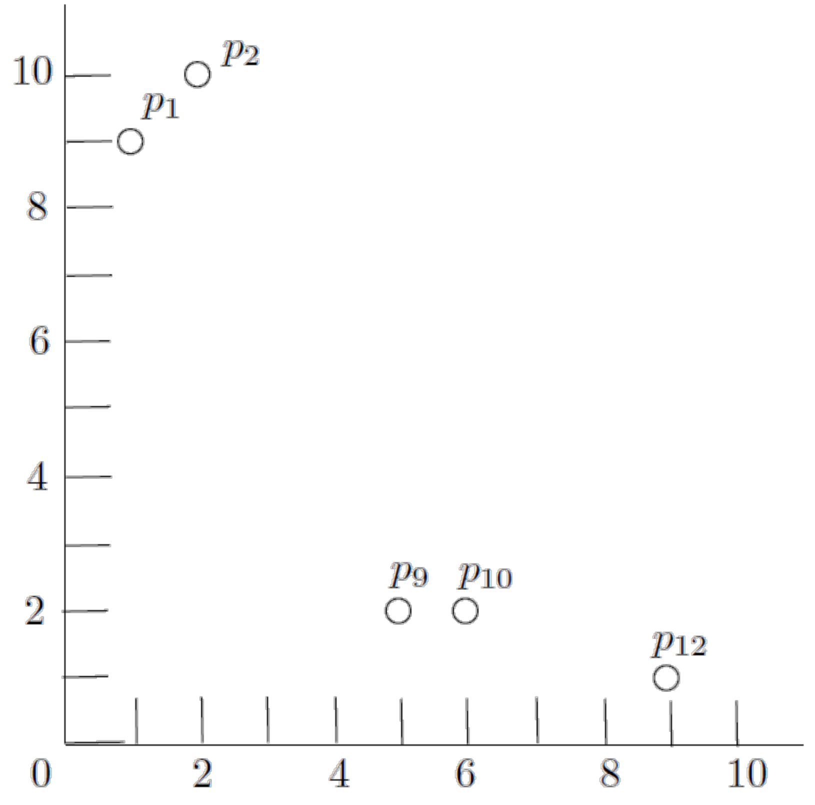
+ Skyline with NN

- Let p be the (Euclidean) NN of the origin of the data space, among all the points in the dataset P. Then, p must be in the skyline of P
  - Proof
    - Suppose it is not true, then p is dominated by at least another point p'
    - This, however, means that p'must be closer to the origin than p
    - Contradict
  - Use NN as a Black Box
    - Repeat the following operations until R-tree is empty
    1. Find the NN p of the origin of the data space. Include p into the skyline
    2. Delete p from the tree, as well as all the points that are dominated by p

+ Skyline with NN Example

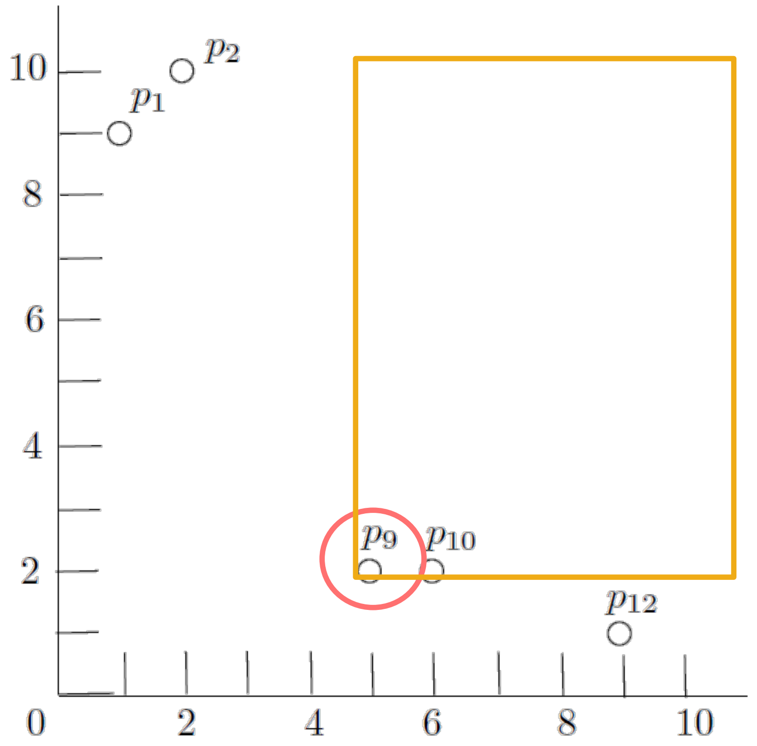
- The first NN query finds p_8
- The figure on the right shows the dataset after the corresponding deletions

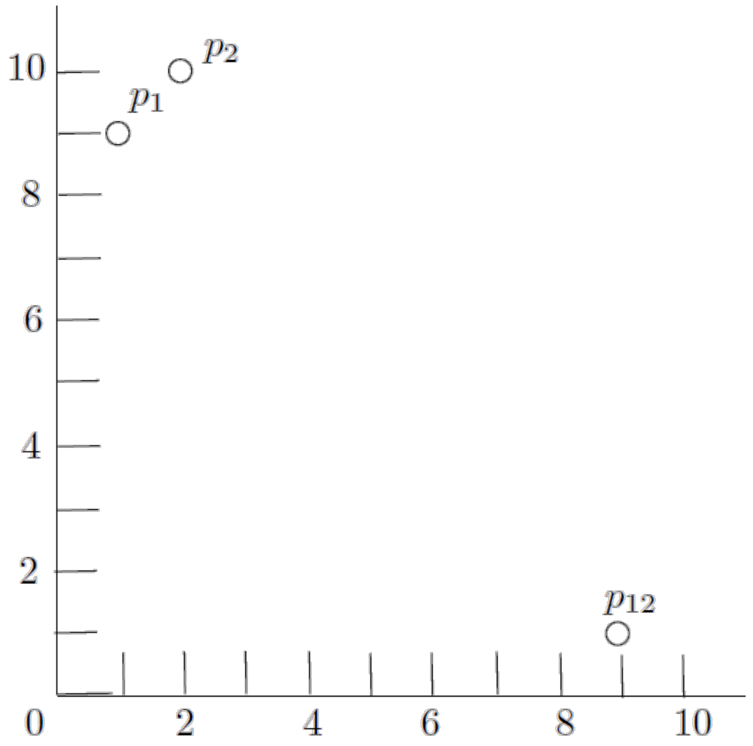




+ Skyline with NN Example

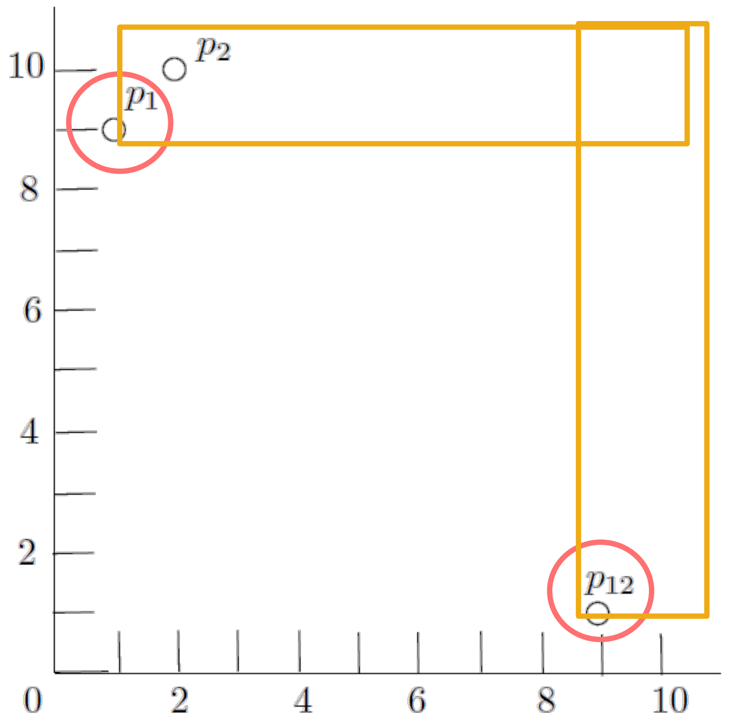
- The second NN query finds p_9
- The figure on the right shows the dataset after the corresponding deletions

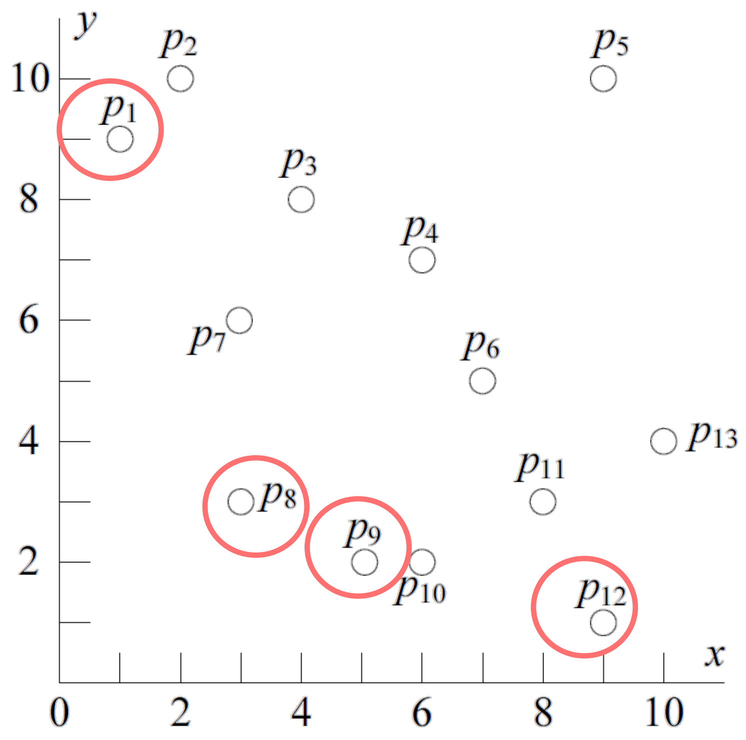




+ Skyline with NN Example

- The third NN query finds p_1 and p_{12}
  - The dataset becomes empty after corresponding deletions
  - Final skyline points=$\{p_8, p_9, p_1, p_{12}\}$





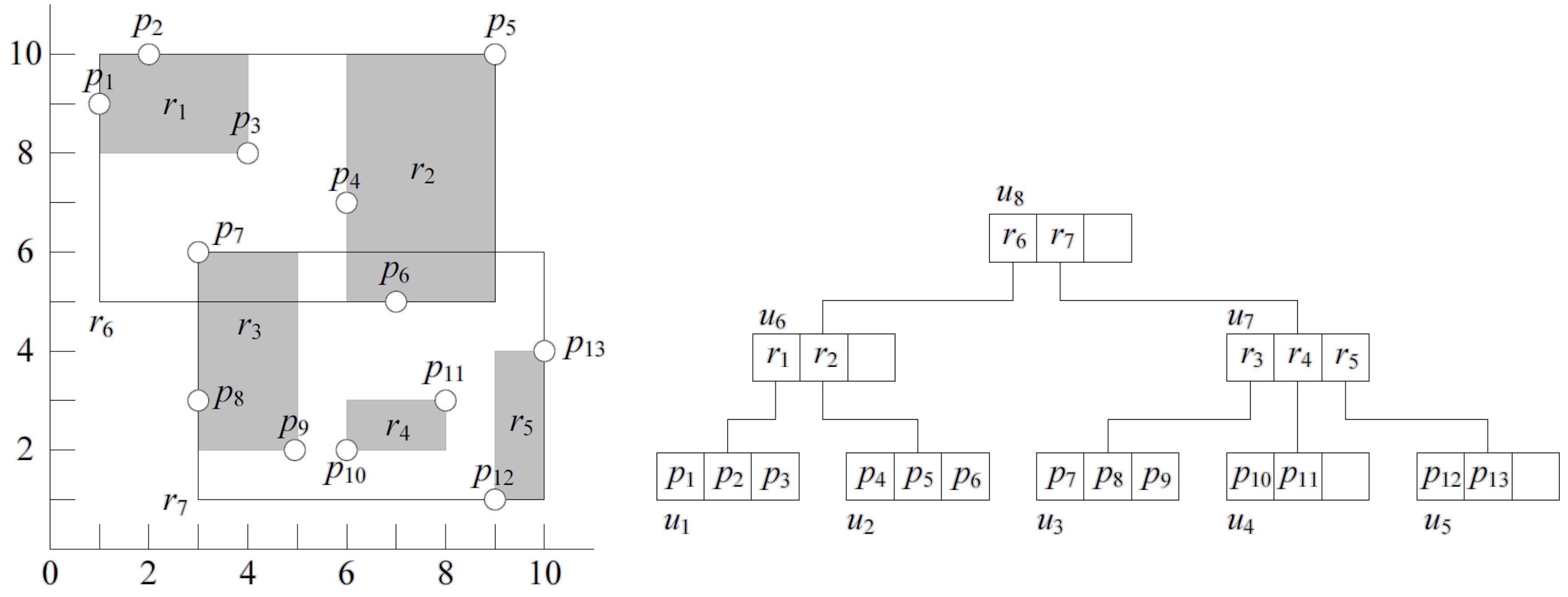
+ Branch and Bound Skyline (BBS)

- The previous algorithm must access all the points
- BBS
  - A variation of the BF algorithm
  - It accesses the nodes of the R-tree in ascending order of their $MinDist$s from the origin
  - Different from NN search
    - Once MBR in its entirely is dominated by a skyline point already found, it can be pruned

+ Branch and Bound Skyline (BBS)

- Access the root, put the MBRs in a sorted list

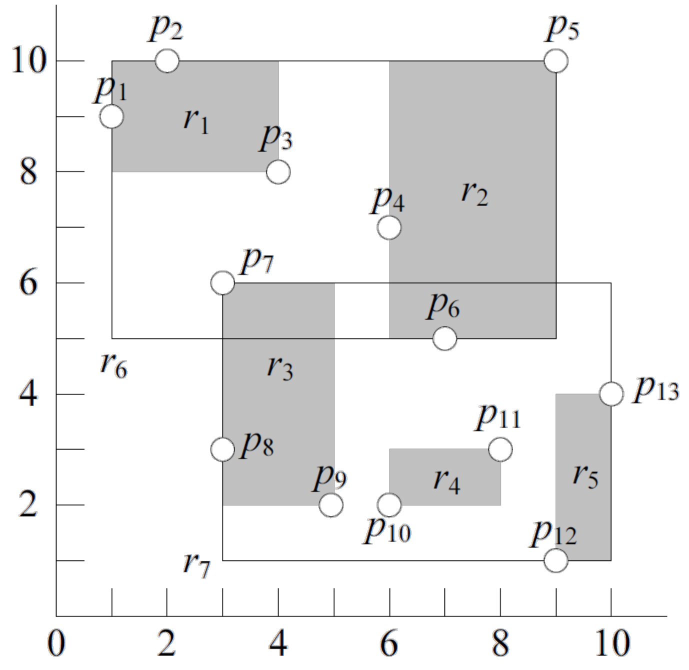
$r_7, \sqrt{10}$
$r_6, \sqrt{26}$

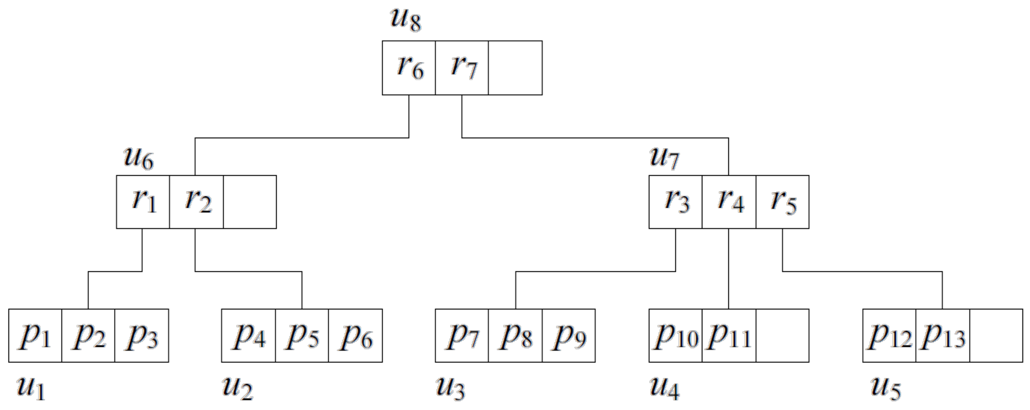


+ Branch and Bound Skyline (BBS)

- Visit u_7
  - Insert r_3, r_4, r_5 into the heap

$r_3, \sqrt{13}$
$r_6, \sqrt{26}$
$r_4, \sqrt{40}$
$r_5, \sqrt{82}$



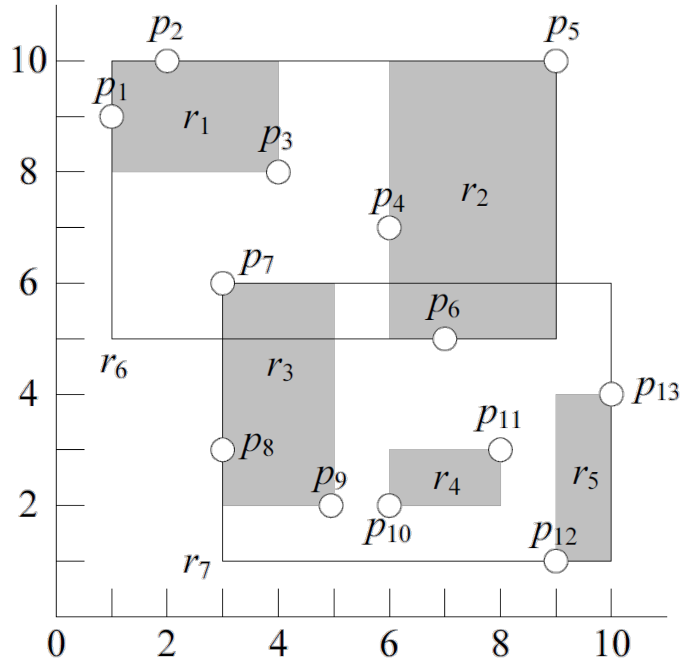


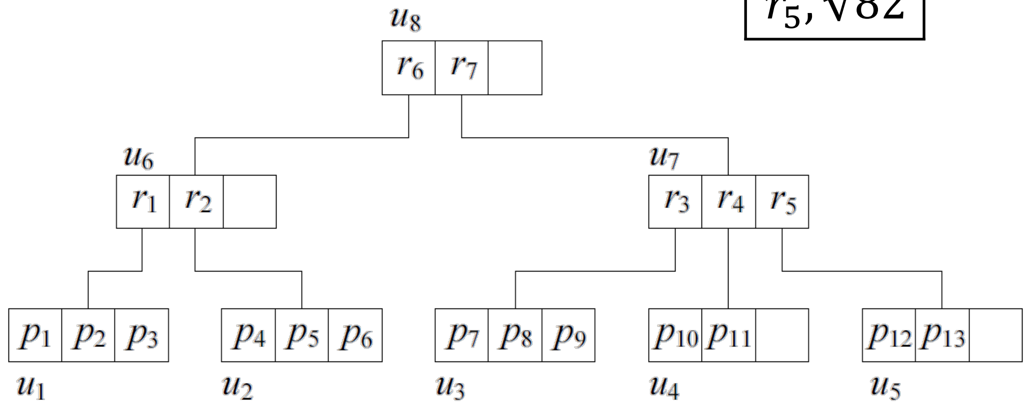
+ Branch and Bound Skyline (BBS)

- Visit u_3
  - Insert all its point into the list

$r_3, \sqrt{13}$
$r_6, \sqrt{26}$
$r_4, \sqrt{40}$
$r_5, \sqrt{82}$

$p_8, \sqrt{18}$
$r_6, \sqrt{26}$
$p_9, \sqrt{29}$
$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_5, \sqrt{82}$



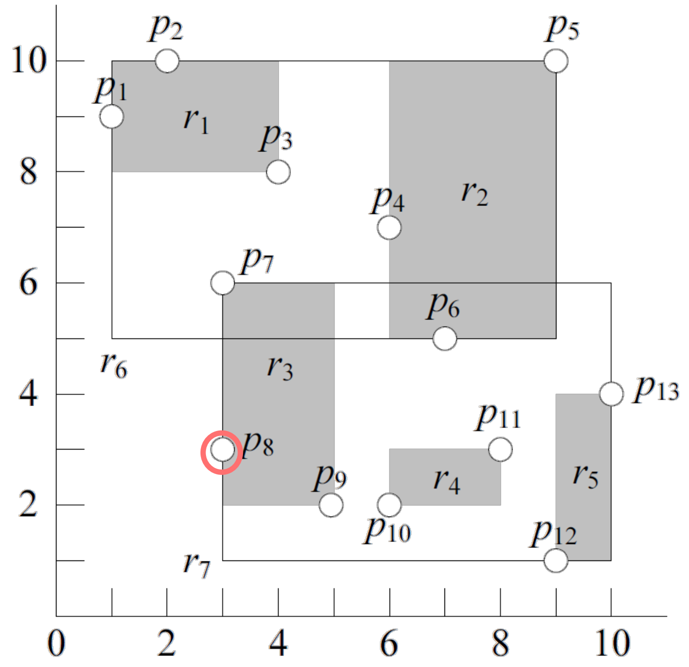


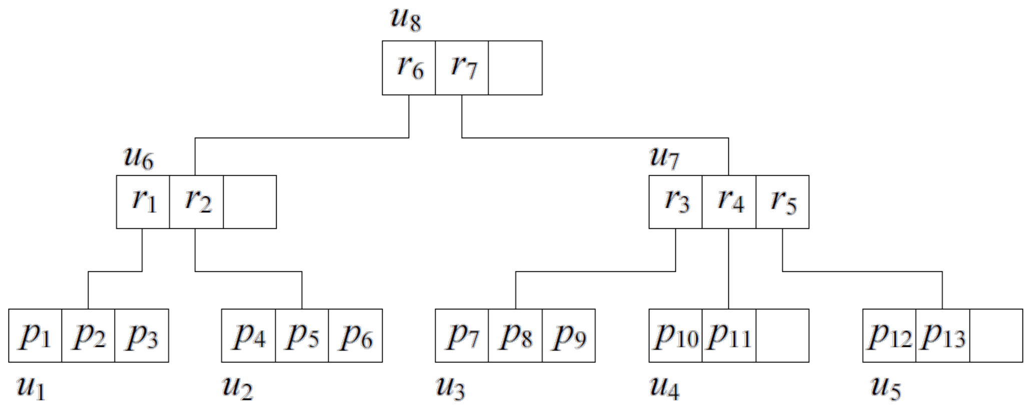
+ Branch and Bound Skyline (BBS)

- Remove p_8 from the list
  - This is the first skyline point found

$p_8, \sqrt{18}$
$r_6, \sqrt{26}$
$p_9, \sqrt{29}$
$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_5, \sqrt{82}$

$r_6, \sqrt{26}$
$p_9, \sqrt{29}$
$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_5, \sqrt{82}$



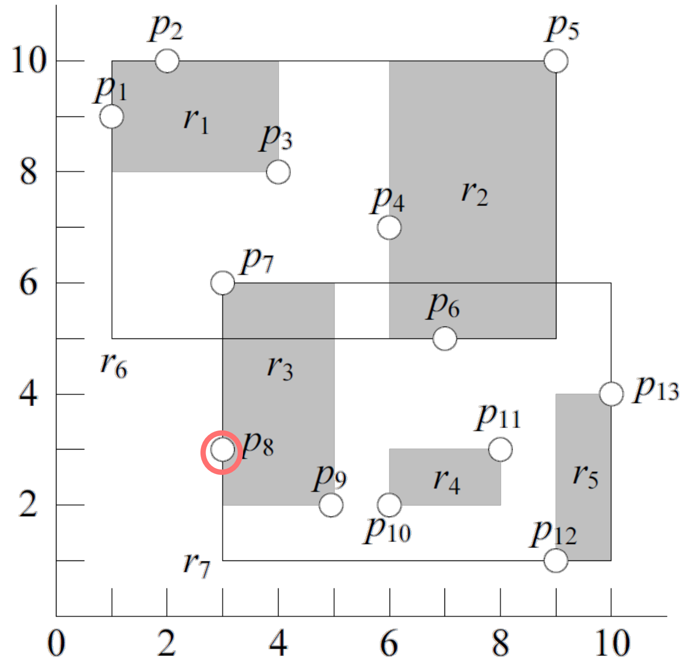


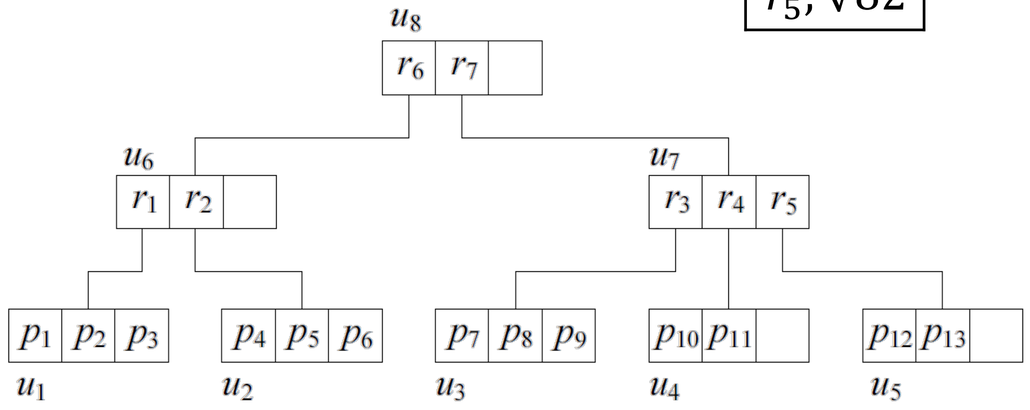
+ Branch and Bound Skyline (BBS)

- Access u_6
  - Insert r_1 and r_2 in the list

$r_6, \sqrt{26}$
$p_9, \sqrt{29}$
$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_5, \sqrt{82}$

$p_9, \sqrt{29}$
$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_2, \sqrt{61}$
$r_1, \sqrt{65}$
$r_5, \sqrt{82}$



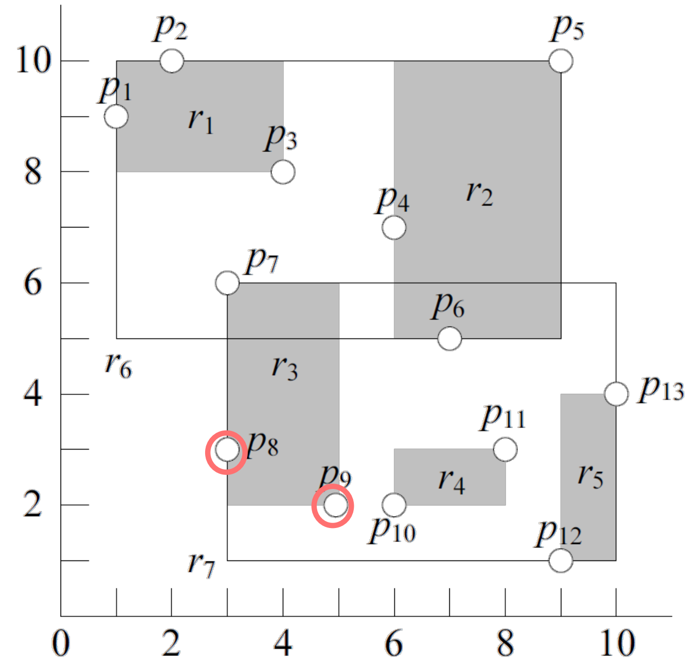


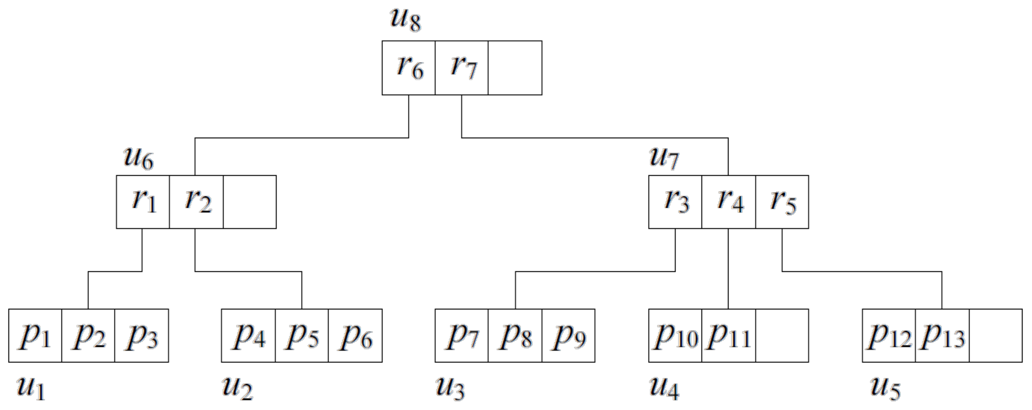
+ Branch and Bound Skyline (BBS)

- Visit p_9
  - Second skyline point

$p_9, \sqrt{29}$
$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_2, \sqrt{61}$
$r_1, \sqrt{65}$
$r_5, \sqrt{82}$

$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_2, \sqrt{61}$
$r_1, \sqrt{65}$
$r_5, \sqrt{82}$



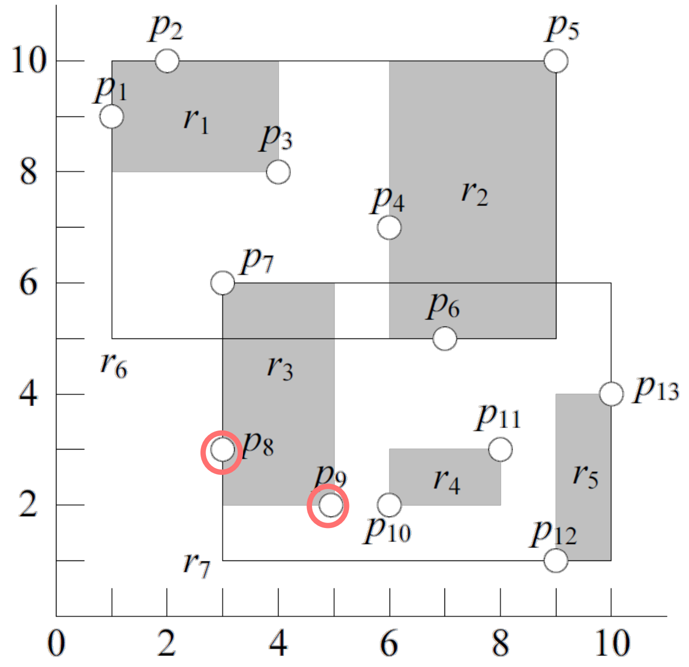


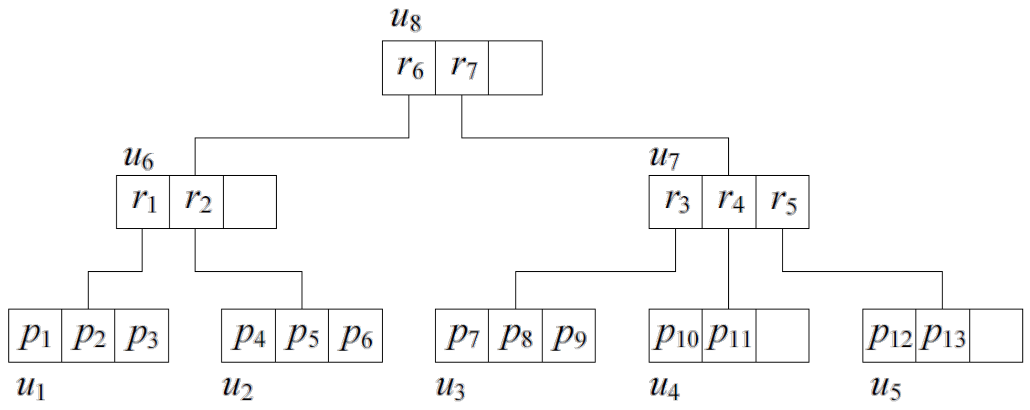
+ Branch and Bound Skyline (BBS)

- r_4 is discarded
  - Lower-left corner is dominated by p_9
  - No point in r_4 can possibly be in the skyline
- p_7 and r_2 are also discarded

$r_4, \sqrt{40}$
$p_7, \sqrt{45}$
$r_2, \sqrt{61}$
$r_1, \sqrt{65}$
$r_5, \sqrt{82}$

$r_1, \sqrt{65}$
$r_5, \sqrt{82}$





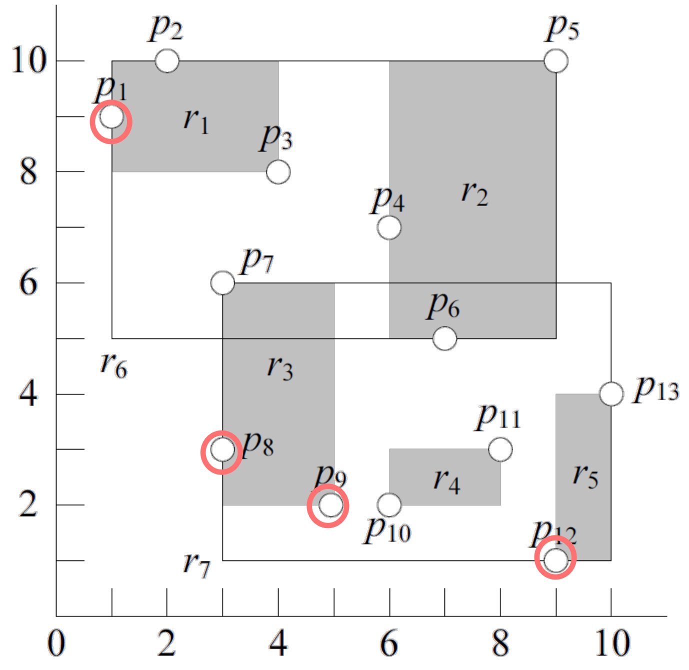
+ Branch and Bound Skyline (BBS)

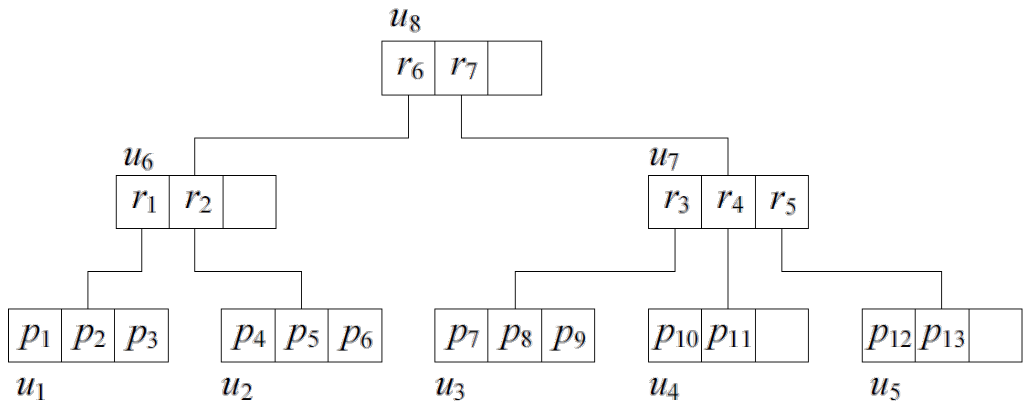
- Access u_1
  - p_3 can be discarded
  - p_1 is a skyline
  - Find the last skyline point p_{12} in r_5

$r_1, \sqrt{65}$
$r_5, \sqrt{82}$

$p_3, \sqrt{80}$
$p_1, \sqrt{82}$
$r_5, \sqrt{82}$
$p_2, \sqrt{104}$

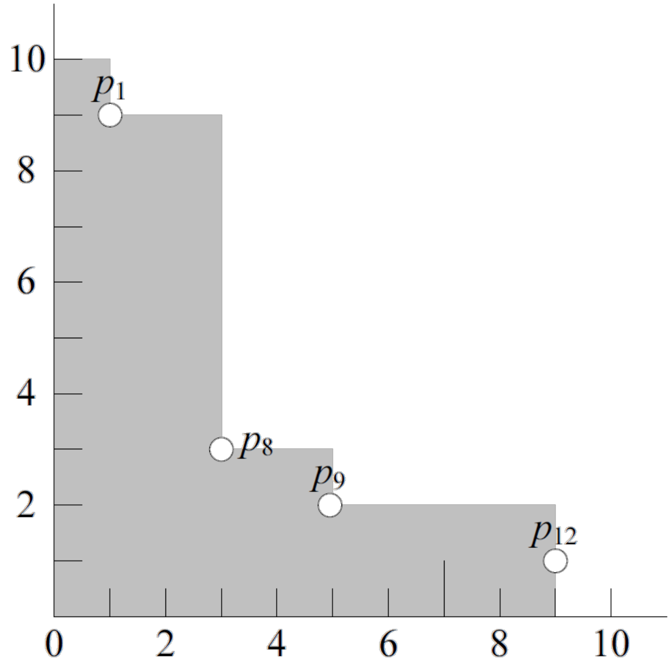
$r_5, \sqrt{82}$
$p_2, \sqrt{104}$





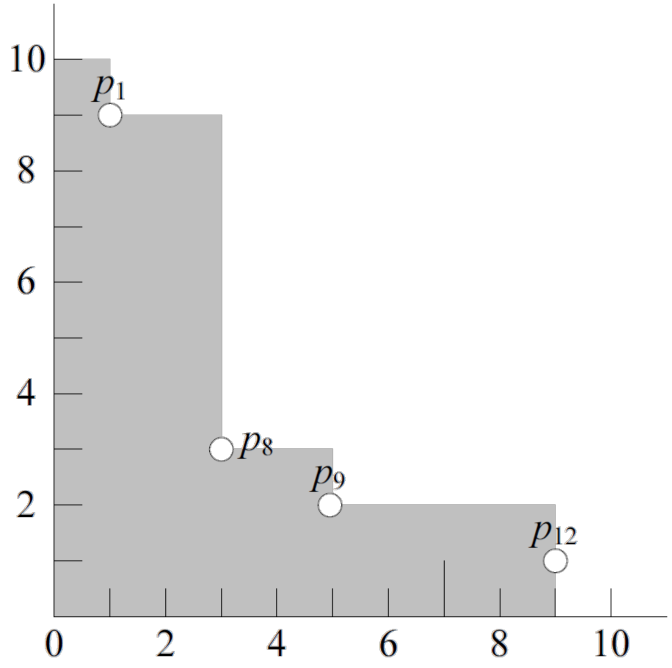
+ Optimality of BBS

- It accesses the least number of nodes among all the algorithms that correctly finds the skyline using the same R-tree without modifying the structure
  - Search Region
    - The union of the points in $\mathbb{R}^d$ that are not dominated by any skyline point
    - Any correct algorithm must access all the nodes whose MBRs intersect the search region
    - BBS accesses only the nodes whose MBRs intersect with the search region



+ Optimality of BBS

- Proof
  - Assume that the algorithm needs to visit a node u whose MBR r is disjoint with the region
    - It follows that a skyline point p dominates the lower-left corner of r
    - p has a smaller $MinDist$ than r
    - The sequence of points/nodes processed by BBS is in ascending order of $MinDist$ to the origin
      - p is processed before u
      - It is impossible to visit u



+ More on Skylines

- Variations
  - Dynamic skyline queries
    - The distance is to a give query point, not fixed
  - Skyline in constrained spaces such as road networks
  - How many they dominate?
    - K points that dominate the largest number of other points
    - The result points are not necessarily in the skyline result
  - Top-k skyline (ranked)
  - Skyline queries with uncertainty
  - Skyline in high dimensional space
  - Skyline in partially ordered space
  - …

+ kNN vs Skyline

- These two types of queries are related in a multidimensional space
  - If an overall ranking function is available (e.g., distance function, similarity function), then kNN can be used
  - Otherwise, skyline may be more useful
- A skyline query returns a superset of any NN queries (based on any liner combination of the attributes)
  - $f(x, y) = w \times x + (1 - w) \times y$ for any weight w
  - This subset can often be very large (too large to be useful)
- These two can be used together (e.g., to give a ranked list, and also more contextual information about the recommendation)

+ Final Words on Spatial Databases

- Spatial databases is one of the most successful database innovations since the relational model
  - Real problems, real solutions, real impacts
  - Foundation to many other complex data management and similarity query processing
- Challenges are still there
  - …so are the opportunities, for researchers and entrepreneurs
- Next…
  - Spatiotemporal databases (moving objects)
  - High dimensional data

+ Assignment 2

- Self enroll on Blackboard with Group
  - Research Track
    - At least three research papers
    - SIGMOD, VLDB, ICDE, ACM Multimedia
      - Google Scholar, UQ library, DBLP, Research Gate,…
  - Implementation Track

  - Each Track has some predefined topics with a limit of 20 students
  - Each Track has 1 $Others$ big group with no limit of student

  - Assessments:
    1. Research Proposal 16:00 5 May
    2. Research Report 16:00 28 May

+ Assignment 2 Research Track

- kNN
  - Classic kNN, Constraint keywords, k furthest, Reverse kNN, Continuous kNN, Aggregate kNN, …
- Skyline
  - Classic Skyline, Reverse Skyline, Constraint Skyline, Top-k Skyline, Group-by Skyline, …
- Trajectory
  - Similarity, Pattern Mining, Compression, Index, Privacy, Map Matching, …
- Multimedia Database
  - Text, Audio, Video, Image, High Dimension, …
- Shortest Path Algorithm
  - Contraction Hierarchy, 2 Hops, Constraint Shortest Path, Time Dependent / Fastest Path,…
- Spatial Data Management
  - Hadoop / Spark / …, Spatial Temporal Data Management, …
- Others…

+ Assignment 2 Implementation Track

- Quadtree
- Kd-Tree
- Z-Order
- R-Tree
- kNN Algorithm
- Skyline Algorithm
- Spatial Database System Performance Testing
- Others…