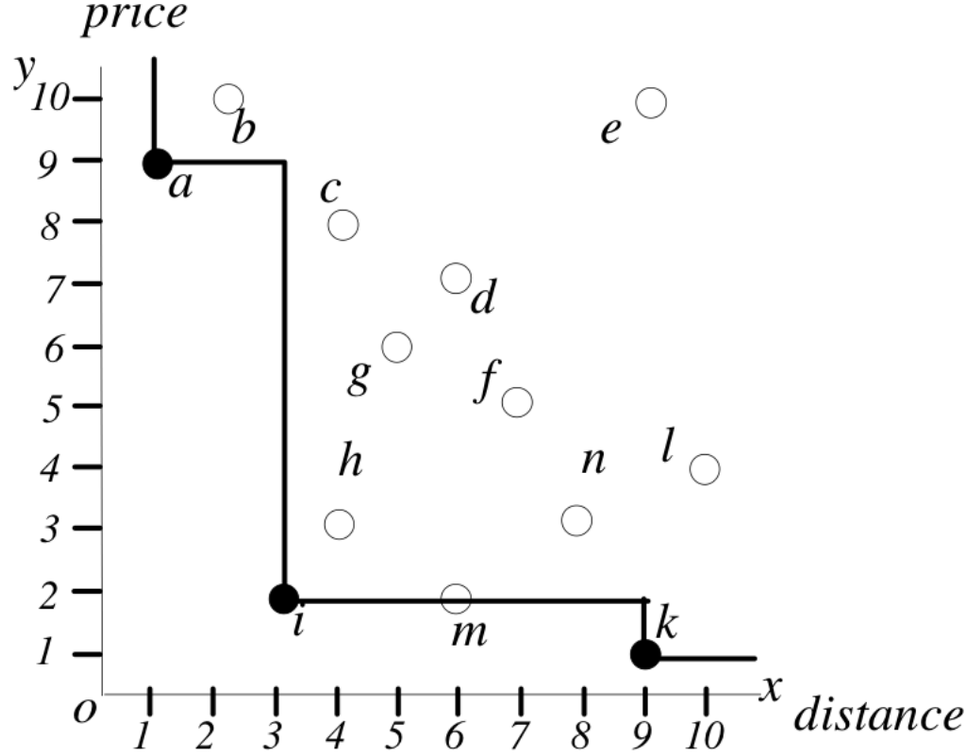
Lab 3

1st Programming assignment introduction

Skyline

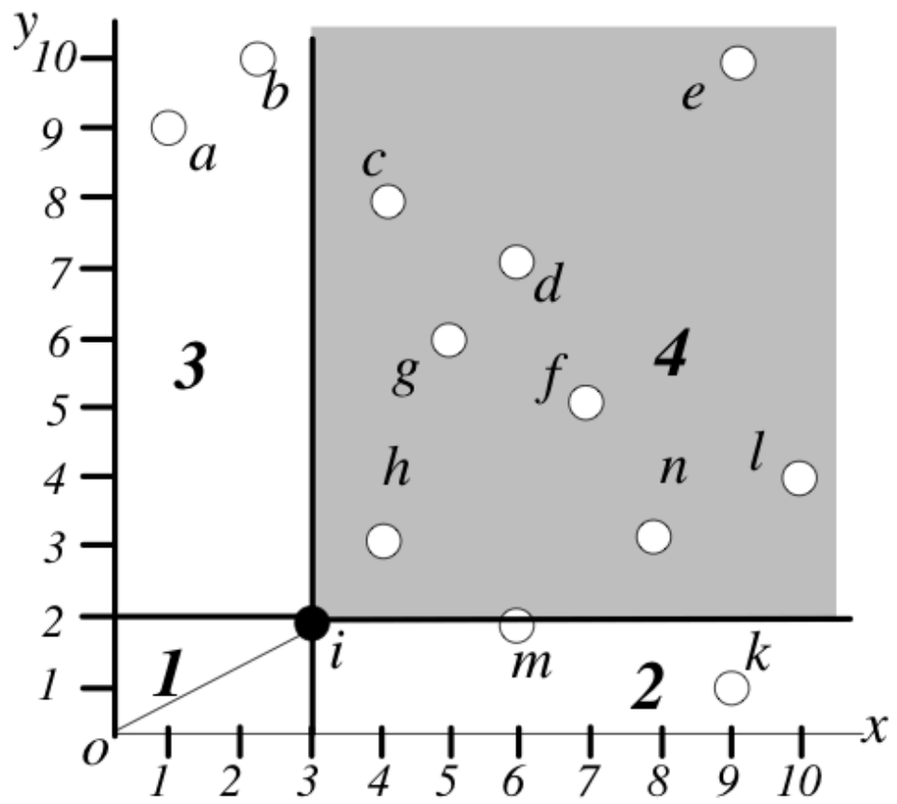
Assuming that we have a set of hotels and for each hotel we store its distance from the beach (*x* axis) and its price (*y* axis)

What is a good choice?



Dominance region of point i

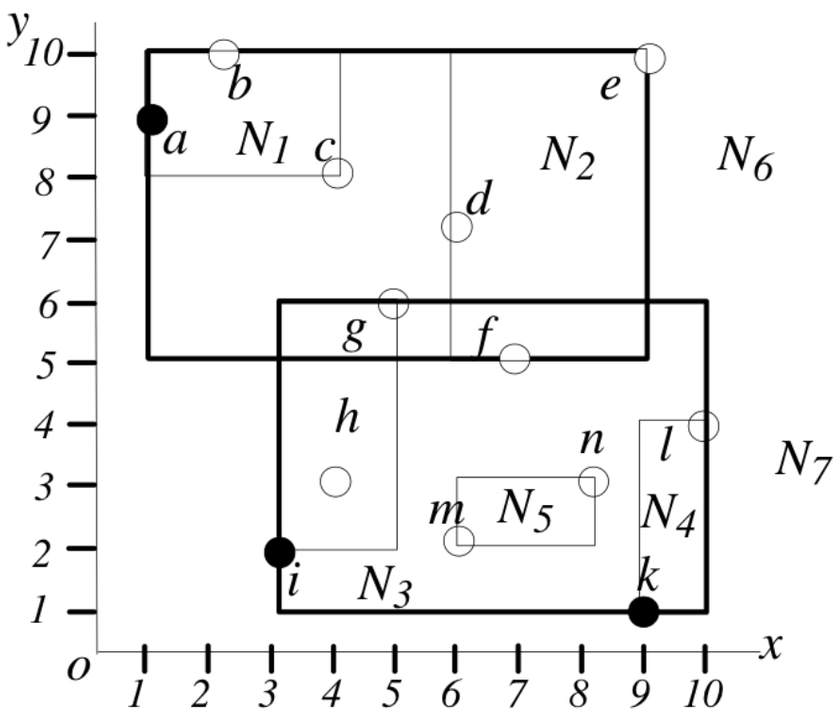
The region in which all of the points contained have a lower value than point i in all dimensions.

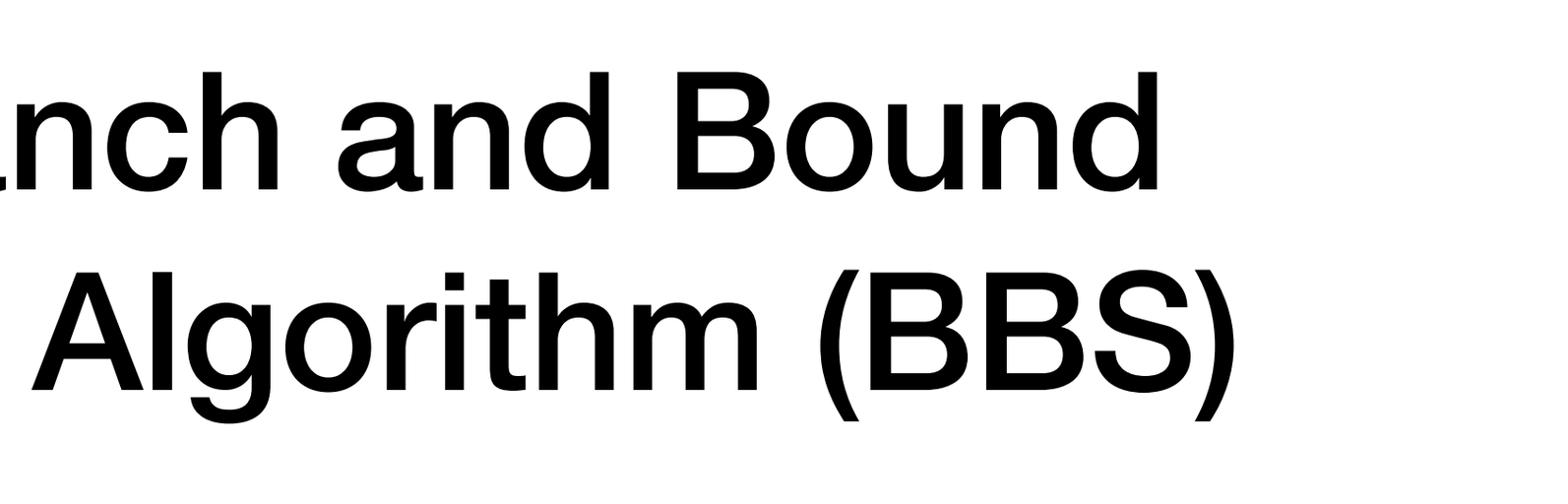


Point i dominates all other points except a, b and k.

(1) Branch and Bound Skyline Algorithm (BBS)

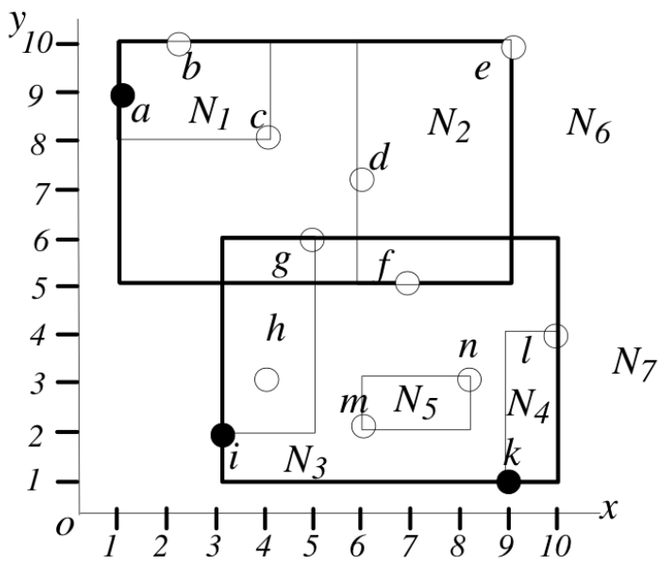
Algorithm based on nearest neighbor search. Can be used with any data portioning method. In the assignment, you will use an R-tree.

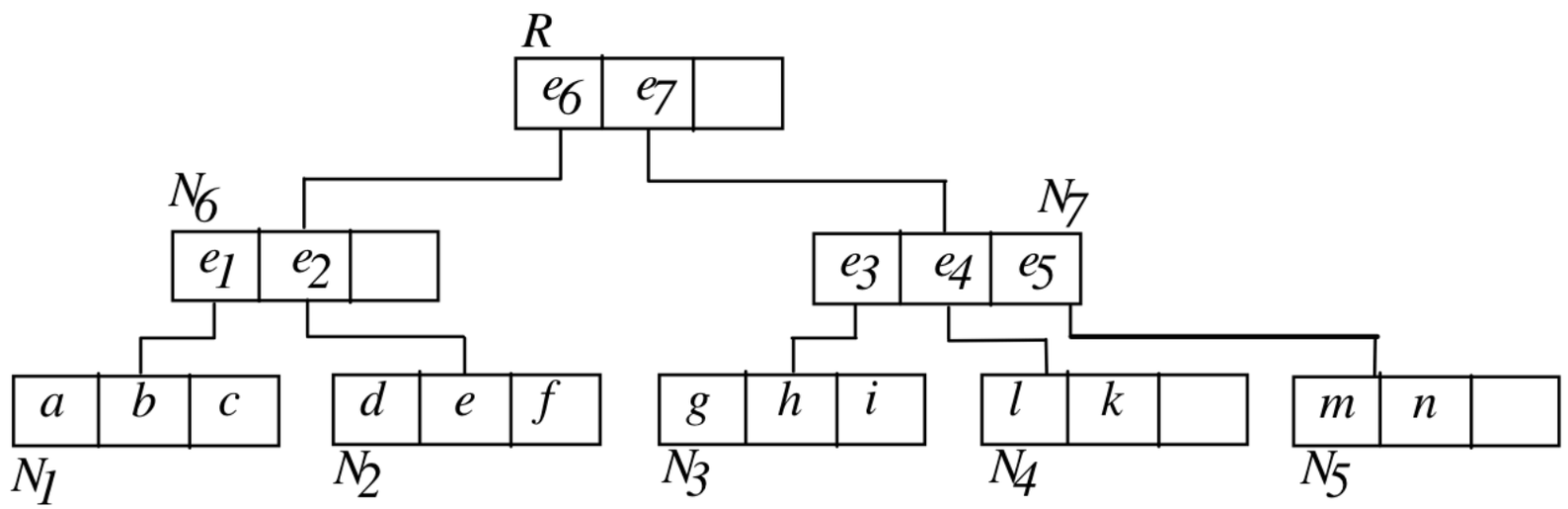




Points in the 2D space are stored in an R-tree as shown in previous labs.

(2) Branch and Bound Skyline Algorithm (BBS)

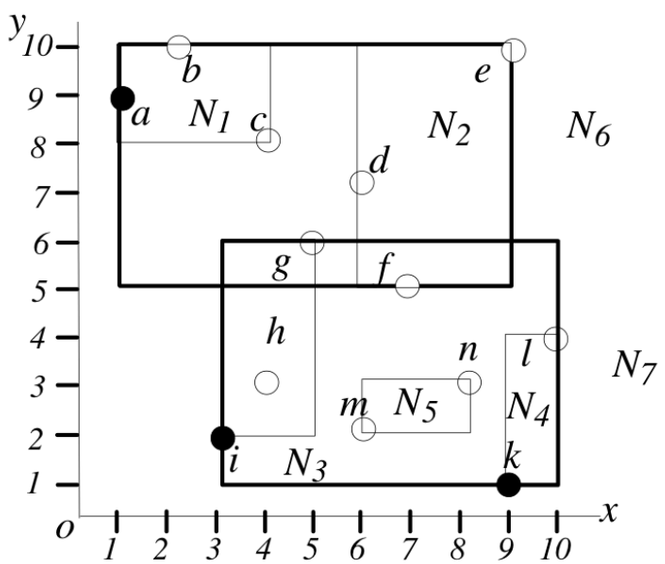


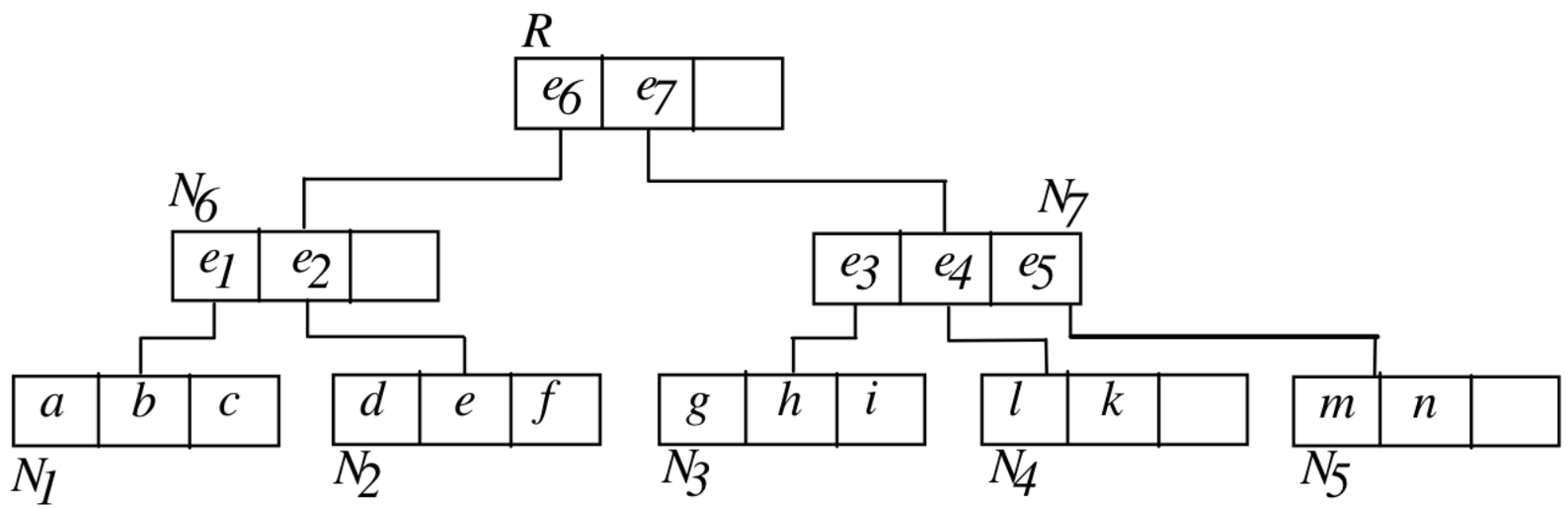


- Starts from the root node of the R-tree and inserts all its entries (e6, e7) in a heap sorted according to their mindist.

Action	heap contents	*S*
access root	$<e_7,4><e_6,6>$	$\varnothing$

(3) Branch and Bound Skyline Algorithm (BBS)

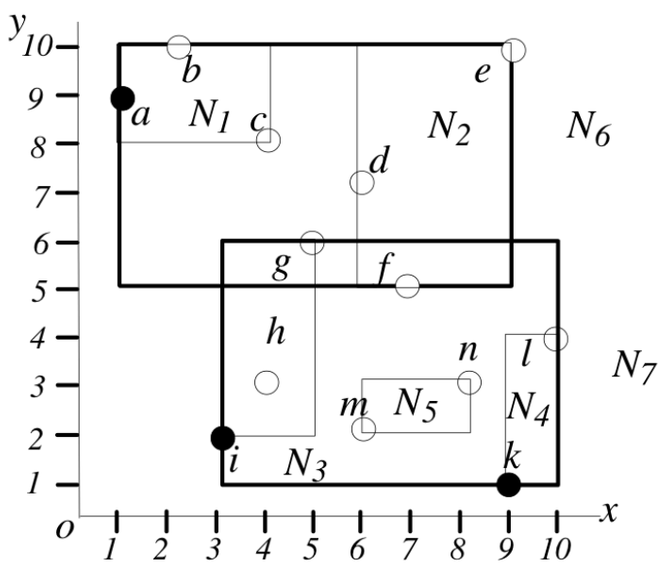


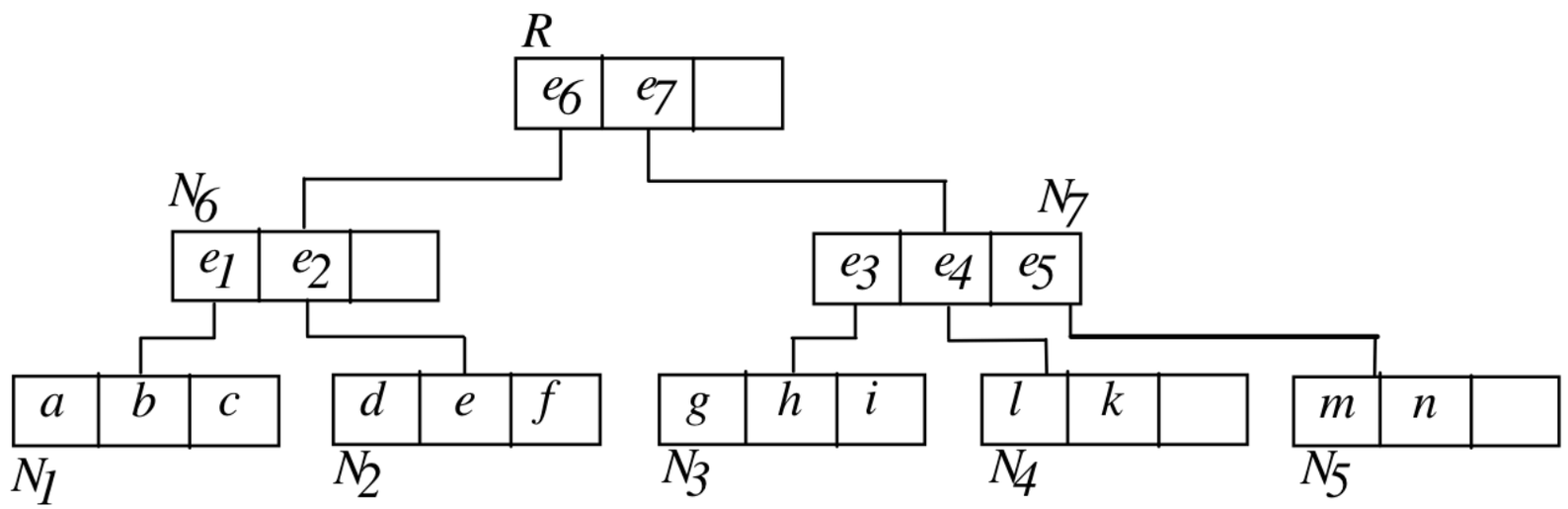


- Then, the entry with the minimum mindist (e7) is expanded.This expansion removes the entry (e7) from the heap and inserts its children (e3, e4, e5)

Action	heap contents	*S*
access root	$<e_7,4><e_6,6>$	∅
expand e_7	$<e_3,5><e_6,6><e_5,8><e_4,10>$	∅

(4) Branch and Bound Skyline Algorithm (BBS)





- The next expanded entry is again the one with the minimum mindist (e3), in which the first nearest neighbor (i) is found

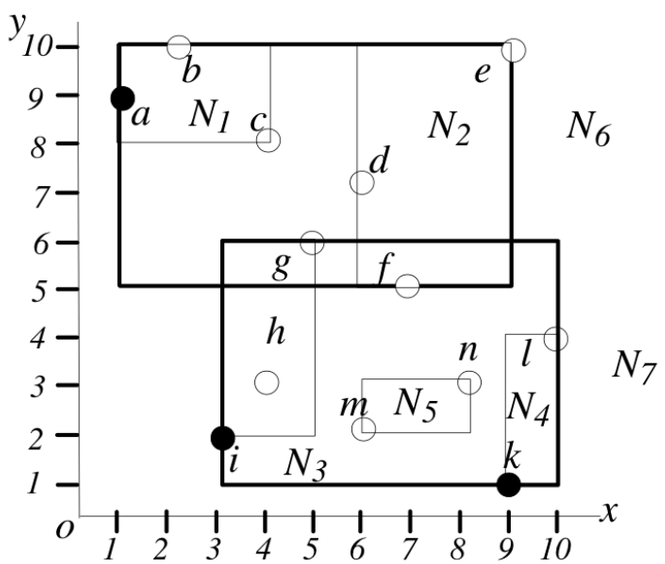
Action	heap contents	*S*
access root	<e_7,4><e_6,6>	∅
expand e_7	<e_3,5><e_6,6><e_5,8><e_4,10>	∅
expand e_3	**<*i*,5>**<e_6,6><*h*,7><e_5,8> <e_4,10><*g*,11>	{*i*}

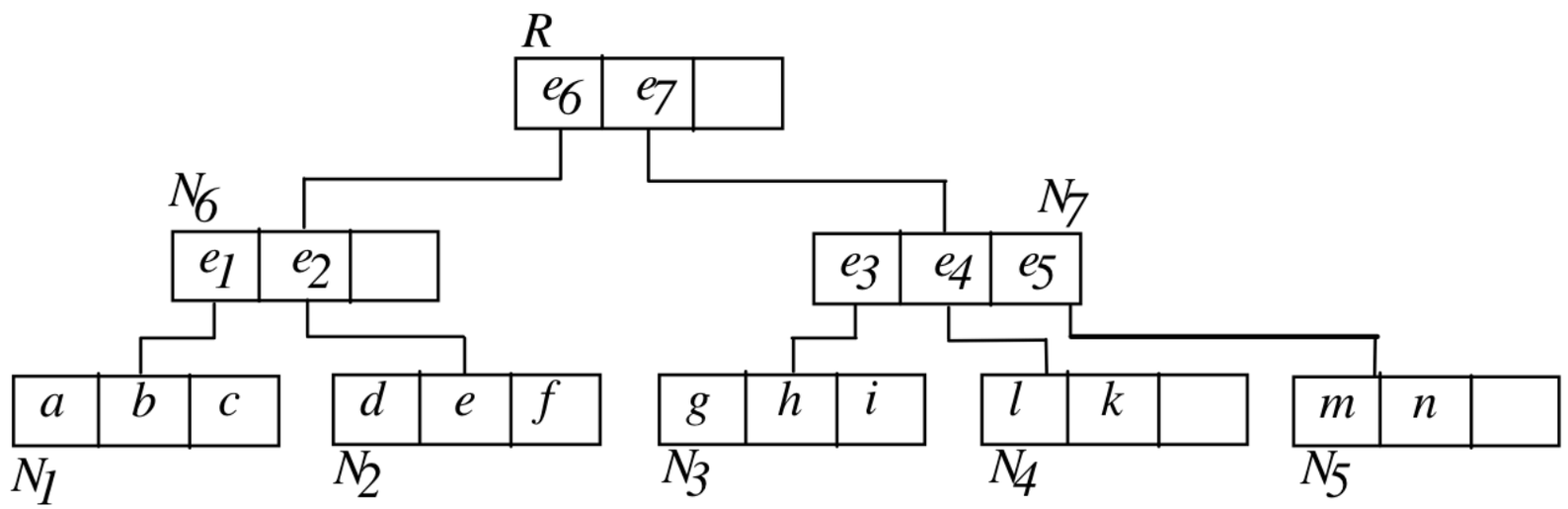
- This point (i) belongs to the skyline, and is inserted to the list S of skyline points.

!

Notice that up to this step BBS behaves like the best-first (BF) nearest neighbor algorithm. Although, BF would now terminate. Why?

(4) Branch and Bound Skyline Algorithm (BBS)

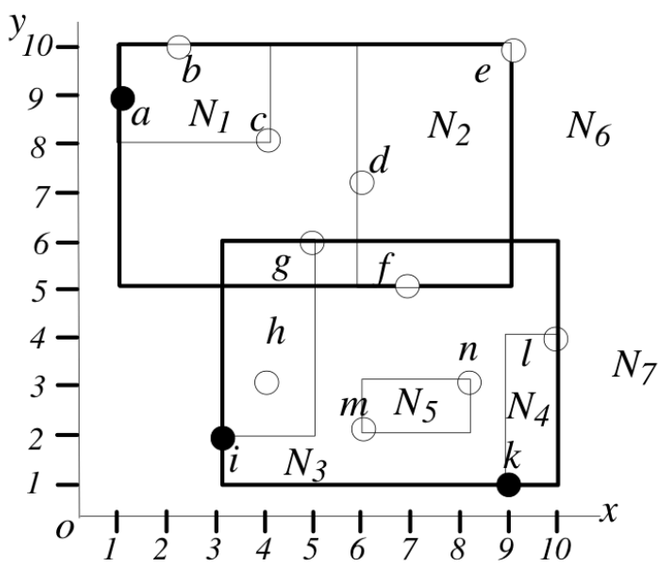


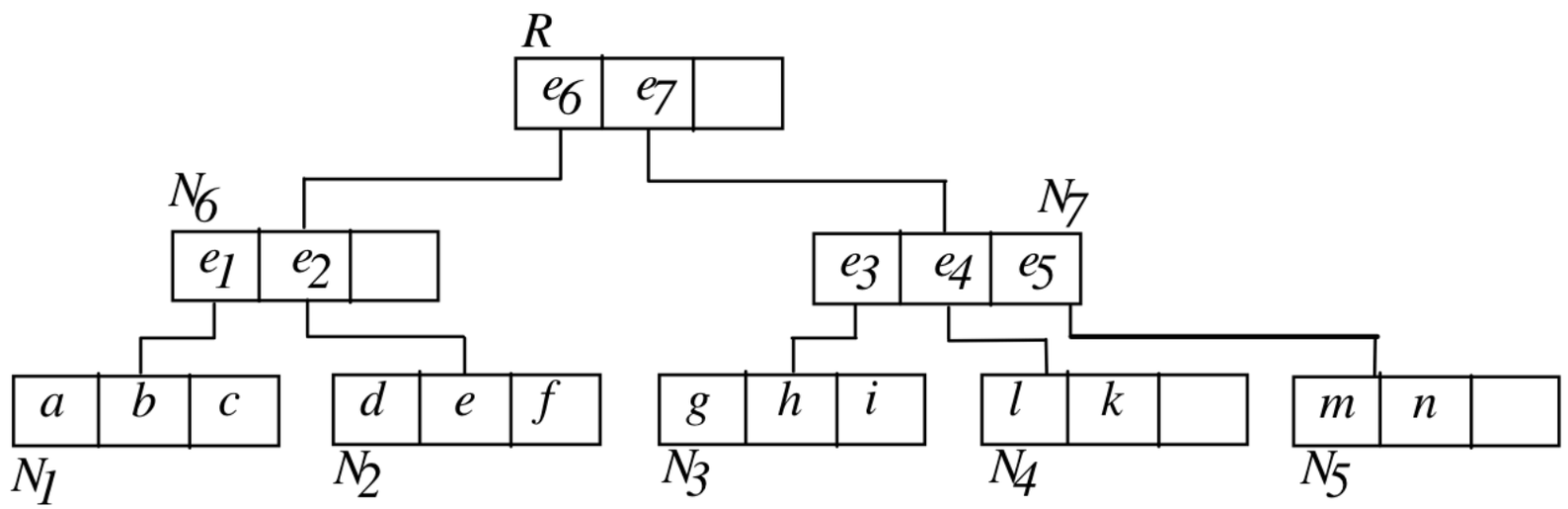


- What is the next step?

Action	heap contents	*S*
access root	<e_7,4><e_6,6>	∅
expand e_7	<e_3,5><e_6,6><e_5,8><e_4,10>	∅
expand e_3	**<*i*,5>**<e_6,6><*h*,7><e_5,8> <e_4,10><*g*,11>	{*i*}

(5) Branch and Bound Skyline Algorithm (BBS)

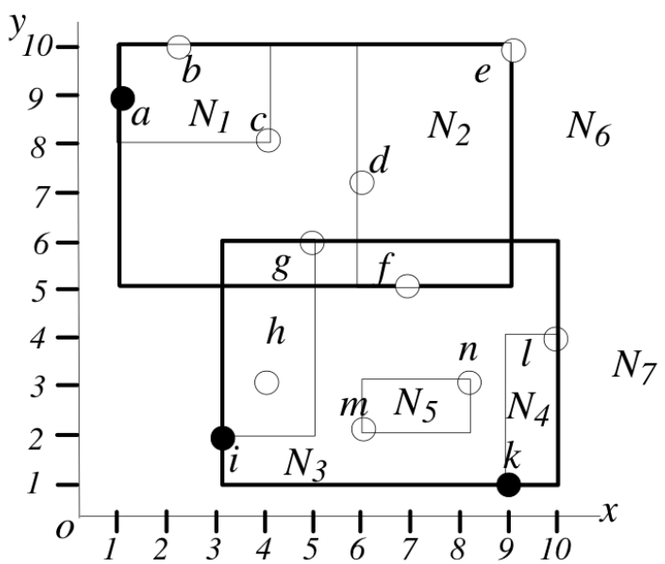


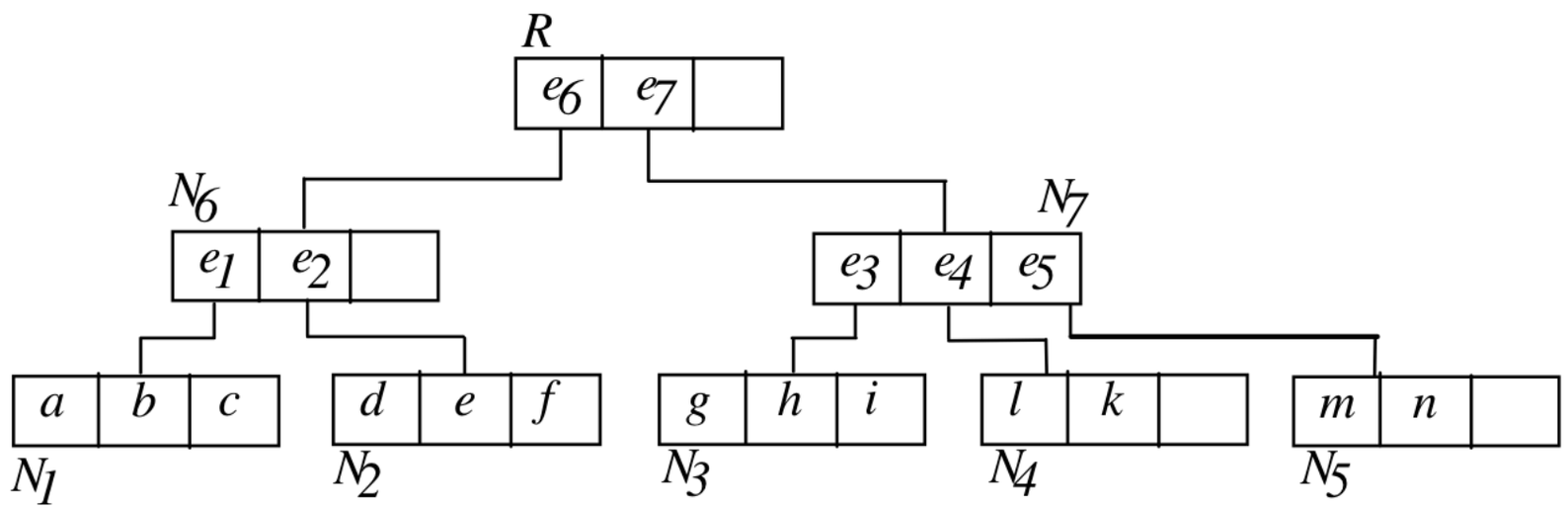


- BBS will proceed because node N6 may contain skyline points (e.g., a). Among the children of e6, however, only the ones that are not dominated by some point in S are inserted into the heap. In this case, e2 is pruned because it is dominated by point i. Entries h, e5 and g are also pruned as they are also dominated by point i.

Action	heap contents	*S*
access root	<e_7,4><e_6,6>	∅
expand e_7	<e_3,5><e_6,6><e_5,8><e_4,10>	∅
expand e_3	**<*i*,5>**<e_6,6><*h*,7><e_5,8> <e_4,10><*g*,11>	{*i*}
expand e_6	~~<*h*,7><e_5,8>~~<e_1,9><e_4,10>~~<*g*,11>~~	{*i*}

(6) Branch and Bound Skyline Algorithm (BBS)

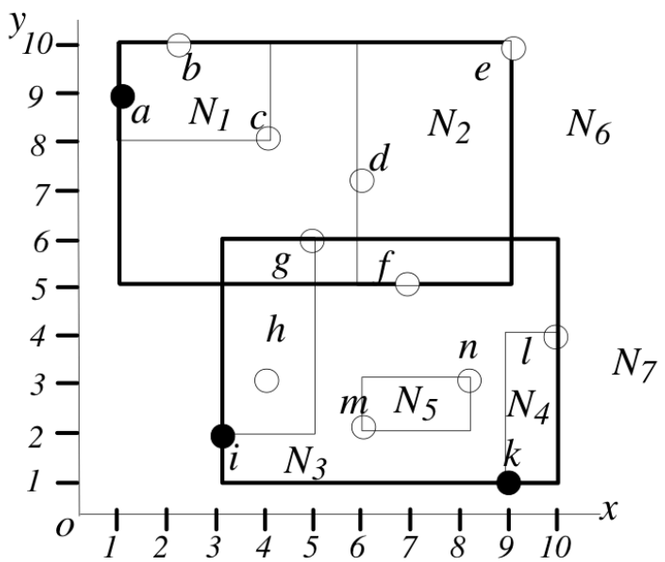


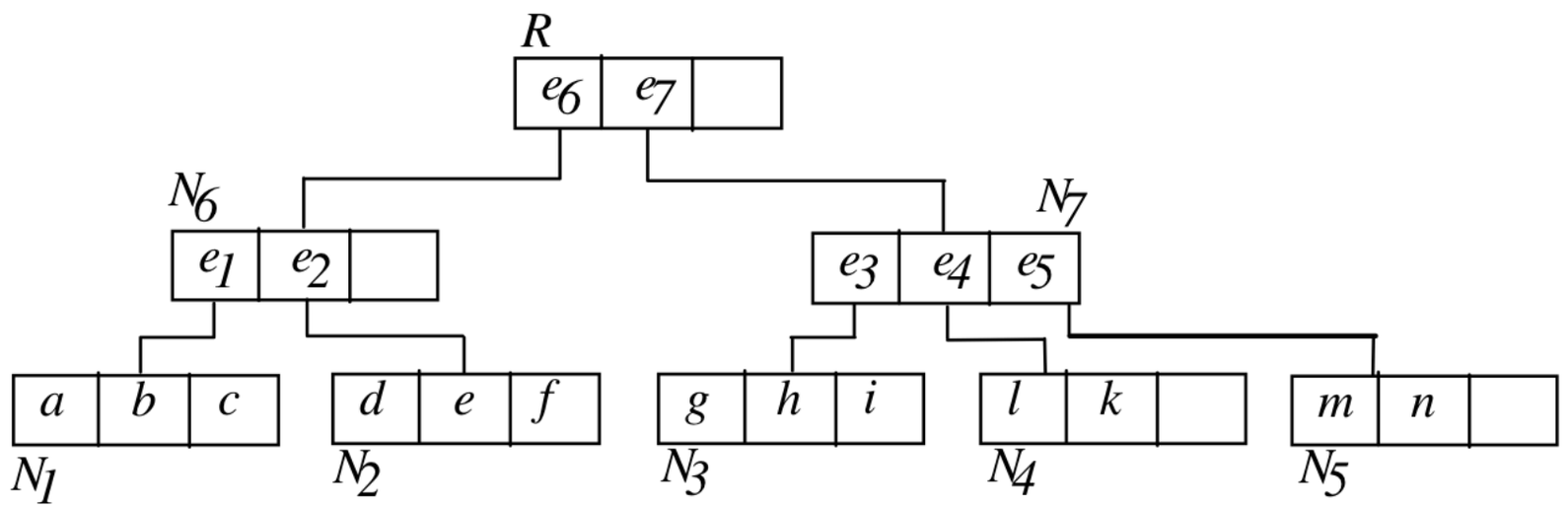


- We expand e1 and prune the points that are dominated by i.

Action	heap contents	*S*
access root	<e_7,4><e_6,6>	∅
expand e_7	<e_3,5><e_6,6><e_5,8><e_4,10>	∅
expand e_3	**<*i*,5>**<e_6,6><*h*,7><e_5,8> <e_4,10><*g*,11>	{*i*}
expand e_6	~~<*h*,7><e_5,8>~~<e_1,9><e_4,10>~~<*g*,11>~~	{*i*}
expand e_1	**<*a*,10>** <e_4,10> <*b*,12>	{*i*,*a*}

(7) Branch and Bound Skyline Algorithm (BBS)



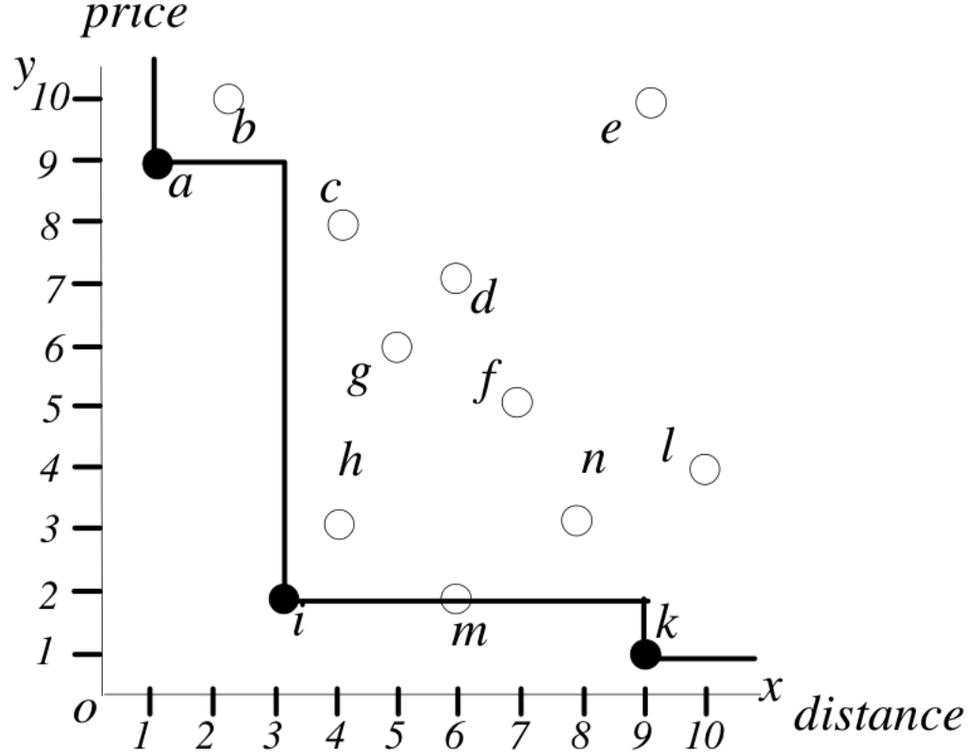


- We continue and we expand e4. We stop when all points are dominated

Action	heap contents	*S*
access root	<e_7,4><e_6,6>	∅
expand e_7	<e_3,5><e_6,6><e_5,8><e_4,10>	∅
expand e_3	**<*i*,5>**<e_6,6><*h*,7><e_5,8> <e_4,10><*g*,11>	{*i*}
expand e_6	~~<*h*,7><e_5,8>~~<e_1,9><e_4,10>~~<*g*,11>~~	{*i*}
expand e_1	**<*a*,10>** <e_4,10> <*b*,12>	{*i*,*a*}
expand e_4	**<*k*,10>** ~~<*b*,12>~~ ~~<*l*,14>~~	{*i*,*a*,*k*}

(8) Branch and Bound Skyline Algorithm (BBS)

Points a, i, k build the skyline

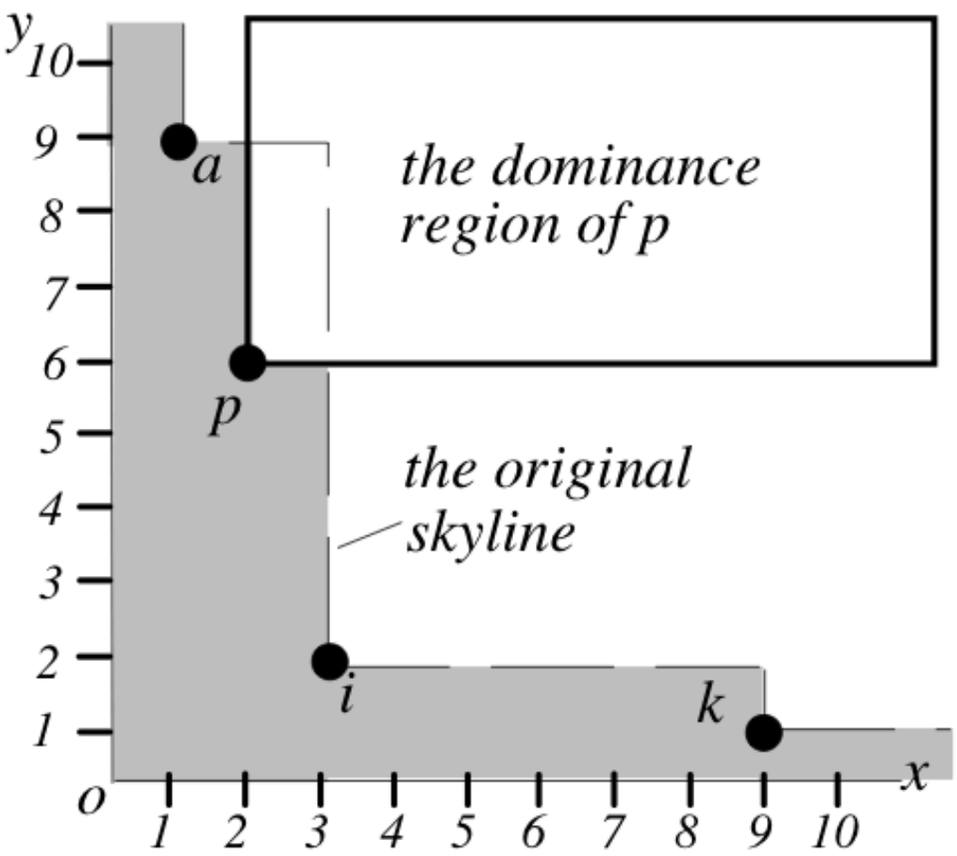


Maintaining the Skyline

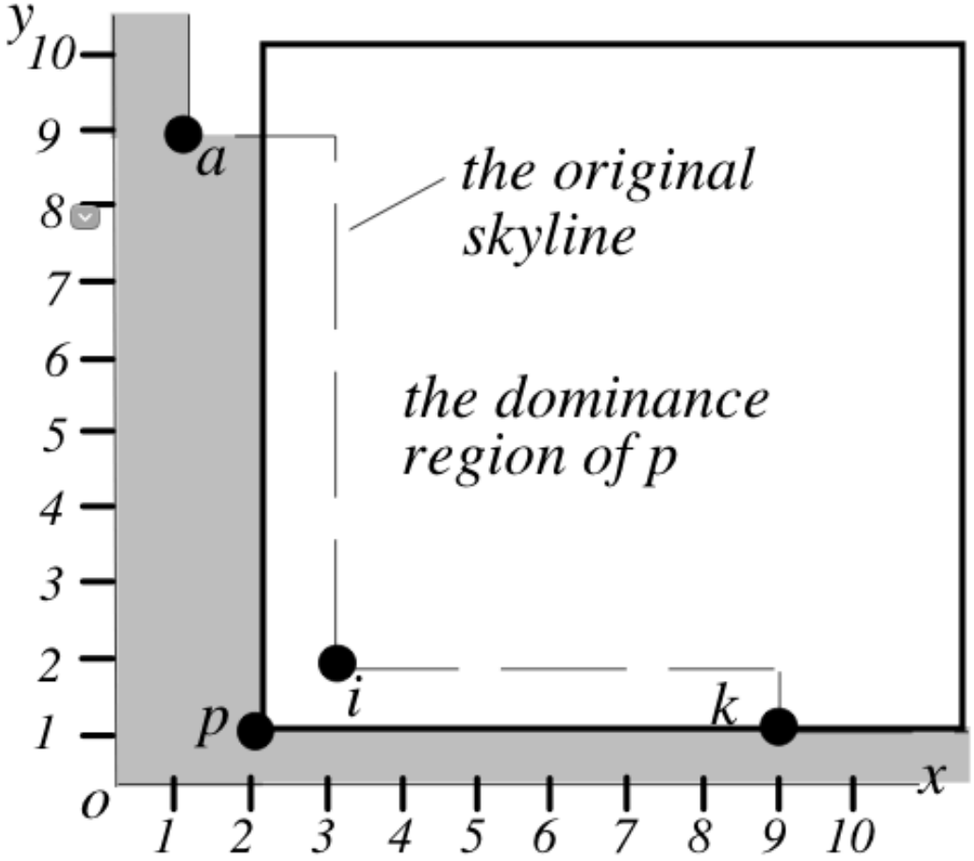
- The skyline may change due to subsequent updates (i.e., insertions and deletions) to the database
- We want to avoid re-computation

Maintaining the Skyline

Insertions



Case 1, No overlapping



Case 2, With overlapping

Maintaining the Skyline

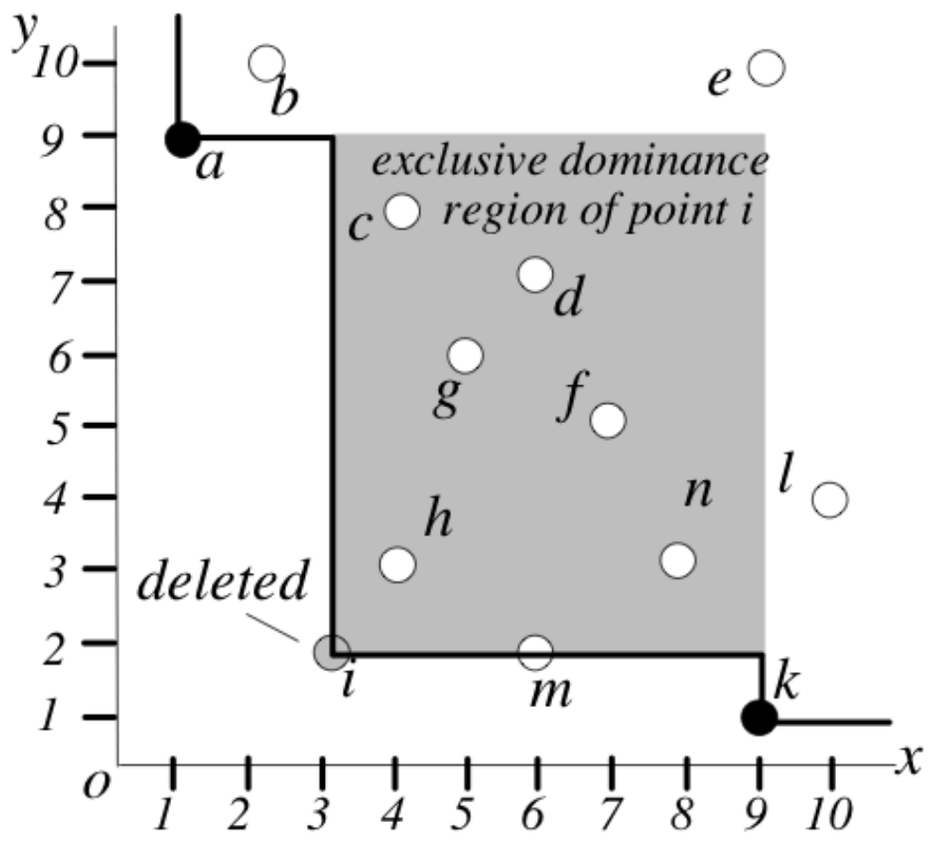
Deletions

- If the point removed is not in the skyline (which can be easily checked by the main-memory R-tree using the point's coordinates), no further processing is necessary.

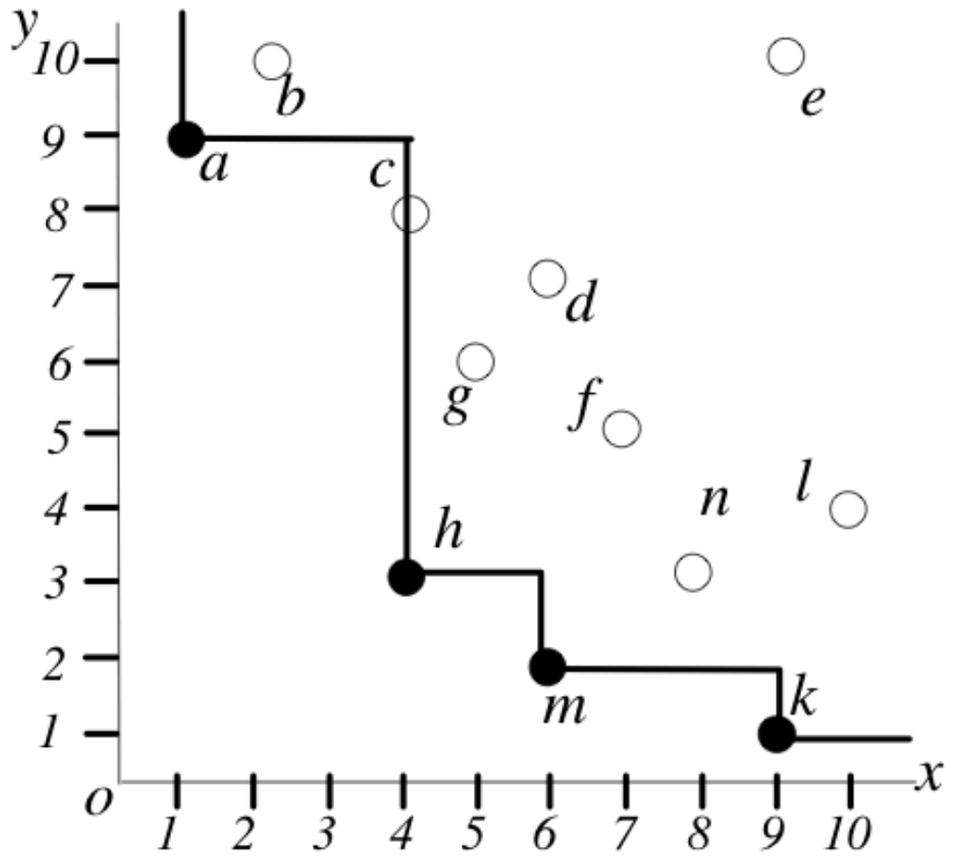
- Otherwise…

Maintaining the Skyline

Deletions



Step 1, find the dominance area of the point that needs to be deleted.



Step 2, find the skyline for that area and merge.