UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Sesión 3.0: k-d Tree

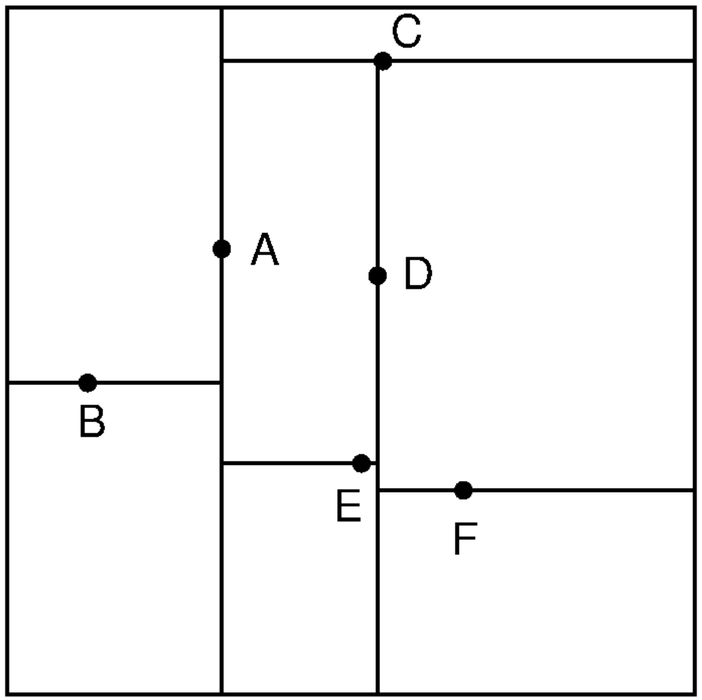
CS3102 EDA

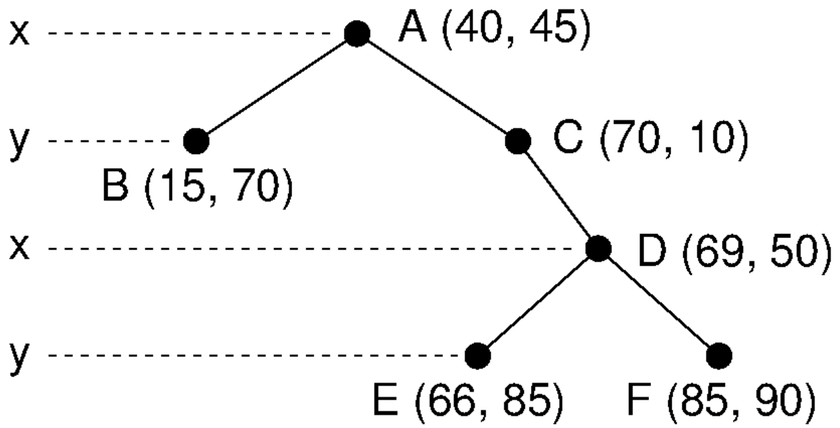
Índice

1. Point k-d Tree

Point *k-d Tree*

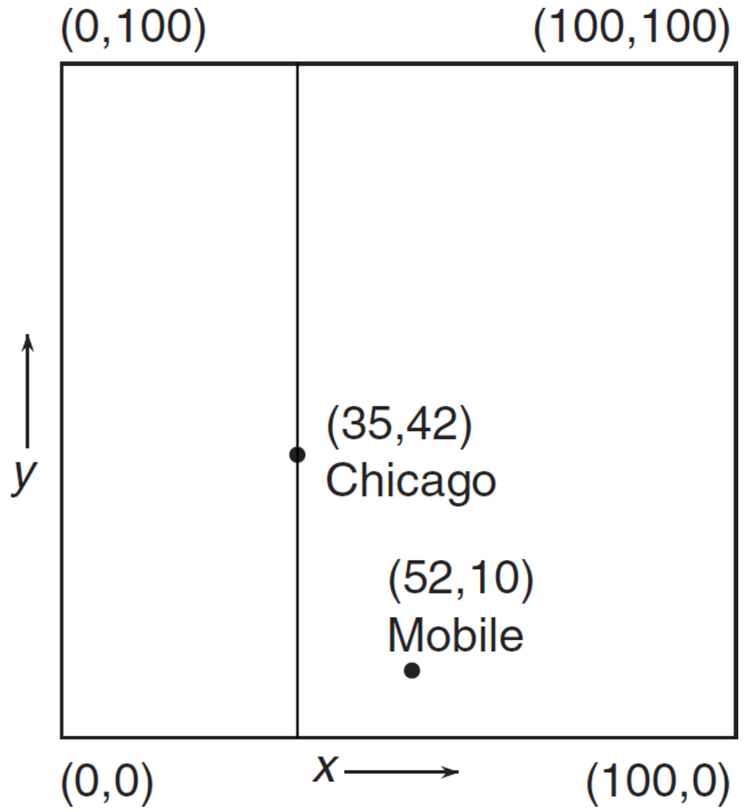
(multidimensional binary search tree, k-dimensional tree)



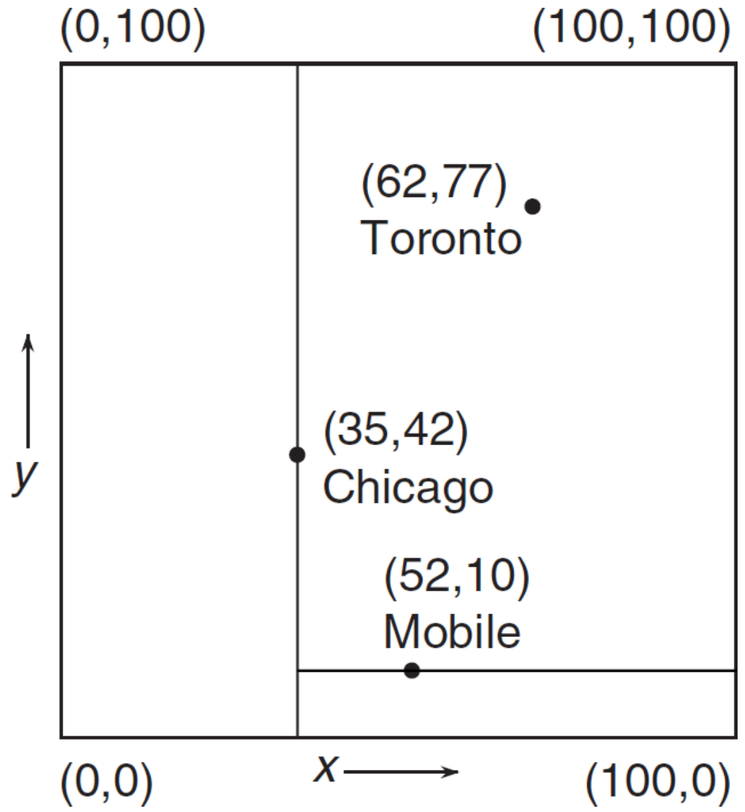


Bentley, Jon Louis (1975) "Multidimensional binary search trees used for associative searching". Communications of the ACM, 1975, vol. 18, no 9, p. 509-517.

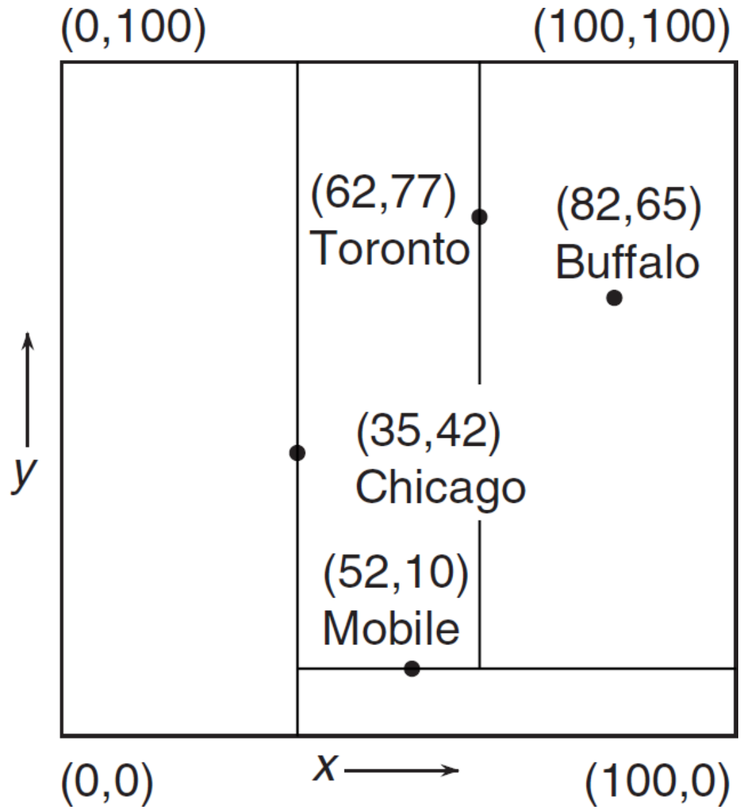
Point *k-d* Tree: *Inserción*



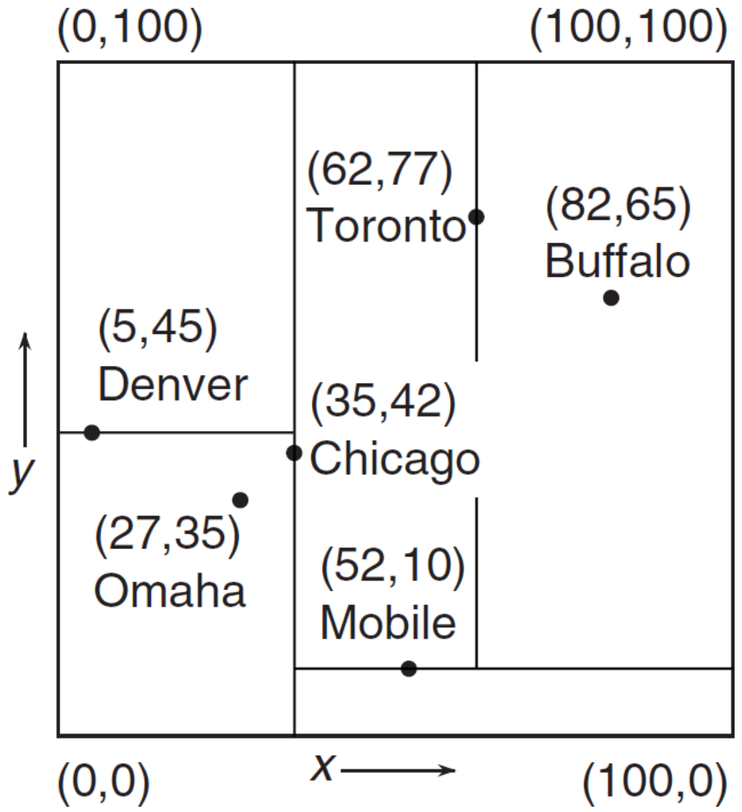
Point *k-d* Tree: *Inserción*



Point *k-d* Tree: *Inserción*



Point *k-d* Tree: *Inserción*



Point *k-d* Tree: *Inserción*

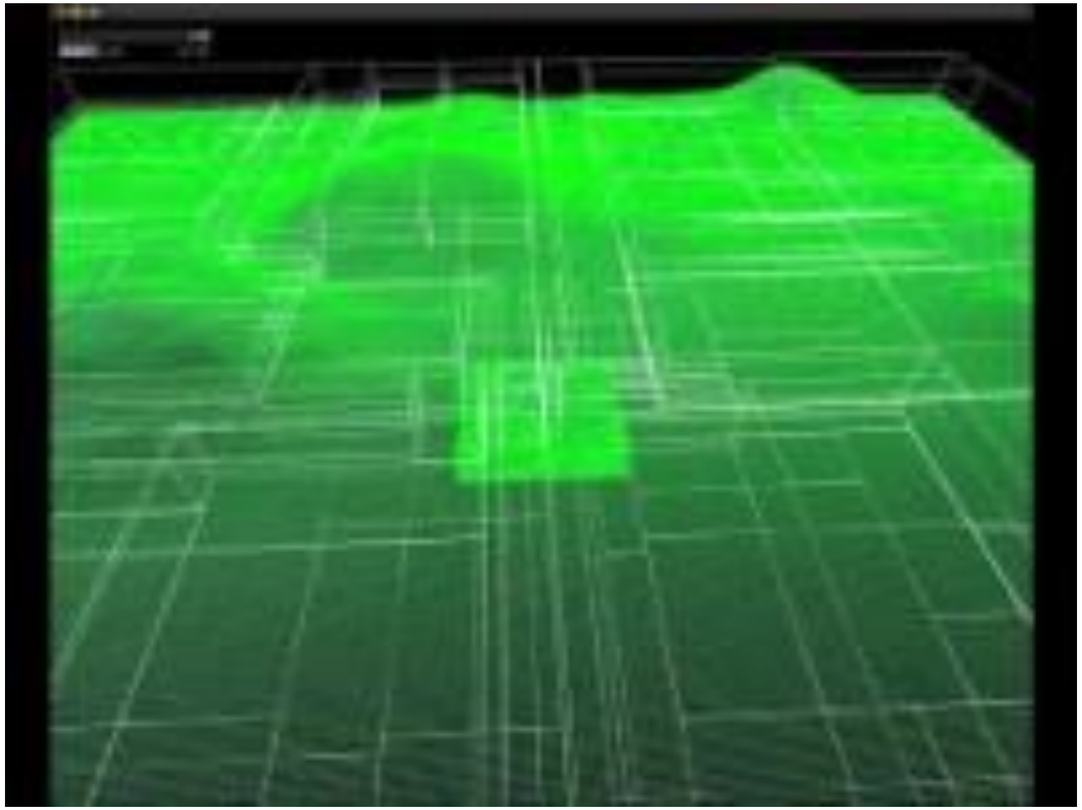
¿Point k-d Tree siempre tiene complejidad logarítmica en búsqueda?

Point *k-d* Tree: *Inserción*

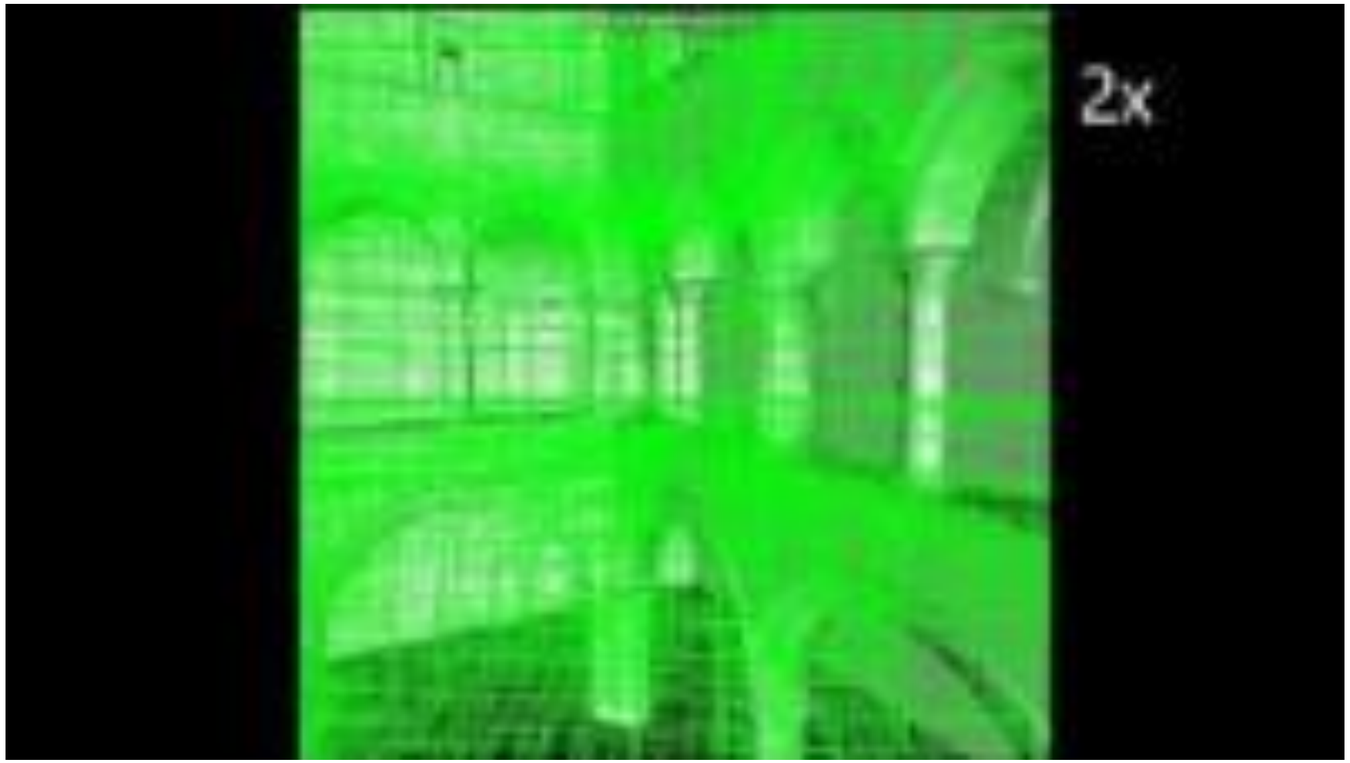
¿Cómo podríamos hacer para que Point k-d Tree tenga complejidad logarítmica en búsqueda?

Idea *general*

Idea *general*

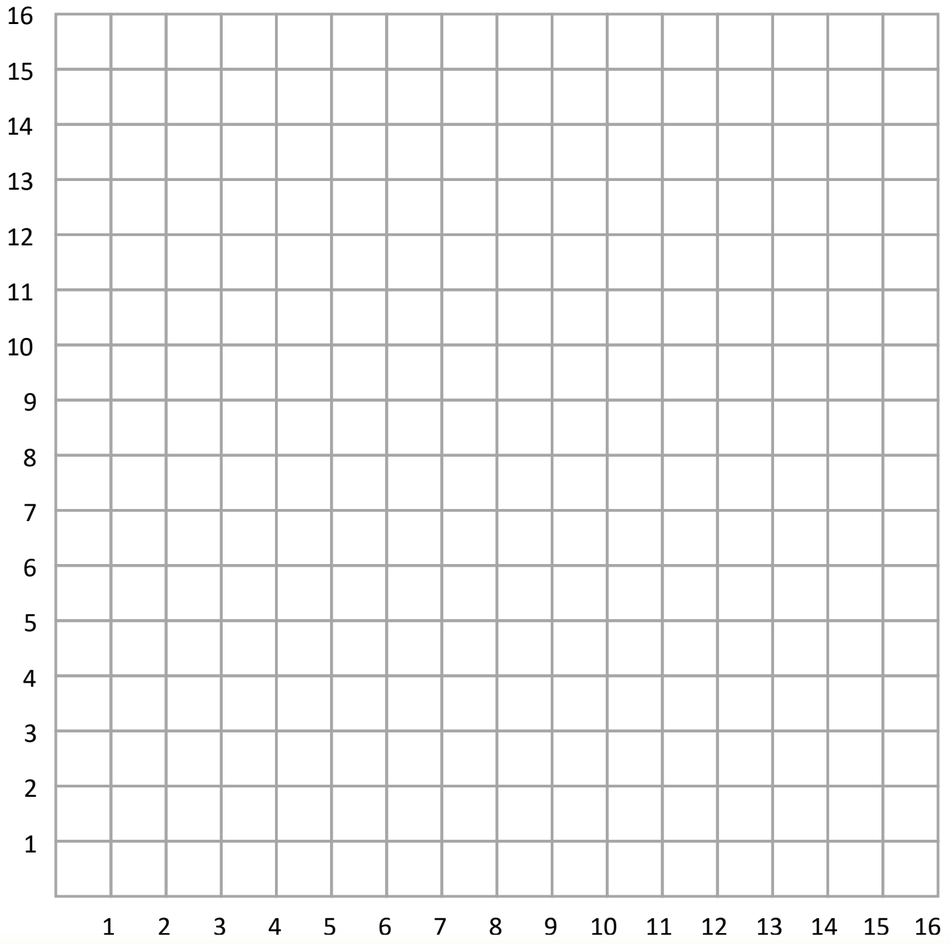


Idea *general*



Point *k-d* Tree: *Inserción*

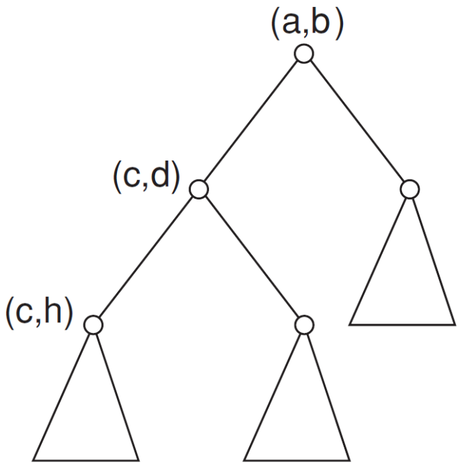
A: (5,9)
B: (16,2)
C: (3,10)
D: (13,7)
E: (11,13)
F: (10,1)
G: (9,15)
H: (1,5)
I: (14,4)
J: (2,12)
K: (15,8)
L: (8,6)



Point *k-d* Tree: *Delete*

¿Cómo borrarían puntos en Point k-d-Tree?

Point *k-d* Tree: *Delete*

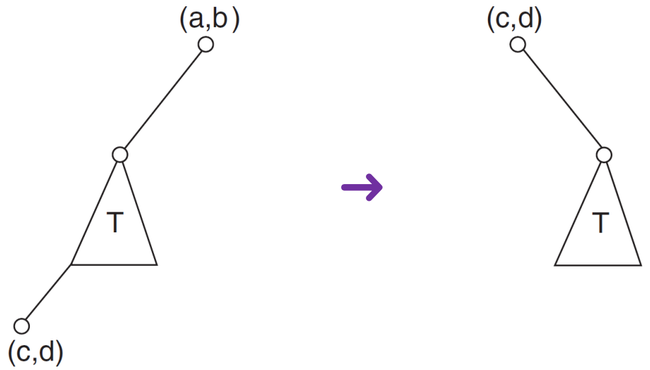


Si el punto se ubica en una hoja

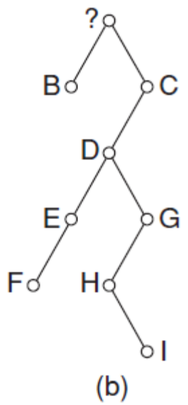
Se borra ➜ **NULL.**

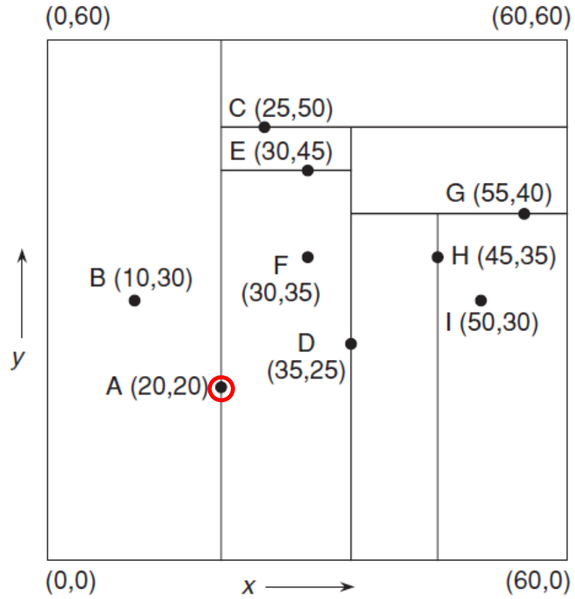
Si el punto se ubica en un nodo interno

- Sub-árbol derecho existe ➜ Escogemos el menor punto del sub-árbol derecha
- Sub-árbol derecho no existe

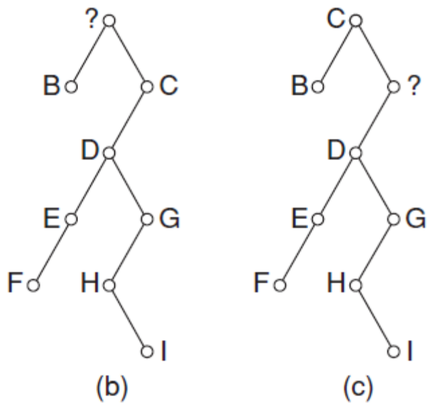


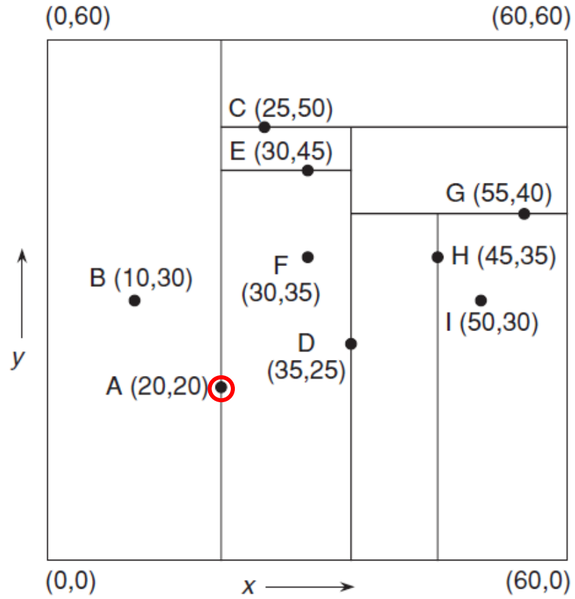
Point *k-d* Tree: *Delete*



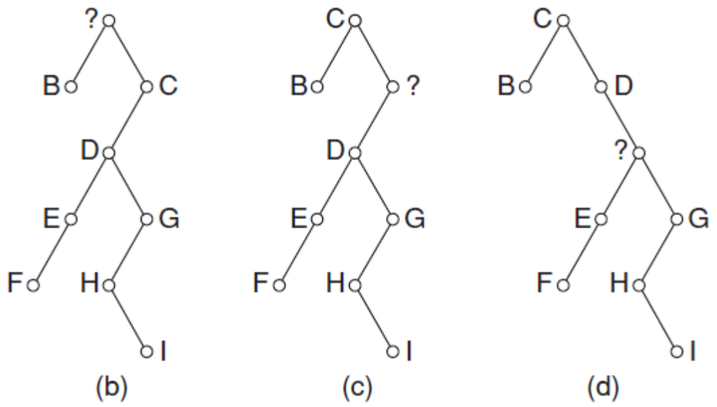


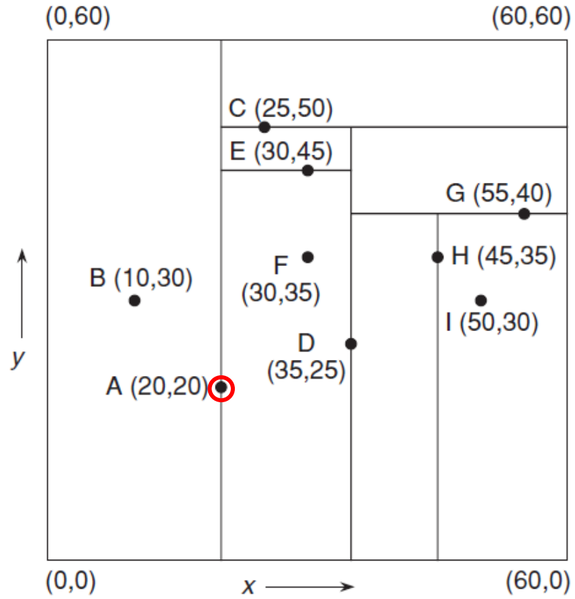
Point *k-d* Tree: *Delete*



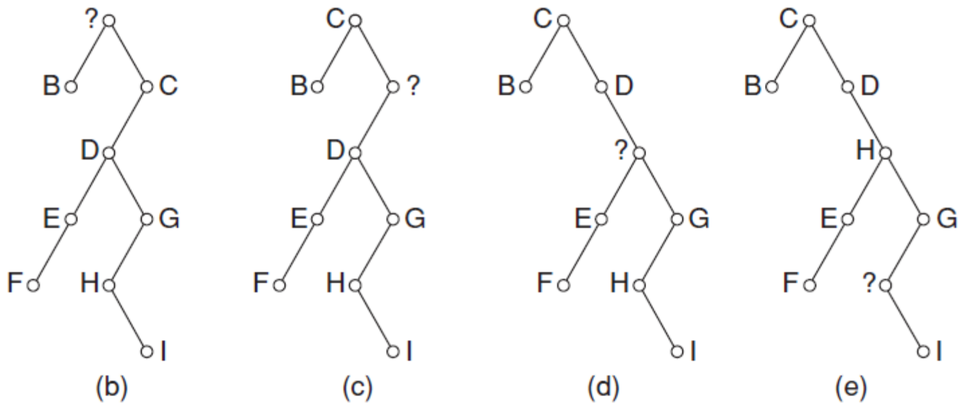


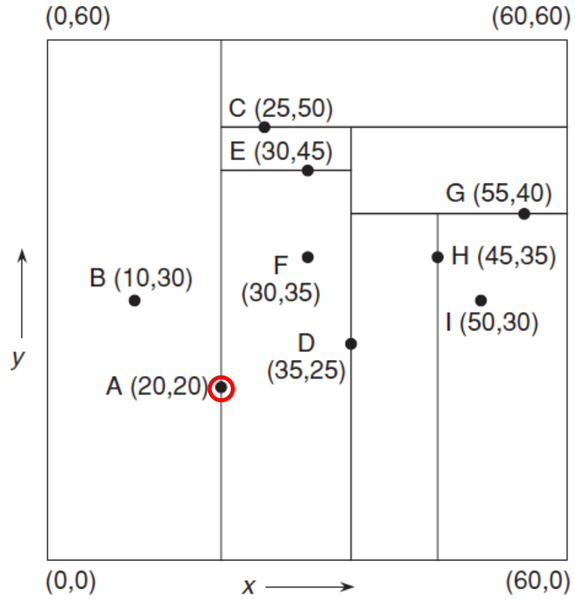
Point *k-d* Tree: *Delete*



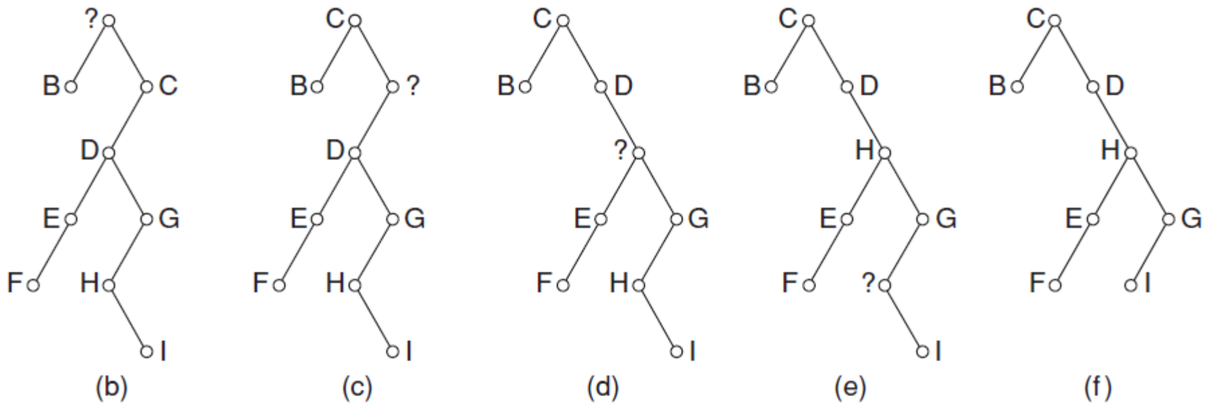


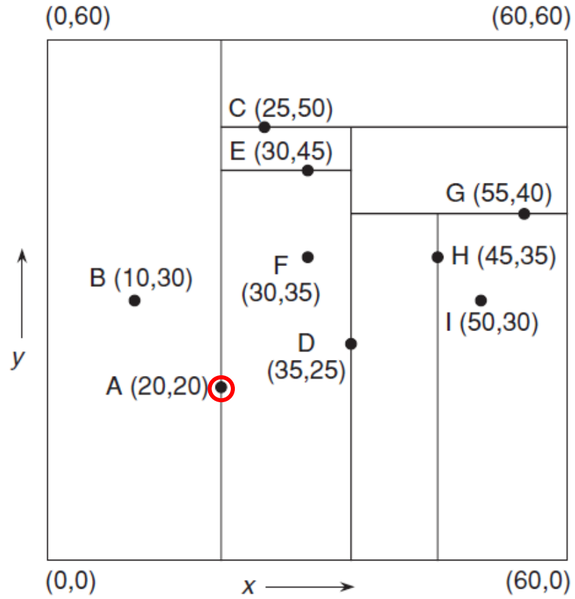
Point *k-d* Tree: *Delete*



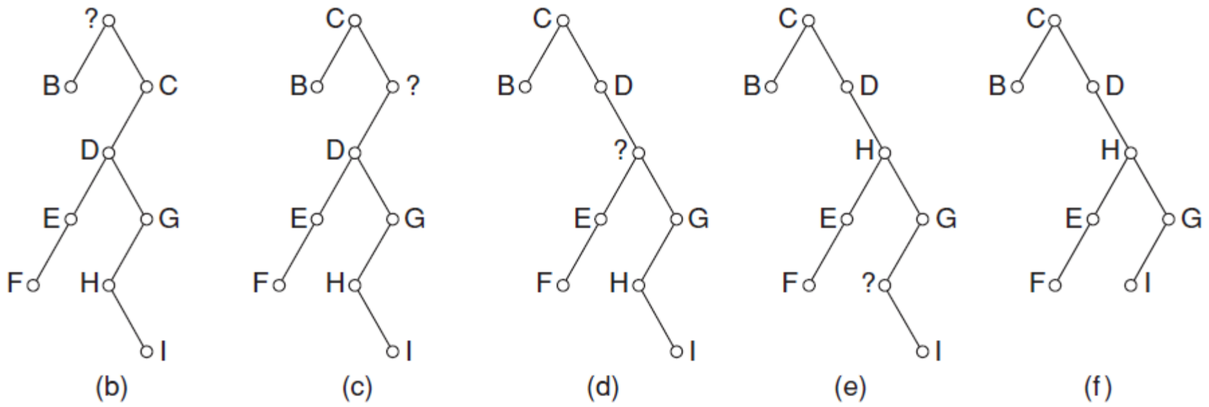


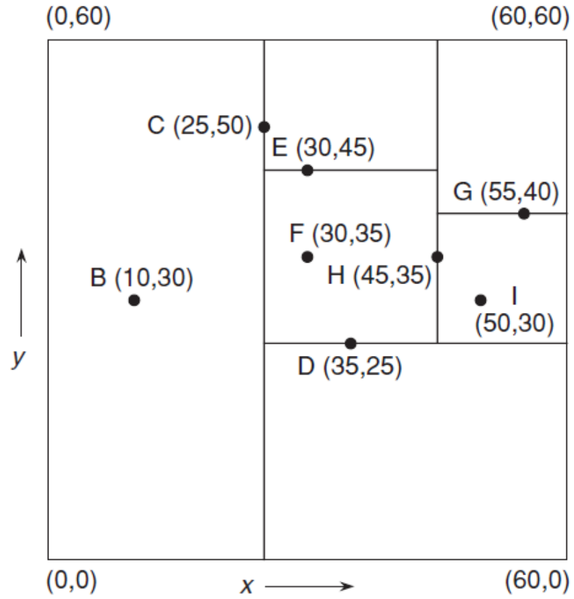
Point *k-d* Tree: *Delete*





Point *k-d* Tree: *Delete*





UTEC

UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA