Examen Parcial

Graded

Student

Paolo Vasquez Grahammer

Total Points

19.5 / 25 pts

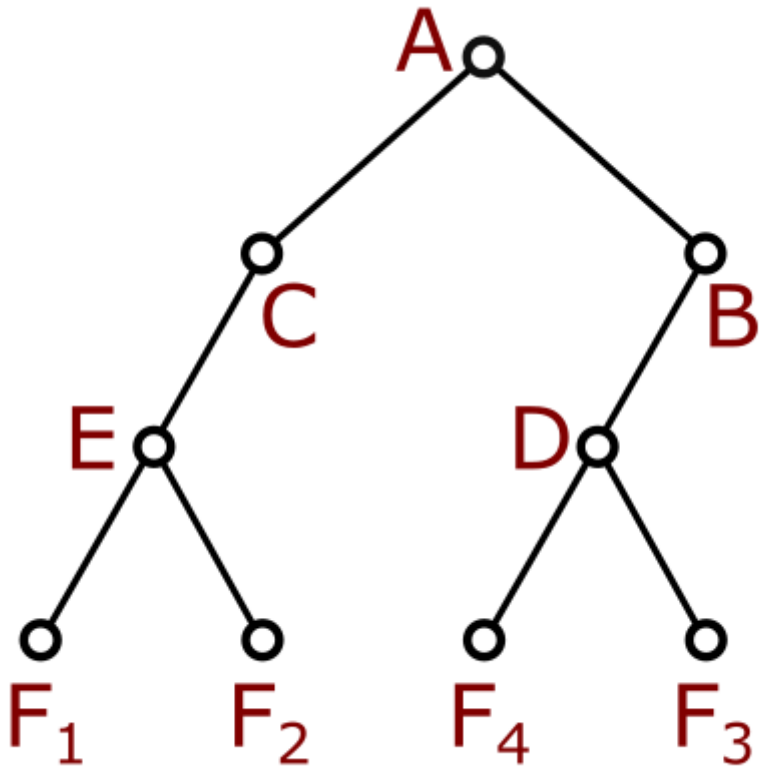
Question 1

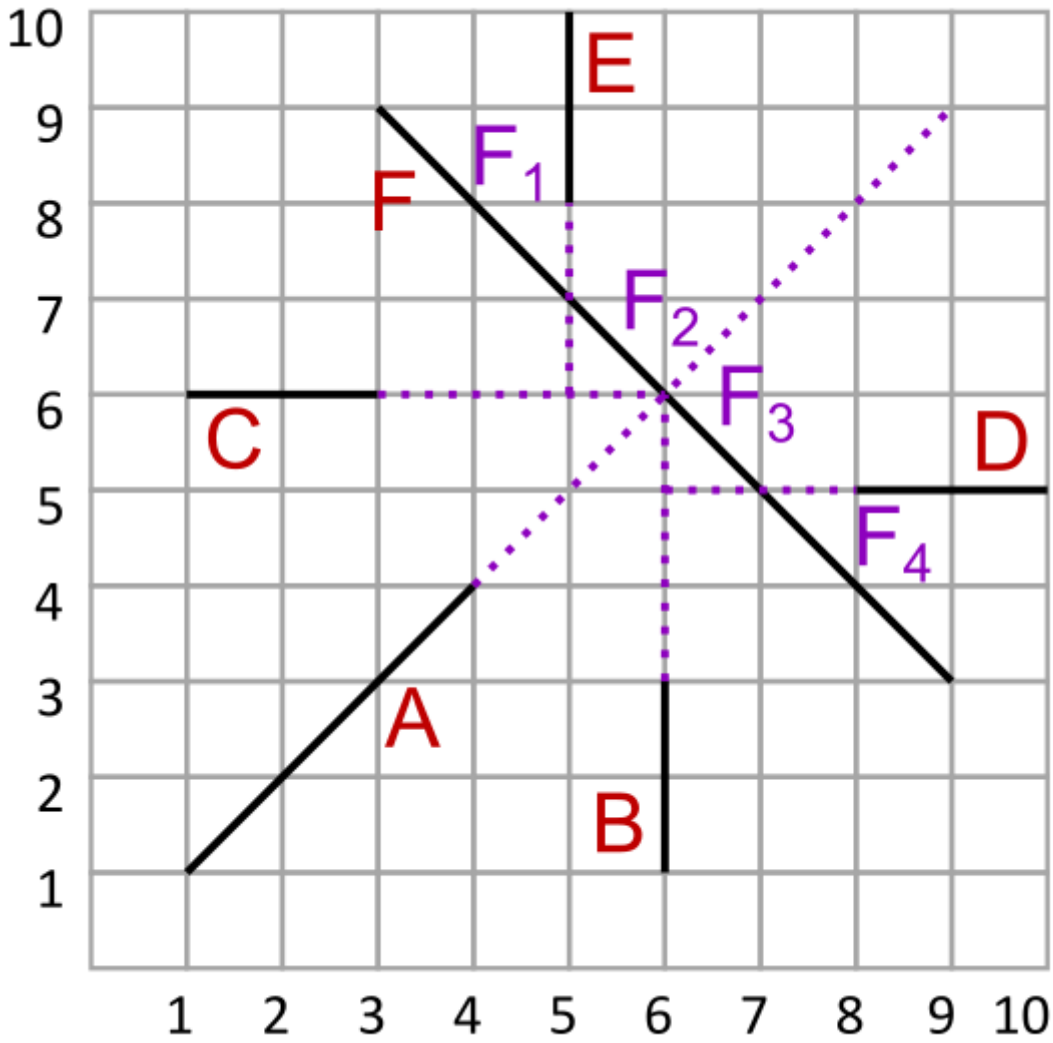
BSP-tree

Resolved **1.5** / 2 pts

+ 2 pts Correcto!

+ 0 pts





+ 1 pt Click here to replace this description.

✔ **+ 1.5 pts** Click here to replace this description.

Regrade Request **Submitted on: Jun 07**

Segun como defino las normales (se pueden ver dibujadas las flechas) el arbol estaría correctamente armado. Demuestro que a partir de planos con sus normales, puedo estructurar correctamente un BSP-Tree.

Pero seguiste todas las normas, solo fallaste ahi :'c

Reviewed on: Jun 17

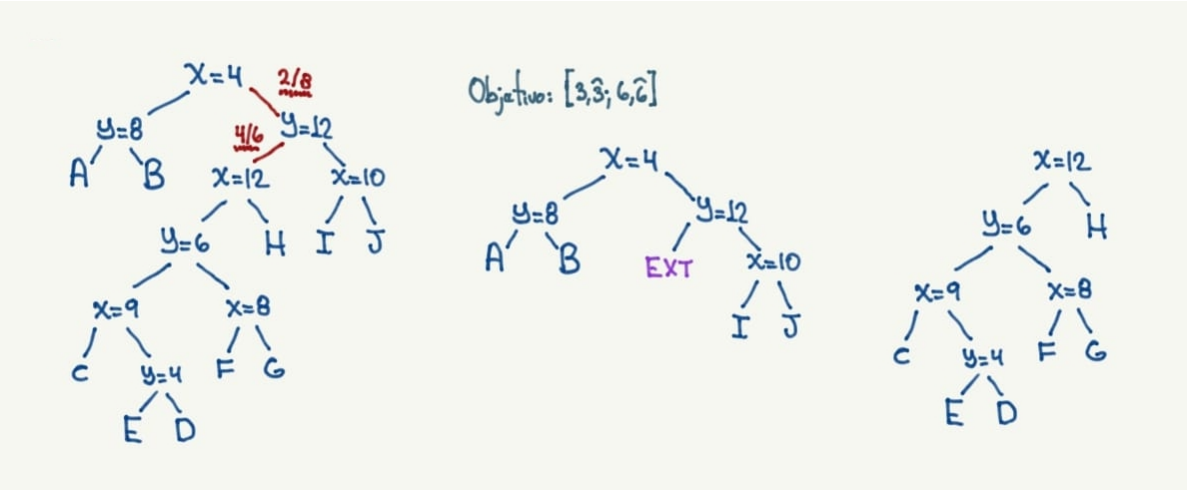
Question 2

hB-Tree

1 / 2 pts

+ 2 pts **Correcto!**

+ 0 pts



✔ + 1 pt Click here to replace this description.

+ 1.5 pts Click here to replace this description.

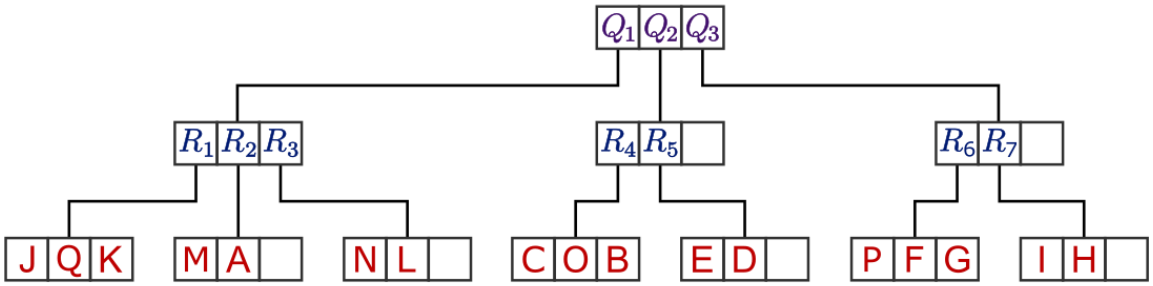
Question 3

Dynamic Hilbert R-Tree

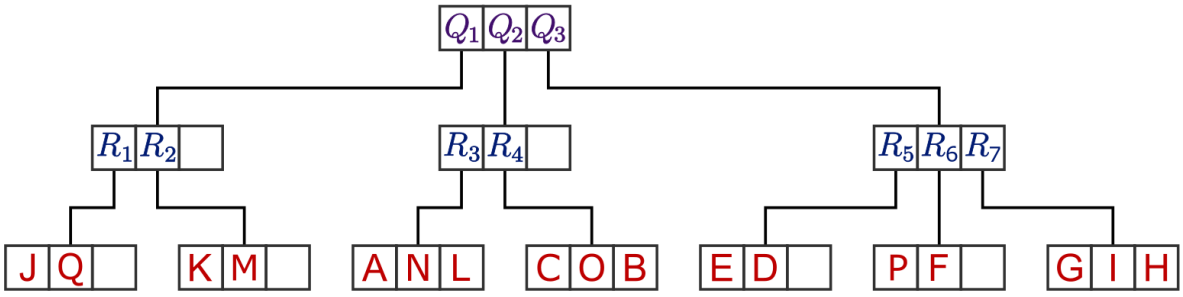
2.5 / 2.5 pts

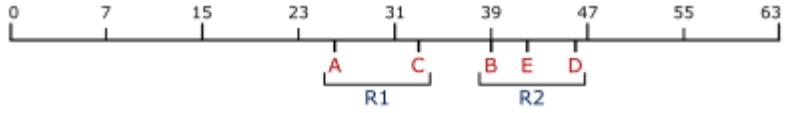
✔ **+ 2.5 pts** **Correcto!**

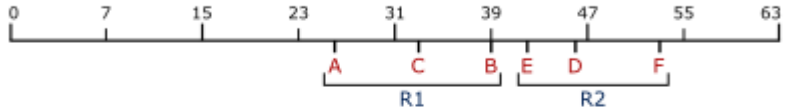
+ 0 pts La otra solución:

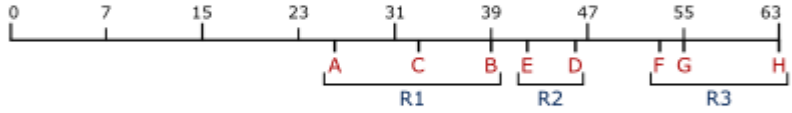


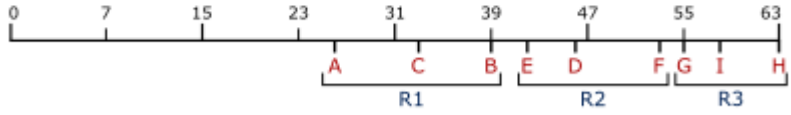
+ 0 pts

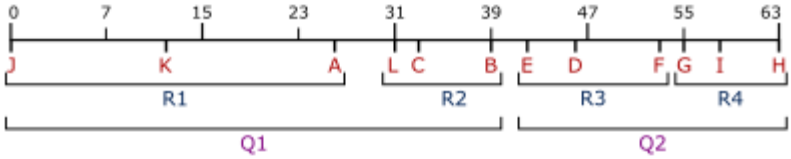


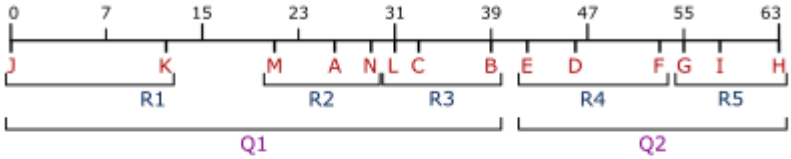


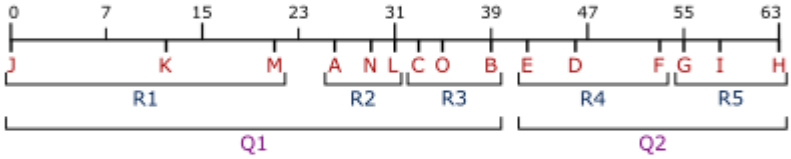


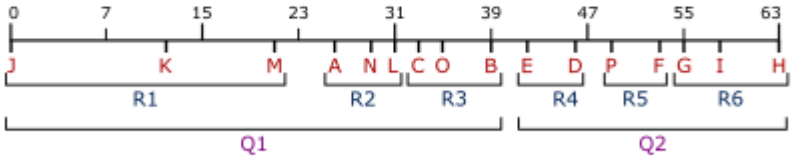


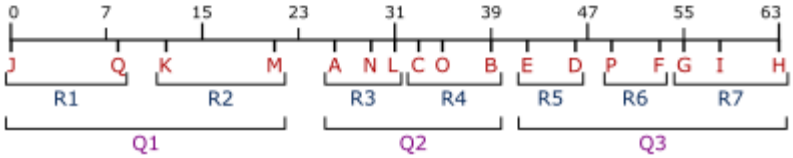












+ 1.5 pts Click here to replace this description.

+ 2 pts Click here to replace this description.

+ 1 pt Click here to replace this description.

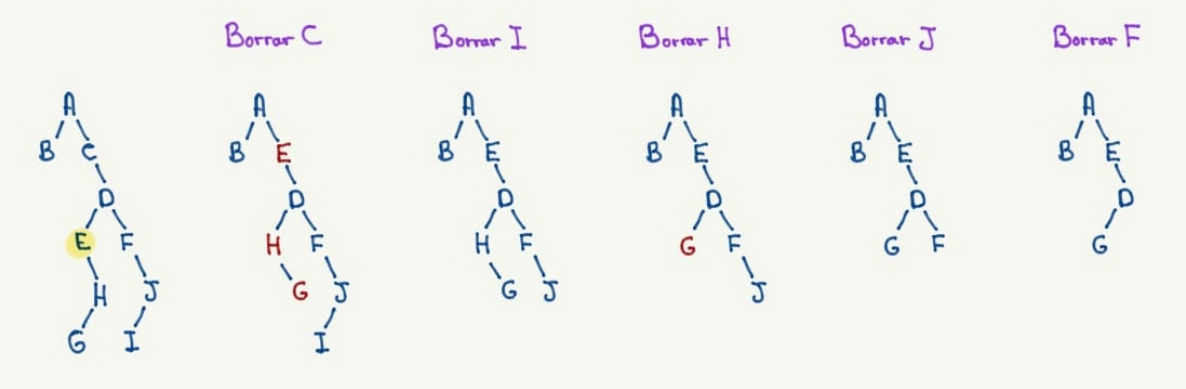
Question 4

Point k-d-Tree: Borrado

2 / 2 pts

✔ **+ 2 pts** **Correcto!**

+ 0 pts



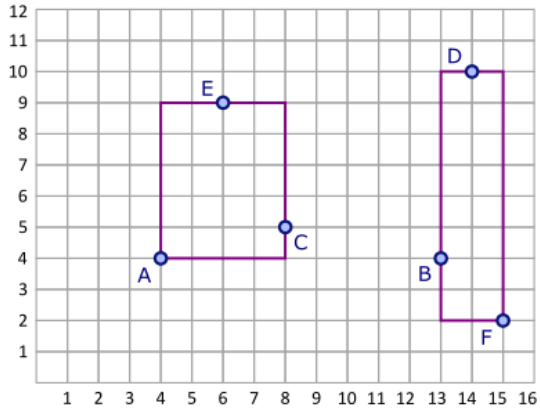
Question 5

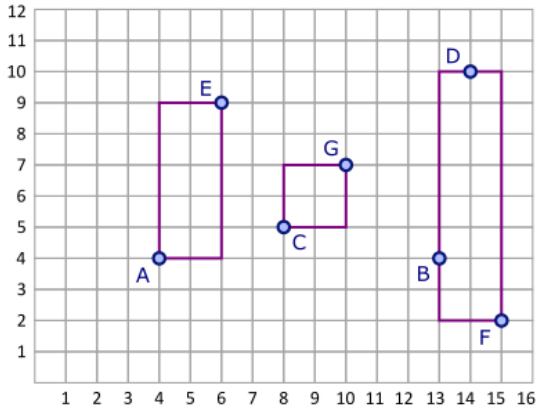
R-tree

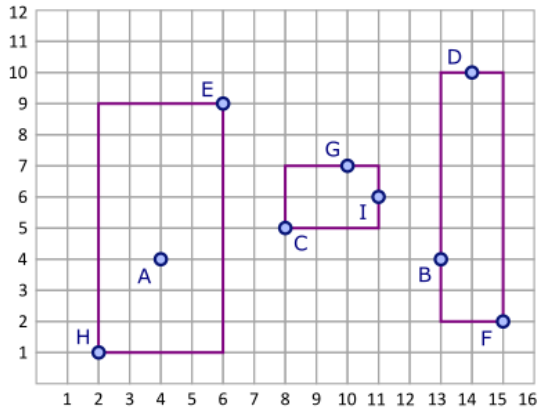
2.5 / 2.5 pts

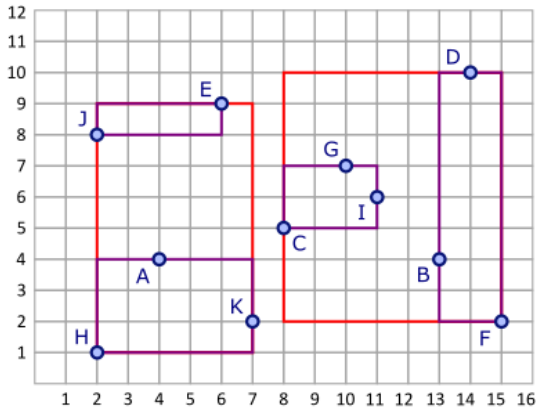
✔ **+ 2.5 pts Correcto!**

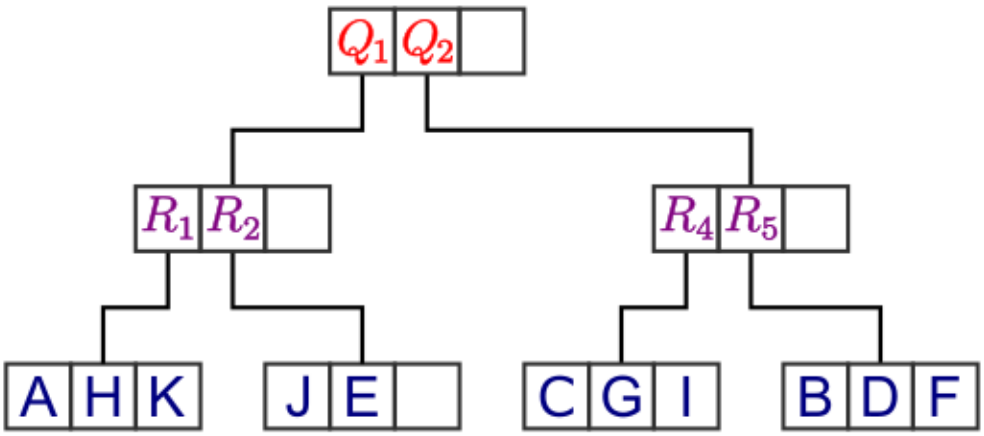
+ 0 pts











+ 1 pt Click here to replace this description.

+ 0.5 pts Click here to replace this description.

Te equivocaste F con I

Question 6

h-index

2.5 / 2.5 pts

✔ **+ 2.5 pts** **Correcto!**

+ 0 pts El mínimo dimensión de cuadrícula que puede aceptar índices 46 y 160 es 256×256 (mínima potencia de dos superior a dichos números).
Si bien hay infinitas posibles dimensiones de cuadrícula, la relación entre los puntos se mantendrá constante.
Entonces, trabajemos con una cuadrícula 256×256

La posiciones de $h = 46$ son $(4, 6)$
La posiciones de $h = 160$ son $(12, 12)$

Entonces la distancia es $\sqrt{8^2 + 6^2} = \sqrt{100} = 10$

+ 1.5 pts Click here to replace this description.

+ 1 pt Click here to replace this description.

Question 7

R-tree rectangles

2.5 / 2.5 pts

✔ **+ 2.5 pts** **Correcto!**

+ 0 pts Dado que queremos determinar la profundidad máxima del árbol (el peor de los casos), consideramos que cada nodo interno tiene al menos m hijos.
La profundidad máxima D es cuando cada nodo interno tiene solo m hijos. Entonces $N \leq m^D$, por tanto hacemos $D = \log_m N$.
En una búsqueda de rango, en el peor caso, tendríamos que examinar todos los nodos.

- En el nivel 1: examinamos 1 rectángulo (la raíz).
- En el nivel 2: examinamos m rectángulos.
- En el nivel 3: examinamos m^2 rectángulos.
- $\ldots$
- En el nivel $D - 1$: examinamos m^{D-2} rectángulos.
- En el nivel D: examinamos $M \times m^{D-2}$ rectángulos (en el peor de los casos, estos nodos estarían llenos).

Por lo tanto, el número máximo de rectángulos es:
$1 + m + m^2 + \cdots + m^{D-2} + M \cdot m^{D-2}$

Resolviendo la serie geométrica:
$\frac{m^{D-1} - 1}{m - 1} + M \cdot m^{D-2}$, donde $D = \log_m N$.

Reemplazando:
$$\frac{\frac{N}{m} - 1}{m - 1} + M \cdot \frac{N}{m^2} = \frac{N - m}{m \cdot (m - 1)} + \frac{M \cdot N}{m^2}$$

+ 1.5 pts Click here to replace this description.

+ 0.5 pts Click here to replace this description.

Question 8

Point k-d-Tree: kNN

Resolved 3 / 3 pts

✔ **+ 3 pts** **Correcto!**

+ 0 pts

```
def k_nearest_neighbors(root, point, k):
    neighbors = []
    pq = PriorityQueue()
    pq.put((0, root))

    while not pq.empty():
        distance, node = pq.get()

        if node is None:
            continue

        current_distance = euclidean_distance(point, node.point)

        if len(neighbors) < k:
            heapq.heappush(neighbors, (-current_distance, node.point))
        else:
            if current_distance < -neighbors[0][0]:
                heapq.heappop(neighbors)
                heapq.heappush(neighbors, (-current_distance, node.point))

        axis = len(point) % len(node.point)

        if point[axis] < node.point[axis]:
            pq.put((abs(point[axis] - node.point[axis]), node.left))
            pq.put((abs(point[axis] - node.point[axis]), node.right))
        else:
            pq.put((abs(point[axis] - node.point[axis]), node.right))
            pq.put((abs(point[axis] - node.point[axis]), node.left))

    return [point for _, point in sorted(neighbors, reverse=True)]
```

+ 2 pts Click here to replace this description.

+ 1.5 pts Click here to replace this description.

+ 0.5 pts Click here to replace this description.

Falta el caso base!

Regrade Request **Submitted on:** Jun 07

Me indicas que falta el caso base, con el que normalmente se detiene la llamada recursiva. Básicamente con el if (node->left != nullptr) y el analogo para el node->right, y sus respectivos condicionales dentro, manejo si la recursividad debe seguir dandose o parar. Como la funcion que defino es void, si no entra a ningun condicional simplemente detiene la recursividad.

mmm no me queda claro, pero te creo

Reviewed on: Jun 17

Question 9

Adaptive k-d-Tree

1 / 1 pt

✔ + 1 pt Correcto!

+ 0 pts Respuesta: b) Puede mejorar la eficiencia en datos sesgados o agrupados, ya que las divisiones se ajustan dinámicamente para minimizar la varianza dentro de cada subregión, resultando en un tiempo de búsqueda potencialmente $O(\log n)$.

Question 10

Representacion explicita

1 / 1 pt

✔ + 1 pt Correcto!

+ 0 pts Respuesta: c) Respuesta más rápida a las consultas de características.

Question 11

Point QuadTree

0 / 1 pt

+ 1 pt Correcto!

✔ + 0 pts Respuesta: b) Las celdas de los nodos hoja se vuelven extremadamente pequeñas, aumentando la altura del árbol.

Question 12

Median-split k-d-Tree

0 / 1 pt

+ 1 pt Correcto!

✔ + 0 pts Respuesta: c) La eficiencia de la búsqueda disminuye debido a divisiones ineficaces, lo que puede resultar en tener que recorrer todas las hojas del árbol.

Question 13

PR QuadTree

0 / 1 pt

+ 1 pt Correcto!

✔ + 0 pts Respuesta: c) Proporciona un ordenamiento que mantiene la proximidad espacial, reduciendo el número de nodos visitados.

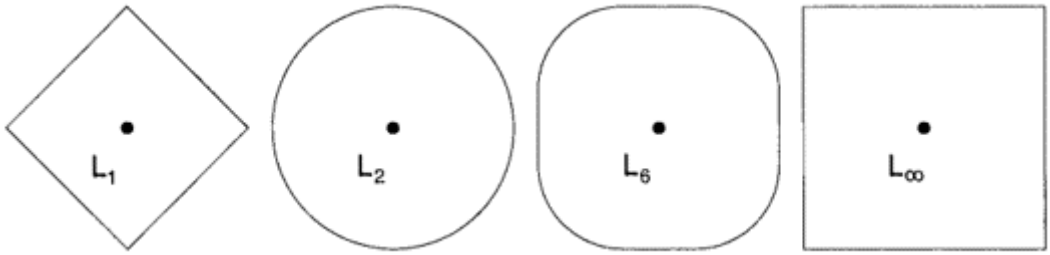
Question 14

Minkowski

0 / 1 pt

+ 1 pt **Correcto!**

✔ **+ 0 pts** **Respusta: b)** $\pi_1 = \pi_\infty > \pi_2$



Si el radio es 1, entonces:

- $\pi_1 = \frac{4*(|1-0|+|0-1|)}{2} = 4$
- $\pi_2 = 3.1416\ldots$
- $\pi_\infty = \frac{4*(\max(-1,1)+\max(1,1))}{2} = 4$

Estructura de datos Avanzados

Examen Parcial
Pregrado 2024-1

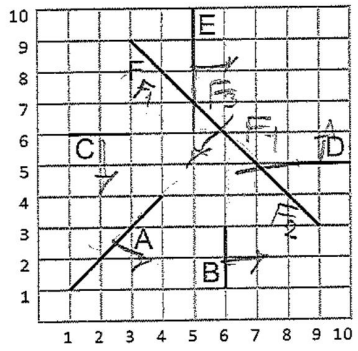
Profesor: Victor Flores Benites

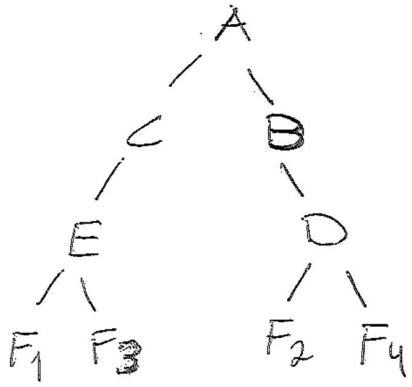
Apellidos: Vasquez Grabonmer Nombres: Paolo

Sección: 1 Fecha: 20/05/2024 Nota:

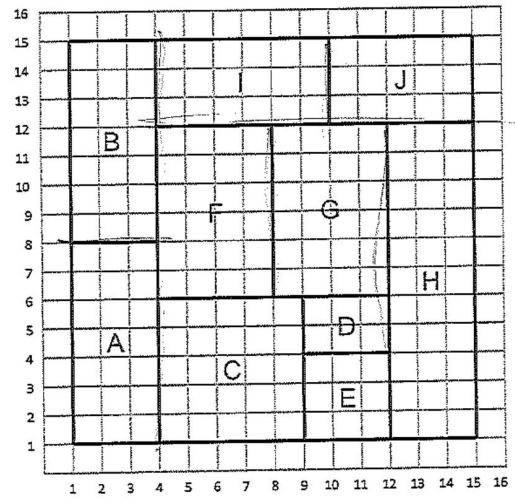
Indicaciones:
La Duración es de **120 minutos.**
La evaluación consta de **9 preguntas.**

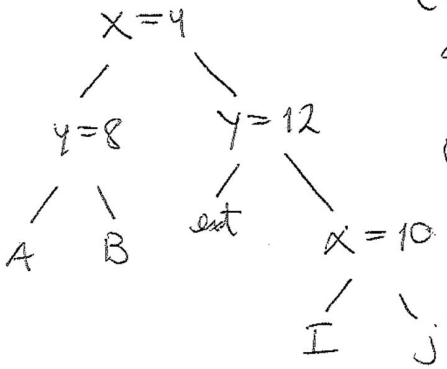
Pregunta 1 (2 puntos)
Inserte los segmentos en orden alfabético a un BSP-Tree. De como respuesta el árbol resultante.





Pregunta 2 (2 puntos)
La siguiente figura muestra un nodo sobrecargado de un hB-tree. Deberá ejecutar el algoritmo de Split sobre dicho nodo y dar como respuesta los sub-árboles de los nodos resultantes.

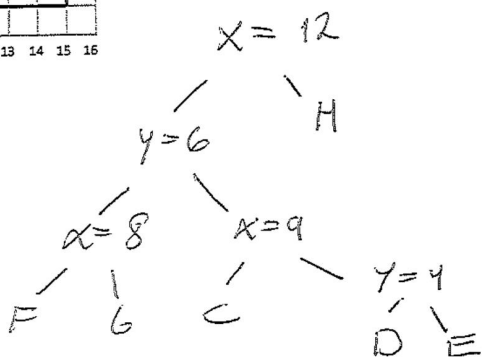




(En caso sea necesario arbol completo, en cuaderno) (al final no alcanzo tiempo) (si lo puse)

[3.33 ; 6,66]

Test



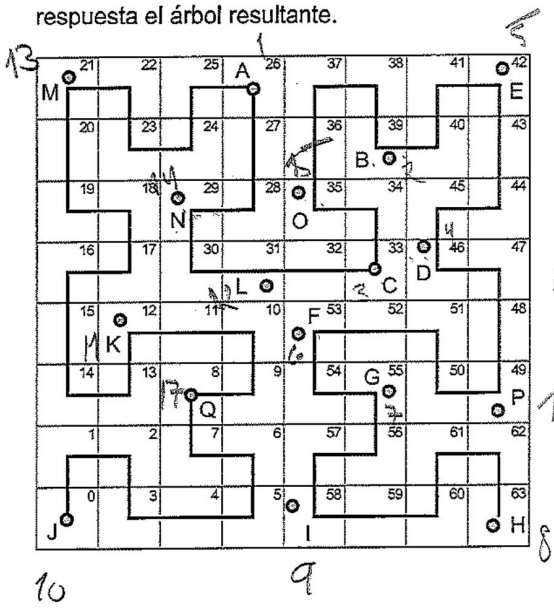
Estructura de datos Avanzados

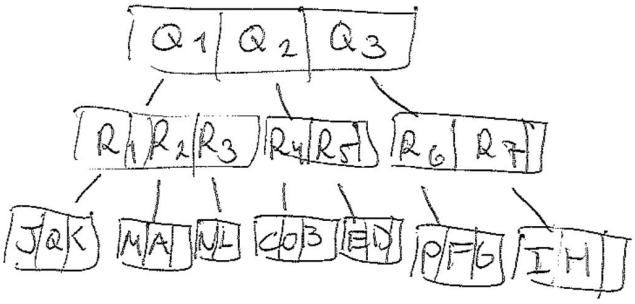
Examen Parcial

Pregrado 2024-1

Pregunta 3 (2.5 puntos)

Inserte los puntos en orden alfabético a un Dynamic Hilbert R-Tree ($M = 3$, $m = 2$). De como respuesta el árbol resultante.

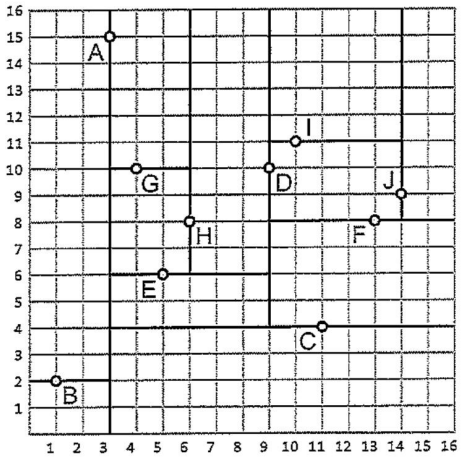


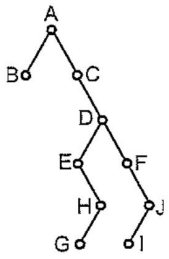


(ojala este bien)

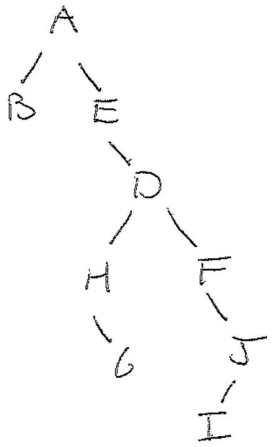
Pregunta 4 (2 puntos)

En el k-d-Tree mostrado, borre los puntos C, I, H, J, F en el orden indicado. De como respuesta el árbol resultante.

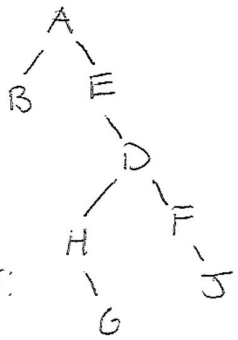




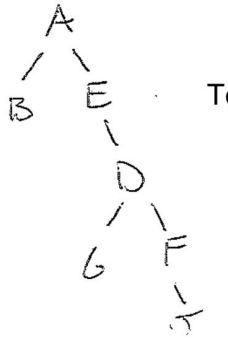
1 Del C:



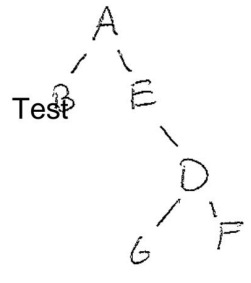
2. Del I:



Del H:

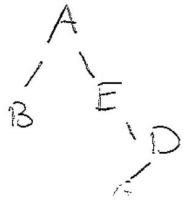


4. Del J:



Test

5. Del F:



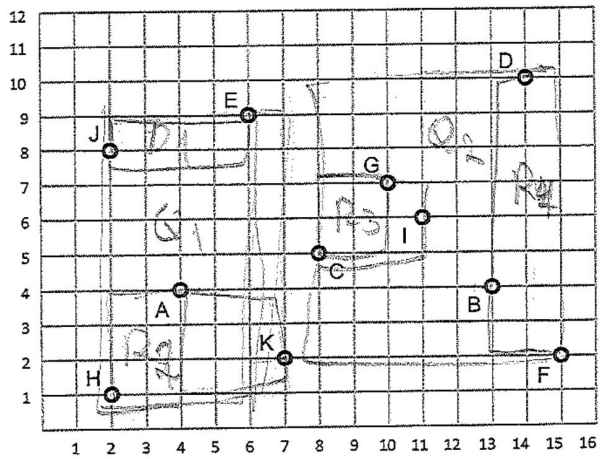
Estructura de datos Avanzados

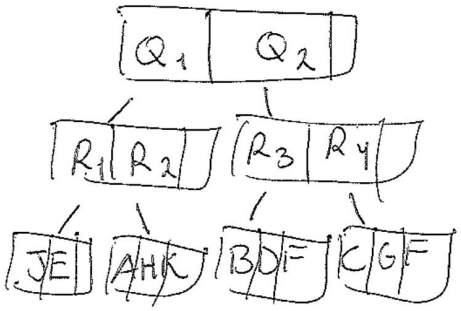
Examen Parcial

Pregrado 2024-1

Pregunta 5 (2.5 puntos)

Inserte los puntos en orden alfabético a un R-Tree ($M = 3$, $m = 2$). De como respuesta el árbol resultante.





Pregunta 6 (2.5 puntos)

Sean los puntos A y B indexados en celdas con h-index 46 y 160 respetivamente. Calcule la distancia euclidiana de A y B.

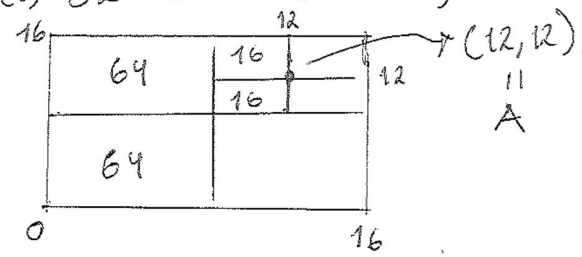
La curva de Hilbert tiene forma: ⊓ ; 46 y 160

↳ grilla de $2^4 \times 2^4$ (16×16)

a) h-index 160:

160 − 128 = 32 (1)

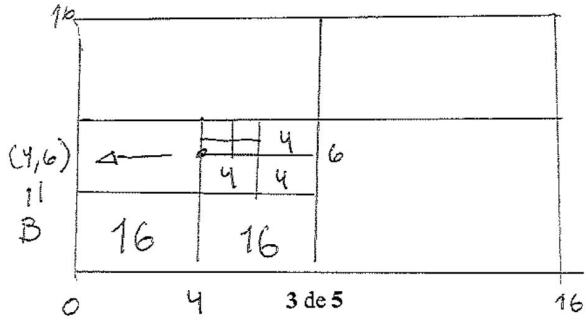
(1) 32 − 32 = 0 (cuadrado)



b) h-index 46:

46 − 32 = 14 (1)

(1) 14 − 12 = 2 (en ese sector)



Test

$|A-B| = \sqrt{(12-4)^2 + (12-6)^2} = 10$

(dist. euclidiana)

(en el sector 0 la curva tiene forma ⊐)

Pregunta 7 (2.5 puntos) (continuación en cuadernillo)

Suponga que se insertan N puntos a un R-tree en el que cada nodo interno puede tener a lo sumo M hijos y al menos m hijos. ¿Cuál es el número máximo de rectángulos de búsqueda que podría tener que examinar durante una búsqueda de rango?

- El maximo se da en el peor caso, es decir se visitan todos los nodos, y el arbol es super profundo (todos los nodos tienen m hijos excepto los más profundos que tienen M hijos)
- Entonces la altura es: $H = \log_m(N)$
- Entonces el recorrido será algo asi:
  1. Se visita la raiz: $m^0 = 1$ nodos
  2. La raiz tiene m hijos, entonces en ese nivel visitamos $m^1 \cdot m$ nodos
  3. Repetimos esta logica hasta el nivel $H-2$, ya que los ultimos nodos internos tienen M hijos, es decir, se visitan $M \cdot m^{H-2}$ nodos.

Pregunta 8 (3 puntos)

Proponga un algoritmo de k vecinos más cercanos para Point k-d-tree con complejidad $O(\log n)$ en el mejor de los casos.

: P. queue → (dist, pair < point, point >) : val = point

- P. queue will reorganize following dist: $dist_i < dist_{i+1} \ldots$

```
void KNN (K, &Pqueue, node) : &Pque is result
if (node->left != nullptr):
    if (Pqueue.size < K):
        P.que.insert (dist (node->val, node->left->val), pair (node, node->left))
        KNN (K, Pqueue, node->left)

    else:
        if dist (node->val, node->left->val) < Pqueue.dist[ind]
            Pqueue.insert (dist (node->val, node->left->val), pair (node, node->left))
        Pqueue.reshape (K)
        KNN (K, Pque, node->left)
// analogous to node->right (en el cuadernillo)
```

Pregunta 9 (6 puntos)

En las siguientes preguntas, marque la respuesta correcta:

1. ¿Qué impacto tiene la adaptación de las divisiones en un Adaptive k-d Tree sobre la eficiencia de las consultas de vecino más cercano?
   a) Siempre reduce el tiempo de consulta a $O(1)$ debido a la mejor partición del espacio, aunque a costa de mayor uso de memoria para mantener las divisiones.
   b) Puede mejorar la eficiencia en datos sesgados o agrupados, ya que las divisiones se ajustan dinámicamente para minimizar la varianza dentro de cada subregión, resultando en un tiempo de búsqueda potencialmente $O(\log n)$.
   c) Aumenta la complejidad de la búsqueda a $O(n)$ en el peor de los casos, debido a la sobrecarga computacional de calcular las variaciones óptimas en cada nivel del árbol.
   d) No afecta la eficiencia de las consultas, ya que la adaptación solo se realiza durante la construcción del árbol y no durante las búsquedas.

2. ¿Qué ventaja tiene la representación explícita sobre la implícita en las consultas de características?:
   a) Menor uso de memoria.
   b) Mayor velocidad en la identificación de objetos.
   c) Respuesta más rápida a las consultas de características.
   d) Facilita la agregación de bloques.

3. ¿Qué problema surge al utilizar un Point QuadTree en un espacio donde los puntos están distribuidos de manera uniforme pero densa?
   a) Los nodos internos se vuelven demasiado grandes para ser manejados eficientemente.
   b) Las celdas de los nodos hoja se vuelven extremadamente pequeñas, aumentando la altura del árbol.
   c) Las consultas de rango se vuelven lineales en tiempo debido a la falta de balanceo.
   d) Los nodos hoja se llenan rápidamente y requieren una división constante.

4. ¿Cuál es una desventaja de utilizar Median-split k-d Tree en un conjunto de datos con una distribución muy sesgada?
   a) La estructura del árbol se vuelve inevitablemente no balanceada, lo que hace que todas las búsquedas se vuelvan lineales.
   b) Aunque la construcción es eficiente, las búsquedas de vecinos cercanos se vuelven lineales en tiempo debido a la alta varianza en cada partición.
   c) La eficiencia de la búsqueda disminuye debido a divisiones ineficaces, lo que puede resultar en tener que recorrer todas las hojas del árbol.
   d) La construcción del árbol se vuelve exponencial en tiempo, debido a la necesidad de recalcular medianas en cada partición.

5. ¿Cómo puede una curva de Hilbert mejorar la eficiencia de una consulta kNN en un PR QuadTree?
   a) Minimiza el solapamiento de nodos, mejorando la poda durante las consultas.
   b) Divide el espacio de manera uniforme, asegurando que todos los cuadrantes tengan la misma densidad de puntos.
   c) Proporciona un ordenamiento que mantiene la proximidad espacial, reduciendo el número de nodos visitados.
   d) Mejora la localización de puntos de alta densidad en el espacio, facilitando la búsqueda.

6. Dada la distancia Minkowski L_p, se define π_p como la relación entre el perímetro y el diámetro de la circunferencia. Entonces:
   a) $\pi_1 = \pi_2 > \pi_\infty$
   b) $\pi_2 < \pi_1 = \pi_\infty$
   c) $\pi_1 = \pi_2 < \pi_\infty$
   d) $\pi_2 > \pi_1 = \pi_\infty$