Comenzado el	jueves, 17 de octubre de 2024, 15:16
Estado	Finalizado
Finalizado en	jueves, 17 de octubre de 2024, 16:45
Tiempo empleado	1 hora 28 minutos
Calificación	**15,00** de 20,00 (**75**%)

Pregunta **1**

Correcta

Se puntúa 0,50 sobre 0,50

Multiple choice (Spatial DS) (Out1, Out2, Out6)

Son estrategias para ordenar los octantes de un octree

I. Nearest Neighbor

II. Z-order

III. Range Query

IV. Bounding Box

V. Todas las anteriores

Seleccione una:

- a. VVFFF
- b. VVVVV
- c. FVFFF ✓
- d. VVFFF
- e. FFFFF

Respuesta correcta

La respuesta correcta es: FVFFF

Pregunta **2**

Correcta

Se puntúa 0,50 sobre 0,50

True or False (Spatial DS) (Out1, Out2, Out6)

La maldición de la dimensionalidad se refiere únicamente a como las dimensiones hacen mas difícil el calculo de las distancias entre los datos a medida que estos crecen dimensionalmente.

Seleccione una:

- Verdadero
- Falso ✓

La respuesta correcta es 'Falso'

Pregunta **3**

Correcta

Se puntúa 0,50 sobre 0,50

La maldición de la dimensionalidad se refiere al conjunto de fenómenos que aparece tras el crecimiento exponencial del espacio de los datos.

Seleccione una:

- Verdadero ✓
- Falso

La respuesta correcta es 'Verdadero'

Pregunta **4**

Incorrecta

Se puntúa 0,00 sobre 0,50

True or False (Spatial DS) (Out1, Out2, Out6)

El pr-quadtree tiene como estrategia la partición del espacio, tomando como centro de dicha partición la posición del punto que se está insertando.

Seleccione una:

- Verdadero ✗
- Falso

La respuesta correcta es 'Falso'

Pregunta **5**

Correcta

Se puntúa 0,50 sobre 0,50

True or False (Spatial DS) (Out1, Out2, Out6)

El Kd-tree es una estructura de datos que me permite indizar únicamente datos en 2d y 3d (d=dimensiones).

Seleccione una:

- Verdadero
- Falso ✓

La respuesta correcta es 'Falso'

Pregunta **6**

Correcta

Se puntúa 0,50 sobre 0,50

True or False (Spatial DS) (Out1, Out2, Out6)

El árbol R+ (R+ tree) toma en consideración el área, perímetro y la sobreposición de las nuevas regiones a ser creadas durante su función de split (Node splitting)

Seleccione una:

- Verdadero
- Falso ✓

La respuesta correcta es 'Falso'

Pregunta **7**

Incorrecta

Se puntúa 0,00 sobre 0,50

True or False (Spatial DS) (Out1, Out2, Out6)

En el árbol R+ (R+ tree) el parámetro *fill-factor* me ayuda a limitar la cantidad de nodos que tendrá dicho nivel de árbol.

Seleccione una:

- Verdadero ✗
- Falso

La respuesta correcta es 'Falso'

Pregunta **8**

Correcta

Se puntúa 0,50 sobre 0,50

True or False (Spatial DS) (Out1, Out2, Out6)

La función **PickSeeds** del árbol R (R-tree) tiene como objetivo encontrar cuales son los mejores objetos/datos para dividir en dos un nodo con *overflow*.

Seleccione una:

- Verdadero ✓
- Falso

La respuesta correcta es 'Verdadero'

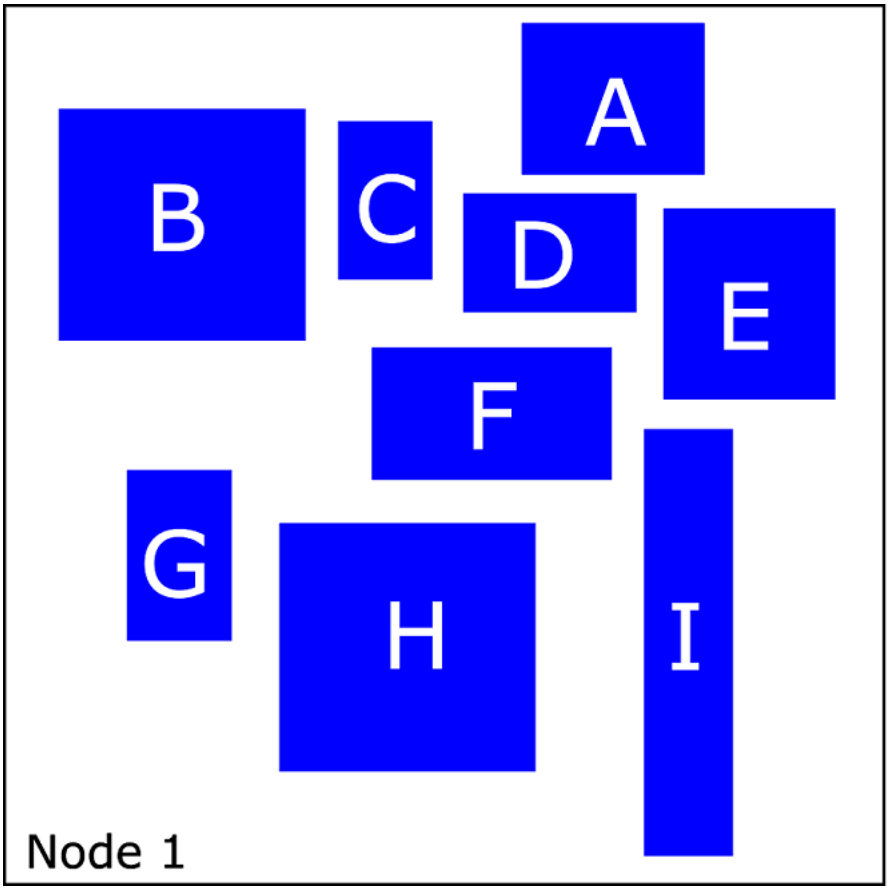
Pregunta **9**

Finalizado

Se puntúa 4,00 sobre 7,00

Theory (Out1, Out2, Out5)

Se tiene el siguiente nodo (Node 1) de un R+ Tree, conteniendo los siguientes rectangulos:



(x_{min}, y_{min}) , $(x_{max}, y_{m}$

A = (144,699) , (208,75

B = (-18,640) , (69,721

C = (79,662) , (112,71

D = (124,650), (185,69

E = (195,619), (255,68

F = (92,591) , (176,63

G = (5,534) , (42,594

H = (59,488) , (149,11

I = (188,458), (219,608

Se ha llegado a un overflow de dicho nodo, y lo que se quiere hacer ahora es un SPLIT NODE llamando al método PACK (y sus funciones internas) para solucionar dicho. Describir el **paso a paso** (esto involucra también la llamada a las funciones internas) de cual sería el resultado de dicha partición, usando un ***fill-factor*** igual a **4 y alguno de los abordajes presentados en clase para la selección de la mejor área**. Puede hacer referencia a los códigos de lineas de cada función dentro del artículo original (p.ej. PA1, SW1, P1).

NOTA: Puede usar la caja a texto a continuación para responder o usar un archivo adjunto (en formato .txt)

para hacer split node en un r´tree con un factor llenado de 4 usando el metodo pack el proceso sería el siguiente

overflow en el nodo: Al llegar a un overflow del nodo (más de 4 entradas en este caso), es necesario dividir el nodo en dos nodos.

lamada al método SplitNode: Se invoca el método SplitNode, el cual busca una partición adecuada para dividir el nodo en dos subregiones disjuntas.

uso del algoritmo Partition: Dentro del método SplitNode, se usa la rutina Partition para encontrar una partición óptima. Esta rutina divide el espacio en dos subregiones mediante una línea de corte paralela a uno de los ejes (x o y). Los criterios de selección de la mejor partición pueden ser:

vecinos más cercanos.

Desplazamiento mínimo en x e y.

Cobertura mínima del espacio total por las dos subregiones.

Número mínimo de rectángulos divididos.

Selección del eje de partición: La rutina Sweep se utiliza para barrer el espacio a lo largo de los ejes x o y hasta que se encuentren 4 rectángulos (el valor del fill-factor). A partir de ahí, se evalúan las posibles particiones basadas en los criterios mencionados.

propagación de la división hacia arriba y abajo: Si se divide un nodo intermedio, las particiones deben propagarse hacia los nodos inferiores (dividiendo recursivamente). Si el nodo es hoja, los rectángulos que se solapan con la partición se colocan en ambos nuevos nodos.

Actualización de los padres: Si el nodo dividido tiene un padre, este también debe actualizarse con los nuevos nodos creados. Si el nodo es raíz, se crea una nueva raíz con dos hijos.

ej10.cpp

Comentario:
Se ha proporcionado la secuencia de pasos de los algoritmos, sin embargo en ningún momento se muestran los valores que daría como resultado el pack en cada uno de los nuevos nodos. Para esto es que se colocaron los valores de "x" y "y" de cada rectángulo.

Pregunta **10**

Finalizado

Se puntúa 8,00 sobre 9,00

Lab (Out1, Out2, Out6)

Usar UNICAMENTE el siguiente compilador Online:

https://cpp.sh/

Se te proporciona un vector llamado rectangles donde rectangles[i] = [li, hi] indica que el rectángulo i-ésimo tiene una longitud de li y una altura de hi. La esquina inferior izquierda de cada rectángulo se encuentra en las coordenadas (0, 0) y la esquina superior derecha en (li, hi).

También se te da un vector llamado points, donde points[j] = [xj, yj] representa un punto con coordenadas (xj, yj).

Escribe la función countRectangles en C++ que calcule cuántos rectángulos contienen cada punto dado. La función debe devolver un vector c, donde c[j] es el número de rectángulos que contienen al punto j. Se dice que un rectángulo contiene un punto si el punto (xj, yj) cumple con las siguientes condiciones: 0<= xj <= li y 0 <= yj <= hi. Ten en cuenta que los puntos que se encuentran exactamente en los bordes de un rectángulo también se consideran contenidos por dicho rectángulo.

La función está definida de la siguiente manera (ver archivo adjunto):

```
vector<int> countRectangles(vector<vector<int>>& rectangles, vector<vector<int>>& points) {
    // Tu codigo va  aqui
}
```

Restricciones de los datos de prueba:

- 1 <= li, xj <= 10^9: Las longitudes y las coordenadas en el eje x pueden ser muy grandes.
- 1 <= hi, yj <= 100: Las alturas y las coordenadas en el eje y están limitadas a 100.

La función **countRectangles debe tener una complejidad menor a O(p*r)** donde p es la cantidad puntos en el vector points y r es la cantidad de elementos en el vector rectangles.

Usa el archivo adjunto para escribir tu función. Indica la complejidad de tu solución.

https://drive.google.com/file/d/1F4H2JVPlVLatPpZKixNVBeq56m5e_2KV/view?usp=sharing

Ejemplos:

Input: rectangles = [[1,2],[2,3],[2,5]], points = [[2,1],[1,4]]

Output: [2,1]

Explicación: El punto [2, 1] está contenido en los dos primeros rectángulos, por lo que el resultado para este punto es 2. El punto [1, 4] solo está contenido en el tercer rectángulo, por lo que el resultado para este punto es 1.

Input: rectangles = [[1,1],[2,2],[3,3]], points = [[1,3],[1,1]]

Output: [1,3]

Explicación: El punto [1, 3] solo está contenido en el tercer rectángulo. El punto [1, 1] está contenido en todos los rectángulos.

Entregables:

- Archivo .cpp con la implementación
- Complejidad de la solución en la caja de texto

la complejidad seria O(p*r) p= puntos, r=rectangulos

ej10.cpp

Comentario:
Complejidad incorrecta