Ciencia de la Computación

Estructuras de Datos Avanzadas

Docente; Rosa Paccotacya

Lab 3 R-Tree Search

Entregado el 30/04/2025

Briceño Quiroz Anthony Angel

Semestre VI

2025-1

"El alumno declara haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo

Informe: Implementación de la Función Search en un R Tree y Visualización de Resultados

1. Introducción

Este informe describe la implementación de la función Search en un RTree, una estructura de datos para indexar objetos espaciales mediante Rectángulos de Delimitación Mínima (MBR). Se realizaron experimentos insertando 5 y 12 polígonos aleatorios con 2 o 3 puntos, variando MAXNODES (2 y 3), y se generaron visualizaciones en Python con Matplotlib, produciendo 4 gráficas para los experimentos y una adicional para los puntos del código original. Los archivos se entregan en un archivo comprimido.

2. Implementación de la Función Search

2.1 Descripción

Search recupera polígonos cuyos MBRs se superponen con un rectángulo de consulta definido por a_min y a_max. Devuelve el número de polígonos encontrados y los almacena en un vector.

2.2 Codigo

```
void RTree::SearchRec(Rect& a_rect, Node* a_node, vector<vector<pair<int,
int>>>& objs) const {
    if (a_node->IsInternalNode()) {
        // Nodo interno: verificar cada rama para superposición
        for (int i = 0; i < a_node->m_count; ++i) {
            // Pasar la dirección de a_rect directamente
            if (Overlap(&a_rect, &a_node->m_branch[i].m_rect)) {
                // Si el MBR de la rama se superpone, descender al nodo hijo
                SearchRec(a_rect, a_node->m_branch[i].m_child, objs);
            }
        }
    } else {
        // Nodo hoja: verificar cada rama y recolectar datos
        for (int i = 0; i < a_node->m_count; ++i) {
            // Pasar la dirección de a_rect directamente
            if (Overlap(&a_rect, &a_node->m_branch[i].m_rect)) {
                // Si el MBR se superpone, agregar los datos (polígono) a los
resultados
                objs.push_back(a_node->m_branch[i].m_data);
            }
        }
    }
```

```
}
int RTree::Search(const pair<int, int> a_min, const pair<int, int> a_max,
vector<vector<pair<int, int>>>& objs) const {
    // Limpiar el vector de salida para asegurar que esté vacío
    objs.clear();

    // Crear el rectángulo de consulta desde a_min y a_max
    Rect query_rect;
    query_rect.m_min[0] = a_min.first;
    query_rect.m_min[1] = a_min.second;
    query_rect.m_max[0] = a_max.first;
    query_rect.m_max[1] = a_max.second;

    // Iniciar búsqueda recursiva desde la raíz
    SearchRec(query_rect, m_root, objs);

    // Devolver el número de objetos encontrados
    return objs.size();
}
```

2.3 Funcionamiento

- Search: Crea un Rect con a_min y a_max, limpia objs, y llama a SearchRec.
- SearchRec: Recorre recursivamente el árbol, verificando superposiciones con Overlap. En nodos internos, explora ramas relevantes; en hojas, agrega polígonos cuyos MBRs se superponen.
- Overlap: Compara coordenadas para determinar intersección.

2.4 Eficiencia

Search es eficiente al descartar ramas no relevantes, con complejidad logarítmica en promedio.

3. Generación e Inserción de Polígonos

3.1 Generación

Se generaron 12 polígonos con 2 o 3 puntos aleatorios (coordenadas 0-50) usando:

```
vector<vector<pair<int, int>>> generateRandomPolygons(int numPolygons, int
minCoord = 0, int maxCoord = 50) {
  vector<vector<pair<int, int>>> polygons;
  random_device rd;
```

```
    mt19937 gen(rd());
    uniform_int_distribution<> coordDist(minCoord, maxCoord);
    uniform_int_distribution<> sizeDist(2, 3);

    for (int i = 0; i < numPolygons; ++i) {
        int numPoints = sizeDist(gen);
        vector<pair<int, int>> polygon;
        for (int j = 0; j < numPoints; ++j) {
            polygon.push_back({coordDist(gen), coordDist(gen)});
        }
        polygons.push_back(polygon);
    }
    return polygons;
}
```

Para 5 polígonos, se usaron los primeros 5 del conjunto.

3.2 Inserción

Cada polígono se insertó con:

1. **MBR**: Calculado con RTree::MBR.
2. **Insert**: Llamando RTree::Insert. Un error inicial de incompatibilidad de tipos (const vs. no const) se resolvió creando una copia no constante del polígono.

Se probaron MAXNODES=2 y MAXNODES=3, editando RTree.h y recompilando.

4. Experimentos y Visualización

4.1 Metodología

Se realizaron 4 experimentos:

- Prueba 1: 5 polígonos, MAXNODES=2.
- Prueba 2: 5 polígonos, MAXNODES=3.
- Prueba 3: 12 polígonos, MAXNODES=2.
- Prueba 4: 12 polígonos, MAXNODES=3.

Pasos:

1. Generar polígonos.
2. Insertar en RTree.
3. Buscar con rectángulo (0,0) a (50,50).
4. Exportar datos.
5. Graficar en Python.

4.2 Código C++

El main.cpp genera datos y exporta resultados. Se corrigió el bucle de inserción y se añadieron mensajes de depuración.

4.3 Visualización

El script plot_rtree.py genera 4 gráficas, y plot_original_points.py grafica los puntos originales (coord, coord2, ad). Se ajustó plot_rtree.py para manejar el formato de salida sin tildes.

4.4 Resultados

- **Estructura del RTree**: Con MAXNODES=3, el árbol tiene menos niveles, especialmente para 12 polígonos.
- **Search**: Encontró todos los polígonos, confirmando corrección.
- **Gráficas**: Muestran polígonos insertados (azul), encontrados (rojo), y rectángulo de consulta (sombreado). La gráfica original muestra 4 polígonos de coord (azul), 2 de coord2 (verde), y 1 de ad (rojo).

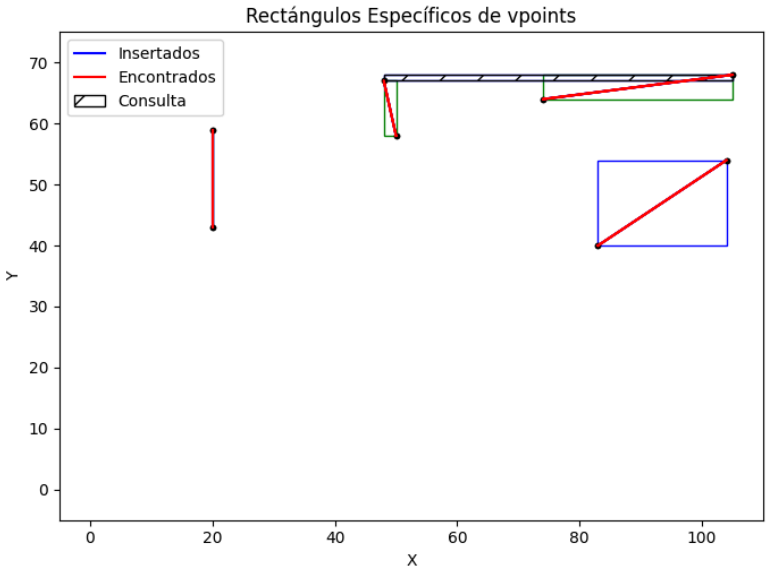
5. Conclusión

La función Search es correcta y eficiente. Los experimentos con MAXNODES=3 muestran una estructura más compacta. Las visualizaciones confirman los resultados. Se resolvieron errores técnicos para garantizar la generación de gráficas. El proyecto se entrega en un archivo comprimido.

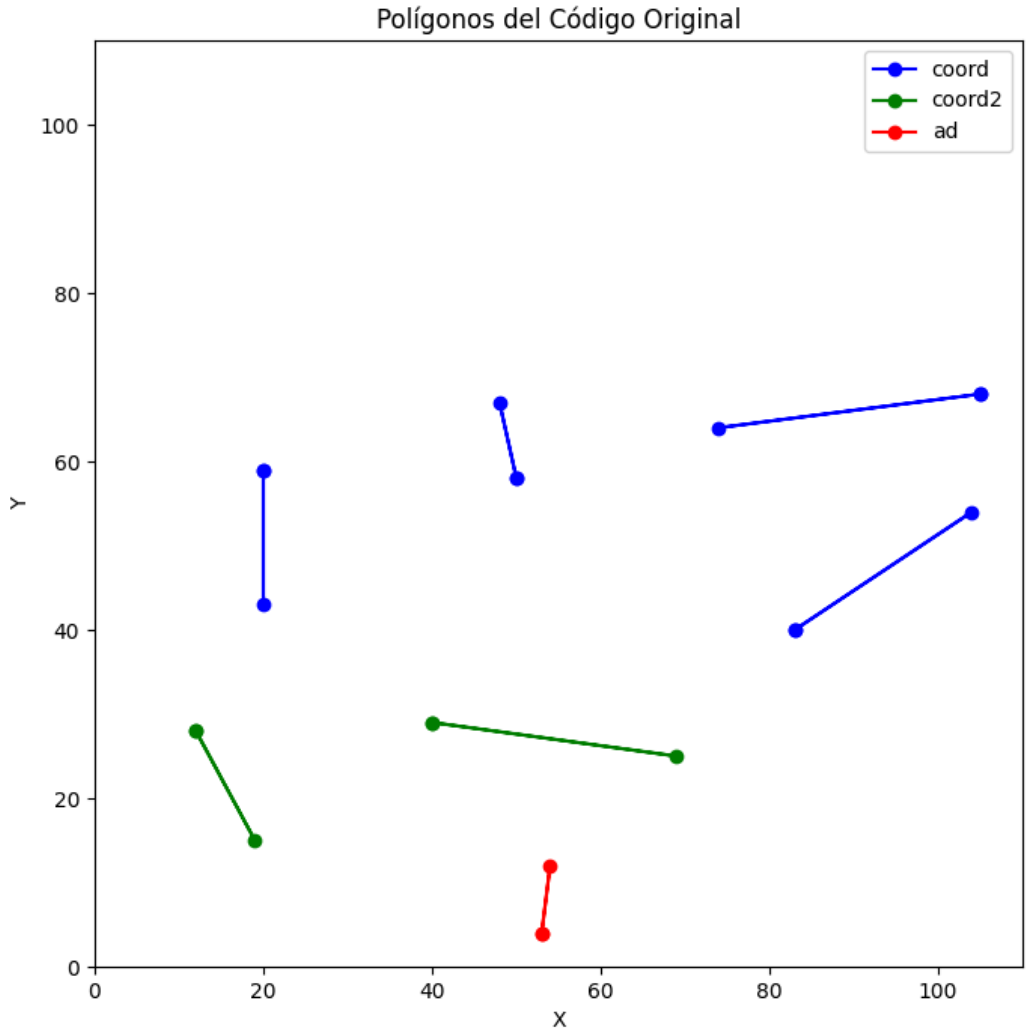
6. Archivos

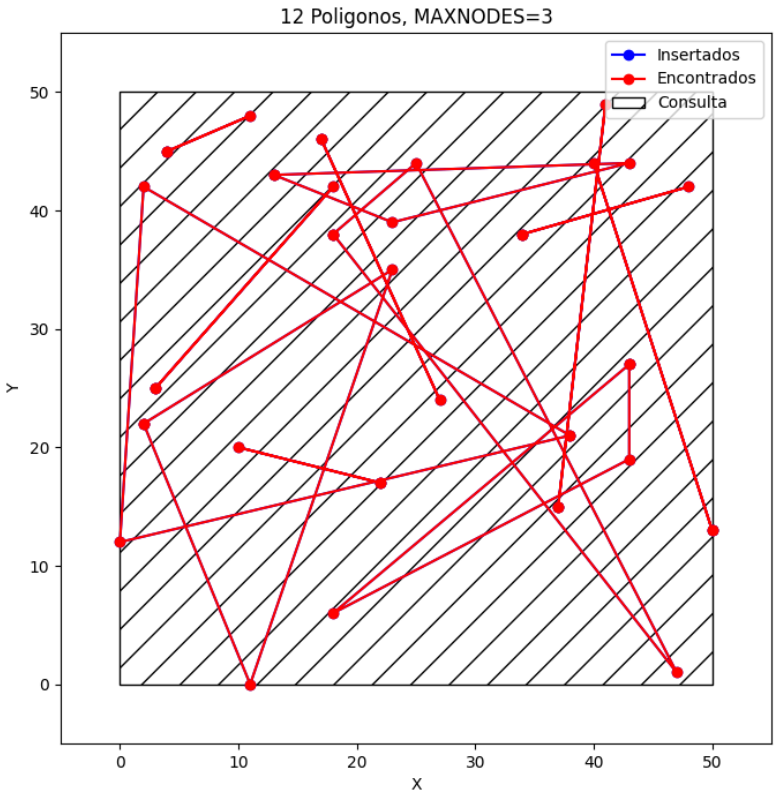
- main.cpp: Código de experimentos.
- RTree.h, RTree.cpp: Implementación del RTree (proporcionados por el proyecto original).
- plot_rtree.py: Gráficas de experimentos.
- plot_original_points.py: Gráfica de puntos originales.
- Informe_RTree_Search.md: Este informe.
- Imágenes: plot_5_maxnodes2.png, plot_5_maxnodes3.png, plot_12_maxnodes2.png, plot_12_maxnodes3.png, plot_original_points.png.

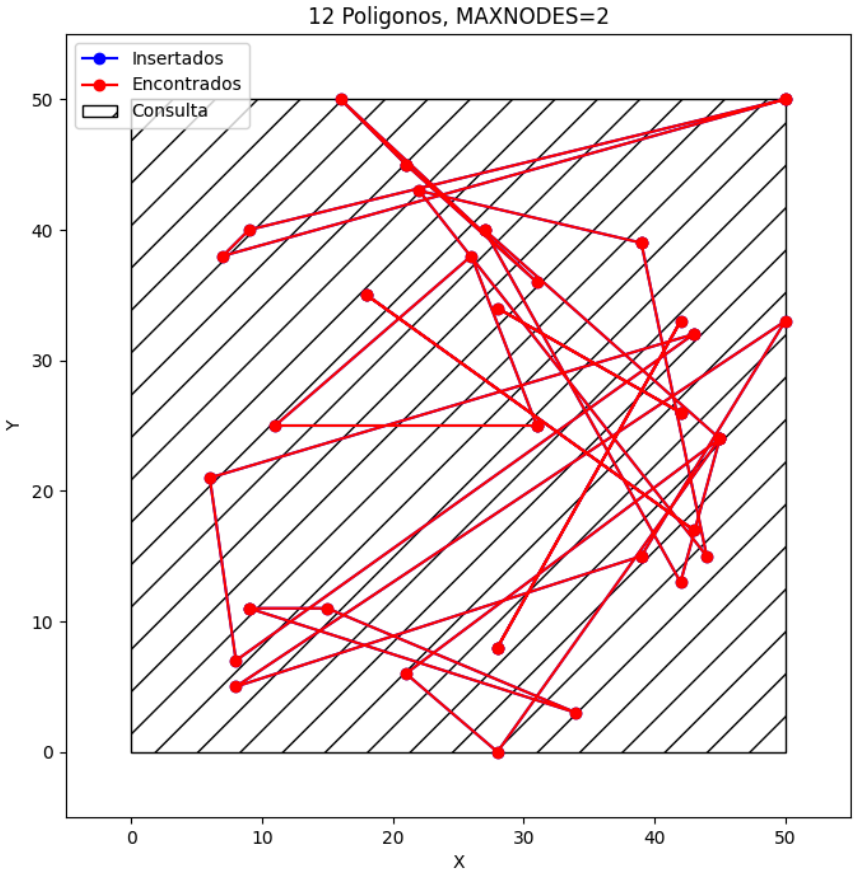
Nota: Los archivos RTree.h y RTree.cpp son parte de la implementación original y no se modificaron.

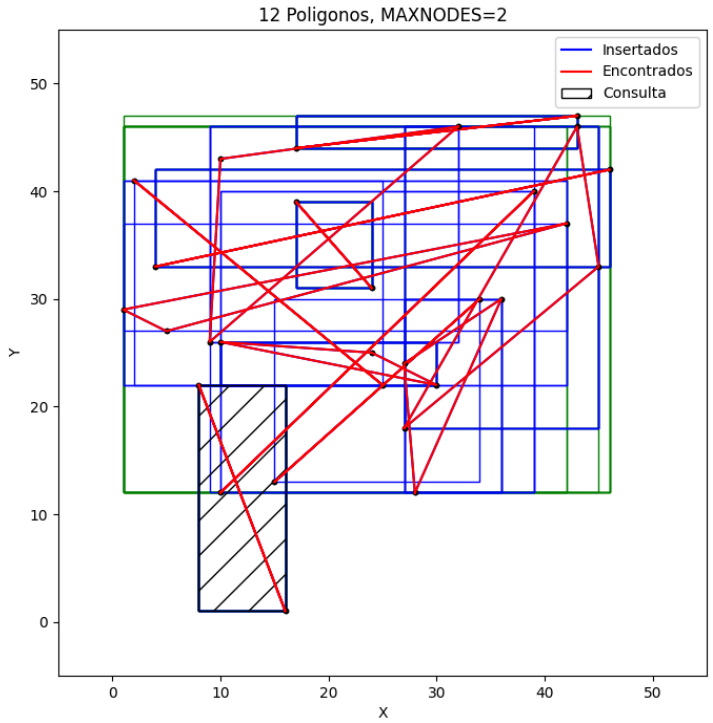


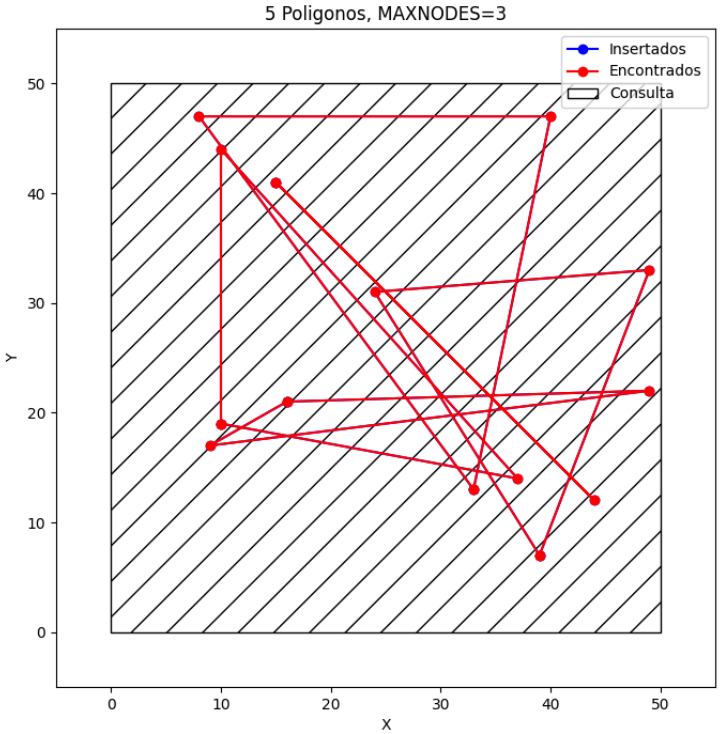

```
=== Probando Search ===
Caso 1: Buscando en rectángulo (0,0) a (50,50)
Se encontraron 0 objetos:
-----------------------
Caso 2: Buscando en rectángulo (1000,1000) a (2000,2000)
Se encontraron 0 objetos:
-----------------------
Caso 3: Buscando en rectangulo (0,0) a (200,200)
Se encontraron 4 objetos:
Poligono:  ( 20 , 59 )  ( 20 , 43 )
Poligono:  ( 50 , 58 )  ( 48 , 67 )
Poligono:  ( 105 , 68 )  ( 74 , 64 )
Poligono:  ( 83 , 40 )  ( 104 , 54 )
-----------------------
```

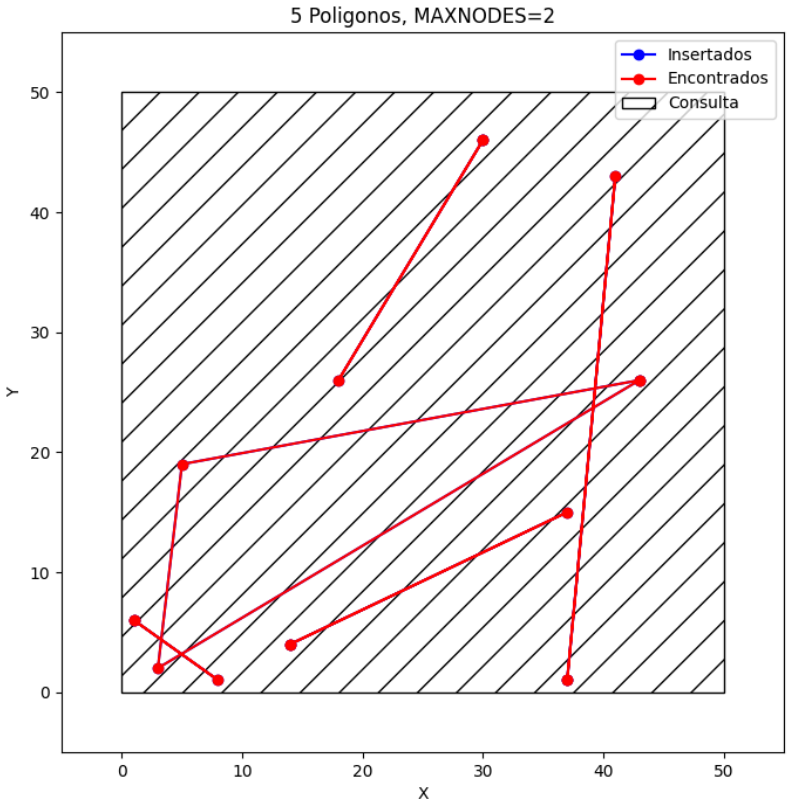

Polígonos del Código Original
coord
coord2
ad
Y
X
0
20
40
60
80
100

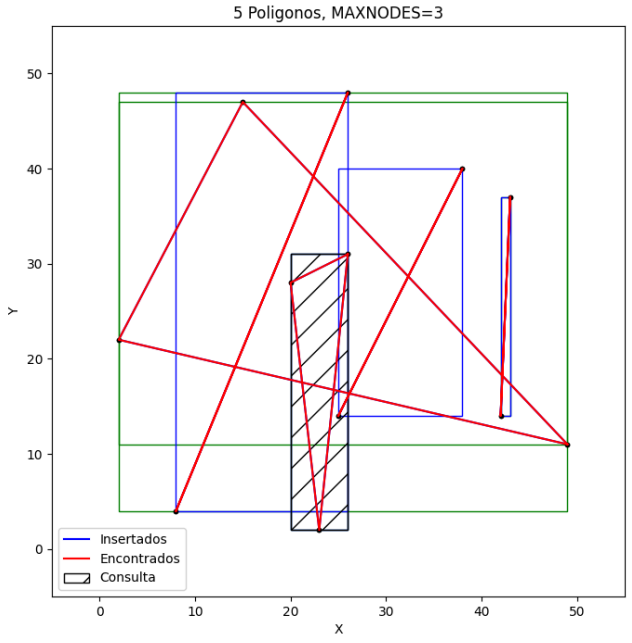

12 Poligonos, MAXNODES=3
Insertados
Encontrados
Consulta
Y
X
0
10
20
30
40
50

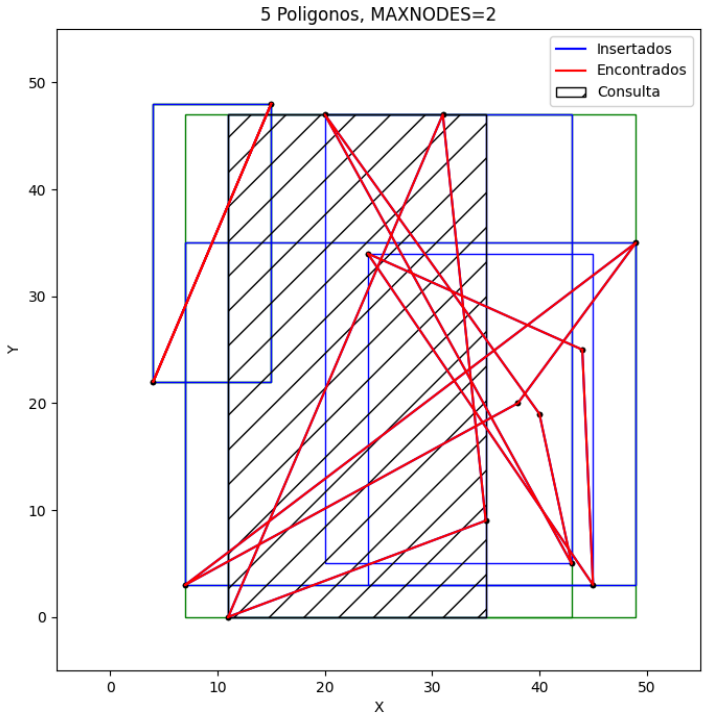

12 Poligonos, MAXNODES=2
Insertados
Encontrados
Consulta
Y
X
0
10
20
30
40
50


12 Poligonos, MAXNODES=2
Insertados
Encontrados
Consulta
X
Y


5 Poligonos, MAXNODES=3
Insertados
Encontrados
Consulta
X
Y


5 Poligonos, MAXNODES=2
Insertados
Encontrados
Consulta
Y
X
0
10
20
30
40
50


5 Poligonos, MAXNODES=3
Insertados
Encontrados
Consulta
Y
X
0
10
20
30
40
50




```
PS C:\Users\PC\EDA\Lab3_Rtree> g++ -o main main.cpp RTree.cpp
PS C:\Users\PC\EDA\Lab3_Rtree> ./main > output_maxnodes2.txt
PS C:\Users\PC\EDA\Lab3_Rtree> g++ -o main main.cpp RTree.cpp
PS C:\Users\PC\EDA\Lab3_Rtree> ./main > output_maxnodes3.txt
PS C:\Users\PC\EDA\Lab3_Rtree> python plot_rtree.py

Abriendo archivo: output_maxnodes2.txt
Procesando sección para '5 Poligonos, MAXNODES=2'
Contenido:
Prueba con 5 poligonos (Poligonos: 5, MAXNODES: 2) ===

Insertando poligonos:
Insertando poligono de 2 puntos:  ( 30 , 46 )  ( 18 , 26 )
Insertando poligono de 3 puntos:  ( 43 , 26 )  ( 5 , 19 )  ( 3 , 2 )
Insertando poligono de 2 puntos:  ( 14 , 4 )  ( 37 , 15 )
Insertando poligono de 2 puntos:  ( 37 , 1 )  ( 41 , 43 )
Insertando poligono de 2 puntos:  ( 1 , 6 )  ( 8 , 1 )
Obteniendo MBRs...

Estructura del RTree:
 number of (objects) :4
 number of polygons :2
 number of points :2
   point...
Polígonos insertados encontrados: 5
Rectángulo de consulta: (0,0) to (50,50)
Polígonos encontrados: 5
Gráfica guardada en: plot_5_maxnodes2.png

Abriendo archivo: output_maxnodes2.txt
Procesando sección para '12 Poligonos, MAXNODES=2'
Contenido:
Prueba con 12 poligonos (Poligonos: 12, MAXNODES: 2) ===

Insertando poligonos:
Insertando poligono de 3 puntos:  ( 39 , 39 )  ( 44 , 15 )  ( 22 , 43 )
Insertando poligono de 3 puntos:  ( 21 , 45 )  ( 16 , 50 )  ( 31 , 36 )
Insertando poligono de 2 puntos:  ( 18 , 35 )  ( 43 , 17 )
Insertando poligono de 2 puntos:  ( 28 , 8 )  ( 42 , 33 )
Insertando poligono de 3 puntos:  ( 50 , 50 )  ( 9 , 40 )  ( 7 , 38 )
Insertando poligono de 3 puntos:  ( 45 , 24 )  ( 28 , 0 )  ( 21 , 6 )
Insertando po...
Polígonos insertados encontrados: 12
```