Tarea 1

CC4102 - Diseño y Análisis de Algoritmos

Estudiantes: - Sebastián Burgueño Merino
- Arianne Peña Cortés
- Agustín Solís Meza
Profesor: Gonzalo Navarro
Profesor Auxiliar: Diego Salas

Fecha de entrega: 09 de Mayo de 2024

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

1.1. Presentación

En el presente informe se trata el proceso de programación y experimentación de dos métodos de construcción de un M-Tree (paper original) a partir de puntos en un plano bidimensional relacionados mediante la distancia euclidiana. Los métodos que serán usados y comparados serán: Ciaccia-Patella (paper) y Sexton-Swinbank (paper). En concreto, se busca hacer una comparación de los tiempos y recursos invertidos en los procesos de construcción y búsqueda de ambos métodos al aplicarlos a distintas cantidades de puntos, esto con el fin de determinar cuál de los dos es mejor.

1.2. Resumen

Este informe detalla el desarrollo y evaluación de dos métodos de construcción de M-Trees, el Método Ciaccia-Patella (CP) y el Método Sexton-Swinbank (SS). Se exploran los algoritmos y estructuras de datos empleados en cada método, destacando sus enfoques y diferencias fundamentales, especialmente en cómo gestionan la formación de clusters y la minimización de superposiciones. Además, se describen los datos utilizados en los experimentos, incluyendo los parámetros de configuración del M-Tree y el entorno de sistema. Los resultados de las pruebas revelan diferencias en tiempos de construcción, accesos a disco durante las búsquedas y eficacia en encontrar puntos, con gráficos que ilustran estas comparativas. Concluimos con una evaluación de los resultados frente a las hipótesis planteadas y sugerencias para futuras investigaciones que podrían mejorar la eficiencia y efectividad de estos métodos.

1.3. Hipótesis

Basado en un un primer análisis de los métodos presentados y comparando sus principales diferencias, se propone la siguiente hipótesis:

El método Ciaccia-Patella (CP), con su enfoque directo en la formación de clusters mediante la selección de samples aleatorios y su redistribución, es más efectivo para la construcción rápida y organizada de árboles M-Tree. Esto se debe a su capacidad para minimizar rápidamente la superposición entre las regiones espaciales de los nodos durante la construcción inicial del árbol, lo que resulta en una estructura más compacta y equilibrada.

Por otro lado, el método Sexton-Swinbank (SS), que enfatiza la optimización de los clusters basándose en los medoides y ajusta los pares más cercanos durante su ejecución, se espera que sea más efectivo para las operaciones de búsqueda. Esto se atribuye a su habilidad para estructurar el árbol de manera que reduce el número de nodos necesarios para examinar durante las búsquedas, debido a su enfoque en minimizar los

radios cobertores y maximizar la separación entre los nodos.

2. Desarrollo de los Algoritmos y Estructuras de Datos

2.1. Presentación de Algoritmos

2.1.1. Método Ciaccia-Patella (CP)

El **Método CP** implementa un enfoque iterativo y recursivo donde los puntos se agrupan alrededor de puntos seleccionados aleatoriamente, denominados *samples*. Comienza con la selección de estos puntos de muestra, luego asigna cada punto del conjunto al *sample* más cercano, formando grupos que se convertirán en nodos del M-tree. Incluye pasos de redistribución y balance para asegurar que todos los subgrupos mantengan un tamaño mínimo antes de aplicar el proceso de forma recursiva.

2.1.2. Método Sexton-Swinbank (SS)

El **Método SS** adopta un enfoque de clustering basado en la proximidad para organizar los puntos en clústeres. Identifica medoides y combina clusters cercanos hasta que alcanzan tamaños adecuados para las restricciones de capacidad del nodo. Los clusters resultantes se transforman en nodos del árbol, empleando divisiones estratégicas para mantener el balance.

2.2. Estructuras de Datos

Se utilizaron estructuras de datos específicas diseñadas para facilitar la construcción y manipulación eficiente del M-tree:

- **Punto**: Representa cada elemento en el espacio con coordenadas (x, y).

```
struct Punto {
    double x;
    double y;
}
```

- **Entrada**: Contiene un punto, un radio cobertor (cr), y un enlace al nodo hijo.

```
struct Entrada {
Punto p;
double cr;
struct Nodo *a;
}
```

- **Nodo**: Forma la unidad básica del M-tree, compuesto por varias entradas y puede ser un nodo hoja o interno.

```
struct Nodo {
vector<Entrada> entradas;
int B = 4096/sizeof(Entrada);
}
```

- **MTree**: La estructura principal que representa el árbol completo, iniciando desde la raíz.

```
struct MTree {
Nodo *raiz;
}
```

2.3. Diferencias entre los Algoritmos y sus Estructuras

2.3.1. Método Sexton-Swinbank (SS)

Descripción del Método: El Método SS utiliza un enfoque de clustering basado en la proximidad para organizar los puntos en clusters que posteriormente se transforman en nodos del M-tree. El proceso sigue varios pasos clave para asegurar que los nodos se formen adecuadamente y mantengan un tamaño entre los límites definidos (b y B).

Funciones Clave:

1. `Cluster()`: Esta función agrupa los puntos en clusters que respetan los límites de tamaño. Utiliza un algoritmo de agrupamiento basado en la proximidad de los puntos y la identificación de medoides para minimizar el radio cobertor.
2. `OutputHoja()`: Crea nodos hoja del M-tree a partir de los clusters finales. El medoide del clúster se establece como el punto de referencia, y el radio cobertor se calcula como la distancia máxima desde el medoide a cualquier otro punto en el cluster.
3. `OutputInterno()`: Forma nodos internos combinando múltiples nodos hoja o internos. Calcula un nuevo medoide y radio cobertor que encapsula todos los nodos contenidos.

Otras funciones: Además de de las Funciones Clave pedidas en el enunciado, se implementaron las siguientes funciones auxiliares:

1. `encontrarVecinoMasCercano()`: Identifica el cluster más cercano a un cluster dado, excluyendo el propio cluster.
2. `encontrarParesMasCercanos()`: Implementa un método de fuerza bruta para encontrar el par de clusters más cercanos entre ellos. Este luego es implementado para su versión optimizada.
3. `closestPair()`: Utiliza una estrategia de división y conquista para encontrar el par de clusters más cercano de manera más eficiente que el enfoque de fuerza bruta. Este método es especialmente útil para grandes conjuntos de datos, ya que reduce significativamente el número de comparaciones necesarias mediante la organización recursiva y el análisis de los clusters.
4. `crearCluster()`: Crea un nuevo cluster con una capacidad inicial especificada, permitiendo una gestión eficiente de la memoria.
5. `agregarPuntoACluster()`: Añade un punto a un cluster existente.
6. `combinarClusters()`: Combina dos clusters en uno nuevo, integrando todos los puntos de ambos clusters.

7. `splitMinMaxPolicy()`:Divide un cluster en dos clusters más pequeños, basándose en la política de minimización del máximo radio cobertor.

Orden de Búsqueda: La búsqueda en el M-tree construido mediante SS se inicia en la raíz y se verifica cada entrada del nodo para determinar si su área cubierta se superpone con la región de búsqueda. Esta verificación se hace usando la distancia del punto de consulta al medoide del nodo y comparándola con la suma del radio de búsqueda y el radio cobertor del nodo. Si se superponen, la búsqueda se profundiza en el nodo correspondiente, siguiendo este procedimiento de manera recursiva hasta llegar a los nodos hoja.

2.3.2. Método Ciaccia-Patella (CP)

Descripción del Método: El Método CP implementa un enfoque iterativo y recursivo para el agrupamiento de puntos, que se basa en la selección de puntos representativos o samples, asignación de puntos a estos samples, y una etapa de redistribución y balance para manejar grupos de tamaño subóptimo.

Funciones Clave:

1. `metodoCP()`: Es la función principal que orquesta el proceso de clustering y construcción del árbol. Comienza con la selección de puntos samples y asigna los demás puntos al sample más cercano, formando grupos iniciales que son ajustados y redistribuidos si es necesario.
2. `distanciaEuclidiana()`: Recibe dos puntos (p1 y p2) y devuelve la distancia entre estos.
3. `altura()`: Recibe un nodo y calcula su altura máxima de forma recursiva, recorriendo todos los hijos de sus entradas. Es parte esencial de la siguiente función.
4. `encontrarSubarboles()`: Recibe una entrada, una altura h, un vector de subárboles y un vector de entradas. Esta función busca en los hijos de la entrada todos los subárboles de altura h, guardándolos en el vector de subárboles, además, guarda los puntos raíces de dichos subárboles en el vector de entradas. Estos vectores son utilizados posteriormente para reorganizar los nodos y mantener el balance del árbol.
5. `calcularDistanciaMaxima()`: Recibe una entrada y calcula la máxima distancia euclidiana que hay entre sí misma y cualquier punto de su subárbol. Lo anterior es, en esencia el radio cobertor de esta entrada.
6. `actualizarCR()`: Recibe un nodo n y recorre recursivamente todas sus entradas y subárboles actualizando el radio cobertor de todos los puntos encontrados usando la función anterior.

Orden de Búsqueda: La búsqueda en el árbol M-tree construido por el método CP comienza en la raíz y evalúa cada nodo de manera similar al método SS. La diferencia principal radica en cómo se han construido los nodos: el método CP puede llevar a una estructura de árbol donde los nodos están más optimizados en términos de cobertura espacial debido al proceso iterativo de ajuste y balanceo durante la construcción, lo que potencialmente reduce el número de nodos necesarios para verificar durante una búsqueda.

3. Especificaciones de los datos

Esta sección tiene como objetivo proporcionar una base para la comprensión de los experimentos realizados, permitiendo una evaluación precisa de la efectividad y eficiencia del M-Tree.

A continuación, detallamos las características de los datos de prueba, los parámetros de configuración del árbol, y las condiciones bajo las cuales se realizaron los experimentos. Además, describimos el entorno del sistema donde se ejecutaron las pruebas, incluyendo detalles sobre el sistema operativo, la memoria RAM y la caché, los cuales pueden influir significativamente en el rendimiento de las operaciones de búsqueda y almacenamiento.

Datos Utilizados

- **Tipo de Datos:** Puntos en un espacio bidimensional.
- **Distribución:** Uniformemente distribuidos entre 0.0 y 1.0.
- **Cantidad de Datos:** Pruebas realizadas con tamaños de $n = 2^{10}$ a $n = 2^{25}$.

Parámetros del M-Tree

- **Tamaño de Bloque B:** El tamaño del bloque, denotado como B, se define basado en la capacidad máxima de entradas que un nodo del árbol puede contener, siendo $B = \frac{4096}{\text{sizeof(Entrada)}}$.
- **Tamaño Mínimo de Bloque b:** La mitad del tamaño del bloque, $\frac{B}{2}$, define la capacidad mínima de entradas en un nodo antes de que ocurra una división.

Configuración del Sistema

Para la ejecución de los experimentos relacionados con la implementación y evaluación del M-Tree, se utilizó un dispositivo que cuenta con las siguientes especificaciones:

- **Sistema Operativo:** Windows 11, 64-bit.
- **RAM:** 24 GB.
- **Procesador:** 12th Gen Intel(R) Core(TM) i5-12450H.
- **Tamaño de Caché:**
  - **L1 Cache:** 704 KB.

- **L2 Cache:** 7.0 MB.
- **L3 Cache:** 12.0 MB.

Metodología de Pruebas

- **Repetición de Pruebas:** Cada configuración de experimento se realizó con 100 consultas diferentes para evaluar la variabilidad del rendimiento.
- **Consulta:** Se realizan consultas con un radio de 0.02, buscando puntos dentro de este rango desde un punto de consulta aleatorio.
- **Método de Construcción y Consulta:** Utilización de los métodos SS (Sexton-Swinbank) y CP (Ciaccia-Patella) para la construcción y evaluación de consultas en el árbol.

4. Resultados y Conclusiones

4.1. Resultados Método Ciaccia-Patella (CP)

$n = 2^i$	Tiempo de Construcción (s)	Tiempo Total de Consultas (s)	Media de Accesos	Desviación Estándar de Accesos	Varianza de Accesos
10	0.20979	0.0007072	8.88	1.35115	1.8256
11	0.414385	0.0004225	9.77	3.09469	9.5771
12	0.762475	0.00118	14.79	2.88546	8.3259
13	1.5749	0.0014155	22.39	3.87271	14.9979
14	3.39023	0.0007426	24.05	3.94557	15.5675
15	6.34414	0.0012115	56.14	8.88146	78.8804
16	12.3212	0.0020685	95.83	10.7815	116.241
17	27.0499	0.0073063	158	20.6252	425.4
18	113.671	0.0170727	319.38	45.1213	2035.94
19	160.038	0.0109942	538.57	81.3393	6616.09
20	248.156	0.0158925	936.06	148.791	22138.7
21	556.035	0.0314107	2084.92	333.294	111085
22	1178.63	0.062372	4751.5	699.859	489802
23	2379.17	0.207346	9468.25	1416.69	2.00702e+06

Tabla 1: Resultados del método Ciaccia-Patella (CP) para diferentes tamaños de $n = 2^i$: Tiempos y Accesos

$n = 2^i$	Media de Puntos Encontrados	Desviación Estándar de Puntos Encontrados	Varianza de Puntos Encontrados
10	1.08	1.16344	1.3536
11	2.58	1.42955	2.0436
12	5.32	2.28858	5.2376
13	10.31	3.59359	12.9139
14	20.22	4.61211	21.2716
15	40.64	6.7194	45.1504
16	80.84	11.2033	125.514
17	163.02	17.5767	308.94
18	320.54	38.3829	1473.25
19	638.87	62.7934	3943.01
20	1297.76	94.0214	8840.02
21	2588.72	183.224	33571.1
22	5244.08	251.666	63335.9
23	10449.1	440.822	194324

Tabla 2: Resultados del método Ciaccia-Patella (CP) para diferentes tamaños de n: Puntos Encontrados y Varianzas

n (2î)	Intervalo de Confianza de Accesos	Intervalo de Confianza de Puntos Encontrados
10	[8.61518, 9.14482]	[0.851965, 1.30803]
11	[9.16344, 10.3766]	[2.29981, 2.86019]
12	[14.2244, 15.3556]	[4.87144, 5.76856]
13	[21.6309, 23.1491]	[9.60566, 11.0143]
14	[23.2767, 24.8233]	[19.316, 21.124]
15	[54.3992, 57.8808]	[39.323, 41.957]
16	[93.7168, 97.9432]	[78.6441, 83.0359]
17	[153.957, 162.043]	[159.575, 166.465]
18	[310.536, 328.224]	[313.017, 328.063]
19	[522.627, 554.513]	[626.562, 651.178]
20	[906.897, 965.223]	[1279.33, 1316.19]
21	[2019.59, 2150.25]	[2552.81, 2624.63]
22	[4614.33, 4888.67]	[5194.75, 5293.41]
23	[9190.58, 9745.92]	[10362.7, 10535.6]

Tabla 3: Intervalos de Confianza del método Ciaccia-Patella (CP) para diferentes tamaños de $n = 2^i$

4.2. Resultados Método Sexton-Swinbank (SS)

$n = 2^i$	Tiempo de Construcción (s)	Tiempo Total de Consultas (s)	Media de Accesos	Desviación Estándar de Accesos	Varianza de Accesos
10	1.06885	0.0009667	46.61	14.6328	214.118
11	2.61987	0.0014809	28.43	4.18152	17.4851
12	10.9281	0.0033703	65.81	5.93413	35.2139
13	100.996	0.0092149	106.59	8.75454	76.6419
14	320.189	0.0231093	127.67	35.7556	1278.46
15	1727.53	0.0201782	432.44	135.129	18259.8
16	7256.31	0.0378432	714.94	198.685	39475.7
17	16916.5	0.0821552	1651.54	338.768	114764

Tabla 4: Resultados del método Sexton-Swinbank (SS) para diferentes tamaños de n: Tiempos y Accesos

n (2î)	Media de Puntos Encontrados	Desviación Estándar de Puntos Encontrados	Varianza de Puntos Encontrados
10	1.27	1.00851	1.0171
11	2.18	1.62099	2.6276
12	5.34	2.42165	5.8644
13	10.38	3.70076	13.6956
14	11.39	4.56923	20.8779
15	32.83	11.4578	131.281
16	68.55	16.6008	275.588
17	139.58	26.9856	728.224

Tabla 5: Resultados del método Sexton-Swinbank (SS) para diferentes tamaños de n: Puntos Encontrados y Varianzas

n (2î)	Intervalo de Confianza de Accesos	Intervalo de Confianza de Puntos Encontrados
10	[43.742, 49.478]	[1.07233, 1.46767]
11	[27.6104, 29.2496]	[1.86229, 2.49771]
12	[64.6469, 66.9731]	[4.86536, 5.81464]
13	[104.874, 108.306]	[9.65465, 11.1053]
14	[120.662, 134.678]	[10.4944, 12.2856]
15	[405.955, 458.925]	[30.5843, 35.0757]
16	[675.998, 753.882]	[65.2962, 71.8038]
17	[1585.14, 1717.94]	[134.291, 144.869]

Tabla 6: Intervalos de Confianza del método Sexton-Swinbank (SS) para diferentes tamaños de n

4.3. Gráficos De Comparación

A partir de los resultados, se procedió la elaboración de de una serie de gráficos para visualizar y comparar de manera efectiva los diferentes métodos estudiados.

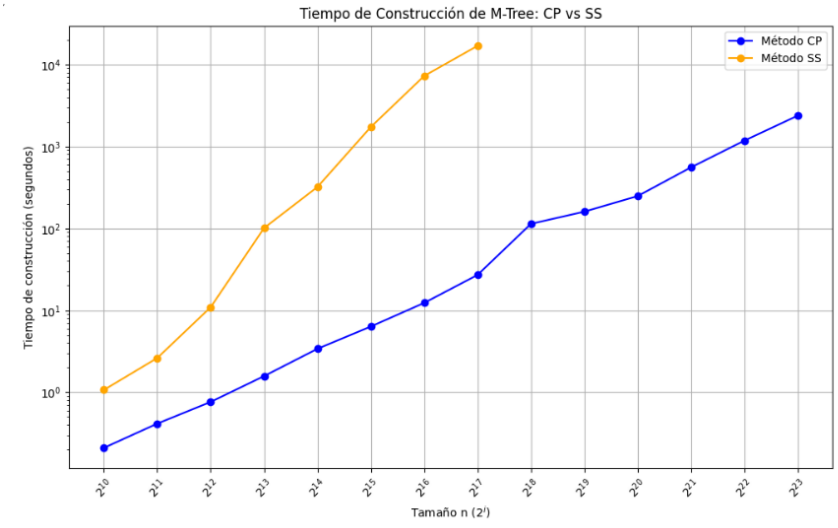

Figura 1: Tiempo de Construcción del M-Tree.

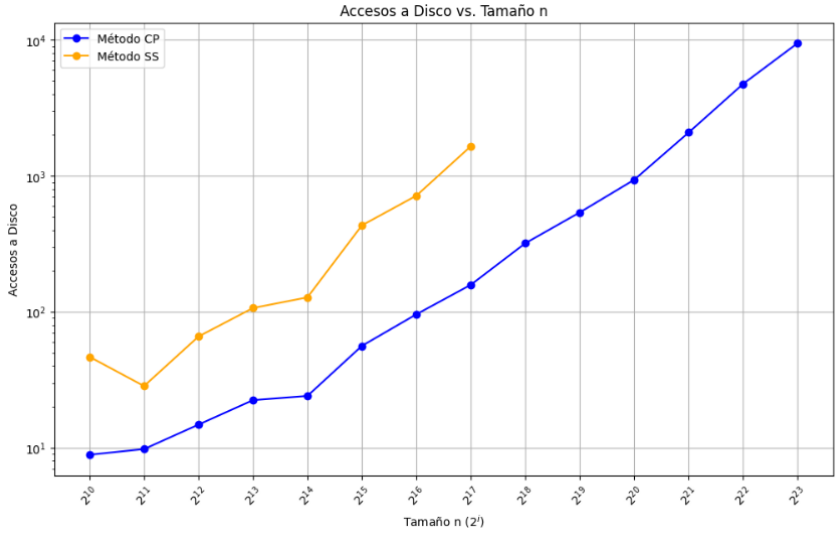

Figura 2: Accesos al Disco vs. Número de Puntos.

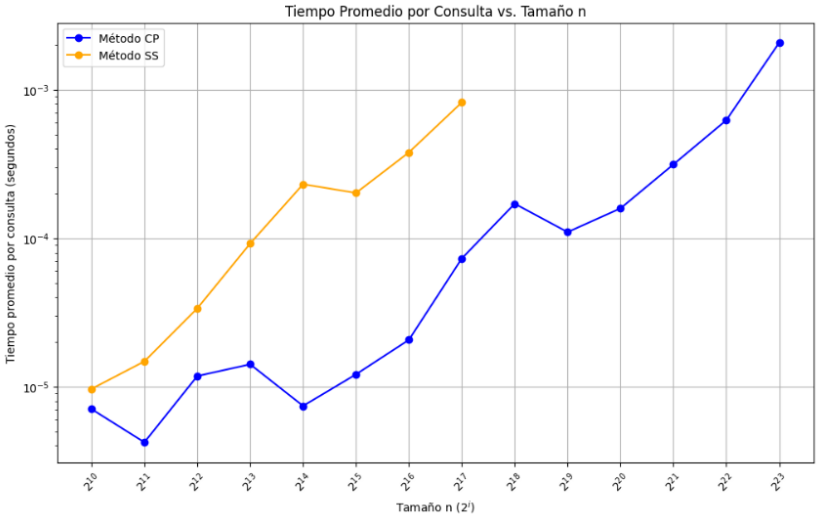

Figura 3: Tiempo Promedio de Consulta vs. Número de Puntos.

4.4. Observaciones Generales

- **Tiempo de Construcción:** El tiempo necesario para construir el M-Tree aumenta significativamente con el tamaño de n para ambos métodos. Sin embargo, el método SS muestra consistentemente tiempos de construcción más altos que CP, especialmente a medida que n aumenta.
- **Tiempo Promedio por Consulta:** El método SS también tiende a tener un tiempo promedio por consulta más alto que el método CP, lo cual sugiere que CP maneja las consultas más eficientemente, posiblemente debido a una estructura de árbol más optimizada con menor superposición de nodos.
- **Accesos a Disco:** Se observa un incremento en el número de accesos a disco con el aumento de n en ambos métodos. El método CP muestra un incremento más gradual en comparación con SS, indicando que sus nodos pueden estar mejor organizados para evitar lecturas innecesarias.

Inferencias y Comentarios

- **Eficacia de los Métodos:** El método Ciaccia-Patella parece más eficiente tanto en la construcción del árbol como en el manejo de consultas, lo que puede atribuirse a su estrategia de clustering y redistribución que minimiza la superposición de nodos y, por ende, reduce el número de accesos a disco necesarios durante las consultas.
- **Costo Computacional:** A pesar de su eficiencia en tiempo de consulta y accesos a disco, el tiempo de construcción sigue siendo un factor importante. Se considera que el tiempo adicional que requiere el método Sexton-Swinbank podría ser un detrimento.
- **Aplicabilidad:** La elección entre Ciaccia-Patella y Sexton-Swinbank podría depender del contexto específico de uso. Por ejemplo, si el tiempo de construcción no es una limitante, pero se prioriza el rendimiento de consulta, Ciaccia-Patella sería preferible. Si las reconstrucciones son infrecuentes y se pueden tolerar tiempos de construcción más largos, Sexton-Swinbank podría considerarse para ciertas aplicaciones dada su robustez en la estructuración de datos más complejos.

En general, el método CP se muestra superior en términos de eficiencia operativa durante las consultas y en la construcción de estructuras menos costosas en términos de acceso a disco. Esto es debido a que el Orden de Complejidad del algoritmo CP es mucho menor en comparación al algoritmo SS.

5. Recapitulación

Recapitulando, el proceso en el que se basó este informe fue la programación de dos métodos de construcción de un M-Tree: Ciaccia-Patella y Sexton-Swinbank. Ambos métodos fueron probados con distintas cantidades de datos, midiendo el tiempo que tomaron en la construcción de la estructura y la búsqueda de puntos en esta. Posteriormente, se recopiló información de la ejecución, en cuanto a tiempo y recursos, y se llevó a cabo un análisis de estos. Finalmente, se concluye que el método CP es superior tanto en los procesos de construcción y búsqueda.

5.1. Hipótesis planteada

Tras un análisis más detallado de los métodos presentados, se llega a la conclusión de que la hipótesis inicial no parece ser adecuada: A pesar de que se planteó que el método Ciaccia-Patella sería más efectivo en la construcción de árboles M-Tree y que el método Sexton-Swinbank sobresaldría en las operaciones de búsqueda, las observaciones indican que el método Ciaccia-Patella podría ser superior en ambos campos. Se debe mencionar que lo anterior puede ser causado porque las soluciones encontradas no son óptimas debido a complicaciones de tiempo y falta de datos que serán detallados más adelante.

5.2. Mejoras a futuro

Se plantean como mejoras a futuro las siguientes:

- Crear un caché capaz de guardar la distancia entre puntos para evitar calcularla cada vez que se necesita.
- Paralelizar funciones como `encontrarParesMasCercanos()` o funciones que busquen la menor/mayor distancia entre elementos, con el fin de aprovechar la capacidad multicore del procesador y acelerar el proceso.

Mencionar que, aún así, debido al conocimiento de la ineficiencia del algoritmo SS sin optimizaciones, debido al análisis inicial del Orden de Complejidad, se optó por no ejecutar el código por más de 20 horas aproximadamente.

5.3. Dificultades encontradas

Primeramente, se debe mencionar como dificultad la inexperiencia tanto con la estructura de datos M-Tree y los métodos CP y SS, lo que provocó que se debiese dedicar una gran cantidad de tiempo en lectura y comprensión de los papers asociados.

Específicamente para el método de Ciaccia-Patella, un factor que influyó en el proceso fue la falta de experiencia con las estructuras ofrecidas por C++, por lo que su implementación fue cambiada muchas veces durante el desarrollo, llegando finalmente a la conclusión de que `vector` y `map` eran las más adecuadas de forma tardía y teniendo poco margen de tiempo para optimizarlo o analizar más detalladamente su correctitud y la precisión de los resultados entregados.

Finalmente, un factor que influyó fuertemente en el tiempo de desarrollo fue que, en primera instancia y durante la primera semana de programación, se decidió usar C como lenguaje para la implementación de los métodos, debido a . Lo anterior provocó que se perdiera tiempo importante de trabajo y, en consecuencia, no se pudieran conseguir todos los resultados necesarios.

5.4. Datos Faltantes

Debido a las dificultades anteriormente mencionadas, el presente informe no contiene datos de la experimentación de Ciaccia-Patella para $n = 24$ y $n = 25$ y para Sexton-Swinbank en $n = [18, 25]$.

5.5. Extensión

Se concluye que el presente informe podría ser extendido mejorando la eficiencia del método Sexton-Swinbank y, además, dándole más tiempo a ambos métodos para su ejecución, hasta obtener la totalidad de resultados.