Proyecto 1

Estructuras de Datos, Primer Semestre 2019

Prof. Diego Seco

Ayudantes: Diego Gatica y Alexander Irribarra

1. Introducción

Los Quadtrees son una estructura de datos jerárquica que se basan en una descomposición recursiva del espacio. Existen numerosas variantes de esta idea, cada una con beneficios en determinados dominios. En este proyecto nos centraremos en una versión de la variante conocida como *point region quadtree* (*PR-quadtree*) para puntos en dos dimensiones. Esta estructura de datos se puede describir como un árbol en el que cada nodo tiene como máximo cuatro hijos. El nodo raíz representa todo el espacio bidimensional y cada uno de sus hijos representa una subregión de igual tamaño, resultante de realizar la bisección en cada dimensión. Por claridad, nombraremos a los cuatro hijos en orden NW (noroeste), SW (suroeste), SE (sureste) y NE (noreste). El espacio se subdivide hasta que toda región contiene un único punto. En la figura 1 se puede ver un ejemplo.

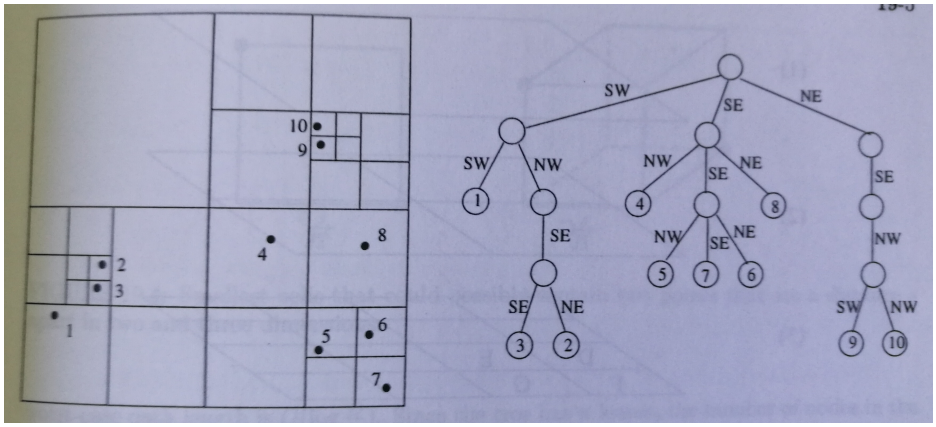

Figura 1: Un conjunto de puntos en dos dimensiones y el correspondiente PR-quadtree

Existen muchas aplicaciones de esta estructura de datos, desde sistemas de información geográfica hasta juegos donde los puntos puede representar posiciones de los personajes en el mundo virtual.

Una alternativa popular es el kd-tree. A diferencia del Quadtree, cuyo particionamiento está guiado por el espacio, en el kd-tree el particionamiento está guiado por los objetos que existen en dicho espacio. Una particularidad de este particionamiento es que va alternando el eje en el cual se realiza; es decir, comienza particionando en una dimensión (ej. eje X), en el segundo nivel particiona en la otra (ej. eje Y), y luego nuevamente en la inicial. Por tanto, la construcción del kd-tree se puede definir fácilmente de manera recursiva mediante un algoritmo que en cada llamada va particionando el conjunto de puntos a la mitad, alternando

el eje de partición. En cada llamada, la mediana del conjunto se almacena en el nodo actual. En la figura 1 se puede ver un ejemplo.

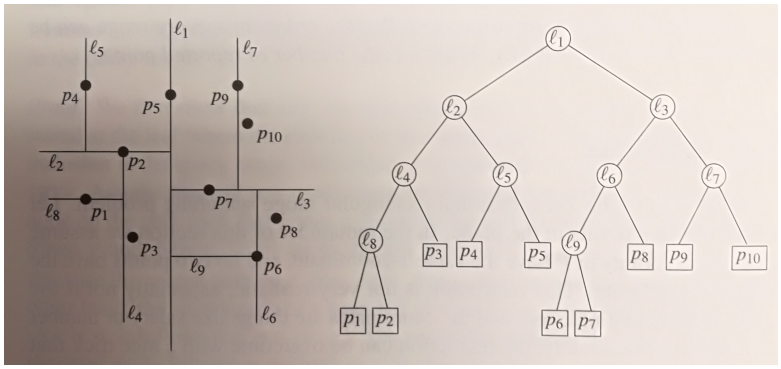

Figura 2: Un conjunto de puntos en dos dimensiones y el correspondiente kd-tree

Por motivos de eficiencia, para la construcción no se deben reordenar los conjuntos después de cada llamada sino que, en su lugar, antes de comenzar la construcción se realiza un preproceso que ordena los puntos tanto por coordenada x como por coordenada y (es decir, se generan dos secuencias ordenadas, cada una de ellas de acuerdo a una dimensión). Dichas secuencias se van dividiendo en cada llamada.

Ambas estructuras de datos soportan varios tipos de consultas. En este proyecto nos restringiremos únicamente a consultas de rango las cuales, dado un rectángulo de consulta definido por la esquina superior izquierda y la esquina inferior derecha, retorna todos los puntos contenidos en dicha consulta. La idea general de los algoritmos de consulta es bajar en el árbol, considerando únicamente aquellas ramas que representan un sub-espacio que interseca con la consulta.

2. Input

Los puntos de entrada del espacio bidimensional se proporcionarán en un archivo (input.txt) con el siguiente formato:

- La primera línea contiene dos números enteros N y M describiendo el tamaño del espacio bidimensional (N: número de filas, M: número de columnas).
- La segunda línea contiene el número de puntos en el espacio P.
- A continuación, P líneas, cada una de ellas conteniendo dos números enteros f_i y c_i describiendo la posición del i-ésimo punto (f_i, c_i) en formato (fila, columna).

3. Preguntas

1. [1 punto] Diseñe el ADT correspondiente al PR-quadtree, el cual debe incluir operaciones para *construir* y *buscar*. La construcción es estática, es decir, se asume que todos los puntos son conocidos al momento de construir la estructura. Buscar realiza una búsqueda de rango que recibe como parámetro el rectángulo de consulta. El informe debe incluir la descripción del ADT, incluyendo el pseudocódigo de los algoritmos.
2. [1 punto] Diseñe el ADT correspondiente al kd-tree, el cual debe incluir operaciones para *construir* y *buscar*. La construcción es estática, es decir, se asume que todos los puntos son conocidos al mo-

mento de construir la estructura. Buscar realiza una búsqueda de rango que recibe como parámetro el rectángulo de consulta. El informe debe incluir la descripción del ADT, incluyendo el pseudocódigo de los algoritmos.

3. [2 puntos] Implemente lo diseñado en el punto anterior e incluya un archivo prueba.cpp que permita verificar el correcto funcionamiento del código.

4. [1 punto] Compare experimentalmente la eficiencia de la operación de búsqueda de ambas implementaciones.

5. [1 punto] La eficiencia de la operación de búsqueda en este ADT es sensible a la distribución de los datos. Es decir, a igual tamaño de espacio y número de puntos, la búsqueda no toma el mismo tiempo si todos los datos están agrupados en una sub-región del espacio, que si están distribuidos uniformemente. Plantee al menos una hipótesis sobre la eficiencia de la búsqueda y valídela experimentalmente. Pista: puede utilizar distribuciones de puntos conocidas como uniforme, Gauss, Zipf, etc.

4. Evaluación

El proyecto se realizará en parejas. Enviar en piazza (mensaje privado a *Instructors*) lo siguiente:

1. Un informe que:
   a) Incluya portada y respuestas a todas las preguntas de la tarea.
   b) Sea claro y esté bien escrito. Un informe difícil de entender es un informe que será mal evaluado aunque todo esté bien implementado. La persona que revise el documento debe poder entender su solución sólo mirando el informe.
   c) Esté en formato pdf.

2. Un archivo comprimido con todos los ficheros fuente implementados para solucionar la tarea. El informe debe hacer referencia a ellos y explicar en qué consiste cada uno.

Fecha de Entrega: viernes 24 de mayo 11:59PM