BSP Tree

Salomé Viana y Daniel Navarrete
Repositorio: https://github.com/salomeviana/Proyecto-Algoritmos

25 de noviembre de 2019

1. Resumen ejecutivo

.

1.1. Descripción final del problema

El objetivo de este proyecto es encontrar una serie de puntos cuyas coordenadas x y y se encuentran en un intervalo definido. Quiere decir lo anterior, que dados unos puntos iniciales y una región rectangular, se deben encontrar los puntos que están dentro en esta de la manera más rápida posible, ya que la solución iterativa de comparación punto por punto es lineal sobre la cantidad de elementos (puntos) sobre la cual se quiere hacer la busqueda. Por este motivo, decidimos estudiar una nueva manera de obtener dichos puntos de una forma más rápida.

1.2. Solución

Teniendo en cuenta este problema, para acotar los puntos se realizó una bipartición binaria del espacio, este método divide la región determinada en dos, en el caso de este proyecto por mitades, de manera recursiva hasta llegar a acotar el punto con una región rectangular de una precisión especifica. Así pues, para dar cuenta de como se realizó la bipartición se implementó una estructura de datos tipo árbol binario: BSP Tree, que almacena la información sobre la partición realizada de tal manera que cada uno de los nodos represente cada una de las particiones hechas. De modo que cuando una región se encuentre en solo un lado de la partición, se obtiene una reducción considerada en la cantidad de puntos que tiene para evaluar.

1.3. Trabajo desarrollado

Se implementó la estructura mencionada anteriormente: BSP Tree con los métodos de inserción y borrado, además de la altura y el tamaño. Támbien se creó el algoritmo de particionamiento del espacio. Por otro lado, se creó un algoritmo para la busqueda de un punto,

y por último el algoritmo que recorre el árbol en busqueda de los puntos que están dentro de una región especificada por el usuario.

2. Funcionalidad de la herramienta

Esta herramienta, como bien se dijo anteriormente, tiene como fin reducir el tiempo de busqueda de puntos que esten dentro en una región específica. Es por esto, que esta se basa en un árbol de particiones, las cuales indican cuales puntos están en esa región y los intervalos que forman dicha partición. Dando así informacíon de vital importancia para que el tiempo de ejecución del algoritmo de busqueda sea menor, pues cada nodo indica hacia que partición o particiones debe seguir buscando para encontrar los puntos deseados.

3. Descripción de la herramienta

La herramienta está compuesta por cinco estructuras (Point, Rectangle, Region, BSPNode, Intervalo) y la clase BSP que utiliza las estructuras enunciadas anteriormente. A continuación serán descritas cada una de las estructuras:

- Point: Está compuesta por la coordenadas x y y que conforman un punto en R^2. Además contiene la sobrecarga de operadores de comparación igualdad ($==$) y de salida de datos ($<<$) y constructor parametrizado y por defecto.
- Region: Está compuesta por un punto de referencia, tipo Point, que corresponde al punto inferior izquierdo del rectángulo que constituye la región, la longitud de este en x, y la longitud en y. Contiene tambien constructor parametrizado y por defecto.
- Interval: Conformado por un valor mínimo y máximo formando así un intervalo cerrado cuyos valores extremos son min y max. Tiene constructor parametrizado y por defecto. Además tiene la sobrecarga del operador de salida de datos ($<<$). Rectangle: Formado por dos intervalos cuyo producto cruz representara el rectángulo.
- BSPNode: Contiene una región tipo Region, un vector de puntos tipo Point que contiene los puntos que se encuentran dentro de la región y los correspondientes punteros a su padre y a sus hijos.

Así pues, la clase BSP utiliza datos de tipo Point, BSPNode, Region y demás estructuras enunciadas anteriormente. La clase BSP está compuesta por un puntero tipo BSPNode a la raíz del árbol de la cual se desprenden los demás nodos, y una precisión que indicará el tamaño del rectángulo que acota el punto. Además de esto, tiene los siguientes métodos:

En cuanto a la construcción del árbol, este cuenta cos dos tipos de constructores:

1. Constructor por defecto: que inicializa la raíz en nullptr y establece una precsión por defecto de 0.5.

2. Constructor parametrizado: Recibe una región, un vector que contiene los puntos presentes en dicha región y la precisión con la que el usuario quiere que sean acotados los puntos.

- Métodos Privados:

    1. *partitionx*: Particiona el espacio en x, es decir, que particiona según la coordenada en x del punto de referencia y la longitud en x de la región representada por el nodo.
    2. *partition y*: Similar a *partition x* pero tomando las coordenadas en y.
    3. *destroy recursive*: Destruye los nodos desde la raíz hasta las hojas de forma recursiva.
    4. *def points x*: Define los puntos que van a estar en cada una de las particiones cuando se particiona sobre el eje x.
    5. *def points y*: Similar a *def points x*, pero cuando se realiza la partición sobre el eje y.

- Métodos Públicos:

    1. *height*: Determina el camino más largo desde la raíz hasta alguna de sus hojas.
    2. *size*: Cuenta la cantidad de nodos que hay en el árbol.
    3. *remove point*: Remueve un punto del árbol.
    4. *insert point*: Inserta un punto en el árbol.
    5. *scan region*: Retorna los puntos que están en dicha región.
    6. *find point region*: Encuentra la región mínima en la que se encuentra el punto buscado.
    7. *display*: imprime el árbol de forma *inorder traversal* con la altura de cada nodo.

Además de las estructuras y la clase BSP, la herramienta contiene una función generadora de puntos aleatorios con una distribución normal y uniforme.

4. Algoritmos y estructuras de datos

Para este proyecto la estructura de datos más importante que se implementó fue el árbol binario BSP que resulta ser muy útil para biparticionar espacios, pues como se dijo anteriormente, cada uno de sus nodos representa una subregión generada luego de realizar una partición. Para el desarrollo de esta se utilizaron los algoritmos de particionamiento (*parititon x* y *partition y*) que defininen las regiones de cada uno de los nodos y los puntos que están contenidos en estas utilizando los métodos privados *def x points* y *def y points*.

Además, los algoritmos más importantes de este proyecto fueron *scanregion*, y find point region, pues el primero es en si la finalidad del proyecto ya que es el que determina cuales

puntos están en la región especificada de una forma más rápida que la lineal. Ahora bien, el *find point region* tambien hace parte de los más importante pues ayuda a determinar la región en la que está el punto buscado, la cual puede ser usada para categorizar algún elemento.

5. Posibles Aplicaciones de la Estructura

Esta estructura se puede utilizar para encontrar elementos de dos llaves, es decir que se categoricen por dos valores o características, haciendolo de una forma rápida y sin tener que comparar punto por punto. Es decir, se puede utilizar una función que genere los puntos representativos de cada elemento de acuerdo a la necesidad del desarrollador o usuario para despues obtener su información oportunamente.