Geometría Computacional
Práctica 04

Profesor:
Adriana Ramírez Vigueras

Ayudante:
José Emiliano Cabrera Blancas

22 de abril de 2014

1. Objetivos:

- Que el alumno implemente un árbol de rangos de dos dimensiones junto con su función de búsqueda.

2. Descripción

Un Árbol de Rangos tiene como objetivo encontrar todos los puntos que se encuentren en una caja dada. Para representar dicho conjunto reutilizaremos la lista doblemente ligada que se implementó para las prácticas anteriores, la única diferencia que van a encontrar es, en lugar de utilizar el tipo *void** se utilizará el tipo *vertex**. La razón es clara, solo necesitamos que la lista doblemente ligada trabaje con puntos.
Otro punto importante para poder implementar el árbol de rangos en dos dimensiones es, primero necesitamos el árbol de rangos en una sola dimensión, dicho árbol de rangos estará representado con un árbol rojo negro. A diferencia de la lista doblemente ligada esta estructura es exactamente igual a la de la práctica dos.(Hint: El árbol rojo negro también sirve para ordenar los puntos con respeto a una coordenada dada.)
En el encabezado *range_tree/range_tree.h* se encuentra la descripción de las funciones que faltan por implementar en el archivo *range_tree/range_tree.c*.

3. *Tests*

Su código debe de pasar los siguientes *tests*:

Tests

```
range_tree.h: two_d_range_query recibe NULL.                [PASS]
range_tree.h: two_d_range_query recibe un arbol vacio       [PASS]
range_tree.h: two_d_range_query recibe un arbol no vacio    [FAIL]
```

El primer *test* verifica que la función *two_d_range_query()* regresa *NULL* cuando recibe *NULL*.
El segundo *test* verifica que la función *two_d_range_query()* regresa una lista vacía cuando recibe un árbol de rangos vacío.
El último *test* verifica que la respuesta que genera la función *two_d_range_query()* sea correcta.

4. ¿Qué debes de programar y dónde comentar tus cambios?

Básicamente debes de completar las funciones que se describen en el encabezado *range_tree/range_tree.h* y son las siguientes:

- **ra_node* build_2d_range_tree_node(list* set_of_points)**: Esta función debe crear de forma recursiva un árbol de rangos de dos dimensiones.
- **ra_tree* build_2d_range_tree(list* set_of_points)**: Esta función es la que se manda a llamar para crear un árbol de rangos de dos dimensiones, y es ayudada por la función anterior.
- **rb_node* find_split_node_1d(rb_tree* tree, double y, double y1)**: Como saben el algoritmo de búsqueda para el árbol de rangos de dos dimensiones hace uso del algoritmo de búsqueda del árbol de una dimensión. Parte fundamental de ambas búsquedas es encontrar el "split_node", esta función búsca ese nodo en un árbol de una dimensión.
- **ra_node* find_split_node_2d(ra_tree* ra_tree, double x, double x1)**: Tiene el mismo proposito de la función anteior pero en un árbol de dos dimensiones.
- **void one_d_range_query(rb_tree* tree, double y, double y1, list* report_points)**: Obtiene los puntos en el rango que se le pasan y los guarda en la lista *report_points*.
- **list* two_d_range_query(ra_tree* ra_tree, double x, double x1, double y, double y1)**: Obtiene los puntos en el rango que se le pasan y regresa la lista de esos puntos.
- **void report_tree(list* list_of_points, rb_node* subtree)**: Es una función auxiliar para agregar los puntos de un subárbol a una lista de puntos.

Cabe mencionar que el árbol de rangos de una dimensión lo deben de hacer con respecto al eje Y y los puntos de los nodos del árbol de rangos de dos dimensiones deben

de estar ordenados con respecto al eje X.

Por último deben completar el código del método *draw_range_query* que se encuentra en el archivo *gui/range_trees.rb*.

Recuerden agregar un archivo *README.txt* donde describan los cambios que hicieron o comentarios que tengan sobre la práctica.

Entrega 7 de mayo de 2014