Memoria Práctica MAR1: Jorge Ortega Carretero 2A

Implementación Montículo Binomial:

La práctica ha sido realizada en Visual Studio 2022, para su ejecución valdrá con disponer VS 2022 para programar en C++. Disponemos de un archivo .h que contiene la estructura del Montículo binomial y sus funciones. Y un archivo .cpp en el que están implementadas las pruebas y el main para ejecutarlo.

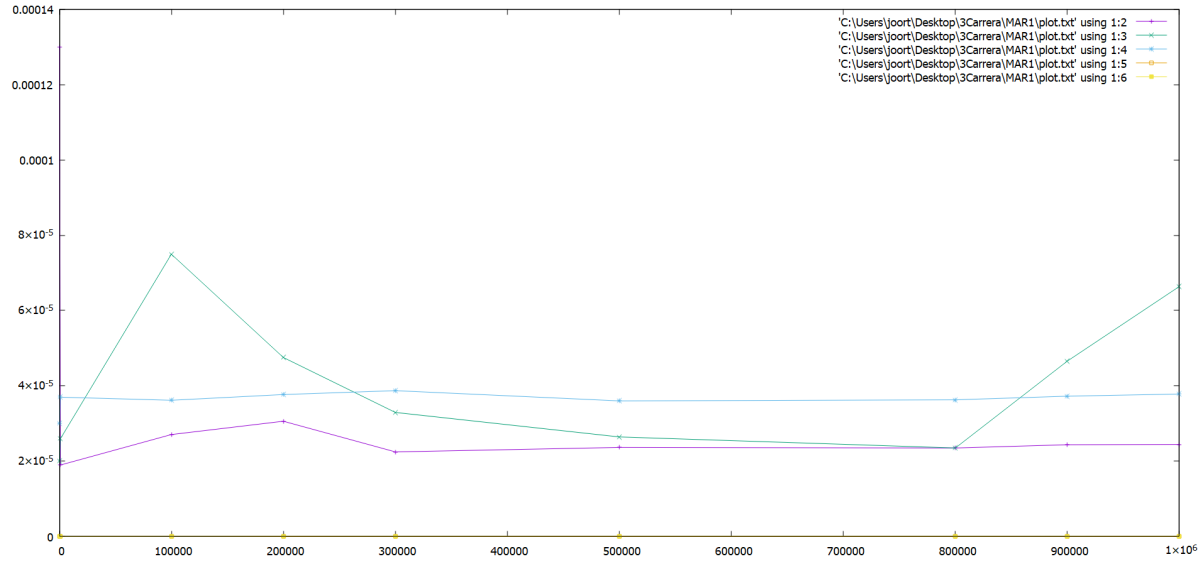
Tiempos de pruebas para x nodos:

Formato: nodos,insertar,borrar,unir,calcMin,Decr Clave

100,0.00013,2e-05,3e-05,0,0
1000,1.9e-05,2.6e-05,3.7e-05,0,0
100000,2.708e-05,7.496e-05,3.618e-05,1.3e-07,1.2e-07
200000,3.0615e-05,4.754e-05,3.771e-05,2e-08,1.3e-07
300000,2.24933e-05,3.295e-05,3.87367e-05,1.66667e-08,1.23333e-07
500000,2.3676e-05,2.645e-05,3.6028e-05,2e-08,1.3e-07
800000,2.35138e-05,2.352e-05,3.63088e-05,2e-08,1.275e-07
900000,2.43811e-05,4.65889e-05,3.72722e-05,2.88889e-08,1.52222e-07
1000000,2.4472e-05,6.6399e-05,3.7835e-05,1.7e-08,1.39e-07

Gráfica Tiempos Pruebas:
La tabla tiene un eje x que son los nodos y un eje y que es el tiempo de ejecución.

Morado: Insertar elementos
Verde: Borrar Mínimo
Azul: Unir
Naranja: Calcular Mínimo
Amarillo: Decrecer Clave



Todas las pruebas han sido realizadas repitiendo un determinado número de veces la operación y calculando la media de estas. Algunas gráficas dan resultados un poco distintos a los esperados ya que al realizarse con números aleatorios se pueden dar casos que el método a realizar no se llegue a ejecutar y disminuye el tiempo de ejecución.

Insertar Elementos: Tenemos que Insertar elementos tiene un coste logarítmico O(log(n)) ya que Crear el nuevo nodo tiene O(1), crear un montículo nuevo y añadirlo es O(1) y unir el nuevo montículo y controlar los grados de los árboles tiene un coste O(log(n)).

Borrar Mínimo: Tiene un coste O(log(n)) ya que mantenemos un puntero al mínimo por lo que acceder a él es O(1), posteriormente tras borrarlo tenemos que unir los hijos del mínimo en un nuevo montículo O(log(n)) y volver a calcular el mínimo del montículo O(log(n)) ya que recorro solo la lista de cabezas. Finalmente unimos los dos montículos y controlamos los grados de árboles. O(log(n))

Unir: Tiene un coste O(log(n1) + log(n2)) + 2) -> O(log(n)) ya que recorro las listas de raíces de los montículos y las voy uniendo en un montículo nuevo. Realizando las comprobaciones necesarias. Recorrer lista mont 1 -> O(log(n1)) y recorrer lista mont 2 -> O(log(n2)). Posteriormente controlamos los grados de los árboles del montículo O(log(n)).

Calcular Mínimo: Tiene un coste O(1) ya que mantenemos el puntero actualizado al mínimo tras cada acción realizada en el montículo.

Decrecer Clave: Tiene coste O(log(n)) ya que suponemos que disponemos ya del puntero al nodo que queremos decrementar su clave. Decrementamos la clave de este O(1) y posteriormente realizamos la flotación hasta cuadrar el nodo en su nueva posición debido a su cambio de clave O(log(n)). Por último calculamos el mínimo del montículo.