Estructuras de dator y álgoritmos II
Tarea 1

Rubén Pérez Palacios Lic. Computación Matemática
Profesor: Dr. Carlos Segura González

27 de febrero de 2022

1. Segment Tree (Modificación)

Puesto que la estructura de datos que implemente la tarea pasada de un Segment Tree partía de la supsición que para obtener el valor de un segmentree se calcula apartir del valor de sus hijos, el cual no es el caso, entonces se modifico la estructura completa.

Para la solución del problema primero tomemos en cuenta que nos preguntan sobre todo nuestro arreglo a y consideremos un bucket b de los elementos de este (nótese que esto es posible puesto que los valores van de 0 a 1000000), donde en la posición $b[i]$ almacenamos todas las posiciones en las que en nuestro arreglo a hay un elemento i. Luego construimos un segmentree sobre este bucket b. Para preguntar el valor del k elemento del arreglo ordenado de a entonces podriamos contestarlo con el Segment Tree, preguntando desde la raíz de este a su hijo izquierdo cuantos inidces i que tiene almacenado, si i es menor a k (estricto puesto que k está indexado en 0) entonces nos dirigimos al hijo izquierdo puesto esto sígnifica que hay al menos k elementos con valor entre el rango que representa el hijo izquierdo por lo tanto el k elemento esta en ese rango, en caso contrario nos dirigimos al hijo derecho pero ahora buscando el valor $k - i$ puesto ya habí i elementos antes en el rango del hijo izquierdo, así recursivamente hasta que llegemos a una hoja y el valor buscado será el valor k. Algo importante es necesitamos una estructura de datos que permita almacenar las posiciones y que nos permita insertar, eliminar y saber cuantos elementos tiene menores a un valor en $O(\log(N))$ lo cual es posible con un arbol binario de busqueda autobalanceado, prodiamos implementarlo con un treap, o un redblack tree el cual fue el usado en esta implementación, el cual ya esta implementado en la librería GNU.

Ahora si tuviesemos para cada subrango un segmentree de buckets podriamos contestar todas las querys en $O(\log(N)\log(M))$, el problema es que necesitariamos $O(M^2(N\log(M)))$ espacio y tomaría actualizar $O(M\log(M)\log(N))$ de las cuales ninguna son viables. Pero notemos que en nuestro forma de ir contestando la query lo que necesitamos saber en cada momento cuantos elementos hay en el rango del hijo izquierdo hay para decidir a donde movernos, pero puede ser que algunos de los índices que contiene el hijo izquierdo no esten dentro del rango que podriamos preguntar pero podemos saber cuantos indices de estos están dentro del rango que estamos preguntando (puesto que le podemos

preguntar al rbt cuantos elemenos son menores que algún valor), entonces con este nuevo valor i podemos ir bajando hasta una hoja y encontrariamos la respuesta. De esta forma solo necesitariamos un solo segmentree, por lo que nuestra memoria sería $O(N \log(M))$ y el tiempo de actualización $O(\log(N) \log(M))$, por lo tanto este algoritmo si resuelve en tiempo el problema.