Programación de estructuras de datos y

algoritmos fundamentales (Gpo 14)

Act 3.2 - Árbol Heap: Implementando una fila priorizada

Alumno:

Alejandro Daniel González Carrillo A01570396

Profesores:
Luis Humberto González Guerra

Monterrey, Nuevo León. 16 de octubre de 2020.

Reflexion: Durante el desarrollo de esta actividad pude realizar una priority queue en donde mis elementos se ordenaban en manera de una max priority queue. En el proceso se hicieron 6 funciones el empty, size, top, print, pop, push. El metodo empty era nomas que regresara un booleano si el tamaño del vector era igual a 0. Su complejidad era O(1) ya que no iteraba sobre el vector. El metodo size obtiene el tamaño del vector y le resta 1 ya que vector siempre inicia el conteo en 0. Su complejidad era O(1) ya que el metodo utilizado datos.size() es una funcion constante. El metodo top regresaba el valor con mayor prioridad de la queue en este caso el primer valor de la queue ya que cada vez que se realizaba un push se ordenaban los elementos. Su complejidad era O(1) debido a que nada mas se esta realizando la busqueda de una posicion en especial y no estamos iterando en el vector. En el metodo push se realiza un push_back al vector con el valor del usuario. Depsues de realizar el push_back realice un algoritmo el cual me permitio ir reordenando los valores cada vez que se realizaba un push_back. La complejidad del metodo push era O(log(n)) debido a que en el while se esta dividiendo entre dos entonces cada vez que se entre al while sera logaritmicamente. El indice va saltando entre dos. El metodo pop que considero fue el más dificil realizaba un intercambio el dato de mayor prioridad con el ultimo dato del vector y luego realizando un pop_back. En este caso tendria el ultimo dato desorganizado, entonces para reorganizarlo era necesario realizar un algoritmo donde entraba la funcion 2n y 2n + 1. En donde debiamos checar que existieran esas posiciones e ir verificando cual de esos dos valores era mayor e intercambiarlo con el valor padre. La complejidad de esta funcion era O(log(n)) debido a que indice del vector iba incrementando logaritmicamente en otras palabras se multiplicaba por 2 cada vez que terminaba el while. El metodo print lo unico que hace era imprimir los valores dentor del vector. Imprimia el valor de mayor prioridad con el metodo top y lo eliminaba con el metodo pop.