Actividad 3.4 **Árbol Heap – Actividad Integral** **Tarea**	TC1031.501: **Programación de estructuras de datos y algoritmos fundamentales** Baldomero Olvera Villanueva	Tecnológico de Monterrey
Integrantes	Esteban Sánchez García A01251440	

Introducción a los Árboles Heap (Heap Binario).

Un Heap binario es otro formato de estructura de datos, similar al árbol binario. Cada dato tiene sub-datos y esos sub-datos tienen sub-datos. Lo interesante de esta estructura de datos es que se puede realizar de diferentes maneras. Puede hacerse con vectores, arreglos, e inclusive con listas ligadas y doblemente ligadas. Las listas Heap cuando se arman con la implementación de priorización tienen dos tipos: Max Heap y Min Heap. El Max Heap es uno donde el primer elemento en la lista siempre será el elemento con mayor valor. Y el Min Heap es el opuesto al Max Heap, el primer elemento de la lista siempre será el elemento con menor valor. En este ejercicio, se creó un código que crea una lista priorizada MaxHeap usando vectores.

Complejidad y Operaciones

A diferencia del árbol binario, esta estructura puede tener una variedad de complejidades dependiendo del método. Cuando se realiza la fila priorizada Heap con arreglos o vectores, las operaciones de insertar y eliminar (*push y pop*) son O(log n). Mientras tanto, si se crea una fila Heap con Listas Doblemente Ligadas, la mayoría si es que no todas las operaciones serán de complejidad O(n). A la costa de ser un programa más largo, por lo general.

A continuación, una tabla explicando de forma resumida la complejidad de cada operación y el porqué son esa complejidad.

Operación	Eficiencia y Razonamiento
IsEmpty() []	Como solo revisa si existe o no al menos un dato en la lista, este siempre será de complejidad O(1)
Insert() []	Siempre que se inserta un dato, se debe mover el elemento hasta la parte superior y recorrerlo en cada parte hasta que se encuentre su lugar en la lista. Es por esto que es Complejidad O(log n)
GetMax() []	Como solo se regresa la variable en la raíz, no se realiza ninguna operación real. Se dice entonces que tiene complejidad O(1)
ExtractMax() []	ExtractMax es la operación más cercana a eliminar. Lo que hace esta operación es eliminar la ráiz y solo la raíz. Cuando este es eliminado, la lista o árbol heap se debe reorganizar por prioridad y encontrar una nueva raíz. La acción de reacomodar datos hace que esta operación tenga una complejidad de O(log n).

IPs infectadas.

La situación problema que se debe afrontar trata de la detección y prevención de bots o IPs “zombies”. Como en este ejercicio se hizo un sistema de navegación de IPs, surge la pregunta de cómo se puede identificar un IP infectado o bot de uno de un usuario real. Bueno, una forma de hacerlo a través de las IPs y de la transferencia de paquetes es a través de un concepto conocido como Rate Limiting. Rate Limiting es el concepto aplicado de limitar la cantidad de información que sale y entra en una red. Si muchos datos salen a una velocidad increíblemente rápida, casi sin límites, se puede asumir que es un bot. Esto es el método que se cree usar en la situación problema, por ahora.

Usos y conclusiones

Después de realizar la actividad, es posible resolver la duda de “¿Qué usos tienen crear heaps o árboles binarios que usen elementos de “priority queue” y priorizan algunos datos?”. Bueno, un ejemplo muy común se puede observar en el algoritmo que usan servicios de GPS para crear rutas óptimas de viaje. También se puede usar para queues en aplicaciones de música, donde se crear una lista de archivos de música priorizados por el usuario, o por que tan popular es una canción con el usuario. Algunas apps de mensajes también implementan esta estructura para determinar que mensajes u objetos se deberían de enviar primero si es que se envía algo que no es un simple mensaje de texto.

En conclusión, cada algoritmo de estructura de dato va apoyando a la creación de otro. Hemos observado como el vector, un formato simple para almacenar datos es elevado con priorización y toma la forma de heaps, construido como si fuese un árbol binario. Entre más elaborados son estas estructuras, más eficiente y más tareas impresionantes se pueden realizar con ellos.

Referencias

- Clinton, D. (9 de septiembre, 2020). “Binary Heap”. Recuperado el 30 de octubre del 2020, de https://www.geeksforgeeks.org/binary-heap/
- KeyCDN (4 de octubre, 2018). “What is Rate Limiting?”. Recuperado el 2 de noviembre del 2020, de https://www.keycdn.com/support/rate-limiting#:~:text=Rate%20limiting%20is%20used%20to,an%20error%20will%20be%20triggered.