Luis Arturo Rendon Inarritu
A01703572

Tecnológico de Monterrey

Reflexión Final de Actividades Integradoras

1. Introducción, 5 actividades integradoras:

Act 1.3 – Conceptos básicos: En esta actividad nos introdujimos por primera vez a algoritmos avanzados para ordenar una estructura, nos enfrentamos a 3 algoritmos no eficientes como Bubble sort, Swap sort y Insertion sort para después compararlos con algoritmos como Quicksort y Mergesort los cuales tienen una eficiencia asombrosa. También empezamos a entender que conllevan realmente las distintas complejidades y el impacto en el resultado final de estas. Los tres primeros algoritmos que mencioné tienen una complejidad de O(n**2) lo que conlleva un peso enorme para cualquier búsqueda, en especial si el vector no es lineal o son otros tipos de estructuras. Y los dos otros algoritmos que vimos tienen una complejidad de O(n log n) lo que los hace muchísimo más eficientes que los primeros 3. Lo interesante aquí es que realmente no es una mejoría como tal de lo que ya usábamos, es más una manera distinta de acercarse al problema, es realmente fabuloso. Ya lo he mencionado en otras actividades, es como la computación cuántica, no es el mejorar lo que ya se tiene, es una aproximación totalmente distinta. Hay algo muy curioso que se mencionó con estos algoritmos, Merge sort a pesar de ser tan rápido ocupa mucho espacio ya que va dividiendo el algoritmo en muchos Sub-vectores (en este caso usamos vectores), esto me interesó ya que no se me ocurrieron muchos casos donde esto pueda ser un problema, al menos hoy en día el hardware ayuda mucho, ahora si tu estructura es kilométrica, si mejor algo así como Quicksort.

Ahora también están los algoritmos de búsqueda, en los cuales realmente el que se me quedó bien grabado, en vez de iterar, es usas Binary search con el cual vamos dividiendo el vector ORDENADO en dos eliminando las mitades en donde lo que buscamos no está. Simple y elegante.

Luis Arturo Rendon Inarritu
A01703572

Act 2.3 – Estructuras lineales: Estas estructuras se me hicieron fascinantes ya que al final son estructuras como los vectores, son lineales, pero ofrecen una ventaja infinita, puedes hacerlas crecer o encogerse a voluntad. El insertar y sacar valores es increíblemente fácil y puedes ir de atrás o adelante (si es una lista doblemente enlazada) con una complejidad de O(1). Al final realmente es lo mismo que un vector que puedes hacer largo o corto a voluntad, pero tienes que recorrerlo igual que un vector, de forma lineal O(n), o puedes usar algoritmos como lo que ya mencioné para buscar y ordenar. Estos se forman creando objetos donde metemos la información y que además tienen apuntadores que apuntan a los nodos vecinos, estos si están al final apuntan a null del lado donde no hay más nodos.

Act 3.3 – BST: Aquí llegan los Árboles de búsqueda binaria, en esta parte nos enfrentamos a estructuras de datos jerárquicas donde de igual manera manejamos nodos que se referencian entre sí, pero lo que cambia es que cada nodo tiene dos nodos como hijos, a la izquierda y a la derecha e inicia con la cabeza que es el primer nodo, el que está hasta arriba. Ahora el chiste es que a la derecha están los números más grandes que el nodo y a la izquierda lo más chicos de esta manera se mantiene ordenado y podemos buscar cualquier dato con búsqueda binaria dividiendo el árbol en 2 dándonos una complejidad de O(n log n). Ahora la diferencia entre el BST y una lista ligada es cómo está la información ordenara ya que en el árbol hay una jerarquía con nodos a la derecha y a la izquierda. También aquí ya se usó la estructura Heap, es diferente al BST ya que no necesariamente mantiene un orden con todos los nodos superiores o los de abajo. Un Max heap el valor de cada nodo es mayor al de todos sus hijos, en el Min heap, pues es lo contrario jejeje, cada nodo es menor a los nodos que tiene como hijos, al insertar valores está padre porque tienes que hundir valores hasta el fondo para ordenar

Act 4.3 – Aquí se pone bien interesante la cosa, en esta actividad usamos a los hermosos Grafos. La verdad donde mi atención se vio completamente atrapada fue cuando me di cuenta que aplicaciones como Google maps o Waze utilizan grafos, igual que nosotros conectan nodos poniéndoles peso para poder calcular la dificultad de ir de un lugar a otro. Los grafos son nodos (los vértices) conectados entre sí, pueden ser

diversas conexiones con cualquier otro nodo creando redes, las conexiones (las aristas) tienen un peso, entonces al moverte de un nodo a otro hay “un costo” por decirlo de alguna forma. Aquí es donde usamos el algoritmo Dijkstra con una complejidad de O(V2) el cual recorre muchos caminos para llegar de un nodo a otro y nos regresa el viaje con menor peso ¡Igual que aplicaciones como Waze o Google maps! ¿Genial no? Ahora también están los Grafos bipartitos los cuales presentan una relación entre nodos, pero… no sé cómo explicarlo, tengo la imagen mental. Puedes dividir los nodos en dos grupos, pintarlos era, y los nodos de un grupo SÓLO están conectados con los nodos del otro grupo, si hay una conexión entre nodos de un mismo grupo ya no funciona.

Act 5.2 – Códigos Hash: Por último, está la Tabla hash, en esta toooodo pasa por un filtro que toma los datos y te arroja como resultado una clave, por dar un ejemplo que tal que mi algoritmo hash es un +2, a todo dato que yo ingrese le va a sumar 2 y ese va a ser su id de posición. Por decir yo ingreso el 0, su id de posición será el 2. Ahora gracias a esto es mucho más rápido encontrar y buscar valores ya que tu le das al algoritmo Hash lo que necesitas encontrar y mediante la función filtro te va a devolver la localidad, entonces ahí estará lo que estás buscando. Lo curioso es que si dos valores que tu ingreses te dan el mismo índice gracias a la función que utilizaste los dos se almacenarán en esa posición, por eso combinamos las Tablas hash y en cada localidad guardamos una lista ligada en donde vamos metiendo los datos. Ahora aquí hay algo interesante, el chiste de la Tabla hash es que puedes acceder a la información con suma facilidad ya que cada cosa tiene su id, o algunas pocas cosas tienen su id, pero que tal si no… que tal que diseñaste mal tu función Hash y todo lo que ingreses, o muchas cosas salen con el mismo id, entonces pierde sentido porque tienes que trabajar todo sobre solo una lista ligada al igual que si no tuvieras una tabla hash, mientras mejor dispersa esté la información, mejor funciona. Lo impactante es que si tienes todo bien distribuido, la búsqueda es de O(1)… Damm….

2. Análisis:

Luis Arturo Rendon Inarritu
A01703572

Creo que en la respuesta anterior me emocioné y expliqué de más entonces resumiré el análisis bastante.

a. ¿Cuáles son las más eficientes?

En búsqueda la búsqueda binaria siempre va a brillar, realmente no hablo de las Tablas hash porque no lo considero casi búsqueda, es como si supieras dónde está cada cosa que necesitas.

En ordenamiento Merge sort gana, aunque en la rara situación donde no tengas espacio, o que tus datos sean millones de millones y los tengas desordenados, entonces mejor Quick sort, aunque es más inestable.
En guardar información, en datos, gana la Hash table ya que te permite recopilar información casi instantáneamente. Ahora si lo que quieres es mantener una jerarquía un árbol de búsqueda binaria, ya sea un Max heap o un Min heap depende lo que necesites funcionaría. Y por último si quieres que las conexiones entre nodos tengan peso y realmente no hay un orden, solo tienes nodos desperdigados, se utilizaría un grafo.

b. ¿Cuáles podrías mejorar? Argumenta cómo lo harías.

Esta pregunta está algo complicada…

A ver voy a elegir una digamos… YA SE, es una locura, pero tomemos una lista doblemente ligada, que tal que conectamos el último nodo al final formando un círculo. Ahora esto está raro, pero podríamos tener como un… digamos una mano mágica que se mueve alrededor y siempre sabemos donde está. Podríamos movernos alrededor del cinturón para recopilar algún dato, o llevémoslo a 3 dimensiones, que tal que es una esfera donde hay puntos y cada uno es un nodo conectado a los colindantes. Podríamos tener varias manos voladoras, o apuntadores, que se muevan alrededor de la esfera para recoger los datos necesarios. Está loco y realmente no se muy bien cómo podría probar ser una mejora, pero el concepto está divertido apoco no.

Luis Arturo Rendon Inarritu
A01703572

3. Conclusiones personales:

   De conclusiones… pues nada, estoy fascinado con el curso. Son maneras de programas a las que nunca me había enfrenado, son otras maneras de pensar y de organizar información, I mean, la gente que pensó en estas cosas está cañona apoco no.

   La verdad el curso me encantó, aprendí, me gustó, me atrapó y sobre todo mis respetos al profesor, le encanta el tema, le sabe excelentemente bien y es un gran maestro. Sabe enseñar, sabe ayudar, sabe apoyar, y es una gran persona, realmente aprendí y le agradezco todo a él. Además, el curso es pesado ¿Y en verano? ¿Y en línea? Lo hizo molto bene. Aprendí, me llevo cada conocimiento sin pasarme nada, estoy feliz.