Proyecto de Investigación

Algoritmo de Dijkstra y

Estudiante: Paola Bonilla Bonilla

B31061

Escuela de Ingeniería Eléctrica, Universidad de Costa Rica.

Profesor: Juan Carlos Coto

3 de julio, 2020.

Introducción

Junto con el almacenamiento de datos, el empleo de algoritmos en el ámbito tecnológico es de suma importancia. No se podría entender, por ejemplo, el enrutamiento de los paquetes en internet sin algoritmos que calculan las distancias mínimas entre los nodos de la red, algoritmos de distancias mínimas en grafos, o sería imposible el comercio electrónico sin los algoritmos criptográficos para proteger las transacciones bancarias.

Se estudiará y comentará la relevancia del algoritmo de Dijkstra por ser voraz y generar uno a uno los caminos de un nodo al resto por orden creciente. Es importante destacar la importante herramienta como parámetro de optimización. Además, se comentará el papel fundamental de algunos conceptos claves para la completa comprensión del algoritmo y entender las principales aplicaciones que han fortalecido las herramientas de programación.

Por otro lado, las estructuras de datos brindan la posibilidad de administrar todo tipo de datos sin ningún tipo de obstáculo, algo que en la actualidad se usa en la red para poder llevar a cabo, por ejemplo, los sistemas de indexado de contenidos. Y también juegan un papel clave en la creación de los mejores algoritmos, así como en su uso con lenguajes de programación que se benefician de ellas. Se comenta la importancia de las estructuras de datos de Árboles rojo-negro.

Algoritmo de Dijkstra

El algoritmo de Dijkstra es un algoritmo de caminos mínimos, ideal para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959. Es un algoritmo iterativo que brinda la ruta más corta desde un nodo inicial particular a todos los otros nodos.

Uno de los principales fundamentos que intenta construir este algoritmo es el principio de optimalidad, este principio aplicado en programación dinámica consiste en que una secuencia óptima de decisiones que resuelve un problema debe cumplir la propiedad de que cualquier subsecuencia de decisiones, que tenga el mismo estado final, debe ser también óptima respecto al subproblema correspondiente. (Real Academia de Ingeniería, 2020)

El algoritmo de Dijkstra de la mano con el principio de optimalidad define en una ilustración que si la distancia más corta entre los vértices *u* y *v* pasa por el vértice *w*, entonces el trayecto que va de *w* a *v* debe ser el más corto entre todos los trayectos que van de *w* a *v (Figura 1)*. De esta manera, se van construyendo sucesivamente los caminos de coste mínimo desde un vértice inicial hasta cada uno de los vértices del grafo, y se utilizan esos trayectos conseguidos como parte de los nuevos caminos.

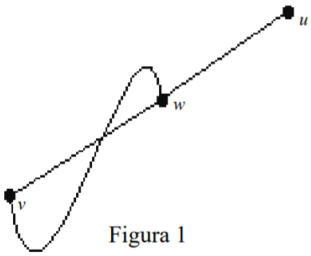

Figura 1

Es importante conocer la relevancia de los grafos en esta investigación porque como se menciona anteriormente, son una parte esencial en la construcción de la definición del algoritmo en este caso.

Un grafo es un conjunto de nodos asociados entre sí mediante arcos, los cuales pueden o no tener peso y pueden o no tener dirección. Los grafos no dirigidos tienen aplicaciones en la representación de estructuras lógicas. Los grafos dirigidos se utilizan para representar y computar flujos; es común utilizar grafos pesados dirigidos para representar la capacidad de flujo en un enlace dentro del sistema, como, por ejemplo: Duración de traslado de un punto a otro. Ancho de banda disponible entre dos centros de datos. Demanda de un recurso particular. (Coto, pg 3, 2020).

Aplicaciones del algoritmo de Dijkstra

En la actualidad el ser humano dispone de mucha tecnología para que la realización de sus tareas cotidianas se facilite y optimice. La aplicación del algoritmo de Dijkstra

cumple un papel fundamental en la cotidianidad de tareas indispensables para diversas áreas, dos de esas aplicaciones son:

- **Encaminamiento de paquetes por los routers:**

  El algoritmo de Dijkstra es de suma importancia en las telecomunicaciones como una simple red telefónica. Los paquetes de información y datos pueden tardar cierta cantidad de tiempo para atravesar cada línea (debido a efectos de congestión, retrasos en las conexiones etc.) En este caso, si se cuenta con una red con costes en los arcos y dos nodos especiales: el nodo de comienzo y el de finalización, el objetivo que cumple es encontrar un camino entre los dos nodos cuyo coste total sea el mínimo.

- **Caminos mínimos en Grafos usando XML y parsers de Java:**

  Cuando se presenta un conjunto de descripciones de operadores, conectores y parámetros de optimización el sistema se encargará de encontrar un camino óptimo de una entrada establecida hasta un tipo de salida especificada aplicando transformaciones específicas en el menor tiempo posible.

  Es importante definir los conceptos conectores y operadores. El camino es una secuencia de operadores y conectores: un operador será aquella unidad de proceso de información llevando a cabo un algoritmo específico como conversores digitales, de formato etc y un conector será cualquier mecanismo a través del cual los operadores se comunican entre sí.

Estructura de datos: Árboles Rojo-Negro

Un árbol rojo-negro es un árbol binario de búsqueda auto balanceado. Un árbol

que busca mantener su altura lo más baja posible en cada momento, es decir, de forma más específica, después de cada inserción o eliminación de un nodo. Un árbol binario rojo-negro tiene la misma estructura que un árbol binario común, con sus elementos menores a la izquierda y elementos mayores a la derecha y además, cumplen las siguientes restricciones:

- Todos los nodos son rojos y/o negros.
- Todas las hojas son negras.
- La raíz siempre es un nodo negro.
- Si un nodo es rojo, sus dos hijos son negros.
- Todos los caminos hasta sus hojas tienen el mismo número de nodos
- negros, esto para cada nodo.

Estos árboles son más complejos que los árboles binarios de búsqueda normales, ya que las inserciones y eliminaciones intentan mantener la condición de árbol balanceado. Sin embargo, permiten que tanto la inserción, la eliminación y la búsqueda tengan siempre un coste O(log(n)), donde n indica el número de nodos.

Lo anterior es de suma importancia porque optimiza el tiempo y facilita en gran medida las búsquedas, a cambio de que las inserciones y las eliminaciones sean más complejas debido a las operaciones necesarias para intentar mantener dicho balanceo.

El proceso de insertar un nodo en un árbol rojo-negro se gestiona de forma similar a un árbol binario de búsqueda porque debe moverse entre los subárboles izquierdo y derecho hasta encontrar una hoja disponible, poniendo el nuevo nodo recién insertado como rojo. Cuando se desea eliminar, también se opera de la misma forma que en los árboles binarios.

No obstante, ambas operaciones pueden violentar las reglas explicadas anteriormente, por lo que después de insertar o eliminar un nodo, es necesario comprobar que el árbol sigue teniendo una estructura correcta y en caso de no ser así, modificar el árbol hasta conseguirlo. Los procedimientos para mantener las propiedades explicadas anteriormente hacen uso de rotaciones para lograr este objetivo. La complejidad de dichas rotaciones no depende del tamaño del árbol, solamente se cambian punteros, por lo que el coste es O(1). Esto, unido a que las operaciones de balanceo sólo se realizan como máximo O(log(N)) veces, hace que no afecte al coste total de las inserciones o eliminaciones del árbol.

Aplicaciones de Estructuras de datos: Árboles Rojos-Negro

En Java y C ++, las estructuras del mapa de la biblioteca generalmente se implementan con un árbol negro rojo. Los árboles negros rojos también son comparables en velocidad a los árboles AVL. Si bien el equilibrio no es tan bueno, el trabajo que se necesita para mantener dicho equilibrio suele ser mejor en un árbol negro rojo.

Los árboles negros y rojos pertenecen a una clase de BST de equilibrio automático y se puede utilizar cualquier árbol de equilibrio automático. Es importante agregar que los árboles rojo-negros se usan ampliamente como tablas de símbolos del sistema en la implementación de lo siguiente:

- Java: java.util.TreeMap, java.util.TreeSet.
- C ++ STL: mapa, multimapa, multiset.
- Kernel de Linux: planificador completamente justo, linux / rbtree.h

Conclusiones

Se puede concluir que los árboles son estructuras bastante complejas con grandes aplicaciones en la ciencia y en la programación convencional. El árbol rojo-negro es una implementación particular de un árbol de búsqueda binaria auto balanceado. Los árboles de búsqueda binarios se utilizan para implementar mapas finitos, donde almacena un conjunto de claves con valores asociados. También puede implementar conjuntos utilizando solo las claves y sin almacenar ningún valor.
En los últimos años este tipo de estructuras han sido utilizadas con mucha frecuencia en la inteligencia artificial, en montones que se utiliza en la implementación de colas de prioridad eficientes, que a su vez se utilizan para procesos de programación en muchos sistemas operativos, entre muchas otras aplicaciones.

Bibliografía

- Real Academia de Ingeniería. (2020). *Bellman's principle of optimality. [Blog].* Recuperado de: http://diccionario.raing.es/es/lema/principio-de-optimalidad-de-bellman
- Sánchez Torrubia. (s.f). *Algoritmo de Dijkstra. Un Tutorial Interactivo.* [PDF]. Recuperado de: http://bioinfo.uib.es/~joemiro/aenui/procJenui/ProcWeb/actas2001/saalg223.pdf
- EncuRed. (s.f.) *Algoritmo de Dijkstra.* [Blog]. Recuperado de: https://www.ecured.cu/Algoritmo_de_Dijkstra#Caracter.C3.ADsticas_del_algoritmo
- Runestone Academy. (2018). *Algoritmo de Dijkstra*. Recuperado de: https://runestone.academy/runestone/static/pythoned/Graphs/ElAlgoritmoDeDijkstra.html
- Coto. (2020). *Grafos. [pdf] Recuperado de:* https://www.dropbox.com/sh/g3zxzg0kxp4z6kc/AAAgRfM7dZ7wJ6_cdQqTaXdKa/Apuntes?dl=0&preview=Lecci%C3%B3n+2020-05-26+Grafos.pdf&subfolder_nav_tracking=1