Investigación sobre los algoritmos de Dijkstra y Floyd y como aplican la estrategia «Divide y Vencerás»

Beatriz Sánchez Delgado.

**Algoritmo de Dijkstra**

Dijkstra es un algoritmo que encuentra la ruta más corta entre un nodo de origen y todos los demás nodos en un grafo con pesos no negativos.

**Introducción Teórica**

Existen numerosos problemas que se pueden formular en términos de grafos. Ejemplos de ello son la planificación de las tareas que completan un proyecto, encontrar las rutas de menor longitud entre dos puntos geográficos, calcular el camino más rápido en un transporte, determinar el flujo máximo que puede llegar desde una fuente a, por ejemplo, una urbanización; entre otros. La resolución de estos problemas requiere examinar todos los nodos o todas las aristas del grafo que representa al problema; sin embargo, existen ocasiones en que la estructura del problema es tal que sólo se necesitan visitar algunos de los nodos o bien algunas de las aristas. Los algoritmos imponen implícitamente un orden en estos recorridos: visitar el nodo más próximo o las aristas más cortas, y así sucesivamente; otros algoritmos no requieren ningún orden concreto en el recorrido.

Cuando se trabaja con grafos dirigidos etiquetados o ponderados con factores de peso no negativos, es frecuente buscar el camino más corto entre dos vértices dados; es decir, el camino que nos permita llegar desde un vértice origen a un vértice destino recorriendo la menor distancia o con el menor costo. Los algoritmos más usados para este fin son: Dijkstra, Floyd y Warshall. Los tres algoritmos utilizan una matriz de adyacencia ponderada o etiquetada: que es la misma matriz de adyacencia utilizada para representar grafos, pero con la diferencia que en lugar de colocar un número "1" cuando dos vértices son adyacentes, se coloca el peso o ponderación asignado a la arista que los une. Con frecuencia en la matriz etiquetada suele utilizarse la siguiente notación: Suponiendo que M[i, representa la matriz de adyacencia, tenemos:

M [i, j] = 0 si i = j.

M [i, j] = 1000 , si no existe un camino de i a j, donde i ≠ j.

M [i, i] = costo de ir del vértice i al vértice j.

Otro nombre con el cual suele llamarse a una matriz de adyacencia etiquetada es matriz de distancias o matriz de costos. El problema de buscar un camino más corto entre dos nodos dados se puede resolver mediante un algoritrmo voraz conocido como Algoritmo de Dijkstra.

**Algoritmo de Dijkstra.**

El algoritmo de Dijkstra es un algoritmo eficiente (de complejidad (n), donde es el número de vértices) que sirve para encontrar el camino de coste mínimo desde un nodo origen a todos los demás nodos del grafo. Fue diseñado por el holandés Edsger Wybe Dijkstra en 1959. Este algoritmo es un tipico ejemplo de algoritmo ávido, que resuelve lbs problemas en sucesivos pasos, seleccionando en cada paso la solución más óptima con el objeto de resolver el problema. El fundamento sobre el que se basa este algoritmo es el principio de optimizar. si el camino más corto entre los vértices «u» y "v pasa por el vértice «w», entonces la parte del camino que va de a debe ser el camino más corto entre todos los caminos que van de «w» a «v». De esta manera, van cônstruyendo sucasivamente los caminös de coste mínimo desde un vétice inicial hasta cada uno de los vértices del grafo, y se utilizan los caminos conseguidos como parte de los nuevos caminos. Dicho en otras palabras: "Dado un grafo a cuyos arcos se han asociado una serie de pesos, se define el camino de coste mínimo de un vértice «u» a otro "v', como el camino donde la suma de los pesos de los arcos que lo forman es la más baja entre las de todos los caminos posibles de «u» a «v»".

El algoritmo de Dijkstra en cada paso selecciona un vértice «v» cuya distancia es desconocida, entre los que tiene la distancia más corta al vértice origen «s», entonces el camino más corto de «s» a «v» ya es conocido y marca el vértice como ya conocido. Así, sucesivamente se van marcando nuevos vértices hasta que estén todos marcados; en ese momento es conocida la distancia mínima del origen «s» al resto de los vértices.

Entre las condiciones más importantes que deben considerarse para aplicar el algoritmo están:

Las aristas deben tener un peso no negativo.

El grafo debe ser dirigido y por supuesto ponderado.

Una posible aplicación de este algoritmo se presenta cuando se desea encontrar la ruta más corta entre dos ciudades; cada vértice representa una ciudad y las aristas representan la duración de los vuelos.

Para la explicación general se tomará como referencia los siguientes pasos:

1. Seleccionar vértice de partida, es decir un origen.

2. Marcar el punto de partida como el punto de inicio.

3. Determinar los caminos especiales desde el nodo de partida, es decir, el de inicio.

4. Camino especial es aquel que solo puede trazarse a través de los nodos o vértices ya marcados.

5. Para cada nodo no marcado, se debe determinar si es mejor usar el camino especial antes calculado o si es mejor usar el nuevo camino especial que resulte al marcar este nuevo nodo.

6. Para seleccionar un nuevo nodo no marcado como referencia, deberá tomarse aquel cuyo camino especial para llegar a él es el mínimo, por ejemplo si anteriormente marqué el nodo o vértice 2, el cual tiene dos nodos adyacentes 3 y 4 cuyo peso en la arista Corresponde a 10 y 5 respectivamente, se tomará como nuevo nodo de partida el 4, ya que el peso de la arista o camino es menor.

7. Cada camino mínimo corresponde a la suma de los pesos de las aristas que forman el camino para ir del nodo principal al resto de nodos, pasando únicamente por caminos especiales, es decir nodos marcados.

**Ventajas**:

Eficiente para encontrar la ruta más corta desde un nodo de origen.

Funciona con grafos con pesos no negativos.

**Desventajas**:

No funciona con grafos con pesos negativOs.

No encuentra todas las rutas más cortas entre todos los pares de nodos.

**Algoritmo de Floyd**

Floyd es un algoritmo que encuentra la ruta más corta entre todos los pares de nodos en un grafo con pesos no negativos.

**Introducción Teórica**

En el vasto campo de la teoria de grafos y la resolución de problemas relacionados con la búsqueda de caminos más cortos, el algoritmo de Floyd-Warshall emerge como una herramienta invaluable. Este algoritmo, desarrollado por Robert Floyd en 1962 y posteriormente refinado por Stephen Warshall en 1962, se erige como una solución eficiente y versátil para el problema de encontrar las rutas más cortas entre todos los pares de nodos en un grafo ponderado dirigido o no dirigido. Su aplicabilidad no se limita a un dominio especifico, lo que lo convierte en una herramienta fundamental en diversas disciplinas, desde la planificación de redes hasta la optimización de rutas en sistemas de transporte.

Uno de los problemas centrales abordados por el algoritmo de Floyd-Warshall es el de encontrar el camino más corto entre cada par de nodos en un grafo ponderado. Esta tarea adquiere relevancia en escenarios donde se busca optimizar el tiempo, costo o cualquier otro criterio medido en teminos de las aristas que conectan los nodos.

**Funcionamiento**

1. Inicialización:

Se crea una matriz de distancias D con dimensiones n xn donde n es el número de nodos en el grafo. o Se inicializa la matriz D con las distancias entre los nodos directamente conectados. Si no hay conexión entre dos nodos, se asigna infinito como distancia. Se crea una matriz de predecesores P con las mismas dimensiones que Se inicializa la matriz Pcon -1 en todas las posiciones, indicando que aún no se conocen los predecesores en las rutas más cortas.

2. Iteración sobre nodos intermedios:

Se itera sobre todos los nodos del grafo (k) como nodos intermedios.

3. Actualización de distancias:

Para cada par de nodos i y j se verifica si la distancia entre ellos se puede mejorar pasando por el nodo intermedio k.

Si la distancia entre i y j a través de k (D[i][k] + D[k]) es menor que la distancia axtual entre i y j (D[i][j]) se actualiza la distancia en D [i][j] con la nueva distancia.

Si la distancia se actualiza, también se actualiza j en la ruta más corta desde i en la matriz P [i][j] con valor k.

4. Repetición:

Se repiten los pasos 2 y 3 para todos los nodos intermedios (k).

5. Resultado:

Al finalizar la iteración, la matriz D contiene las distancias más cortas entre todos los pares de nodos en el grafo.

La matriz P contiene los predecesores en las rutas más cortas, lo que permite reconstruir la ruta más corta entre dos nodos cualesquiera.

**Curiosidad**

Una curiosidad interesante sobre el algoritmo de Floyd es que, aunque fue descrito inicialmente en 1959 por Bernard Roy, es comúnmente atribuido a Robert Floyd, quien lo popularizó en 1962. Además, el algoritmo de Floyd-Warshall, como también se le conoce, es un ejemplo clásico de la técnica de programación dinámica, y su aplicación va más allá de la teoría de grafos, siendo utilizado en áreas como la bioinformática para la comparación de secuencias biológicas

**Ventajas:**

Encuentra todas las rutas más cortas entre todos los pares de nodos.

Funciona con grafos con pesos no

negativos.

**Desventajas:**

Menos eficiente que Dijkstra para encontrar la ruta más corta desde un nodo de origen. No funciona con grafos con pesos negativas.

**Aplicación de la Estrategia «Divide y Vencerás» en Dijkstra y Floyd.**

La estrategia Divide y Vencerás es una técnica de diseño de algoritmos que consiste en dividir un problema en subproblemas más pequeños, resolver los subproblemas recursivamente y luego combinar las soluciones para obtener la solución al problema original. Tanto Dijkstra como Floyd utilizan la estrategia Divide y Vencerás para encontrar las rutas más cortas en un grafo. A continuación, se explica cómo se aplica esta estrategia en cada algoritmo:

**Dijkstra**

En Dijkstra, el problema se divide en encontrar la ruta más corta desde el nodo de origen a cada nodo adyacente. Se puede considerar cada nodo adyacente como un subproblema.

1. Dividir: Se selecciona el nodo con la distancia más corta no visitado. Este nodo se convierte en el nuevo nodo de origen.

2. Vencer: Se calcula la ruta más corta desde el nuevo nodo de origen a cada uno de sus nodos adyacentes no visitados.

3. Combinar: Se actualizan las distancias de los nodos adyacentes al nodo seleccionado, si se encuentra una ruta más corta. Este proceso se repite hasta que todos los nodos estén visitados. Al final, se obtiene la ruta más corta desde el nodo de origen a todos los demás nodos.

Floyd

En Floyd, el problema se divide en encontrar la ruta más corta entre todos los pares de nodos pasando por un nodo intermedio. Se puede considerar cada nodo intermedio como un subproblema.

1. Dividir: Se selecciona un nodo intermedio.

2. Vencer: Se calcula la ruta más corta entre todos los pares de nodos pasando por el nodo intermedio.

3. Combinar: Se actualizan las distancias entre todos los pares de nodos, si se encuentra una ruta más corta pasando por el nodo intermedio. Este proceso se repite para todos los nodos intermedios. Al final, se obtiene la ruta más corta entre todos los pares de nodos.

**Ventajas de la estrategia «Divide y Vencerás» en Dijkstra y Floyd**

La estrategia Divide y Vencerás ofrece varias ventajas en la implementación de Dijkstra y Floyd:

Simplifica la resolución de problemas complejos: Divide el problema en subproblemas más pequeños y manejables.

Mejora la eficiencia: Permite resolver los subproblemas de forma independiente y luego combinarlos para obtener la solución final.

Facilita la recursividad: Permite aplicar el mismo algoritmo a los subproblemas, lo que simplifica la implementación.

En conclusión, los algoritmos de Dijkstra y Floyd son ejemplos fundamentales de la aplicación de la estrategia de divide y vencerás en la resolución de problemas computacionales complejos. Aunque estos algoritmos no aplican divide y vencerás de la manera clásica como lo hacen otros algoritmos como MergeSort o QuickSort, su diseño incorpora principios relacionados que optimizan el cálculo de caminos mínimos.

**Conclusión**

El algoritmo de Dijkstra se centra en resolver el problema de encontrar el camino más corto desde un único nodo origen a todos los demás nodos en un grafo ponderado y dirigido. Utiliza una técnica de relajación de aristas que, aunque no divide el problema en subproblemas más pequeños de forma explícita, sí conquista gradualmente el grafo al encontrar el camino más corto a cada nodo paso a paso, aplicando una estrategia de selección y evaluación sistemática. Por otro lado, el algoritmo de Floyd-Warshall aborda el problema de los caminos más cortos entre todos los pares de nodos en un grafo. Este algoritmo sí emplea una forma de divide y vencerás implícita al descomponer el problema en etapas, donde cada etapa considera los caminos que pasan por un conjunto incremental de nodos intermedios. De esta manera, el algoritmo conquista el problema global a través de la solución de problenas más pequeños y manejables, combinando sus resultados para llegar a la solución final.

Ambos algoritmos reflejan la potencia del paradigma divide y vencerás, demostrando que incluso cuando la división del problema no es evidente, los principios de esta estrategia pueden ser adaptados para desarrolar soluciones eficientes a problemas de optimización en grafos. Así, Dijkstra y Floyd representan la adaptabilidad y la ingeniosidad del diseño de algoritmos, mostrando que la estrategia divide y vencerás es más un marco de pensamiento que una receta fija, permitiendo su aplicación en una amplia gama de contextos computacionales.