TALLER N°3

El problema del máximo flujo

usach

Taller de programación 2-2024

Fecha: 06/01/2025

Autor: Vicente Aninat

TALLER N°3

El problema del máximo flujo

Explicación breve del algoritmo

El algoritmo elegido para resolver el problema del máximo flujo es el algoritmo de Dinic. Este, a partir de un grafo correctamente definido, puede encontrar la capacidad máxima de flujo desde un vértice fuente hasta un vértice sumidero. La forma en que hace esto es que a partir del vértice fuente, se calculan niveles incrementales hasta llegar al vértice sumidero con una búsqueda en anchura, como si de contar los pisos que subir en un edificio se tratase. La razón de estos niveles es que al recorrer el grafo, naturalmente siempre serán priorizados los caminos que acerquen más a uno hacia su objetivo. Una vez con el grafo nivelado, se realizan búsquedas en profundidad para encontrar caminos hasta el vértice sumidero a través de las aristas del grafo, una vez se llega al sumidero se define el flujo que pasa por aquel camino según la arista con menor capacidad. El proceso se repite hasta que no sea posible encontrar más caminos válidos, entonces el resultado del máximo flujo es la suma del flujo de los caminos calculados durante la ejecución del algoritmo.

Heurísticas o técnicas utilizadas

El algoritmo Dinic utiliza principalmente algoritmos de búsqueda en profundidad y búsqueda en anchura.

El algoritmo de búsqueda en anchura es esencial en todo lo relacionado a operaciones de grafos, se basa en que a partir de un vértice se visitan sus vértices vecinos y luego los vecinos de estos. De esta forma es posible determinar por ejemplo si se puede llegar a todos los vértices de un grafo a partir de un vértice inicial, lo que es sumamente útil para nivelar los vértices del grafo en este problema. Su complejidad algorítmica es de O(V + A), dónde V es el número de vértices y A es el número de aristas.

El algoritmo de búsqueda en profundidad, de importancia equivalente a la búsqueda en anchura, se basa en que a partir de un vértice se visita un vértice vecino y a partir de este otro vecino, marcando cada vértice visitado. Una vez no se puede seguir, se retrocede para continuar su camino por otro vecino hasta agotar completamente el grafo. A diferencia de la búsqueda en anchura y en relación al problema, la búsqueda en profundidad con la ayuda de los niveles de los vértices del grafo resulta útil para encontrar caminos directos y eficientes desde el vértice fuente y el sumidero.

Funcionamiento del programa

Al ejecutar el código, se despliega un menú por consola desde el cual se puede elegir una de entre cuatro opciones: Cargar un grafo, visualizar el grafo, calcular el máximo flujo del grafo o salir del programa.

Cargar un grafo: Al elegir cargar un grafo, se solicita ingresar el nombre del archivo dentro de la carpeta Graphs que quiere cargar, esto sin la extensión de archivo.
Luego, el archivo se lee de la siguiente forma:

1) Se verifica el acceso correcto al archivo
2) Se extrae la primera línea del archivo, el listado de vértices fuente y sus contenidos se almacenan en un arreglo de datos.
3) Se extrae la segunda línea del archivo, el listado de vértices sumidero y sus contenidos se almacenan en un arreglo de datos.
4) Se extraen las aristas, el resto de líneas del archivo y se ingresan al grafo.
5) Se crea un vértice súper fuente que conecte a todos los vértices del listado de vértices fuente y de esa forma solo haya un vértice fuente en el grafo.
6) Se crea un vértice súper sumidero que conecte a todos los vértices del listado de vértices fuente y de esta forma sólo haya un vértice sumidero en el grafo.
7) Queda almacenado el grafo dentro de la memoria del programa.

Visualizar un grafo: Al elegir visualizar un grafo, se muestra por consola las aristas del grafo mostrando los vértices origen y destino, el flujo actual, el flujo máximo, si son fuente, sumidero o normales y si son una arista residual.

Calcular flujo máximo: Al elegir el cálculo del flujo máximo, se llamará al algoritmo Dinic para resolver el problema, destacando también el tiempo que demoró en hacerlo.

Salir: Al elegir salir, se termina la ejecución.

Aspectos de implementación y eficiencia

Las tres operaciones en el programa que presentan exigencias al momento de ejecución son las referentes al conteo de vértices, la lectura de archivo y la resolución del problema. Una de las implementaciones de eficiencia es el uso del tipo de dato set para el almacenamiento de los vértices, lo que reduce los tiempos de inserción de O(n) a O(log n), que en grandes volúmenes de datos es una diferencia sumamente significativa.

Uno de los principales aspectos de implementación es que la clase Dinic que resuelve el problema hereda de una clase abstracta GrafoBase, la cual define los aspectos generales que necesita cualquier algoritmo de resolución. De esta forma es posible implementar con facilidad otros algoritmos sin necesidad de redundancias en atributos o métodos.

Ejecución del código

Para ejecutar el código, se debe ingresar por consola de comandos el comando ‘make’, esto compilará automáticamente todos los archivos del programa. Adicionalmente se puede ingresar el comando ‘make clean’ para borrar los archivos de compilación.

Para ejecutar el programa, ingresar ‘./main’ por consola de comandos. También están presentes archivos de prueba de las clases utilizadas, para ejecutarlos ingresar ‘./Test+Clase a probar’, por ejemplo: ‘TestDinic’.

Las librerías utilizadas para el programa son: queue, string, sstream, iostream, vector, list, chrono, fstream, algorithm y set.

Bibliografía

GeeksforGeeks. (2023, 13 marzo). *Dinic’s algorithm for Maximum Flow*. GeeksforGeeks.

https://www.geeksforgeeks.org/dinics-algorithm-maximum-flow/

GeeksforGeeks. (2025, 2 enero). *Breadth First Search or BFS for a Graph*. GeeksforGeeks.

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

Anexos