UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERÍA

CURSO: MA475 - MATEMÁTICA COMPUTACIONAL

PROBLEMA DEL FLUJO MÁXIMO

Integrantes	Código
Prieto Mantari, Leonardo Fabrizzio Junior	u202319949
Ríos Pacheco, Héctor Javier	u20231c540
Hermoza Quispe, Jude Alessandro	u202318220
Huapaya Vargas, Daniel José	u202312230
Palacios Valentin, Sebastian Wilfredo Yosue	u20231B463

Profesor:

Venegas Palacios, Edgar Kenny

UPC, septiembre 2024

Contenido

Flujo Máximo

El problema de flujo máximo es uno de los pilares más importantes y fundamentales dentro de la teoría de grafos, debido a su relevancia en la resolución de problemas complejos en diversos campos como la optimización de redes, la planificación de recursos y la ingeniería. Este concepto a la vez es aplicable en situaciones del mundo real, como la distribución eficiente de productos, el transporte de datos a través de redes de comunicación o la gestión de sistemas de abastecimiento de agua y energía, demostrando de esta manera la versatilidad y practicidad que posee.

El objetivo de este problema es determinar la máxima cantidad de flujo que puede transportarse desde un punto de origen hacia un punto de destino, esto a través de una red compuesta por nodos y aristas. Cada arista posee una capacidad máxima la cual no puede ser excedida, introduciendo restricciones en el flujo total que puede atravesar la red. De este modo, la finalidad es maximizar el flujo total mientras se respetan estas limitaciones, optimizando el uso de esta red.

En este proyecto, nos proponemos desarrollar un algoritmo que implemente una solución al flujo máximo. Utilizaremos técnicas de grafos como el algoritmo de Ford-Fulkerson, que buscará iterativamente caminos disponibles en la red hasta llegar al tope del flujo ofreciendo así un camino óptimo y eficiente.

Objetivos

El objetivo del proyecto es implementar un algoritmo eficiente para resolver problemas de flujo máximo, utilizando el algoritmo expuesto en el capítulo 10.2 de Johnsonbaugh (2005). u otros algoritmos adecuados, y asegurar que la aplicación funcione en diversos tipos de redes con distintas tipologías y tamaños.

Maximizar el flujo de redes con restricciones de capacidad, garantizando que los recursos sean utilizados de manera óptima y eficiente en cada escenario. Manejar redes pequeñas y simples para visualizar el paso a paso del algoritmo.

Crear una app capaz de manejar redes grandes y complejas, permitiendo su aplicación en casos reales de industrias como transporte, telecomunicaciones o suministro de energía.

Desarrollar una interfaz clara e intuitiva para que usuarios sin conocimientos técnicos profundos en teoría de grafos puedan utilizar la aplicación para resolver problemas de flujo máximo. Permitir que el usuario sea capaz de insertar un grafo manualmente en un lienzo.

Marco Teórico

Teoría de grafos

Un grafo está compuesto por un conjunto de nodos (también llamados vértices) conectados por aristas (o arcos), estas representan las relaciones o conexiones entre ellos. Los grafos pueden ser dirigidos o no dirigidos, según la direccionalidad de las aristas.

Problema de Flujo Máximo

El problema de flujo máximo consiste en encontrar la cantidad máxima de flujo que puede trasladarse desde un nodo de origen a un nodo destino a través de una red, respetando las capacidades de las aristas. Este problema es fundamental en la teoría de grafos y tiene múltiples aplicaciones en optimización y logística.

El flujo máximo busca distribuir el flujo por la red de tal forma que:

- El flujo a través de cada arista no exceda su capacidad.
- Se maximice el flujo total que llega al “sumidero” desde la “fuente”.

Conservación del Flujo

Una de las características más importantes en las redes de flujo es la conservación del flujo. Esto implica que, para cualquier nodo de la red que no sea el origen ni el destino, el flujo que entra en el nodo debe ser igual al flujo que sale. Esta propiedad garantiza que el flujo se distribuye de manera constante a lo largo de la red, sin pérdidas ni acumulaciones en los nodos intermedios.

La conservación del flujo se puede expresar mediante una ecuación que indica que la suma de los flujos entrantes en un nodo es igual a la suma de los flujos salientes. Esto asegura que el flujo se distribuye de manera eficiente entre las aristas conectadas a ese nodo.

Teorema de Conservación del Flujo

Dado un flujo **F** en una red, el flujo que sale del nodo de origen **a** es igual al flujo que llega al nodo de destino **z**. De manera matemática, ello se expresa como:

$$\sum_i F_{ai} = \sum_i F_{iz}$$

Este teorema asegura que el flujo no se pierde ni se acumula en los nodos intermedios. En términos de grafos, si tomamos un vértice **v** que no es ni el origen ni el destino, el flujo entrante es igual al flujo saliente, es decir:

$$\sum_i F_{iv} = \sum_i F_{vi}$$

Teorema del Corte Mínimo

El teorema de flujo máximo y corte mínimo establece que en cualquier red, el valor de un flujo máximo es igual a la capacidad de un corte mínimo. Esto significa que podemos demostrar que un flujo es máximo si encontramos un corte cuya capacidad coincida con el valor del flujo. Muchos de los resultados en teoría de grafos derivan directamente de este teorema. Un ejemplo de aplicación es el algoritmo de Ford-Fulkerson, que es uno de los temas que queremos analizar en esta tesina.

Capacidades de las Aristas

Cada arista en una red tiene una capacidad máxima, que limita la cantidad de flujo que puede transportar. Las capacidades son restricciones fundamentales que deben respetarse al calcular el flujo máximo. Si se excede la capacidad de una arista, se produce un "cuello de botella" en la red, lo que limita el flujo total que puede llegar al nodo destino.

El cálculo del flujo máximo implica identificar estos cuellos de botella y distribuir el flujo de

manera que se maximice el uso de las aristas con capacidad disponible.

Algoritmos de Flujo Máximo

Para resolver el problema de flujo máximo, se han desarrollado varios algoritmos eficientes. Entre ellos destacan:

Algoritmo de Ford-Fulkerson: Este es uno de los algoritmos más conocidos para encontrar el flujo máximo en una red. Se basa en la idea de encontrar rutas incrementales desde el origen al destino en las que se pueda aumentar el flujo. El algoritmo sigue encontrando caminos de aumento hasta que no sea posible incrementar más el flujo.

Algoritmo de Edmonds-Karp: Este es una implementación más específica del algoritmo de Ford-Fulkerson que utiliza búsqueda en anchura (BFS) para encontrar los caminos de aumento. Es más eficiente en redes grandes, ya que garantiza encontrar el flujo máximo en tiempo polinomial (Johnsonbaugh, 2005).

Aplicaciones

El problema del flujo máximo tiene una amplia gama de aplicaciones prácticas en diversas industrias:

Optimización de redes de tráfico: Permite calcular el máximo número de vehículos que pueden circular por una red de carreteras sin provocar atascos.

Distribución de energía: Optimiza el transporte de electricidad a través de una red de distribución, maximizando el suministro a los usuarios finales.

Redes de telecomunicaciones: Ayuda a gestionar el tráfico de datos en redes de comunicación, asegurando que la mayor cantidad posible de información fluya sin sobrecargar los enlaces.

Redes sociales o análisis de redes: En el análisis de redes sociales, maximizar el flujo de información o influencia entre diferentes nodos puede ser clave para comprender la propagación de ideas, marketing viral o detección de comunidades.

Programa

En esta entrega presentaremos un programa al 100% implementado en JavaScript que resuelve el flujo máximo. Hemos visto conveniente el uso de este lenguaje para manipular la creación del dibujo de los grafos manualmente. Para trabajar en un entorno colaborativo se ha realizado el código en GitHub, y se ha desplegado en una página web para una mayor comodidad para el usuario.

Figura 1

Red de acoplamiento arbitraria dentro del programa

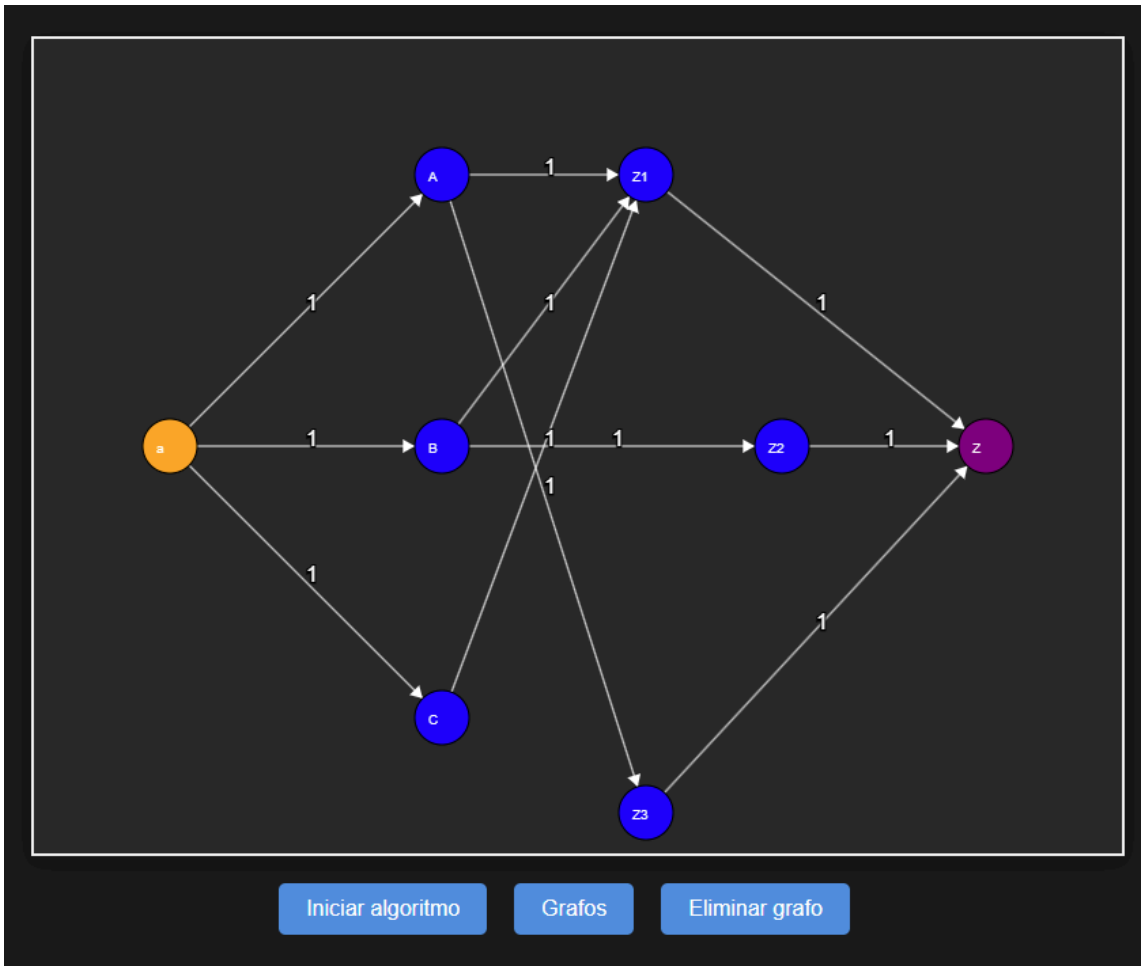
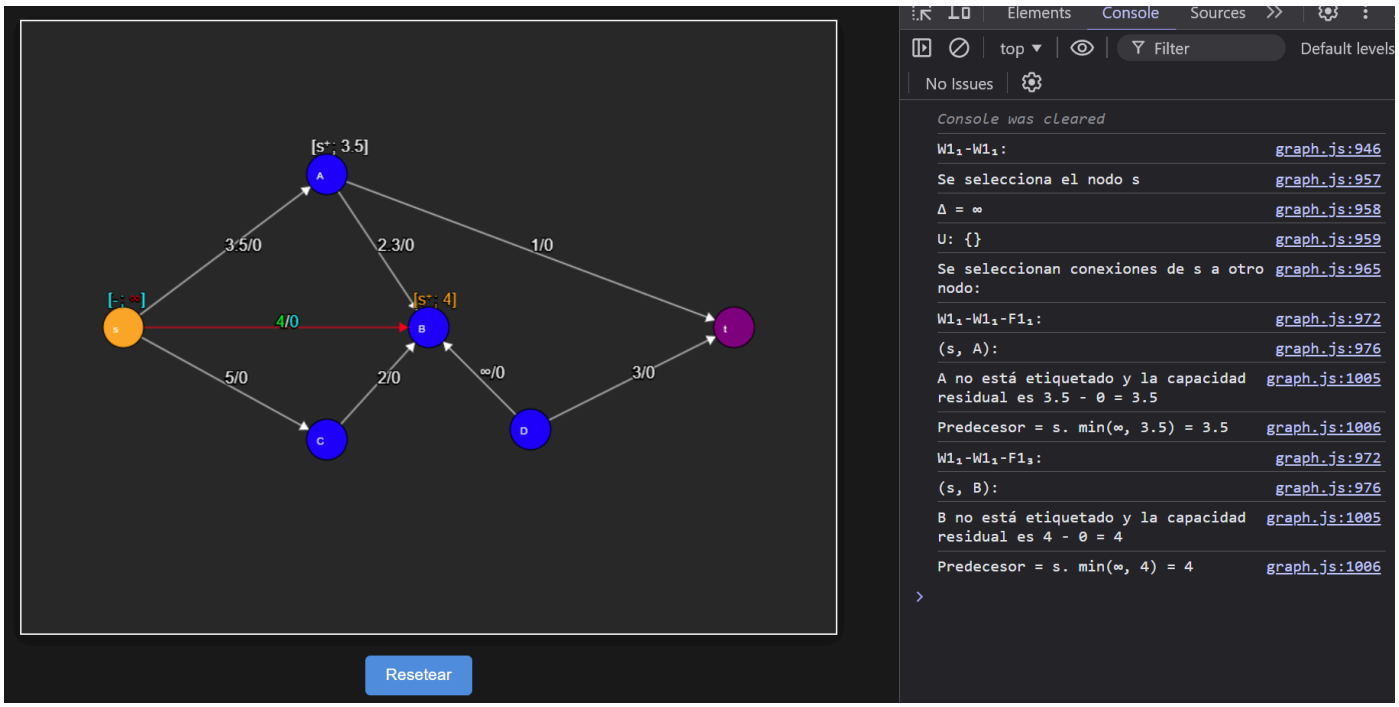


Figura 2

Proceso de etiquetado



Nota. El detalle del algoritmo se abre dándole click a la página, presionando F12 y en la opción de consola.

Figura 3

Proceso detallado de la construcción de una trayectoria

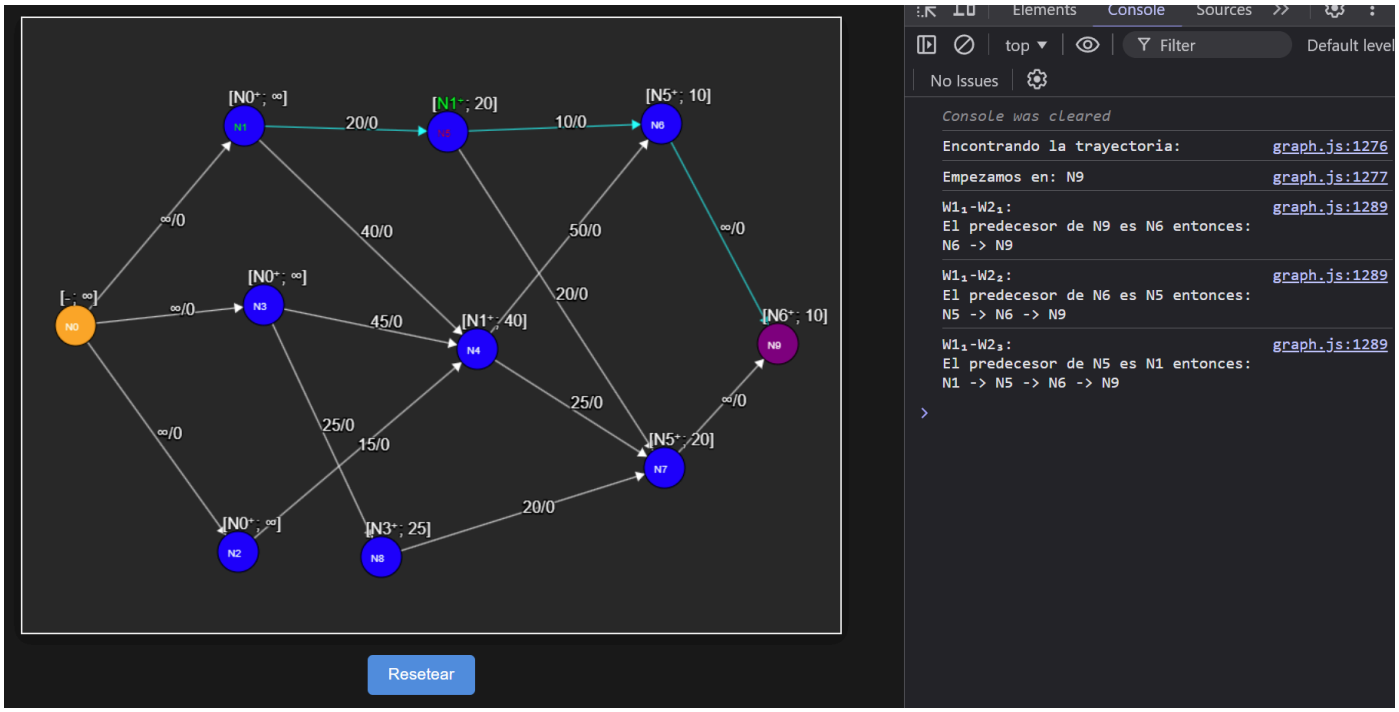


Figura 4

Proceso de incremento de flujo de una trayectoria

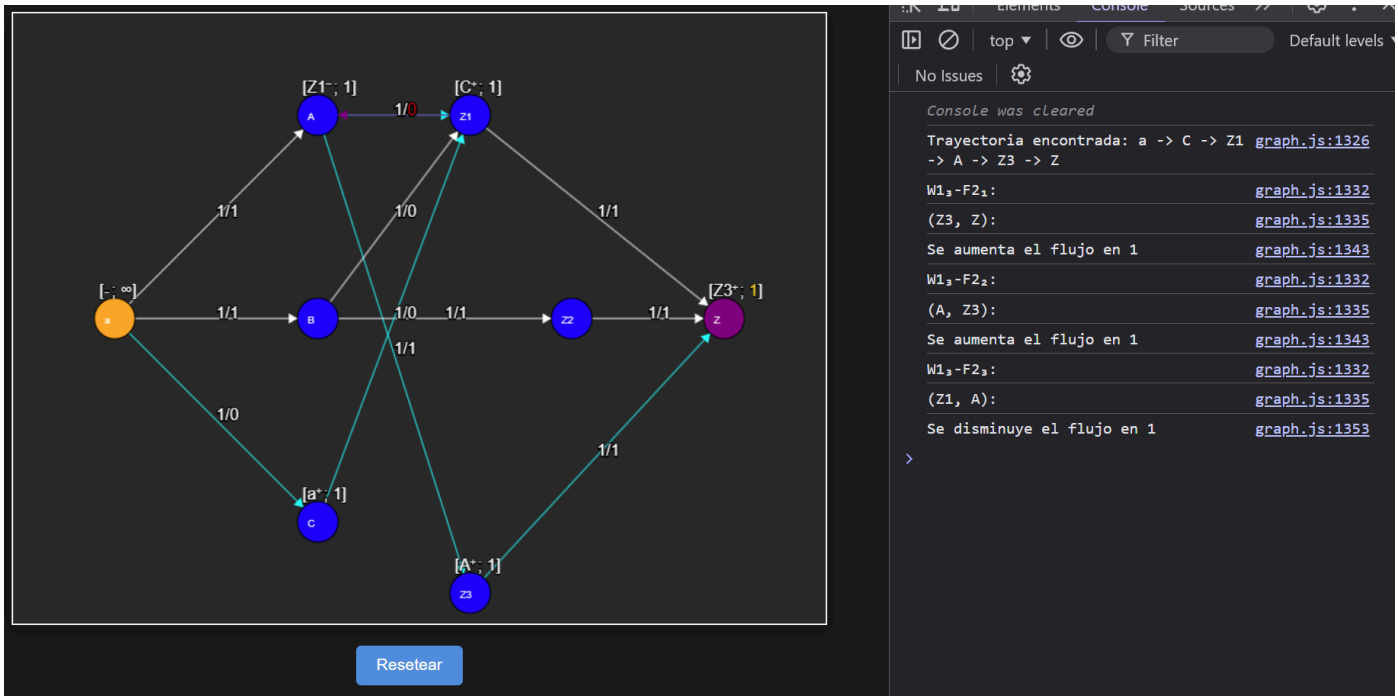
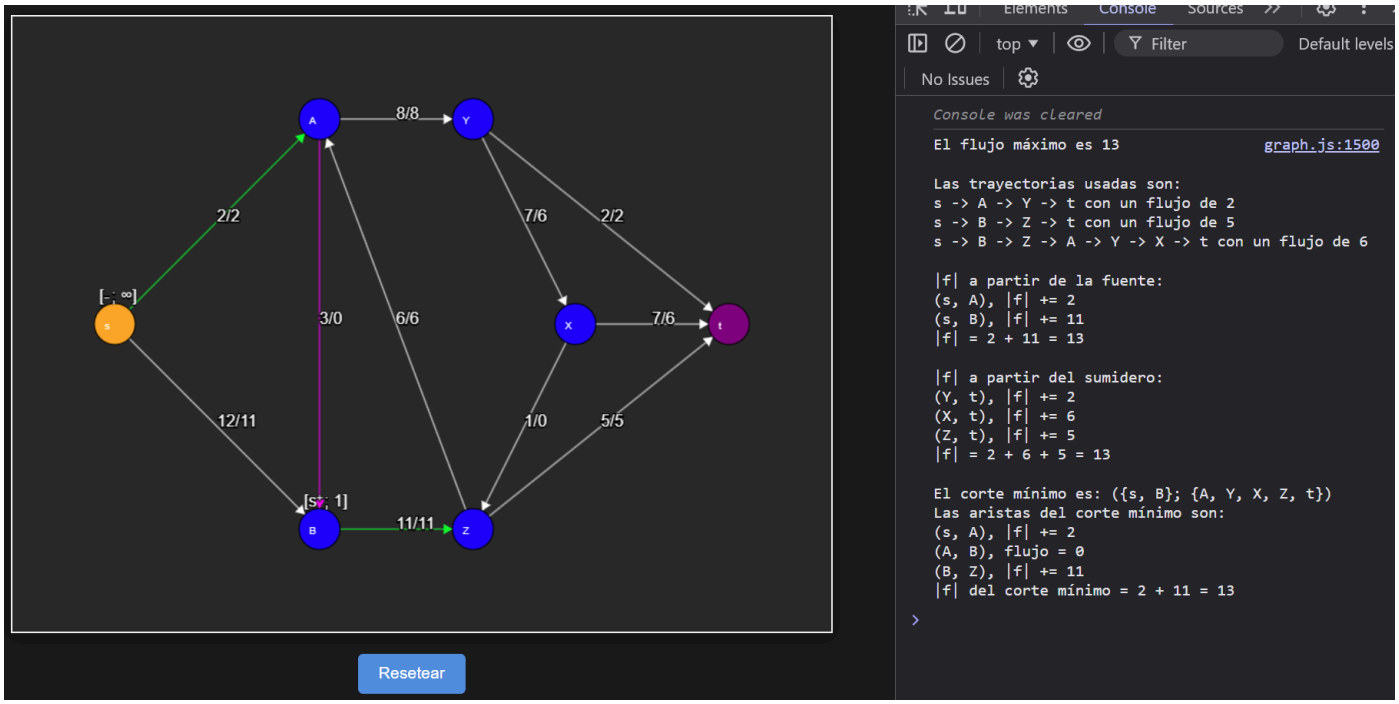


Figura 5

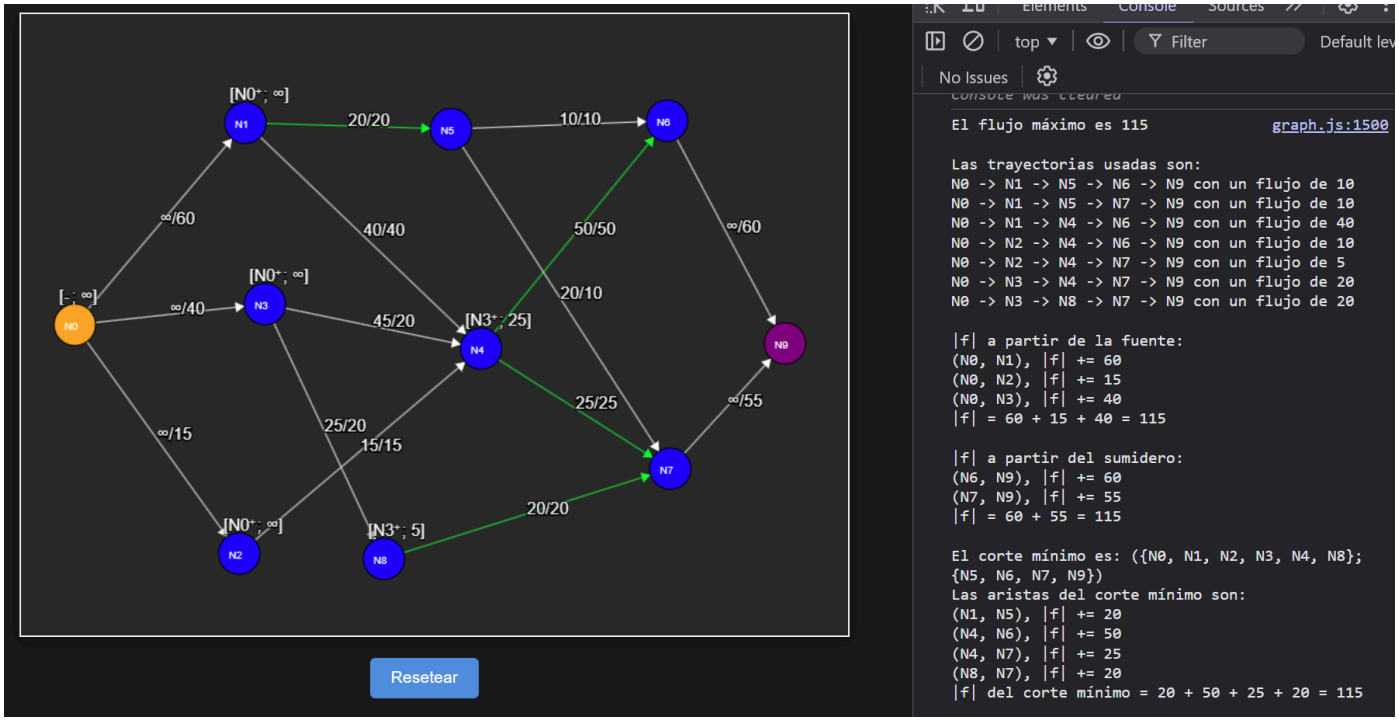
Grafo resultante con un flujo máximo y una arista inapropiada



Nota. *A* → *B* y *A* pertenece a los nodos no etiquetados, es por eso que se usa el término "inapropiada".

Figura 6

Red resultante con un superorigen y un superdestino



Conclusiones

La implementación del algoritmo de Ford-Fulkerson, y la variante de Edmonds-Karp, demostró ser adecuada para resolver problemas de flujo máximo en redes de diferentes tamaños y complejidades. La utilización de la búsqueda en anchura (BFS) para encontrar caminos de aumento ha asegurado una solución en un tiempo eficaz, lo cual es particularmente relevante en redes grandes, garantizando que se llegue a un flujo máximo de manera óptima.

Asimismo, la capacidad del algoritmo para manejar restricciones de capacidad en las aristas y respetar la conservación del flujo demuestra que la implementación ha sido precisa y rigurosa con las restricciones, maximizando la utilización de las capacidades disponibles en la red.

La herramienta desarrollada permite a los usuarios crear manualmente grafos, insertar aristas y definir capacidades, demostrando su versatilidad para adaptarse a distintas configuraciones de red. Esta característica es esencial para abordar aplicaciones reales en diversas industrias, como la optimización de redes de transporte, la distribución de energía, o la gestión de tráfico de datos en redes de telecomunicaciones.

La implementación del algoritmo mediante una interfaz gráfica consiguió una visualización clara y accesible de los grafos generados, permitiendo observar el comportamiento del flujo en cada iteración del algoritmo. La interfaz creada facilita el uso del programa incluso a personas sin conocimientos profundos en teoría de grafos, cumpliendo el objetivo de accesibilidad y simplicidad en la interacción.

Finalmente, el proyecto ha validado el aprendizaje en teoría de grafos y algoritmos de optimización, con la implementación exitosa del algoritmo de Ford-Fulkerson en JavaScript. La creación de una interfaz web a su vez ha demostrado la capacidad de aplicar conceptos teóricos a soluciones prácticas, consolidando los conocimientos matemáticos en la práctica.

Evidencia de coevaluación

Integrantes	Código	Nota de Coevaluación
Prieto Mantari, Leonardo Fabrizzio Junior	u202319949	2
Ríos Pacheco, Héctor Javier	u20231c540	2
Hermoza Quispe, Jude Alessandro	u202318220	2
Huapaya Vargas, Daniel José	u202312230	2
Palacios Valentin, Sebastian Wilfredo Yosue	u20231B463	2

Participación en Reuniones:

1. **Prieto Mantari, Leonardo Fabrizzio Junior:** La coordinación fue realizada mediante chat y una reunión final a la que asistió.
2. **Ríos Pacheco, Héctor Javier:** La coordinación fue realizada mediante chat y una reunión final a la que asistió.
3. **Hermoza Quispe, Jude Alessandro:** La coordinación fue realizada mediante chat y una reunión final a la que asistió.
4. **Huapaya Vargas, Daniel José:** La coordinación fue realizada mediante chat y una reunión final a la que asistió.
5. **Palacios Valentin, Sebastian Wilfredo Yosue:** La coordinación fue realizada mediante chat y una reunión final a la que asistió.

Calidad del Trabajo:

1. **Prieto Mantari, Leonardo Fabrizzio Junior:** Cumplió con las expectativas de lo necesario y solicitado.
2. **Ríos Pacheco, Héctor Javier:** Cumplió con las expectativas de lo necesario y solicitado.
3. **Hermoza Quispe, Jude Alessandro:** Cumplió con las expectativas de lo necesario y solicitado.
4. **Huapaya Vargas, Daniel José:** Cumplió con las expectativas de lo necesario y solicitado.
5. **Palacios Valentin, Sebastian Wilfredo Yosue:** Cumplió con las expectativas de lo necesario y solicitado.

Colaboracion y Cooperación:

1. **Prieto Mantari, Leonardo Fabrizzio Junior:** Leonardo colaboró con la resolución de conflictos y propuso soluciones efectivas durante el desarrollo del trabajo.
2. **Ríos Pacheco, Héctor Javier:** Hector estuvo para ayudar con el trabajo y el informe del grupo.
3. **Hermoza Quispe, Jude Alessandro:** Jude siempre ayudo en sus horas disponibles para colaborar con el avance del trabajo e informe.
4. **Huapaya Vargas, Daniel José:** Daniel siempre estuvo para el grupo para ayudar al equipo con el código y los requisitos de entrega
5. **Palacios Valentin, Sebastian Wilfredo Yosue:** Sebastian siempre estuvo dispuesto a ayudar a sus compañeros, facilitó la comunicación de requisitos para la entrega del trabajo y el informe.

Responsabilidad y Cumplimiento de las Tareas:

1. **Prieto Mantari, Leonardo Fabrizzio Junior:** Leonardo cumplio a tiempo con sus tareas y responsabilidades a tiempo
2. **Ríos Pacheco, Héctor Javier:** Hector cumplio a tiempo con sus tareas y responsabilidades a tiempo
3. **Hermoza Quispe, Jude Alessandro:** Daniel cumplio a tiempo con sus tareas y responsabilidades a tiempo
4. **Huapaya Vargas, Daniel José:** Daniel cumplio a tiempo con sus tareas y responsabilidades a tiempo
5. **Palacios Valentin, Sebastian Wilfredo Yosue:** Sebastian cumplio a tiempo con sus tareas y responsabilidades a tiempo

Bibliografía

El Taller De TD. (8 de enero de 2022). Algoritmo Depth First Search (DFS) para búsqueda en grafos: Explicación, ejemplos y código [Archivo de video]. YouTube. https://www.youtube.com/watch?v=iE8m53SxhDc&t=967s&ab_channel=ElTallerDeTD

Jariasf. (2014, octubre 18). *Algoritmo DFS usando Recursion en C++* [Código en C++]. GitHub. https://github.com/jariasf/Online-Judges-Solutions/blob/master/Algorithms/C%2B%2B/Graph%20Theory/Algoritmo%20DFS%20usando%20Recursion.cpp

Johnsonbaugh, R. (2005). *Matemáticas discretas* (6a ed., Cap. 10). Pearson Educación.

Ribeiro, F., et al (2014). A proposal to find the ultimate pit using Ford Fulkerson algorithm. *Rem: Revista Escola de Minas, 67(4)*, 389-395. https://www.scielo.br/j/rem/a/FD9pwRJWZsL54TPqsBSJWkK/?format=pdf&lang=en