

CASO 1: FORD-FULKERSON



Maria Paula Rodríguez Ruiz

Optimización y Simulación
Dpto. de Ingeniería de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
3 de Marzo del 2025

Objetivo: Realizar paso a paso el algoritmo Ford-Fulkerson para obtener el flujo máximo que es posible enviar por la siguiente red

Nodo Origen: 1, Nodo Destino: 15, Nodos Intermedios 2-14

Nodo de Origen	Nodo de Destino	Capacidad	Nodo de Origen	Nodo de Destino	Capacidad
1	2	35	7	10	20
1	3	14	7	11	21
1	4	25	7	12	20
1	5	16	8	11	28
1	8	21	8	12	38
2	5	20	9	10	23
2	6	36	9	13	22
2	8	26	10	11	37
3	5	18	10	13	37
3	6	27	10	14	11
3	7	24	11	14	14
4	6	11	11	15	14
4	7	39	12	13	13
4	8	25	12	15	15
5	9	38	13	14	37
5	11	18	13	15	18
6	9	23	14	15	11
6	10	11			

I. Formular el modelo matemático que permita hallar el flujo máximo de la red.

Dado un grafo dirigido $G = (V, A)$ con capacidades C_{ij} en cada arco $(i, j) \in A$ se desea maximizar el flujo total desde el nodo fuente s (nodo 1) hasta el nodo sumidero t (nodo 15).

La formulación es la siguiente:

$$\text{Maximizar: } Z = \sum_{j: (1,j) \in A} f_{1j}$$

$$\text{Sujeto a: } 0 \leq f_{ij} \leq c_{ij} \quad \forall (i, j) \in A,$$

$$\sum_{j: (i,j) \in A} f_{ij} - \sum_{j: (j,i) \in A} f_{ji} = 0 \quad \forall i \in V \setminus \{1, 15\}$$

Esta formulación garantiza que el flujo que sale del nodo fuente se maximice, respetando las capacidades de los arcos y manteniendo la conservación del flujo en los nodos intermedios.

II. Encontrar la solución óptima del modelo por Python y explicarla

Usando Python, se realizó el algoritmo completo y se verificó que el flujo máximo en la red es de 58 unidades, lo que es consistente con el teorema de flujo máximo y corte mínimo.

Solución:

<https://colab.research.google.com/drive/1TMvrOPNzRRVwqqSZb8P7NO63yKDfVfOL?usp=sharing>

Este código implementa el algoritmo Ford-Fulkerson para hallar el flujo máximo en una red, utilizando 7 estrategias distintas para encontrar caminos aumentantes. La red se define con un conjunto de nodos y arcos con capacidades específicas, y el objetivo es maximizar el flujo desde el nodo fuente (1) hasta el nodo sumidero (15).

Pasos Clave:

1. **Inicialización de la Red Residual:** Se crea una copia del grafo original, donde cada arco tiene la capacidad original y se añade el arco inverso (si no existe) con capacidad 0. Esta red residual se actualiza a medida que se envía flujo.
2. **Búsqueda del Camino Aumentante:** Se exploran caminos desde la fuente al sumidero usando diferentes estrategias que serán detalladas más adelante.
3. **Actualización de la Red y Acumulación del Flujo:** Una vez encontrado un camino aumentante, se determina el flujo posible (el mínimo de las capacidades a lo largo del camino, o cuello de botella). Se "envía" ese flujo, actualizando las capacidades en la red residual (se resta en el sentido original y se suma en el sentido inverso). Este proceso se repite hasta que ya no se encuentra ningún camino aumentante.

4. **Registro y Comparación:** Durante cada iteración se guarda un log con el camino encontrado, el flujo enviado y el estado de la red residual. Finalmente, se muestra una tabla resumen que compara el flujo total y el número de iteraciones (rutas) necesarias para cada estrategia.

III. Realización del Algoritmo Ford-Fulkerson

Previo al desarrollo del algoritmo vale la pena aclarar dos conceptos de lo que me valí para resolver el caso:

1. ¿Qué Hace el Algoritmo Ford-Fulkerson?

El **Algoritmo Ford-Fulkerson** trabaja de la siguiente manera:

1. **La red residual:** Se inicia con el grafo original y a medida que se envía flujo por un camino, se reduce la capacidad de los arcos en la dirección del flujo y se incrementa la capacidad en la dirección opuesta. Esto forma la *red residual*, que indica la capacidad restante para enviar más flujo.
2. **Caminos Aumentantes:** Se busca iterativamente un camino (llamado *camino aumentante*) desde el nodo fuente s hasta el nodo sumidero t en la red residual, donde todos los arcos tengan capacidad positiva. Ahí, se identifica el *cuello de botella* del camino, es decir, la mínima capacidad disponible a lo largo de dicho camino y se "envía" esa cantidad de flujo y se actualiza la red residual.
3. **Terminación:** El proceso se repite hasta que ya no se puede encontrar ningún camino aumentante. La suma de los flujos enviados a través de todos los caminos es el flujo máximo.

2. DFS vs BFS en la búsqueda de caminos

1. **DFS (Búsqueda en Profundidad):** En ella se explora un camino hasta lo más profundo posible antes de retroceder, y su elección del siguiente nodo depende del orden en que se exploran los vecinos, por lo que permite implementar diversas estrategias modificando el orden (menor, mayor, aleatorio).
2. **BFS (Búsqueda en Anchura):** Aquí se explora primero todos los nodos a una distancia determinada del nodo fuente, garantizando encontrar el camino más corto (en número de aristas), por lo que al ser sistemático, en algunos casos, puede reducir el número de iteraciones. Es así como decidí implementar 7 estrategias en total, agrupadas de la siguiente manera:

Estrategias Basadas en DFS

1. DFS: Menor Capacidad

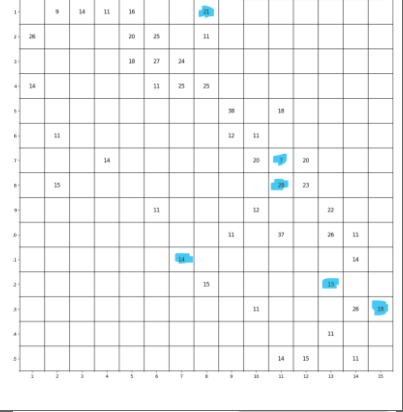
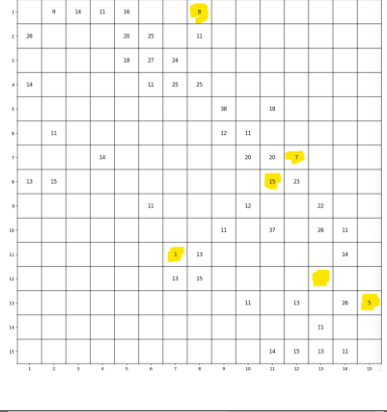
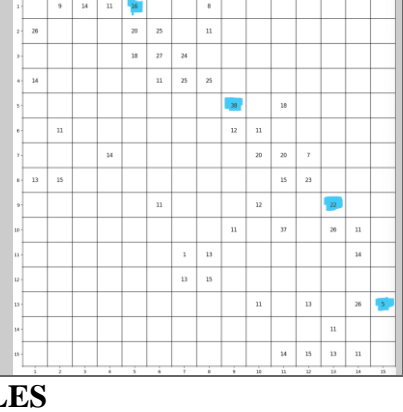
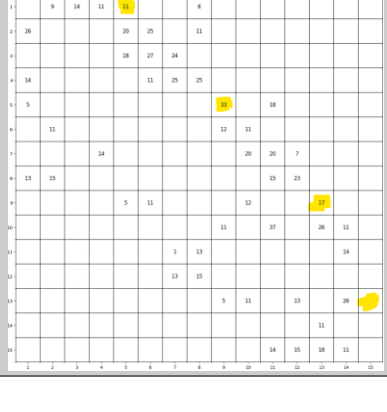
- **Descripción:**
En cada nodo, la búsqueda en profundidad (DFS) ordena a los vecinos disponibles de forma ascendente según la capacidad residual. Esto significa que se priorizan los arcos con **menor capacidad disponible**.
- **Objetivo y Justificación:**
La idea es "agotar" primero aquellos arcos que tienen menor capacidad, ya que estos suelen representar los cuellos de botella en la red. Al saturar estas rutas

tempranamente, se puede obtener una reducción en el número total de iteraciones necesarias.

○ **Ventajas y Desventajas:**

- *Ventaja:* Puede llegar rápidamente a saturar las rutas críticas y, en algunos casos, minimizar el número de pasos.
- *Desventaja:* Si el camino de menor capacidad no es el más prometedor para alcanzar el sumidero, se podrían tomar rutas subóptimas y, en redes muy heterogéneas, esta estrategia podría forzar iteraciones adicionales.

RUTA A	MATRIZ ORIGEN	MATRIZ RESIDUAL	RUTA OBTENIDA	FLUJO QUE SE ENVIÓ
1			[1, 2, 6, 9, 10, 13, 14, 15]	11
2			[1, 4, 7, 11, 15]	14
3			[1, 2, 8, 12, 15]	15

4			[1, 8, 11, 7, 12, 13, 15]	13
5			[1, 5, 9, 13, 15]	5
TOTALES			5	58

2. DFS: Mayor Capacidad

○ Descripción:

Esta variante utiliza DFS pero, en cada nodo, se ordenan los vecinos en forma descendente según la capacidad residual. Es decir, se escoge primero el arco con **mayor capacidad disponible**.

○ Objetivo y Justificación:

Se intenta maximizar el flujo enviado en cada iteración al aprovechar arcos con mucha capacidad. La expectativa es que, al enviar grandes cantidades de flujo, se pueda alcanzar el flujo máximo rápidamente.

○ Ventajas y Desventajas:

- *Ventaja:* Potencial para enviar flujos grandes en cada ruta.
- *Desventaja:* Aunque se envíe mucho flujo en etapas iniciales, puede ocurrir que más adelante se encuentre un cuello de botella que obligue a realizar iteraciones adicionales, aumentando el número total de rutas.

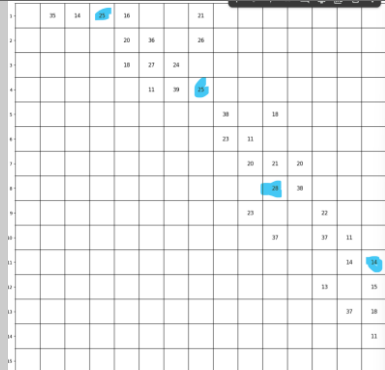
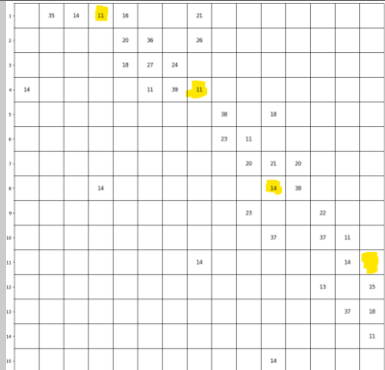
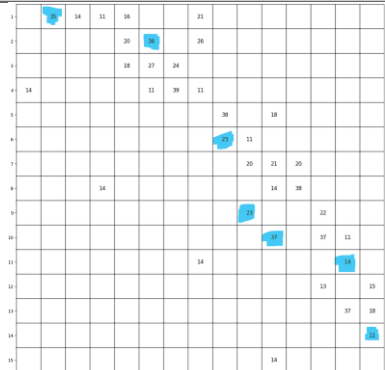
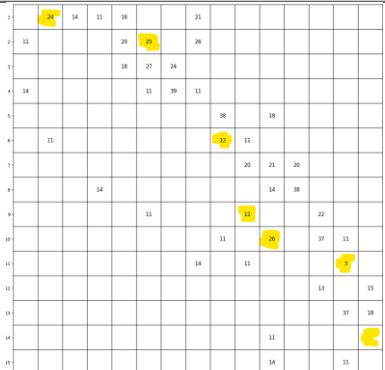
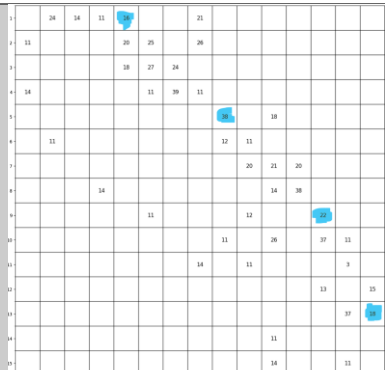
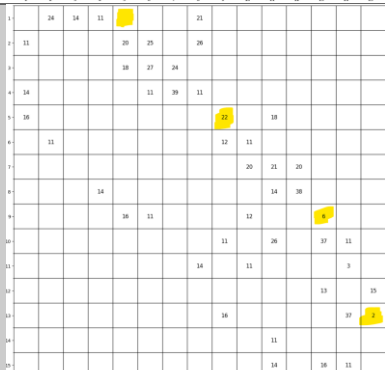
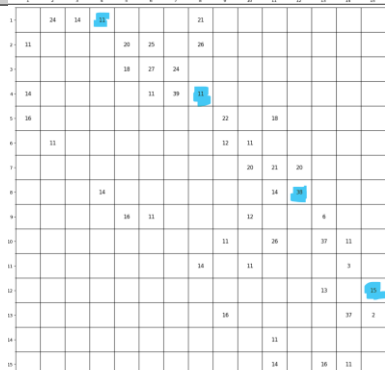
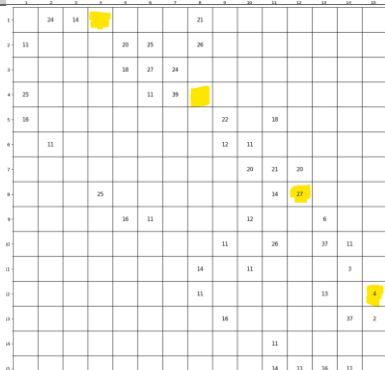
RUT A	MATRIZ ORIGEN	MATRIZ RESIDUAL	RUTA OBTENIDA	FLUJO QUE SE ENVIÓ
1			[1, 3, 5, 11, 15]	14
2			[1, 5, 11, 14, 15]	4
3			[1, 5, 3, 7, 12, 13, 15]	12
4			[1, 8, 11, 14, 15]	7

5			[1, 8, 11, 5, 3, 7, 12, 13, 15]	1
6			[1, 8, 11, 5, 3, 7, 12, 15]	1
7			[1, 8, 11, 5, 9, 13, 15]	5
8			[1, 8, 11, 5, 9, 13, 12, 15]	7

9		[1, 4, 6, 10, 14, 11, 5, 9, 13, 12, 15]	4
10		[1, 4, 6, 10, 14, 11, 8, 12, 15]	3
TOTALES		10	58

3. DFS: Aleatorio

- **Descripción:**
En esta estrategia, la DFS selecciona de forma aleatoria entre los vecinos disponibles, sin seguir un orden específico basado en la capacidad.
- **Objetivo y Justificación:**
La aleatoriedad puede ayudar a evitar caer sistemáticamente en caminos subóptimos. Dado que la red puede presentar múltiples rutas con características similares, la selección aleatoria permite explorar diversas posibilidades.
- **Ventajas y Desventajas:**
 - *Ventaja:* Puede descubrir rutas alternativas y evitar estancamientos en decisiones repetitivas.
 - *Desventaja:* La falta de un criterio fijo puede llevar a resultados variables de una ejecución a otra, lo que puede implicar un mayor número de iteraciones en algunos casos.

RUTA A	MATRIZ ORIGEN	MATRIZ RESIDUAL	RUTA OBTENIDA	FLUJO QUE SE ENVIÓ
1			[1, 4, 8, 11, 15]	14
2			[1, 2, 6, 9, 10, 11, 14, 15]	11
3			[1, 5, 9, 13, 15]	16
4			[1, 4, 8, 12, 15]	11

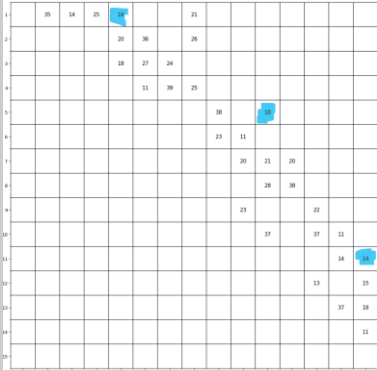
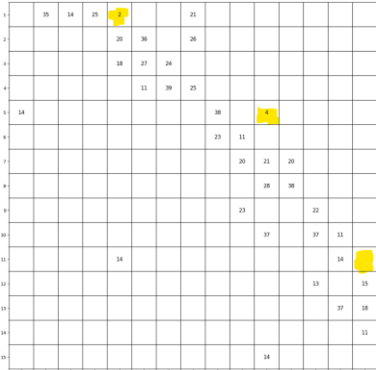
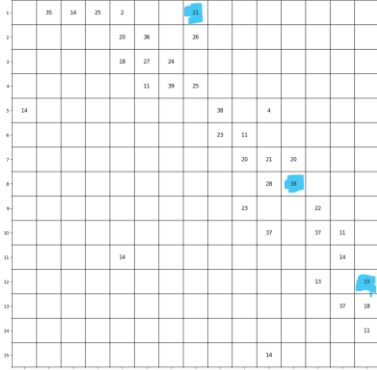
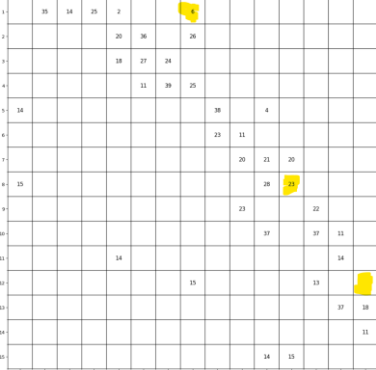
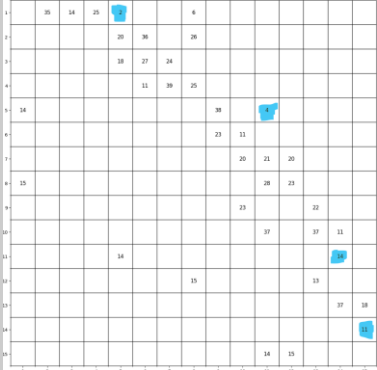
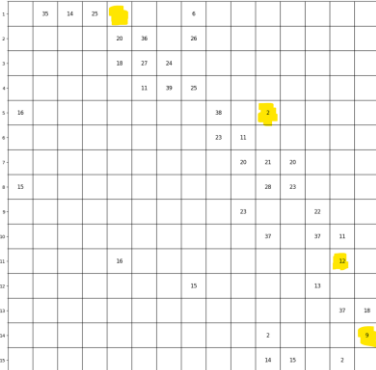
5			[1, 8, 11, 10, 13, 15]	2
6			[1, 3, 6, 9, 5, 11, 8, 4, 7, 12, 15]	4
TOTALES			6	58

Estrategias Basadas en BFS Ordenado

Las estrategias BFS aquí se han modificado para ordenar a los vecinos por capacidad, similar a lo que se hace en DFS, pero manteniendo la naturaleza de búsqueda en anchura (BFS), que garantiza la exploración de caminos con el menor número de aristas.

4. BFS: Ordenado (Menor Capacidad)

- **Descripción:**
En cada nivel de la búsqueda, se ordenan los vecinos disponibles en forma ascendente según la capacidad residual, de manera que se prioricen aquellos con menor capacidad.
- **Objetivo y Justificación:**
Al igual que en DFS: Menor Capacidad, se intenta saturar los arcos críticos (cuellos de botella) que pueden limitar el flujo.
- **Ventajas y Desventajas:**
 - *Ventaja:* Al buscar el camino más corto en términos de número de aristas, esta estrategia puede reducir la cantidad de rutas totales.
 - *Desventaja:* El ordenamiento ascendente puede no siempre favorecer la acumulación de grandes flujos si la ruta más corta tiene, en ocasiones, capacidades muy bajas.

RUT A	MATRIZ ORIGEN	MATRIZ RESIDUAL	RUTA OBTENID A	FLUJ O QUE SE ENVI O
1			[1, 5, 11, 15]	14
2			[1, 8, 12, 15]	15
3			[1, 5, 11, 14, 15]	2

4			[1, 8, 12, 13, 15]	6
5			[1, 3, 5, 11, 14, 15]	2
6			[1, 3, 5, 9, 13, 15]	12
7			[1, 4, 6, 10, 14, 15]	7
TOTALES			7	58

5. BFS: Ordenado (Mayor Capacidad)

- **Descripción:**
En este caso, la búsqueda en anchura ordena a los vecinos de forma descendente, priorizando aquellos con **mayor capacidad residual**.
- **Objetivo y Justificación:**
Se busca aprovechar las rutas que permitan enviar grandes volúmenes de flujo, combinando la ventaja del camino corto (por BFS) con la capacidad alta en cada arco.
- **Ventajas y Desventajas:**
 - *Ventaja:* Puede lograr enviar grandes cantidades de flujo en rutas que son, al mismo tiempo, cortas en número de aristas.
 - *Desventaja:* Dependiendo de la estructura de la red, esta estrategia podría omitir caminos que, aunque tengan menor capacidad en algunos arcos, en conjunto ofrezcan mejores oportunidades para aumentar el flujo global.

RUTA A	MATRIZ ORIGEN	MATRIZ RESIDUAL	RUTA OBTENIDA	FLUJO QUE SE ENVÍO
1			[1, 8, 12, 15]	15
2			[1, 5, 11, 15]	14

3		[1, 8, 11, 14, 15]	6
4		[1, 5, 9, 13, 15]	2
5		[1, 2, 6, 9, 13, 15]	16
6		[1, 4, 7, 11, 14, 15]	5
TOTALES		6	58

6. BFS: Ordenado (Aleatorio)

○ Descripción:

Aquí, en cada nivel de BFS, los vecinos se ordenan de forma aleatoria.

○ Objetivo y Justificación:

Introduce variabilidad en la exploración de la red, permitiendo que se descubran rutas que podrían ser pasadas por alto con un ordenamiento fijo.

○ Ventajas y Desventajas:

- *Ventaja:* Puede ofrecer resultados diferentes en cada ejecución, lo cual en algunos escenarios permite hallar rápidamente caminos eficientes.
- *Desventaja:* La aleatoriedad puede conducir a una mayor variabilidad en el número de iteraciones y, en algunos casos, a un mayor número de rutas requeridas.

RUT A	MATRIZ ORIGEN	MATRIZ RESIDUAL	RUTA OBTENIDA	FLUJO QUE SE ENVIÓ
1			[1, 8, 11, 15]	14
2			[1, 8, 12, 15]	7

3		[1, 4, 8, 12, 15]	8
4		[1, 5, 11, 14, 15]	11
5		[1, 5, 9, 13, 15]	5
6		[1, 4, 8, 12, 13, 15]	13
TOTALES		6	58

Estrategia Heurística:**7. Best-First Search**○ **Descripción:**

Esta estrategia combina dos criterios clave para evaluar un camino aumentante en la red residual:

▪ **Capacidad del Cuello de Botella:**

Se refiere a la menor capacidad disponible a lo largo de un camino. Un valor alto indica que el camino permite enviar una gran cantidad de flujo.

▪ **Longitud del Camino:**

Se mide en número de aristas. Un camino más corto es preferible, ya que generalmente implica menos iteraciones y menor complejidad al actualizar la red residual.

La idea es asignar a cada camino un puntaje que penalice caminos largos y premie aquellos que permiten enviar mayor flujo.

La fórmula utilizada es: $Puntaje = \alpha \times (capacidad\ del\ cuello\ de\ botella) - \beta \times (longitud\ del\ camino - 1)$, con $\alpha = 1$ y $\beta = 1$

○ **¿Cómo Funciona en la Práctica?**▪ **Evaluación de Caminos:**

Para cada camino candidato que se va construyendo, se calcula el cuello de botella (el mínimo de las capacidades de todos sus arcos) y se toma en cuenta la cantidad de aristas que componen el camino.

▪ **Priorización:**

Se utiliza una cola de prioridad (simulada mediante un heap) para almacenar los caminos candidatos. Cada entrada de la cola incluye:

1. El *puntaje* del camino (almacenado de forma negativa para simular un max-heap).
2. El camino en sí (lista de nodos).
3. El cuello de botella del camino.

▪ De esta forma, el camino con el puntaje más alto (es decir, el que maximiza el flujo potencial y minimiza la longitud) se extrae primero.

▪ **Expansión del Camino:**

A partir del camino extraído, se expande explorando los vecinos del último nodo en el camino. Para cada vecino con capacidad disponible, se calcula el nuevo cuello de botella (mínimo entre el cuello actual y la capacidad del arco hacia el vecino) y se forma un nuevo camino extendido. Se vuelve a calcular el puntaje para este nuevo camino y se añade a la cola de prioridad.

▪ **Retroceso Implícito:**

Si un camino extendido no conduce eventualmente al sumidero, el heap garantiza que se explorarán otras alternativas con puntajes más altos. De este modo, se evita quedar atrapado en caminos largos o subóptimos, ya que el puntaje penaliza la longitud del camino.

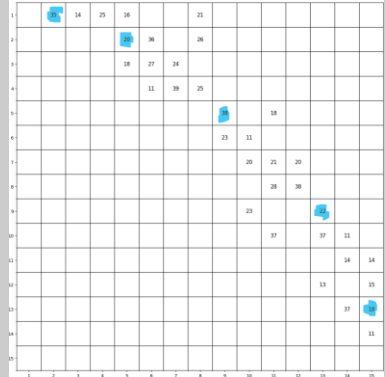
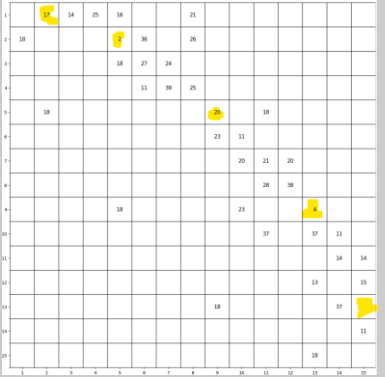
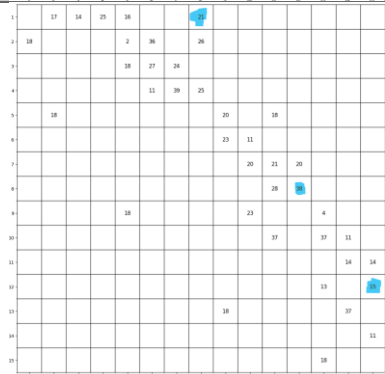
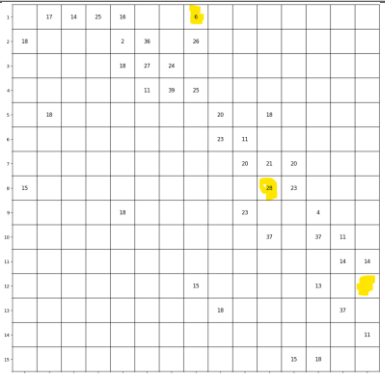
▪ **Selección del Camino Aumentante:**

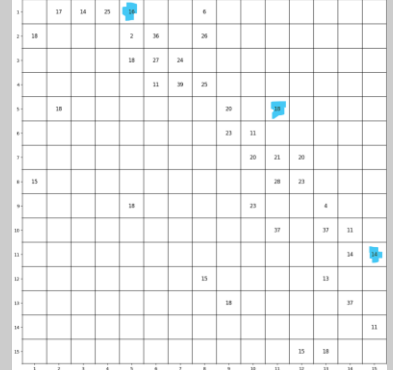
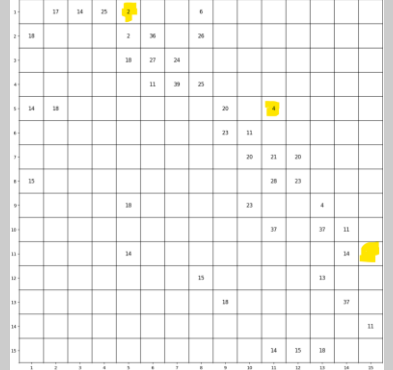
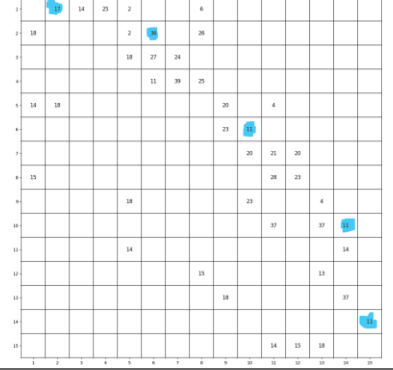
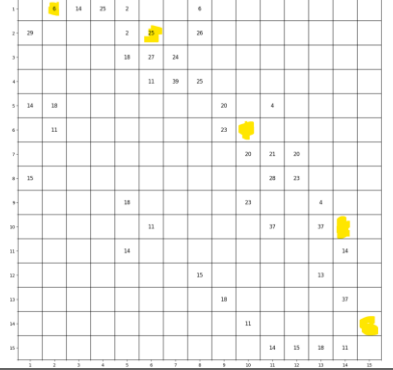
El proceso se repite hasta que se extrae de la cola un camino que llegue al

nodo sumidero. Este camino se utiliza para enviar flujo, se actualiza la red residual y se registra la iteración.

○ **Ventajas y Desventajas:**

- *Ventaja:* Esta estrategia no se enfoca únicamente en maximizar el flujo (como la estrategia de mayor capacidad) ni únicamente en minimizar la cantidad de aristas (como una búsqueda estrictamente en anchura). En su lugar, busca un balance que permita enviar una buena cantidad de flujo en caminos relativamente cortos.
- *Ventaja y Desventaja:* Los parámetros α y β se pueden ajustar según la topología de la red. Si se incrementa α en relación a β , se priorizará enviar mayor flujo, incluso si el camino es un poco más largo. Si se incrementa β , se penalizarán fuertemente los caminos largos, favoreciendo rutas más directas. Por lo que, la eficacia de la estrategia depende del ajuste adecuado de estos parámetros.

RUT A	MATRIZ ORIGEN	MATRIZ RESIDUAL	RUTA OBTENID A	FLUJ O QUE SE ENVI O
1			[1, 2, 5, 9, 13, 15]	18
2			[1, 8, 12, 15]	15

3			[1, 5, 11, 15]	14
4			[1, 2, 6, 10, 14, 15]	11
TOTALES			4	58

IV. Resultados y Conclusiones

ESTRATEGIA	FLUJO TOTAL OBTENIDO	CANTIDAD DE RUTAS OBTENIDAS
1 - DFS: MENOR CAPACIDAD	58	5
2 - DFS: MAYOR CAPACIDAD	58	10
3 - DFS: ALEATORIO	58	6
4 - BFS: MENOR CAPACIDAD	58	7
5 - BFS: MAYOR CAPACIDAD	58	6
6 - BFS: ALEATORIO	58	6
7- BEST-FIRST SEARCH	58	4

A partir del análisis de la tabla mostrada previamente se puede decir que:

- **Flujo Máximo:** Todas las estrategias alcanzan el flujo máximo de 58, lo que es coherente con el teorema de flujo máximo y corte mínimo
- **Eficiencia en Iteraciones:** La estrategia heurística (Best-First Search) es la más eficiente, ya que logró el flujo máximo en solo 4 iteraciones.

- **Comparación DFS vs. BFS:**
 - Las estrategias DFS pueden ser muy sensibles al orden de exploración y, en algunos casos, requieren más iteraciones.
 - Las estrategias BFS, al buscar el camino más corto en términos de aristas, suelen reducir el número de iteraciones, aunque la elección del orden (menor, mayor o aleatorio) también afecta el desempeño.
- **Estrategia Heurística:** Al incorporar tanto la capacidad del cuello de botella como la longitud del camino, se obtiene un método más equilibrado. Ajustando α y β se puede adaptar la estrategia a distintas topologías de red, lo que en este caso permitió obtener el mejor rendimiento en términos de rutas.

En conclusión, el algoritmo Ford-Fulkerson, al aplicarse con diferentes estrategias de búsqueda, siempre alcanza el flujo máximo (58 unidades en este caso), pero la eficiencia (número de iteraciones) varía considerablemente. La estrategia heurística Best-First Search resulta ser la más prometedora para este ejemplo, lo que sugiere que es una opción a considerar en redes más complejas.