

UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD  
DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA,  
ELECTRÓNICA, INFORMÁTICA Y MECÁNICA  
INGENIERÍA INFORMÁTICA Y DE SISTEMAS



---

## Laboratorio 2 - Edmonds-Karp

---

### Integrantes:

Ciro Gabriel Callapiña Castilla  
134403

Jhon Esau Pumachoque Choquenaira  
210940

Ian Logan Will Quispe Ventura  
211359

Luis Manuel Tinoco Ccoto  
204807

Jorge Enrique Zegarra Rojas  
161534

### Docente:

Lauro Enciso Rodas

### Curso:

Algoritmos Avanzados

Cusco - Perú  
2024

# 1 Guía de usuario

Ejecutar el archivo con extensión .exe en windows, no es necesario instalar ninguna dependencia.

## 1.1 Requisitos previos

Al ser un programa con la extensión '.exe' no es necesario ningun requisito para su ejecución, más que ejecutarlo en un sistema Windows de preferencia Windows 10 como mínimo.

## 1.2 Uso básico

Ejecutar el programa haciendo doble click en el archivo 'edmondsKarp.exe' o también desde la consola de Windows (cmd), posicionandonos en la carpeta donde se encuentra el programa, con ayuda del comando cd (change directory), para ejecutar el programa simplemente escribiremos ./edmondsKarp.exe.

## 1.3 Características principales

- La aplicación implementa el algoritmo de Edmonds-Karp que es una variante del algoritmo de Ford-Fulkerson para calcular el flujo máximo en una red de flujo. La principal diferencia entre estos dos algoritmos radica en la elección de la búsqueda de camino aumentante.
- El usuario puede indicar sus datos según sus preferencias.

## 1.4 Ejemplo de uso

1. Completa los campos propocionados en la interfáz gráfica, que son:
  - Número de nodos
  - Nodo fuente
  - Nodo sumidero
  - Aristas (u v capacidad), debe ser ingresado con un espacio entre los números y dar Enter despues de cada arista
2. Haga clic en "Calcular flujo Máximo" para ejecutar el algortimo de Edmonds-Karp
3. Enseguida puede hacer clic en el boton "Mostrar Grafo" para ver la implementación gráfica de este algoritmo.

# 2 Implementación

## 2.1 Implementación del algoritmo

Este programa implementa el algoritmo de Edmonds-Karp para calcular el flujo máximo en una red de flujo. Veremos que hace cada parte del código.

- La clase edge Define la estructura de un arco en el grafo, con el nodo de destino (to), la capacidad (capacity) y el flujo (flow).
- La Función addEdge agrega una arista dirigida al grafo, especificando el nodo de origen, el nodo de destino y la capacidad de la arista.
- La Función bfs realiza una búsqueda de camino de aumento utilizando el algoritmo Breadth-First Search (BFS) para encontrar un camino desde el nodo fuente hasta el nodo sumidero.
- La Función edmondsKarp implementa el algoritmo de Edmonds-Karp para calcular el flujo máximo en la red de flujo. Utiliza BFS para encontrar caminos de aumento y actualizar el flujo en cada iteración hasta que ya no sea posible encontrar más caminos de aumento.

La implementación del código a continuación.

Listing 1: Código de Python para el algoritmo de Edmonds-Karp

---

```

1 from collections import deque
2
3 class Edge:
4     def __init__(self, to, capacity, flow):
5         self.to = to
6         self.capacity = capacity
7         self.flow = flow
8
9 def addEdge(graph, u, v, capacity):
10     graph[u].append(Edge(v, capacity, 0))
11
12 def bfs(graph, parent, source, sink):
13     n = len(graph)
14     visited = [False] * n
15     queue = deque()
16     queue.append(source)
17     visited[source] = True
18
19     while queue:
20         u = queue.popleft()
21
22         for e in graph[u]:
23             v = e.to
24             if not visited[v] and e.capacity - e.flow > 0:
25                 parent[v] = u
26                 visited[v] = True
27                 queue.append(v)
28
29     return visited[sink]
30
31 def edmondsKarp(graph, source, sink):
32     n = len(graph)
33     parent = [-1] * n
34     maxFlow = 0
35
36     while bfs(graph, parent, source, sink):
37         pathFlow = float('inf')
38         v = sink
39         while v != source:
40             u = parent[v]
41             for e in graph[u]:
42                 if e.to == v:
43                     pathFlow = min(pathFlow, e.capacity - e.flow)
44                     break
45             v = parent[v]
46
47         v = sink
48         while v != source:
49             u = parent[v]
50             for e in graph[u]:
51                 if e.to == v:
52                     e.flow += pathFlow
53                     for backEdge in graph[v]:
54                         if backEdge.to == u:
55                             backEdge.flow -= pathFlow
56                             break
57                     break
58             v = parent[v]
59
60         maxFlow += pathFlow
61
62     return maxFlow

```

---

## 2.2 Implementación de la interfaz gráfica

La interfaz gráfica tiene una ventana principal donde se alojan los campos de datos donde el usuario podrá indicar el número de nodos, el nodo fuente, el nodo sumidero, y las aristas necesarias. Ya con estos datos ingresados se puede ejecutar el algoritmo para determinar el flujo máximo y también se puede ver el grafo generado en base a los datos dados por el usuario. Esta interfaz es intuitiva y de fácil uso, para que la interacción con el algoritmo Edmonds-Karp sea la más sencilla posible. El código a continuación.

Listing 2: Código de la implementación de la interfaz gráfica

```
1 import tkinter as tk
2 from tkinter import messagebox
3 from edmonds_karp import Edge, addEdge, edmondsKarp
4 import networkx as nx
5 import matplotlib.pyplot as plt
6 from collections import deque
7
8 def clear_interface():
9     for widget in graph_frame.winfo_children():
10         widget.destroy()
11
12 def calculate_max_flow():
13     try:
14         numNodes = int(numNodes_entry.get())
15         source = int(source_entry.get())
16         sink = int(sink_entry.get())
17
18         graph = [[] for _ in range(numNodes)]
19
20         edges_info = edges_entry.get("1.0", tk.END).strip().split("\n")
21         for edge_info in edges_info:
22             u, v, capacity = map(int, edge_info.split())
23             addEdge(graph, u, v, capacity)
24
25         maxFlow = edmondsKarp(graph, source, sink)
26         messagebox.showinfo("Resultado", f"Flujo máximo calculado: {maxFlow}")
27
28         global G
29         G = nx.DiGraph()
30         for i in range(numNodes):
31             G.add_node(i)
32
33         for u in range(numNodes):
34             for edge in graph[u]:
35                 G.add_edge(u, edge.to, capacity=edge.capacity, flow=edge.flow)
36
37     except Exception as e:
38         messagebox.showerror("Error", f"Ocurrió un error: {e}")
39
40 def show_graph():
41     try:
42         pos = nx.spring_layout(G)
43         nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size
44                 =2000, arrows=True)
45
46         edge_labels = {(u, v): f"{d['flow']}/{d['capacity']}" for (u, v, d)
47                         in G.edges(data=True)}
48         nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
49
50         plt.title("Grafo con Flujo Máximo")
51         plt.show()
52     except NameError:
```

```

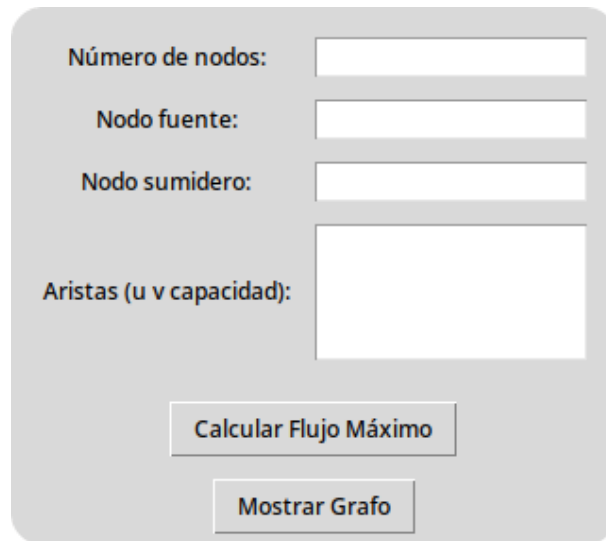
51         messagebox.showerror("Error", "Primero calcula el flujo máximo")
52
53     root = tk.Tk()
54     root.title("Calculculo de Flujo Máximo")
55
56     graph_frame = tk.Frame(root)
57     graph_frame.pack(padx=10, pady=10)
58
59     numNodes_label = tk.Label(graph_frame, text="Número de nodos:")
60     numNodes_label.grid(row=0, column=0, padx=5, pady=5)
61     numNodes_entry = tk.Entry(graph_frame)
62     numNodes_entry.grid(row=0, column=1, padx=5, pady=5)
63
64     source_label = tk.Label(graph_frame, text="Nodo fuente:")
65     source_label.grid(row=1, column=0, padx=5, pady=5)
66     source_entry = tk.Entry(graph_frame)
67     source_entry.grid(row=1, column=1, padx=5, pady=5)
68
69     sink_label = tk.Label(graph_frame, text="Nodo sumidero:")
70     sink_label.grid(row=2, column=0, padx=5, pady=5)
71     sink_entry = tk.Entry(graph_frame)
72     sink_entry.grid(row=2, column=1, padx=5, pady=5)
73
74     edges_label = tk.Label(graph_frame, text="Aristas (u v capacidad):")
75     edges_label.grid(row=3, column=0, padx=5, pady=5)
76     edges_entry = tk.Text(graph_frame, height=4, width=20)
77     edges_entry.grid(row=3, column=1, padx=5, pady=5)
78
79     calculate_button = tk.Button(root, text="Calcular Flujo Máximo", command=
        calculate_max_flow)
80     calculate_button.pack(padx=10, pady=5)
81
82     show_graph_button = tk.Button(root, text="Mostrar Grafo", command=show_graph)
83     show_graph_button.pack(padx=10, pady=5)
84
85     root.mainloop()

```

---

### 3 Ejemplo de uso

Al ejecutar el programa visualizaremos una ventana como esta:



Número de nodos:

Nodo fuente:

Nodo sumidero:

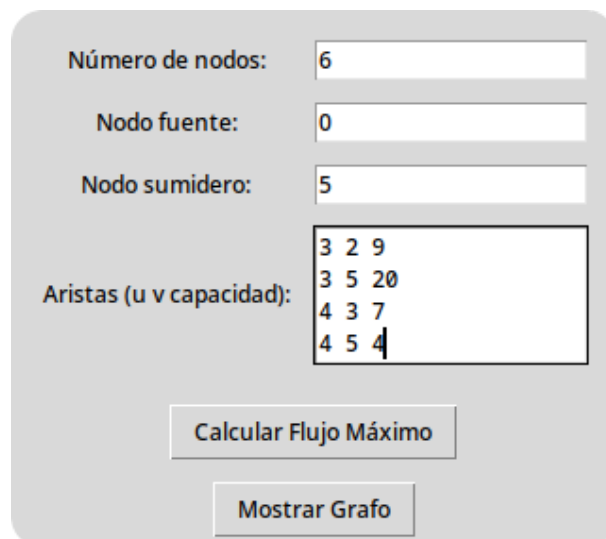
Aristas (u v capacidad):

Calcular Flujo Máximo

Mostrar Grafo

Figure 1: Ventana inicial

Donde ingresaremos los datos del número de nodos, el nodo fuente, el nodo sumidero y las aristas, que deben ser ingresadas en base a "u", "v" y "capacidad" con un espacio entre cada uno de estos datos y dar un Enter para saltar a la siguiente línea donde ingresaremos la siguiente arista. Aquí un ejemplo (Datos sacados de la figura 26.5 del libro de "Introduction to Algorithms por Thomas H. Cormen 2da. Edición.):



Número de nodos: 6

Nodo fuente: 0

Nodo sumidero: 5

Aristas (u v capacidad):  
3 2 9  
3 5 20  
4 3 7  
4 5 4

Calcular Flujo Máximo

Mostrar Grafo

Figure 2: Datos de ejemplo

Una vez ingresados los datos, podemos dar click en "Calcular FLujo Máximo" y obtendremos:

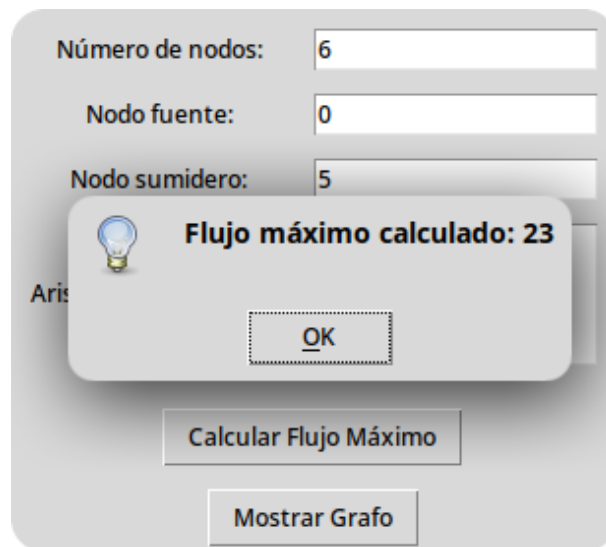


Figure 3: Flujo máximo

Para ver el grafo generado damos click en "Mostrar Grafo", se habrá otra ventana con el gráfico deacuerdo a los datos que porporcionamos.

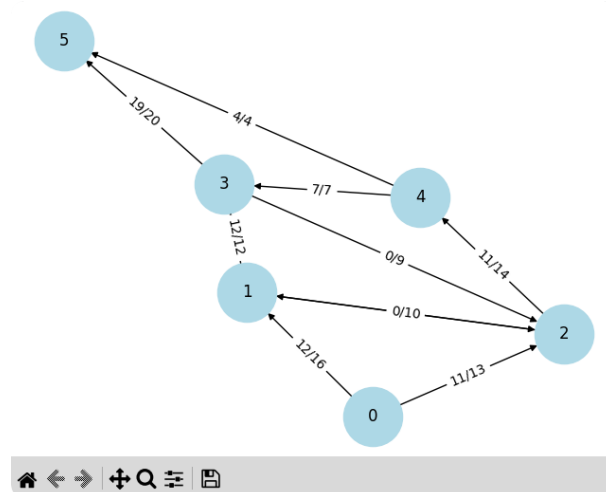


Figure 4: Grafo generado

En ele grafo se observan los nodos indicados por el usuario además del fujo actual y la capacidad máxima de cada arista.

Este programa usa el algoritmo de Edmonds-Karp para calcular la cantidad máxima de flujo que puede pasar desde el nodo fuente al nodo sumidero, respetando las capacidades de los arcos y cumpliendo con la conservación del flujo en los nodos intermedios.