A continuación se presentan los resultados de ejecución con cada archivo de entrada para cada uno de los 3 algoritmos.

distances5.txt:

```
C:\Users\juand\OneDrive\Desktop\UNIANDES\2025-20\DALGO\Tareas\Tarea 4\Tarea4-Dalgo\Parte 1>python parte_1.py
Algoritmo: Dijkstra
 Tiempo total de ejecución: 0.000000 ms
Algoritmo: Bellman-Ford
 Tiempo total de ejecución: 1.001120 ms
Algoritmo: Floyd-Warshall
 Tiempo total de ejecución: 0.000000 ms
```

distances100.txt:

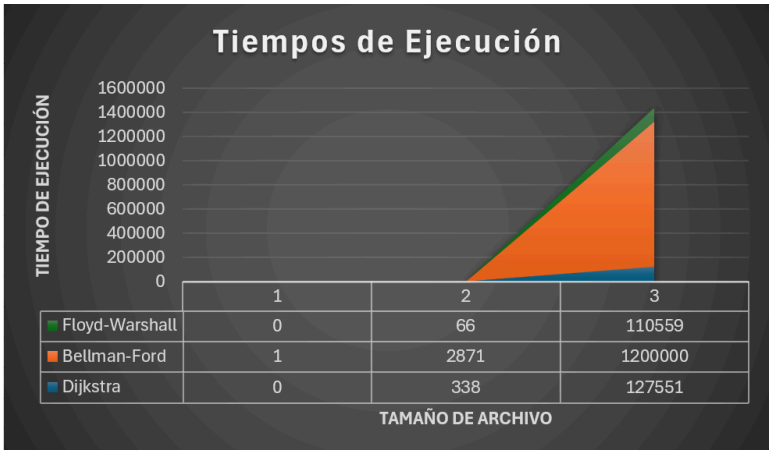
```
C:\Users\juand\OneDrive\Desktop\UNIANDES\2025-20\DALGO\Tareas\Tarea 4\Tarea4-Dalgo\Parte 1>python parte_1.py
Algoritmo: Dijkstra
 Tiempo total de ejecución: 338.149309 ms
Algoritmo: Bellman-Ford
 Tiempo total de ejecución: 2871.866703 ms
Algoritmo: Floyd-Warshall
 Tiempo total de ejecución: 66.768646 ms
```

distances1000.txt:

```
C:\Users\juand\OneDrive\Desktop\UNIANDES\2025-20\DALGO\Tareas\Tarea 4\Tarea4-Dalgo\Parte 1>python parte_1.py
Algoritmo: Dijkstra
 Tiempo total de ejecución: 127551.762581 ms
```

```
C:\Users\juand\OneDrive\Desktop\UNIANDES\2025-20\DALGO\Tareas\Tarea 4\Tarea4-Dalgo\Parte 1>python parte_1.py
Algoritmo: Floyd-Warshall
 Tiempo total de ejecución: 110559.099436 ms
```

Análisis de resultados



Como se puede ver en la gráfica de resultados, el algoritmo de recorrido de grafos que más tiempo tardó en ejecutarse fue **Bellman-Ford**, de hecho no terminó de procesar el archivo **"distances1000.txt"**, solo se sabe que duró más de 20 minutos (1200000 ms) que fue el tiempo que se le dió para que arrojara un resultado. Con respecto a **"Dijkstra"** y **"Floyd-Warshall"** se obtuvieron resultados bastante similares, con diferencias de apenas 16 segundos aprox. siendo **"Floyd-Warshall"** la implementación más rápida para la solución del problema. Así las cosas se concluye que el orden de 1 a 3 siendo 1 el algoritmo que mejor se comporta con archivos de volumen grandes y 3 el algoritmo que peor lo hace, se concluye lo siguiente:

1. **Floyd-Warshall.** **2. Dijkstra** **3. Bellman-Ford**