Dmytro Vernyuk

Practica 5: Bellman-Ford

Para ejecutar las pruebas, el código dispone de un generador aleatorio de grafos dirigidos valorados. En general, cada vértice tendrá una arista como mínimo, saliente o entrante, y una arista como máximo con el mismo origen y destino. Las aristas pueden ser recursivas (solo si existe otra arista entrante o saliente) y con costes que varían desde 9 hasta -9.

La generación de grafos aleatorios depende de la variable global N_VERTICES (por defecto =5). Esta indica cuantos números de vértices tendrán los grafos generados. Si queremos cambiar ese valor desde la terminal, utilizamos la tecla 9 e ingresamos un valor que sea mayor que 1 y menor o igual a 5000.

Para generar un grafo aleatorio ingresamos el numero 2 en el menú de la aplicación. Para cada grafo generado se nos muestra automáticamente el resultado por pantalla con todos los vértices y las aristas (si N_VERTICES <= 10) y si queremos imprimirlo manualmente utilizamos la tecla 4 (para cualquier caso). La **nomenclatura** de cada arista es de: arista "**x** -> **y** de coste **z**", donde **x** es el origen, **y** es el destino y **z** es el valor de la arista.

Para la resolución propia del problema utilizamos la tecla 1, siempre y cuando hallamos generado un grafo antes. Todas las resoluciones parten siempre del vértice 0 y puede darse el caso que en algunos grafos sea imposible llegar a uno o varios vértices, dando infinito positivo en las distancias menos en el propio vértice inicial (siempre con distancia 0). Cuando terminemos de resolver el grafo, se nos imprimirá por pantalla el tiempo utilizado por el algoritmo y el propio resultado con todas las distancias (si N_VERTICES <= 10) y si queremos imprimirlas manualmente utilizamos la tecla 3 (para cualquier caso). Las distancias pueden ser cero, positivas, negativas, infinitos positivos e infinitos negativos (en el caso de que tengamos un bucle con valores negativos altos). La **nomenclatura** de cada distancia es de: "vértice **x** -> **y**", donde **x** es el destino e **y** es el coste de ida desde el nodo 0 hasta el nodo **x**.

Algunos ejemplos sencillos de 5 vértices:

```
Vertice 0:
        arista 0 -> 3 de coste 9
        arista 0 -> 0 de coste 7
        arista 0 -> 4 de coste 9
        arista 0 -> 2 de coste 7

Vertice 1:
        arista 1 -> 0 de coste 5

Vertice 2:

Vertice 3:
        arista 3 -> 1 de coste -2
        arista 3 -> 3 de coste 5
        arista 3 -> 4 de coste 3
        arista 3 -> 2 de coste -6

Vertice 4:
        arista 4 -> 2 de coste 9
```

```
Tiempo total 0.0023 ms
Distancias desde el vertice 0:
        vertice 0 -> 0
        vertice 1 -> 7
        vertice 2 -> 3
        vertice 3 -> 9
        vertice 4 -> 9
```

```
Vertice 0:
        arista 0 -> 3 de coste -6

Vertice 1:
        arista 1 -> 2 de coste -3
        arista 1 -> 3 de coste 5
        arista 1 -> 0 de coste -3
        arista 1 -> 4 de coste 0

Vertice 2:
        arista 2 -> 0 de coste 3
        arista 2 -> 1 de coste -6

Vertice 3:
        arista 3 -> 4 de coste -2

Vertice 4:
        arista 4 -> 4 de coste -4
```

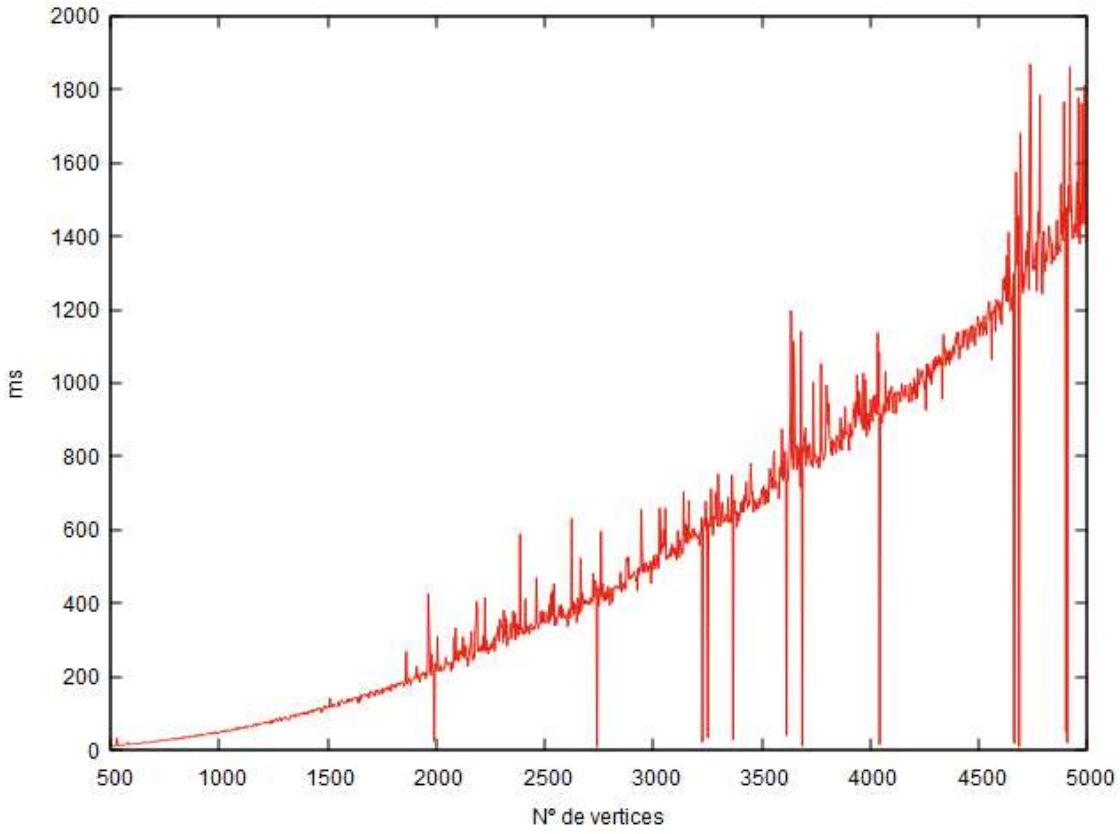
```
Tiempo total 0.0035 ms
Distancias desde el vertice 0:
        vertice 0 -> 0
        vertice 1 -> inf
        vertice 2 -> inf
        vertice 3 -> -6
        vertice 4 -> -inf
```

```
Vertice 0:
        arista 0 -> 1 de coste -6
        arista 0 -> 2 de coste 9
        arista 0 -> 4 de coste -4
        arista 0 -> 3 de coste -7
        arista 0 -> 0 de coste -1

Vertice 1:

Vertice 2:
        arista 2 -> 4 de coste 4

Vertice 3:
        arista 3 -> 3 de coste -7
        arista 3 -> 0 de coste 5
        arista 3 -> 2 de coste 4
        arista 3 -> 4 de coste 0

Vertice 4:
        arista 4 -> 1 de coste 1
```

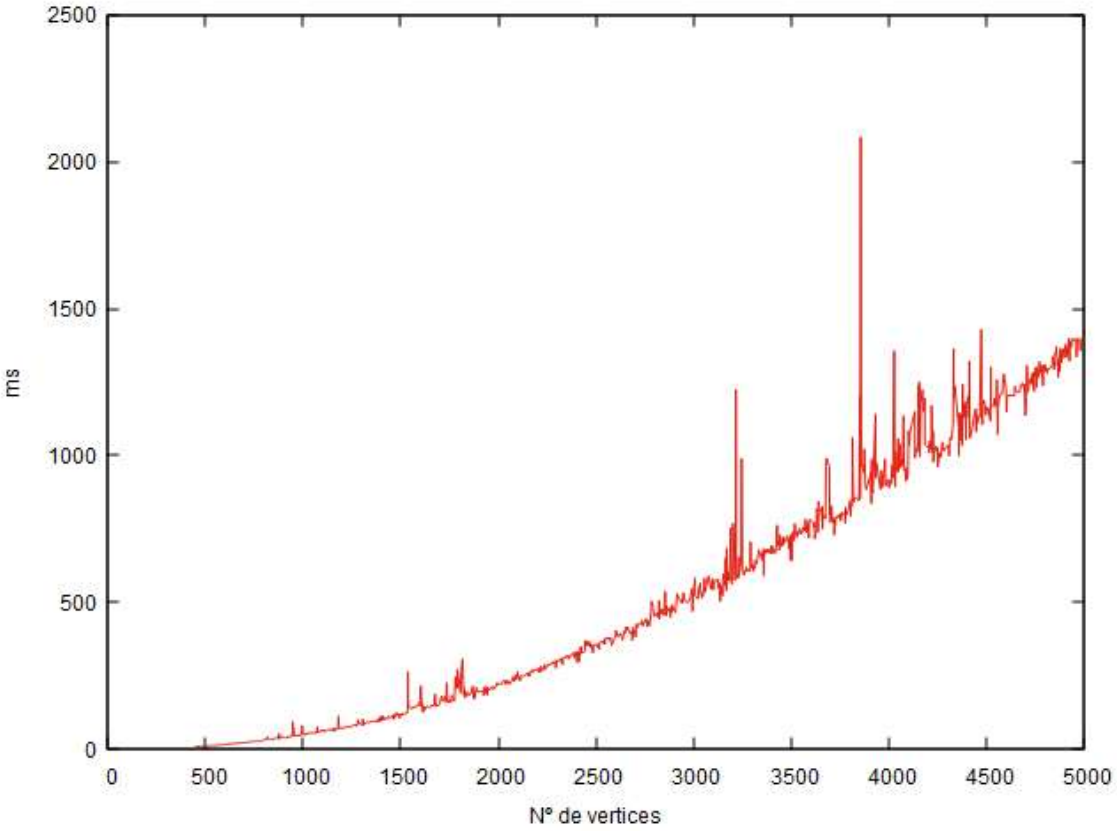
```
Tiempo total 0.0037 ms
Distancias desde el vertice 0:
        vertice 0 -> -inf
        vertice 1 -> -inf
        vertice 2 -> -inf
        vertice 3 -> -inf
        vertice 4 -> -inf
```

Para la generación de tiempos según el tamaño, la aplicación cuenta en el menú con la opción 8, que genera en un archivo “tiempos.txt” los tiempos obtenidos. La subrutina de la opción crea y destruye grafos desde un mínimo de 2 vértices hasta un máximo de 5000 para después ejecutar el algoritmo principal sobre estos y obtener el tiempo de ejecución.

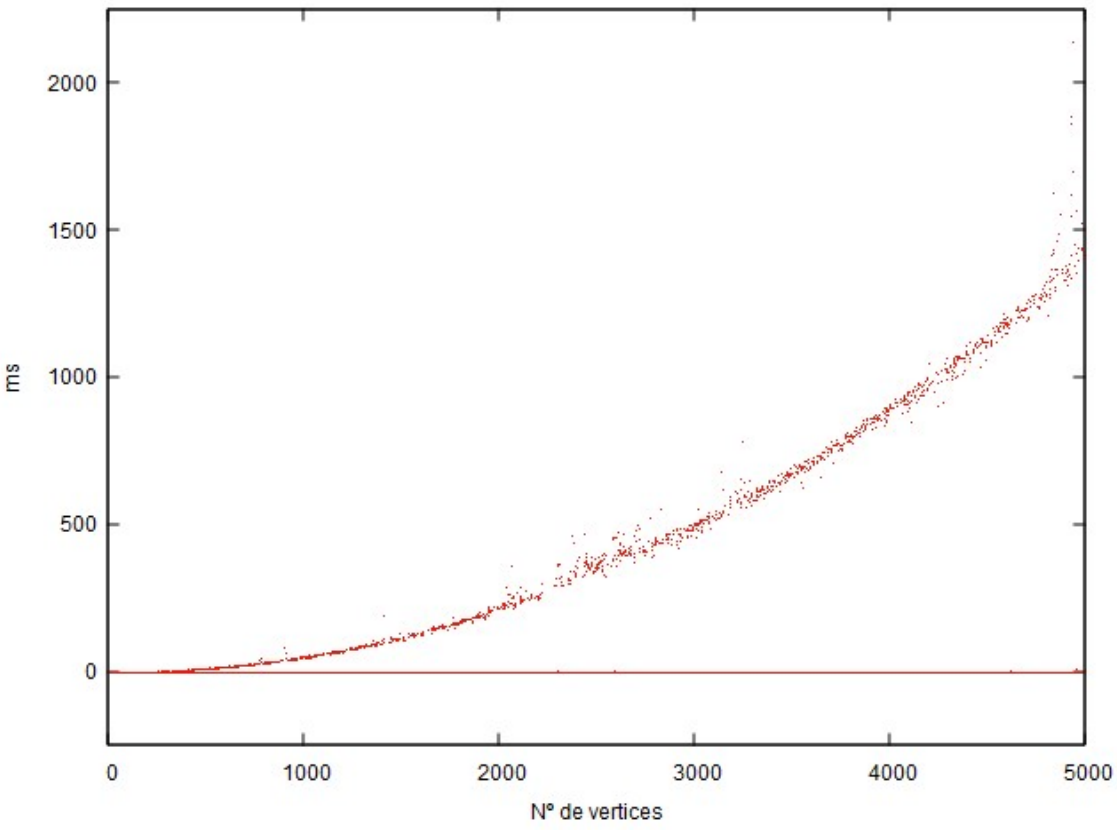
Algunos gráficos de tiempos:



Para este grafico se ha generado grafos con vértices desde 500 hasta 5000 sumando +3 en cada iteración. Antes de 500 casi todas las salidas son igual a cero. Para cada iteración escogíamos 3 resultados mayores o igual que 10 ms y hacíamos la media, la cual utilizamos para el grafico. Este grafico es tal y como se indica en el enunciado.



Para este grafico se ha generado grafos con vértices desde 2 hasta 5000 sumando +1 en cada iteración. Para cada iteración escogíamos los 3 primeros resultados, hacíamos la media y solo la seleccionábamos si esta fuese mayor o igual a 10 ms.



Para este grafico se ha generado grafos con vértices desde 2 hasta 5000 sumando +1 en cada iteración. Para cada iteración escogíamos siempre el primer resultado.

En los gráficos observamos que hay una importante cantidad de resultados con tiempo 0 ms, esto es gracias a la “poda” que hacemos al no mejorarse ni una distancia en cierta iteración de los dos bucles.

En los casos donde no es 0 los tiempos son grandes, pero no tan grandes como podrían. El generador no crea tantas aristas como vértices (en mis pruebas con 5000 vértices cada uno llegaba a lo sumo a 10, aunque teóricamente podrían llegar a tener 5000). Esto explica el porqué de una curva tan poco empinada teniendo en cuenta que el coste es O(VE) y que este depende directamente del número de aristas, además del número de vértices.