Actividad 4.

Se quieren implementar nuevas funcionalidades:

a) `public HashMap<String, Double> pageRank()`
`// Post: el resultado es el valor del algoritmo PageRank para cada actor del grafo`

El algoritmo Page Rank (https://en.wikipedia.org/wiki/PageRank) corresponde a una familia de algoritmos utilizados para asignar de forma numérica la relevancia de los documentos (o páginas web) de un grafo. El sistema PageRank es utilizado por el popular motor de búsqueda Google para determinar la importancia o relevancia de una página web. El algoritmo PageRank calcula la probabilidad de que una persona, accediendo al grafo y siguiendo enlaces, llegue a un nodo en particular. El cálculo de PageRank requiere varios pases, llamados iteraciones, para ajustar el valor aproximado. Inicialmente, todos los nodos reciben la misma probabilidad:

$\forall A \in nodos, PR(A) = \frac{1}{N}$ donde **N** es el número de nodos del grafo.

Después, en cada iteración, se recalcula el valor asociado a cada nodo mediante la fórmula:

$PR(A) = \frac{1-d}{N} + d * \sum_{i=1}^{n} \frac{PR(i)}{C(i)}$ donde:

- **PR(A)** es el PageRank de la página A.
- **d** (damping factor) es un factor de amortiguación que tiene un valor entre 0 y 1 (un valor típico es 0.85)
- **PR(i)** son los valores de PageRank que tienen cada una de las páginas **i** que enlazan a A.
- **C(i)** es el número total de enlaces salientes de la página i (sean o no hacia A).

El proceso continúa hasta que la suma de las diferencias en valor absoluto entre los valores de una iteración y la siguiente es menor a un umbral predeterminado (p. ej. 0.0001).

Como ejemplo, supongamos un grafo formado por cuatro nodos A, B, C y D. B tiene enlaces hacia C y A, C tiene un enlace hacia A, y el nodo D tiene enlaces de salida hacia A, B y C.

Inicialmente cada nodo tendrá un valor de 0.25. En la primera iteración, B transferirá la mitad de su valor, 0.125, a A y la otra mitad a C. C transferirá todo su valor, 0.25, a A. Como D tiene tres arcos salientes, transferirá un tercio de su valor (aproximadamente 0.083) a A, B y C. Después de esta primera iteración, A tendrá un PageRank aproximado de 0.427.

b)
```
public class Par {

   String actor;

   Double pageRank;

   }

ArrayList<Par> ordenarPorPageRank(ArrayList<String> actores)

// Post: la lista se ordenará de mayor a menor por su pagerank, de manera que en las primeras posiciones aparecerán los elementos con pagerank más alto

// Se deberá implementar un método de ordenación (o alternativamente, usar un método de ordenación implementado en el primer trabajo práctico)
```

Se deberá entregar: Programas que implementen lo pedido (ejecutados correctamente). **Se deberá demostrar que el programa funciona realmente con conjuntos de datos no triviales (es decir, procesando miles de líneas del fichero original).** La documentación deberá incluir:

- Ejemplos de ejecución de los métodos pedidos, indicando datos de prueba y resultados, junto con el número de datos usados y de pruebas realizadas. En caso de que sea relevante, también se indicará el tiempo de ejecución de cada prueba.
- Documentación describiendo el problema planteado, las alternativas examinadas, implementaciones y eficiencia.

NOTA: debido a que algunos de los resultados pedidos requerirán un alto tiempo de computación, esto exigirá que los algoritmos funcionen correctamente un tiempo antes de la fecha de entrega, ya que si no no se podrán obtener los resultados.

Además debéis rellenar y entregar el Checklist para verificar que habéis realizado todo lo que se os pide.

AYUDA: sobre el ejemplo del fichero de 20.000 películas diferencias de suma de page rank en cada iteración (orientativamente, cada iteración lleva unos 8 segundos)

iteración:	0	diff:	0.26152610253673697
iteración:	1	diff:	0.0751788597593788
iteración:	2	diff:	0.04379976148760491
iteración:	3	diff:	0.029756028670771123
iteración:	4	diff:	0.0210104361047662
iteración:	5	diff:	0.015256596569045376
iteración:	6	diff:	0.011290116695255066
iteración:	7	diff:	0.008475450588192832
iteración:	8	diff:	0.006433641979389274
iteración:	9	diff:	0.004928753914452251
iteración:	10	diff:	0.0038035525098759535
iteración:	11	diff:	0.002952607776162848
iteración:	12	diff:	0.0023049070671614033
iteración:	13	diff:	0.0018076799804635716
iteración:	14	diff:	0.0014232358528210281
iteración:	15	diff:	0.0011245252041837974
iteración:	16	diff:	8.913000417858815E-4
iteración:	17	diff:	7.084737427111381E-4
iteración:	18	diff:	5.646748366378263E-4
iteración:	19	diff:	4.5124168112586693E-4
iteración:	20	diff:	3.6140971600794405E-4
iteración:	21	diff:	2.9008751845244684E-4
iteración:	22	diff:	2.3327731386682852E-4
iteración:	23	diff:	1.8794160648164107E-4
iteración:	24	diff:	1.517017202904282E-4
iteración:	25	diff:	1.2264761222435073E-4
iteración:	26	diff:	9.929913703454761E-5

número de iteraciones: 27