TP 2: Union-Find

Universidad Nacional de General Sarmiento

Materia: Programacion 3

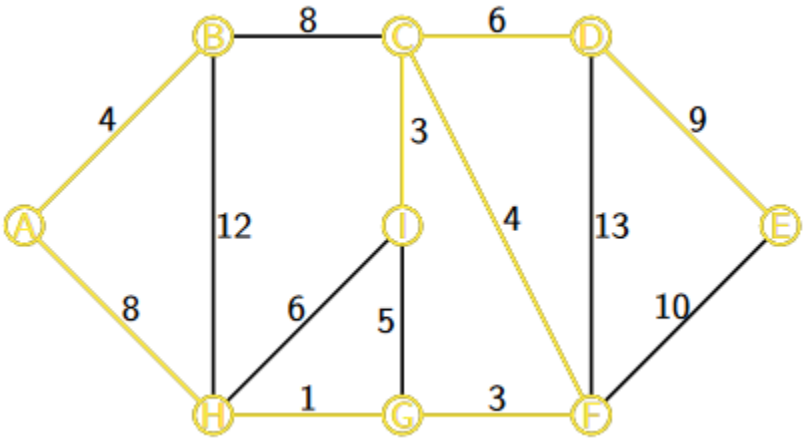
Comision 1 (noche)

Docentes: Patricia Bagnes, Javier Marenco

Integrantes: Richter Alexis Valentin, Nazareno Avalos

1. Implementar el Algoritmo de Kruskal sin Union-Find (es decir, determinando con BFS si dos vertices estan en la misma componente conexa)

El grafo utilizado para este punto fue el de la diapositiva de AGM



2. Implementar el Algoritmo de Kruskal con alguna de las versiones de Union-Find, implementada sobre un arreglo de indices

Tambien se uso el grafo del punto 1 y se eligio path compression

3. Generar grafos aleatoriamente y medir el tiempo de ejecucion de las dos versiones.

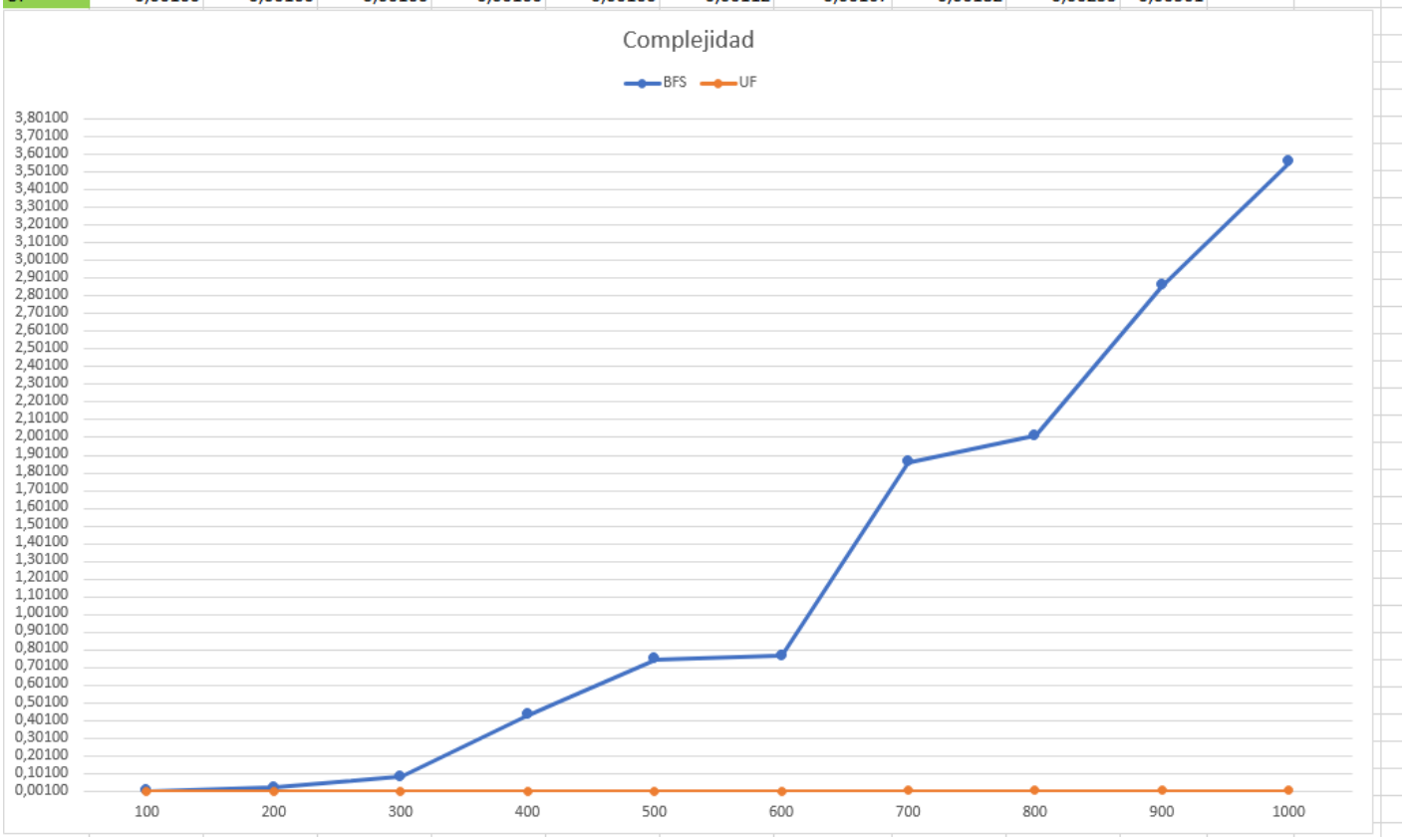
Este punto esta echo todo en la clase TiempoPromedio, se decidio no utlizar el metodo propuesto por el TP para generar grafos randoms

4. Graficar el tiempo promedio de ejecucion en funcion del tamano del grafo

```
Generando grafo...
Procesando AGM con BFS...
Tamanio: 100, seg: 0.003960000000000003

Generando grafo...
Procesando AGM con BFS...
Tamanio: 200, seg: 0.02399

Generando grafo...
Procesando AGM con BFS...
Tamanio: 300, seg: 0.08526000000000002

Generando grafo...
Procesando AGM con BFS...
Tamanio: 400, seg: 0.4334999999999998

Generando grafo...
Procesando AGM con BFS...
Tamanio: 500, seg: 0.7470399999999995

Generando grafo...
Procesando AGM con BFS...
Tamanio: 600, seg: 0.7665900000000003

Generando grafo...
Procesando AGM con BFS...
Tamanio: 700, seg: 1.8594600000000012

Generando grafo...
Procesando AGM con BFS...
Tamanio: 800, seg: 2.00963

Generando grafo...
Procesando AGM con BFS...
Tamanio: 900, seg: 2.860280000000001

Generando grafo...
Procesando AGM con BFS...
Tamanio: 1000, seg: 3.556790000000001
```

```
Generando grafo...
Procesando AGM con UF...
Tamanio: 100, seg: 7.0000000000000001E-5

Generando grafo...
Procesando AGM con UF...
Tamanio: 200, seg: 2.0000000000000012E-4

Generando grafo...
Procesando AGM con UF...
Tamanio: 300, seg: 4.8000000000000034E-4

Generando grafo...
Procesando AGM con UF...
Tamanio: 400, seg: 6.300000000000005E-4

Generando grafo...
Procesando AGM con UF...
Tamanio: 500, seg: 7.700000000000005E-4

Generando grafo...
Procesando AGM con UF...
Tamanio: 600, seg: 0.0011200000000000008

Generando grafo...
Procesando AGM con UF...
Tamanio: 700, seg: 0.0016700000000000011

Generando grafo...
Procesando AGM con UF...
Tamanio: 800, seg: 0.0018200000000000013

Generando grafo...
Procesando AGM con UF...
Tamanio: 900, seg: 0.002530000000000002

Generando grafo...
Procesando AGM con UF...
Tamanio: 1000, seg: 0.0030100000000000023
```

	100	200	300	400	500	600	700	800	900	1000
BFS	0,00396	0,02399	0,08526	0,43349	0,74703	0,76659	1,85946	2,00963	2,86028	3,55679
UF	0,00100	0,00100	0,00100	0,00100	0,00100	0,00112	0,00167	0,00182	0,00253	0,00301



CODIGO...

El codigo esta divido en cuatro packages:

algoritmos, donde se encuentra BFS, Union-Find y Kruskal con la implementacion de BFS y UF, tambien cada clase de negocio tiene su respectivo test

grafo_ListaVecinos, una clase grafo representada por su lista de vecinos y su BFS adaptado para utilizar Lista de Vecinos, solo se utiliza para generar grafos random, tambien tienen sus tests

grafo_MatrizAdyacencia, una clase grafo representada por su matriz de adyancencia, aca tambien esta incluida la clase TiempoPromedio, posee tests

main, tiene una clase main donde se ejecutan todos los puntos del TP

Cada funcion tiene su explicacion dentro del codigo...

Aclaraciones:

La funcion getTodasAristas ordena todas las aristas del grafo, este metodo en principio pareceria O(1) pero es O(n log n) ya que usa colections sort, se ordena antes para evitar errores en el equals de grafo esto se debe porque al paracer dos ArrayList son iguales solo si tienen los mismos elmentos ordenados de la misma manera

Asimismo la clase Arista implementa la interfaz Comparable