Report: GiantBook

By Rasmus Lehmann, Sergiy Isakov & Andreas Tønder

Results

The following table summarizes our results – it shows the average number of random connections needed to make three events occur: the first appearance of a giant component ("giant"), the first time no vertices are isolated ("no isolated") and the first time all vertices are connected in the same component ("connected").

N denotes the number of vertices and *T* denotes the number of times the simulation is run.

N	T	giant	(stddev)	no isolated	(stddev)	connected	(stddev)
100	10^5	7.17×10^1	5.92	2.60×10^2	6.33×10^1	2.62×10^2	6.31×10^1
1000	10^4	6.95×10^2	1.76×10^1	3.75×10^3	6.45×10^2	3.75×10^3	6.44×10^2
10^4	1000	6.93×10^3	5.39×10^1	4.88×10^4	6.45×10^3	4.88×10^4	6.45×10^3
10^5	100	6.93×10^4	1.77×10^2	6.04×10^5	6.54×10^4	6.04×10^5	6.54×10^4
10^6	50	6.93×10^5	4.90×10^2	7.25×10^6	7.25×10^5	7.25×10^6	7.25×10^5
10^7	10	6.93×10^6	1.21×10^3	8.15×10^7	4.11×10^6	8.15×10^7	4.11×10^6

Our main findings are the following: The first thing that happens is that the giant component emerges. Perhaps surprisingly, two of the events seem to happen simultaneously, namely "no isolated" and "connected". Both the emergence of giant component, and the "no isolated" and "connected" events happen at linear time in *N*.

Implementation details

We have based our union-find data type on *WeighedQuickUnionUF.java* from Sedgewick and Wayne: Algorithms, 4th ed., Addison-Wesley (2011). We added a method *erdosRenyi(int p, int q)* by adding the following lines to find all events:

```
178     public void erdosRenyi(int p, int q) {
179         union(p, q);
180         setOfVertices.add(p);
181         setOfVertices.add(q);
182         time++;
183         if (setOfVertices.size() == numberOfVertices && lastIsolatedVertex == -1)
184             lastIsolatedVertex = time;
185         if (size[find(p)] >= (double) numberOfVertices / 2 && giantComponent == -1)
186             giantComponent = time;
187         if (count() == 1 && graphConnected == -1) graphConnected = time;
188     }
```

It starts by calling original *union* method, adds *p* and *q* elements in a *HashSet* and increments *time* by 1. Then in first if-statement it checks whether size of HashSet reached the size of network, and if it so, then assigns time to variable *lastIsolatedVertex*. The second if-statement checks if there is a component that reached at least half of network size. And

the last statement traces variable *count*, which is showing if the network is already connected.

Assuming we never run out of memory or heap space, if we would let out algorithm for detecting the emergence of a giant component run for 24 hours, it could compute the answer for $N = 8{,}93*10^{10}$.

Discussion

We defined the giant component to have size at least αN for $\alpha = 0.693$.