Random Graphs

Optional Project – See The Window

**Ground Rules.** You can work as a group with at most three people. The project would be submitted by the group. Submission should be by April 12. Your results **must** be described in a clear understandable way.

Take \( n \) vertices 1, \ldots, \( n \), initially with no edges. Each step select two random vertices \( x, y \) and add the edge \( \{x, y\} \). Use a Union-Find data structure to keep track of the components and their sizes and (see below) their complexities. At step \( e \) (so now the graph has \( e \) edges) parametrize

\[
e \left( \frac{n^2}{2} \right) = \frac{1}{n} + \frac{\lambda}{n^{4/3}}
\]

Stop the process at (say) \( \lambda = -4, -3, -2, -1, 0, 1, 2, 3, 4 \). For each \( \lambda \) give the top ten component sizes both raw and when divided by \( n^{2/3} \).

What will happen (if you did it correctly!) is that at \( \lambda = -4 \) (barely subcritical) there is an “asteroid belt” with the top sizes fairly close together. During the critical window these asteroids are merging and by \( \lambda = +4 \) (barely supercritical) a “Jupiter” has appeared, a component whose size is substantially larger than the others.

The **complexity** of a component is edges minus vertices plus one. (Trees are zero, unicyclic one, etc.) When two components merge the complexity of the new component is simply the sum of the old complexities. When an edge \( \{x, y\} \) is added with \( x, y \) in the same component, its complexity is incremented by one. What will happen (if you did it correctly!) is that at \( \lambda = -4 \) the complexities of the large components are pretty much all zero but that at \( \lambda = +4 \) the Jupiter will have a moderately large complexity, the other complexities pretty much all zero.

**Repeated Edges.** Occasionally you may select \( \{x, y\} \) that has already been selected. Bummer. But it turns out that this has a negligible effect. So simply ignore it, process this \( \{x, y\} \) the same as the others and implement \( e \) the same as before.

**Movie.** If you are a graphics type – how about a movie of what’s going on! **How many edges** \( n = 10^6 \), if well programmed, should be good. You can compare different \( n \) and many runs.
More Ideas: Let $C_1, C_2, \ldots$ denote the components in decreasing size order.

1. Graph $|C_1|n^{-2/3}$ (averaged over several runs) for various $\lambda$

2. Graph $|C_2|n^{-2/3}$ (averaged over several runs) for various $\lambda$. Here at some point it will get smaller – when $C_2$ merges with $C_1$ the new $C_2$ is the previous $C_3$.

3. Set

$$\chi(G) = \frac{1}{n} \sum |C|^2$$

summed over all components. This is called the susceptibility of the graph. Rather than recomputing this the value of $\chi(G)$ can be updated every time an edge is added. Graph $|C_1|^2/\chi(G)$ as a function of $\lambda$. This gives a measure of the dominance of the Dominant Component.

Open-Ended. Come up with your own interesting data on the process in this region. Read Chapter 11 for ideas. Surprise me!
Union Find Notes

(If you have another way – thats fine too! These are somewhat modified notes that I gave for Fundamental Algorithms last year.)

We want to start with no edges and add random edges $e_1, \ldots, e_E$. Let $x_i, y_i$ be the (random) vertices of $e_i$. To each vertex $x$ we have functions $\pi(x)$ and $\text{SIZE}(x)$, $C(x)$ initially all $\pi(x) \leftarrow x$ and all $\text{SIZE}(x) \leftarrow 1$ and all $C(x) \leftarrow 0$. ($C$ is complexity)

For $i = 1$ to $E$ we set (for notational convenience) $x \leftarrow x_i$, $y \leftarrow y_i$ and do the following:

WHILE $\pi(x) \neq x$
  $x \leftarrow \pi(x)$ (*going down the stairs*)

WHILE $\pi(y) \neq y$
  $y \leftarrow \pi(y)$ (*going down the stairs*)

IF $x = y$ then $C(x) \leftarrow +$

IF $x \neq y$ then DO
  IF $\text{SIZE}(x) \leq \text{SIZE}(y)$ then DO
    $\pi(x) \leftarrow y$
    $\text{SIZE}(y) \leftarrow \text{SIZE}(y) + \text{SIZE}(x)$
    $C(y) \leftarrow C(y) + C(x)$
  OTHERWIZE DO
    $\pi(y) \leftarrow x$
    $\text{SIZE}(x) \leftarrow \text{SIZE}(x) + \text{SIZE}(y)$
    $C(x) \leftarrow C(x) + C(y)$

At any time the $\pi(x)$ will give a rooted forest with $\pi(x) = x$ exactly when $x$ is a root. For the $x$ with $\pi(x) = x$ (ignore others!) $\text{SIZE}[x]$ is the size of the component containing $x$ and $C[x]$ is the complexity. One thing you can do is at various values of $E$ (corresponding to, say, $\lambda = -4, -3, \ldots, +3, +4$) go through all $x$ (but ignore those with $\pi(x) \neq x$ and find the ten (say) largest $\text{SIZE}[x]$. Looking at the $C[x]$ for these ten largest is also quite interesting – for $\lambda = -4$ they should all be zero (possibly 1) but by $\lambda = +4$ the $C[x]$ for the largest component will be of fair size – what is happening is that this biggest component is getting internal edges.