Algorithmic Thinkingby Luay Nakhleh, Scott Rixner, Joe Warren

Overview

Graph exploration (that is, "visiting" the nodes and edges of a graph) is a powerful and necessary tool to elucidate properties of graphs and quantify statistics on them. For example, by exploring a graph, we can compute its degree distribution, pairwise distances among nodes, its connected components, and centrality measures of its nodes and edges. As we saw in the Homework and Project, breadth-first search can be used to compute the connected components of a graph.

In this Application, we will analyze the connectivity of a computer network as it undergoes a cyber-attack. In particular, we will simulate an attack on this network in which an increasing number of servers are disabled. In computational terms, we will model the network by an undirected graph and repeatedly delete nodes from this graph. We will then measure the *resilience* of the graph in terms of the size of the largest remaining connected component as a function of the number of nodes deleted.

Example graphs

In this Application, you will compute the resilience of several types of undirected graphs. We suggest that you begin by collecting and writing code to create the following three types of graphs:

- **An example computer network** - The text representation for the example network is here. You may use this provided code to load the file as an undirected graph (with 1347 nodes and 3112 e dges). Note that this provided code also includes several useful helper functions that you should review.
- **ER graphs** - If you have not already implemented the pseudo-code for creating *undirected* ER graphs, you will need to implement this code. You may wish to modify your implementation of `make_complete_graph` from Project 1 to add edges randomly (again, keep in mind that in Project 1 the graphs were directed, and here we are dealing with undirected graphs).
- **UPA graphs** - In Application 1, you implemented pseudo-code that created DPA graphs. These graphs were directed (The D in DPA stands for directed). In this Application, you will modify this code to generate undirected UPA graphs. The UPA version should add an undirected edge to the UPA graph whenever you added a directed edge in the DPA algorithm. Note that since the degree of the newly added node is no longer zero, its chances of being selected in subsequent trials increases. In particular, you should either modify the `class DPATrial` to account for this change or use our provided UPATrial class.