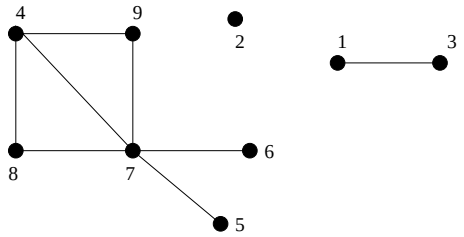
ICFP M2 - Statistical physics 2
Homework n^o 3
A numerical study of random graphs

Guilhem Semerjian

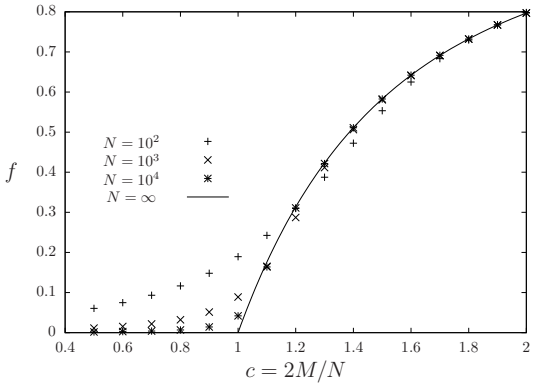
February 2019

The next lecture and TD shall be devoted to the properties of random graphs ; in this homework you are invited to discover by a numerical experimentation some of their phenomenology.

A graph is a set of vertices linked by a set of edges (an edge being an unordered pair of vertices). A connected component of a graph is a maximal subset of vertices that can be reached one from the other by traveling along the edges. One calls size of a connected component the number of vertices it contains. The graph of the figure below has $N = 9$ vertices and $M = 8$ edges ; there are 3 conneced components, one of size 1 (containing the vertex 2) one of size 2 (containing the vertices 1 and 3), and one of size 6 (containing the rest of the graph).



A simple way of generating a graph on N vertices consists in drawing at random M edges, where each edge is chosen uniformly among the $N(N-1)/2$ possible ones. On the plot below one can see the average fraction f of vertices in the largest component (i.e. the size of the largest component of the graph divided by N), as a function of $c = 2M/N$, for different sizes N of the graph.



These curves suggest the appearance of a phase transition at $c = 1$ in the thermodynamic limit. This is indeed the case, the solid line being the analytical prediction that will be derived in the next TD.

The exercise is for you to reproduce the plot above with your own numerical simulation. You need thus to generate random graphs, and identify the connected components to compute the size of the largest one. To perform this task you can either exploit ready-to-use libraries in Python (see for instance the package `networkx`), in which case your python code should only be a few lines long, or implement yourself the algorithms for the generation of random graphs and identification of the largest component. In the latter case you should think of the data structure you can use to represent a graph, in particular to be able to find efficiently the neighbors of a vertex. To identify the connected

components a possible way to proceed is to start with all vertices in an « yet unidentified » state, then put one vertex v in a list, set $i = 1$, and mark the vertex v as « in component i ». As long as the list is non-empty, remove one element w of the list, consider all the neighbors of w, mark those that are still unindentified as « in component i » and add them to the list. When the list is empty you have exhausted the connected component of v. If there remains unindentified vertices at this point, choose one of them, increment i by 1, and start again from this new vertex.