Complex Networks: solutions to the questions in the examination of 16 December 2015

1. List the four main categories into which real-world networks are classified and give a short description of an example in each category.

   Solution: The four main categories are: communcation networks, technological networks, economic networks, biological networks. Chapter 1 of the lectures notes lists examples in each category.

2. 2a. Give the definition of typical distance in a graph G.

   Solution:
   $$H(G) = \frac{\sum_{i,j \in V, i \neq j, i \leftrightarrow j} d(i,j)}{\sum_{i,j \in V, i \neq j, i \leftrightarrow j} 1}$$
   where $d(i,j)$ is the graph distance between vertices i and j, and $i \leftrightarrow j$ means that i and j are connected. This formula says that $H(G)$ is the average distance between two vertices that are drawn randomly from G and are conditoned to be connected.

   2b. When is a sequence of random graphs $(G_n)_{n \in \mathbb{N}}$ called small world?

   Solution: When there exists a constant $K \in (0, \infty)$ such that
   $$\lim_{n \to \infty} \mathbb{P}(H(G_n) \leq K \log n) = 1,$$
   where $\mathbb{P}$ denotes the law of $(G_n)_{n \in \mathbb{N}}$. This says that distances within G_n grow very slowly with n.

   2c. Is the sequence of Erdős-Rényi random graphs $(\mathrm{ER}_n(\lambda/n))_{n \in \mathbb{N}}$ small world for all choices of $\lambda \in (0, \infty)$?

   Solution: No, only for $\lambda \neq 1$. For $\lambda < 1$, all clusters have size $O(\log n)$, in which case the statement is obvious. For $\lambda > 1$, there is one giant component, whose size is $\Theta(n)$, while all other clusters again have size $O(\log n)$. But even within the giant component distances are $O(\log n)$. For $\lambda = 1$, there are many components of size $\Theta(n^{2/3})$, and they have larger distances, namely, $\Theta(n^{1/3})$.

3. 3a. Draw all the possible outcomes of $\mathrm{CM}_4((1,1,2,2))$.

   Solution: There are $(6-1)!! = 15$ ways to pair the half-edges. These lead to only 6 different outcomes.

   3b. What is the probability of each outcome? (Note that different pairings of half-edges may lead to the same outcome.)

   Solution: The 6 different outcomes correspond to $4, 4, 2, 2, 2, 1$ pairings. Hence their probabilities are $\frac{4}{15}, \frac{4}{15}, \frac{2}{15}, \frac{2}{15}, \frac{2}{15}, \frac{1}{15}$.

   3c. Are all the outcomes simple?

Solution: No, 3 outcomes contain one or two self-loops, while 1 outcome contains a double edge. Only 2 outcomes are simple.

4. A real-world network is represented as a simple (i.e., with no multiple edges and self-loops) undirected graph $\mathbf{G}^*$ with $n = 5$ vertices. Chung and Lu decide they want to compare their model with the real-world network. They define their connection probabilities $\{p_{ij}\}$ (with $p_{ii} \equiv 0 \, \forall i$) and, after computing them on the real network, they find that $p_{34} > p_{53}$, $p_{15} > p_{25}$, $p_{43} > p_{14}$, $p_{52} = p_{45}$.

   4a. Find the degree sequence $\vec{k}(\mathbf{G}^*)$ of the real-world network $\mathbf{G}^*$. Explain your result.

   Solution: In the Chung-Lu model, the connection probabilities are $p_{ij} = k_i k_j / 2L$ for $i \neq j$ and $p_{ii} = 0$, where $\{k_i\}_{i=1}^n$ are the degrees and $L = \sum_{i=1}^n k_i/2$ is the total number of links. Therefore $p_{il} > p_{jl}$ implies $k_i > k_j$, and $p_{il} = p_{jl}$ implies $k_i = k_j$ (for $l \neq i, j$). From the (in)equalities given in the text, we can therefore conclude that

$$k_3 > k_1 > k_2 = k_4 > k_5.$$

   Since the degrees are non-negative integers, it is easy to check that the only set of numbers that satisfies the above (in)equalities in such a way that a simple undirected graph can be realized is

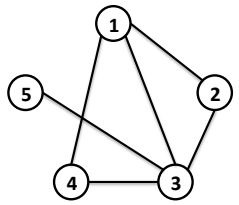
$$4 > 3 > 2 = 2 > 1.$$

   The corresponding degree sequence is:

$$\vec{k}(\mathbf{G}^*) = (3, 2, 4, 2, 1).$$

   4b. Draw $\mathbf{G}^*$. Explain your result.

   Solution: It is easy to check that there is only one graph consistent with the degree sequence found in 4a. This graph, which is necessarily $\mathbf{G}^*$, is:



   4c. Write the matrix $\mathbf{P}$ having the numerical values of $\{p_{ij}\}$ as entries.

   Solution: Given the expression for p_{ij} in the Chung-Lu model and the degree sequence of $\mathbf{G}^*$ (see solution to 4a), the matrix $\mathbf{P}$ is easily

calculated as

$$\mathbf{P} = \frac{1}{2L}\begin{pmatrix} 0 & k_1k_2 & k_1k_3 & k_1k_4 & k_1k_5 \\ k_2k_1 & 0 & k_2k_3 & k_2k_4 & k_2k_5 \\ k_3k_1 & k_3k_2 & 0 & k_3k_4 & k_3k_5 \\ k_4k_1 & k_4k_2 & k_4k_3 & 0 & k_4k_5 \\ k_5k_1 & k_5k_2 & k_5k_3 & k_5k_4 & 0 \end{pmatrix} = \frac{1}{12}\begin{pmatrix} 0 & 6 & 12 & 6 & 3 \\ 6 & 0 & 8 & 4 & 2 \\ 12 & 8 & 0 & 8 & 4 \\ 6 & 4 & 8 & 0 & 2 \\ 3 & 2 & 4 & 2 & 0 \end{pmatrix}.$$

4d. Write the probability of occurrence of $\mathbf{G}^*$ under the model used by Chung and Lu.

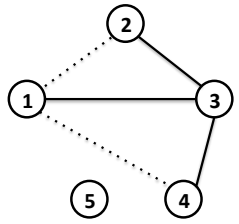
Solution: Since edges are independent in the model, the probability of the graph $\mathbf{G}^*$ is

$$P(\mathbf{G}^*) = \prod_i \prod_{j>i} p_{ij}^{g_{ij}} (1-p_{ij})^{1-g_{ij}} = \frac{25}{1944} \approx 0.01286,$$

where g_{ij} is the entry of the adjacency matrix of $\mathbf{G}^*$ and the numerical values of p_{ij} are taken from the matrix $\mathbf{P}$ calculated in 4c.

4e. Draw all the graphs that have the maximum probability in the model.

Solution: The most likely graphs are those such that an edge between vertices i and j is drawn whenever $p_{ij} > 1/2$ and is *not* drawn whenever $p_{ij} < 1/2$. Edges for which $p_{ij} = 1/2$ can be drawn or not, with no effect on the graph probability. By looking at the entries of $\mathbf{P}$ (see 4c), one can conclude that there are 4 graphs with maximum probability. These graphs can be compactly drawn as



where a solid edge is always drawn, whereas a dashed edge can be drawn or not.

4f. Is $\mathbf{G}^*$ found among the graphs with maximum probability? Explain why.

Solution: No. This is because, given a real-world network $\mathbf{G}^*$, the Chung-Lu model is not constructed in a way that ensures that $P(\mathbf{G}^*) = \max_{\mathbf{G}} P(\mathbf{G})$.

5. Whenever this exam asks to "provide an algorithm," you are requested to do so in a programming language that is close to Python, Java, C, or C++. Here "close" means that we are interested in the algorithmic essence. No

points will be deducted for simple syntactic omissions such as missing semi-colons. The algorithmic meaning, however, should be clear beyond doubt. Do provide declarations of essential variables. Provide algorithmic detail with low level operations, no high level Python functions. See also definition of pseudo-code at the end of this exam.
Of an undirected graph the following edge list is given:

v_1	v_2
1	2
2	3
2	4
2	5
2	6
4	6

5a. Draw the graph.

Solution: Everybody was able to draw this graph correctly!

5b. Provide the adjacency matrix of this graph in algorithmic Pseudo-code.

Solution: Assuming an undirected graph, which is codes with double entries.

```
int am = {{0, 1, 0, 0, 0, 0},
{1, 0, 1, 1, 1, 1},
{0, 1, 0, 0, 0, 0},
{0, 1, 0, 0, 0, 1},
{0, 1, 0, 0, 0, 0},
{0, 1, 0, 1, 0, 0}};
```

5c. What is the mean of the number of edges of the vertices? Provide an algorithm to compute this number.

Solution: Code in C. C++, Java and Python are also ok. No high level Python constructs that obscure the time complexity, though. No comments in code means less points.

```
#define N 6            // dimension, a constant
double mean() {
  int am[N][N] = ...;  // matrix declaration
  int total = 0;       // the sum of the edge count
  for (int i = 0; i < N; i ++) {  // doubly nested loop
    for (int j = 0; j < N; j ++) {
      total += am[i][j];
    }
  }
  return (double)(tot / N); // mean is total dived by number of vertices
```

```
}
```

5d. What is the time complexity of this algorithm? Why?

Solution: $O(n^2)$. Two nested for loops depend on n. Other computations do not depend on n.

5e. Name two strategies to help you ascertain that your program is correct.

Solution: 1. Code inspection by yourself or by someone else
2. Correctness proof using pre and post conditions
3. Use test instances to check the outcome of the algorithm

6. 6a. Provide an algorithm for the Configuration Model based on an adjacency matrix. Describe any assumptions, imperfections or limitations of your program.

Solution: Writing an algorithm for the Configuration Model that addresses all possible problems is actually quite complicated. Below is the most basic version, as was described during our classes. It has a few imperfections, which will be described below.

```
#define D 11
#define N 50

int dist[D];

int degree[N];

int am[N][N];

int main(void) {

  // init am
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      am[i][j] = 0;
    }
  }

  // generate degree sequence with Nk = k^-t
  printf("Degree distribution: ");
  double t = -1.3;
  double c = 20.0;
  for (int k = 1; k < D; k++) {
    dist[k] = (int)floor(c*pow(k, t));
    printf("%d ", dist[k]);
  }
```

```
  printf("\n");

  printf("Degree Sequence: ");
  int i = 0;
  for (int k = 0; k < D; k++) {
    for (int j = 0; j < dist[k]; i++, j++) {
      degree[i] = k;
      printf("%d ", degree[i]);
    }
  }

  // cm: match edges
  for (int i = 1; i < N; i++) { // pick the first vertex
    while (degree[i] > 0) {
      degree[i]--;
      int r = (rand() % (N-1)) + 1;  // pick a random vertex
      if (degree[r] > 0) {  // is there still room to match?
        am[i][r] = 1;
        am[r][i] = 1;
        degree[r]--;
      }
    }
  }

  return 0;
}
```

Assumptions: The computation of the degree sequence is added in the code above for your convenience. It is not necessary to write out in the exam. You may just assume that "degree" contains the correct degree sequence. Only the final for loop (which is labeled with cm) is needed as answer.
Imperfections and Limitations: This code may create only an approximate matching. Self-loops and multi-edges may occur. They should later on be fixed in some way.

6b. What is the computational time complexity of the program?

Solution: This bare-bones CM program consists of a double loop, so the time complexity is $O(n^2)$. However, if you would have put in extra checks for self-loops and multi-edges, or added more rnadomness, the time complexity might be different. So this answer will also depend on the code that you wrote previously.

7. Consider the configuration model $\mathrm{CM}_n(\vec{K})$ with n vertices and degree sequence $\vec{K} = (K_1, \ldots, K_n)$ whose components are i.i.d. random variables

with probability distribution function f given by

$$f(2) = f(3) = \tfrac{1}{2}, \qquad f(k) = 0 \text{ otherwise},$$

conditioned to satisfy $K_1 + \cdots + K_n =$ even. A "hacker" removes each vertex of degree 3 with probability $q \in (0,1)$, independently of other vertices. For what values of q does the "mutilated" random graph percolate for $n \to \infty$ (i.e., the largest connected component has a size of order n with a probability tending to 1 as $n \to \infty$)?

Solution: The mutilated random graph percolates if and only if

$$\bar{\nu} = \frac{\sum_{k\in\mathbb{N}} k(k-1)f(k)\pi(k)}{\sum_{k\in\mathbb{N}} kf(k)} > 1,$$

where $\pi(k)$ is the probability that a vertex with degree k is not removed. In our case, $\pi(3) = 1 - q$ and $\pi(k) = 1$ for all $k \neq 3$. Hence

$$\bar{\nu} = \frac{1 + 3(1-q)}{1 + \frac{3}{2}}.$$

Consequently, $\bar{\nu} > 1$ if and only if $q < \frac{1}{2}$.

8. Consider the contact process with parameter $\lambda \in (0,\infty)$ on the triangle. Let $X(t)$ denote the total number of infections at time t. Define $e(j) = \mathbb{E}(\tau_0 | X(0) = j)$, $j = 0,1,2,3$, where $\tau_0 = \inf\{t \geq 0 \colon\ X(t) = 0\}$ is the time to extinction.

   8a. Write down the transition rates for the continuous-time Markov process $X = (X(t))_{t\geq 0}$.

   Solution: The transition rates $r(i,j)$ from i to j are: $r(3,2) = 3$, $r(2,1) = 2$, $r(1,0) = 1$, $r(1,2) = 2\lambda$, $r(2,3) = 2\lambda$, and all other rates zero.

   8b. Write down four equations linking $e(j)$, $j = 0,1,2,3$.

   Solution: The average time spent in state i is $1/\sum_{j\neq i} r(i,j)$. Hence we have

$$\begin{aligned}
e(0) &= 0,\\
e(1) &= \frac{1}{1+2\lambda} + \frac{1}{1+2\lambda}e(0) + \frac{2\lambda}{1+2\lambda}e(2),\\
e(2) &= \frac{1}{2+2\lambda} + \frac{2}{2+2\lambda}e(1) + \frac{2\lambda}{2+2\lambda}e(3),\\
e(3) &= \frac{1}{3} + e(2).
\end{aligned}$$

   8c. Use these equations to compute $e(j)$, $j = 0,1,2,3$.

   Solution: The first two equations, respectively, the last two equa-

tions combine to give

$$e(1) = \frac{1}{1+2\lambda}\,[1 + 2\lambda e(2)],$$
$$e(2) = \frac{1}{2+2\lambda}\,\left[1 + 2e(1) + 2\lambda\left(\frac{1}{3} + e(2)\right)\right].$$

Substitution of the first into the second yields

$$e(2) = \frac{3}{2} + \frac{4}{3}\lambda + \frac{2}{3}\lambda^2,$$

from which in turn it follows that

$$e(1) = \frac{1}{1+2\lambda}\,\left[1 + 3\lambda + \frac{8}{3}\lambda^2 + \frac{4}{3}\lambda^3\right], \qquad e(3) = \frac{11}{6} + \frac{4}{3}\lambda + \frac{2}{3}\lambda^2.$$

9. A real-world network is represented as a simple (i.e., with no multiple edges and self-loops) undirected graph $\mathbf{G}^*$ with $n = 5$ vertices. Park and Newman, not satisfied with the model used by their colleagues Chung and Lu in Problem 4, decide they want to compare their own model with the real-world network. Park and Newman define their connection probabilities $\{p_{ij}\}$ (with $f(x_i) \equiv x_i$ and $p_{ii} \equiv 0\ \forall i$) and, after computing the hidden variables $\{x_i^*\}$ on the real network using the Maximum Likelihood principle, they find that $p_{34} > p_{53}$, $p_{15} > p_{25}$, $p_{43} > p_{14}$, $p_{52} = p_{45}$.

   Note: This problem refers to Problem 4, but is completely independent of the solution of Problem 4, so the two can be solved in any order.

   9a. Find the degree sequence $\vec{k}(\mathbf{G}^*)$ of the real-world network $\mathbf{G}^*$. Explain your result.

   Solution: In the Park-Newman model, the connection probabilities are $p_{ij} = x_i x_j/(1 + x_i x_j)$ for $i \neq j$ and $p_{ii} = 0$. Therefore $p_{il} > p_{jl}$ implies $x_i > x_j$, and $p_{il} = p_{jl}$ implies $x_i = x_j$ (for $l \neq i, j$). From the (in)equalities given in the text, we can therefore conclude that

$$x_3 > x_1 > x_2 = x_4 > x_5.$$

   Moreover, we know that fixing the hidden variables to the values $\{x_i^*\}$ dictated by the Maximum Likelihood principle leads to the condition $\langle k_i \rangle \equiv \sum_{j \neq i} x_i^* x_j^*/(1 + x_i^* x_j^*) = k_i(\mathbf{G}^*)$ for all i, where $x_i^* \geq x_j^*$ implies $k_i(\mathbf{G}^*) \geq k_j(\mathbf{G}^*)$. We can therefore conclude that

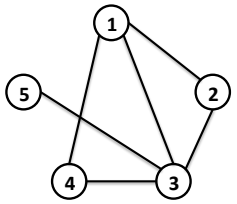
$$k_3 > k_1 > k_2 = k_4 > k_5,$$

   exactly as in 4a. Following the same arguments there, this piece of information is enough to conlude that the degree sequence of $\mathbf{G}^*$ is

$$\vec{k}(\mathbf{G}^*) = (3, 2, 4, 2, 1).$$

9b. Draw $\mathbf{G}^*$. Explain your result.

Solution: It is easy to check that there is only one graph consistent with the degree sequence found in 9a. This graph, which is necessarily $\mathbf{G}^*$, is:



9c. If $\mathbf{P}$ denotes the matrix having the numerical values of $\{p_{ij}\}$ as entries, calculate the marginals of $\mathbf{P}$, defined as the n row sums $\sum_{j=1}^{n} p_{ij}\ \forall i$.

Solution: The Maximum Likelihood condition $\sum_{j\neq i} p_{ij} = k_i(\mathbf{G}^*)$ $\forall i$ directly implies that the i-th row sum of $\mathbf{P}$ coincides with the empirical degree $k_i(\mathbf{G}^*)$. Therefore the n row sums coincide with the entries of the degree sequence $\vec{k}(\mathbf{G}^*) = (3, 2, 4, 2, 1)$.

9d. Calculate the average nearest-neighbour degree $k_i^{\mathrm{nn}}(\mathbf{G}^*)$ for each node i of $\mathbf{G}^*$.

Solution: From the definition of average nearest-neighbour degree and from the topology of $\mathbf{G}^*$ (see 9b), it is easy to calculate

$$\vec{k}^{\mathrm{nn}}(\mathbf{G}^*) = \left(\frac{8}{3}, \frac{7}{2}, \frac{8}{4}, \frac{7}{2}, 4\right).$$

9e. Calculate the clustering coefficient $C_i(\mathbf{G}^*)$ of each node i for $\mathbf{G}^*$. (For nodes with degree 1, conventionally set the clustering coefficient to 0.)

Solution: From the definition of clustering coefficient and from the topology of $\mathbf{G}^*$ (see 9b), it is easy to calculate

$$\vec{C}(\mathbf{G}^*) = \left(\frac{2}{3}, 1, \frac{1}{3}, 1, 0\right).$$

9f. Let $P(\mathbf{G}^*)$ be the probability of occurrence of the real-world network $\mathbf{G}^*$ in the model used by Park and Newman. Write $P(\mathbf{G}^*)$ as a function of the degree sequence of $\mathbf{G}^*$ and discuss qualitatively what happens to $P(\mathbf{G}^*)$ when the parameters $\{x_i\}$ are varied.

Solution: Since edges are independent in the Park-Newman model, the probability of the graph $\mathbf{G}^*$ is

$$P(\mathbf{G}^*) = \prod_i \prod_{j>i} p_{ij}^{g_{ij}} (1 - p_{ij})^{1-g_{ij}} = \left(\prod_i x_i^{k_i(\mathbf{G}^*)}\right) \prod_i \prod_{j>i} \frac{1}{1 + x_i x_j},$$

where g_{ij} is the entry of the adjacency matrix of $\mathbf{G}^*$. The Maximum Likelihood ensures that $P(\mathbf{G}^*)$ is maximized when $x_i = x_i^* \; \forall i$. When each parameter x_i is varied away from x_i^*, the probability of the graph $\mathbf{G}^*$ decreases.

10. 10a. Provide an algorithm to compute the Empirical Average Nearest Neighbor degree. Assume as data structure an adjacency *matrix*.

Solution:

```
#define N ...   // number of vertices
int am[N][N];   // adjacency matrix
int degree[N];
double annd[N];

//compute degrees for each vertex
for (int i = 0; i < N; i ++) {
  degree[i] = 0;
  for (int j = 0; j < N; j ++) {
    degree[i] += am[i][j];
  }
}

//compute average degrees of all neighbors
for (int i = 0; i < N; i ++) { // i loops through all vertices
  annd[i] = 0.0;              // set annd to zero, keep a running total
  int n = 0;                  // count of neighbours
  for (int j = 0; j < N; j ++) { // j loops through all neighbours
    if (am[i][j] == 1) { // are we a neighbour?
      annd[i] += degree[j];
      n++;
    }
  }
  // annd has the total, we need the average
  annd[i] /= n;
}
```

10b. What is the computational time complexity of the program? Why?

Solution: The algorithm contains two double nested loops, so quadratic in n: $O(n)$.

11. 11a. Provide an algorithm to compute the Empirical Average Nearest Neighbour degree. Assume as data structure an adjacency *list*.

Solution: This code is in C. C has no convenient built in list handling. C++, Java, Pyhton do have this. I will just assume a Pseudocode that has a length operation built in, and an is-member function.

```
#define N ...   // number of vertices
```

```
int *al[N];     // adjacency list with a built-in length operation
int degree[N];
double annd[N];

//compute degrees for each vertex
for (int i = 0; i < N; i ++) {
  degree[i] = length(al[i);
}
// really superfluous, just for reasons of clarity

//compute average degrees of all neighbors
//naive version, that does a double loop through all possible vertices.
for (int i = 0; i < N; i ++) { // i loops through all vertices
  annd[i] = 0.0;              // set annd to zero, keep a running total
  int n = 0;                  // count of neighbours
  for (int j = 0; j < N; j ++) { // j loops through all neighbours. faster is pos
    if (is_member(al[i], j)) { // are we a neighbour?
      annd[i] += degree[j];
      n++;
    }
  }
  // annd has the total, we need the average
  annd[i] /= n;
}
```

11b. What is the computational time complexity of the program? Why?

Solution: The algorithm contains two double nested loops, so quadratic in n: $O(n)$.
However, a faster solution to the inner loop is possible (not shown), that loops in m the number of edges, which may be less in a sparse network. Then the complexity becomes $O(nm)$.