CS9668/4438 Internet Algorithmics
Assignment 5
Due December 7

1. In Chord, whenever a new processor p joins the system it needs to compute the addresses that will be stored in its finger table. To do this, processor p needs to know the address of at least one other processor q in the system. Note that q is not necessarily the successor of p.

   - (15 marks) Explain how to compute the address that processor p must store in the i-th entry of its finger table, given that it knows the address of a processor q. Remember that processor q and all other processors in the system, except p, already have computed their finger tables. You need to explain what messages need to be sent and you need to explain why your solution correctly computes the address of the i-th finger of processor p.
   - (30 marks) Write a program in Java using the simulator for distributed systems that computes the addresses that must be stored in the finger table of a processor p that has just joined the system. The program must return a string of the form: "F_1, F_2, ..., F_m", where F_i is the address stored in the i-entry of the finger table.

     Assume that the hash function for the processor addresses is $h_p(ID) = ID \bmod 2^m$, where the ring of identifiers has size 2^m. You can use the code provided in FindFingerTable.java to compute the value of m and the finger tables of all processors. Since the simulator cannot simulate processors joining the peer-to-peer system, in the configuration file there will be a processor p whose finger table has all entries equal to 1 (value 1 is the address of a processor in the network). This processor p is the one that is joining the system and 1 is the address of the only processor q that p knows.

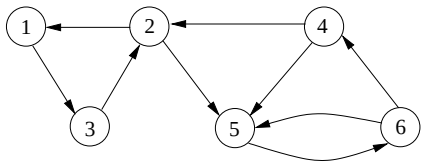
     Once processor p has computed all the addresses in its finger table it must send an END message to its successor and then terminates. END messages are always forwarded to the successor and cause the sending processor to terminate.

     If you cannot write code in Java we will allow you to submit a solution in detailed pseudocode, but you are strongly encouraged to write a program in Java to answer this question.

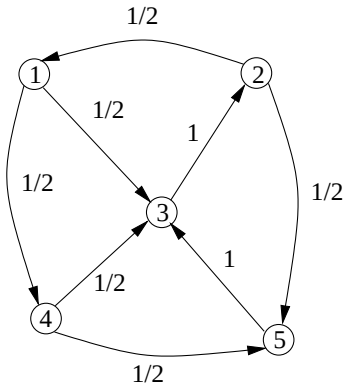
     Note. Your algorithm does not need to update the finger tables of all other processors so they include the address of processor p; you only need to compute the finger table for p.

     Hint. Use the Java code that you wrote for Question 4 of Assignment 4.

2. (8 marks) Compute the (simple) page rank of every node in the following graph. Indicate which method you used to compute the page rank.



3. (7 marks) Compute the stationary distribution of the following Markov chain. Explain how you computed the stationary distribution.



4. In class we talked about the problem of ranking pages to be returned by web search engines. Ideally high-quality content pages should be assigned high ranks, and low-quality content pages should be assigned low ranks.

   Let $r(i)$ denote the page rank of page i, and let n be the total number of pages in the Web graph. Consider the following 3 ways of defining $r(i)$:

   a. (5 marks) $r(i) = 1/n$.
   b. (10 marks) $r(i) = \text{maximum}\{r(j) \mid j \in B(i)\}$, where $B(i)$ is the set of pages that have hyperlinks pointing to page i.
   c. (10 marks) $r(i) = \frac{1}{n} + \sum_{j \in B(i)} r(j)$.
   d. (15 marks) $r(i) = \text{maximum}\{r(j) \mid j \in B(i)\} + \text{minimum}\{r(j) \mid j \in B(i)\}$, where $B(i)$ is the set of pages that have hyperlinks pointing to page i. If $B(i)$ has only one page j, then $\text{maximum}\{r(j) \mid j \in B(i)\} = \text{minimum}\{r(j) \mid j \in B(i)\} = r(j)$.

   For this question you will **assume** that the Web Graph is strongly connected. For each one of the above methods indicate if (i) the page rank always exists and (ii) the method achieves the goal of assigning high rank to high-quality pages and low rank to low-quality pages. You must explain your answers. If a rank function does not work, construct an example showing that either the page rank cannot be computed or that the rank function does not correctly differentiate between high quality content and low quality content pages. Assume that a low quality page has very few references to it and a high quality page has many references to it.

   Recall that the sum of page ranks of all the pages must be 1.