Distributed Algorithms
Fall 2020

Links & Gossip - solutions
1st exercise session, 28/09/2020

Matteo Monti <matteo.monti@epfl.ch>
Jovan Komatovic <jovan.komatovic@epfl.ch>
Graphs

A graph is a couple \((V, E)\) where \(V\) is a set of vertices and \(E \subseteq V^2\) is a set of edges.

Example graph \((V, E)\):

- \(V = \{a, b, c, d, e\}\)
- \(E = \{(a, b), (b, c), (b, e), (e, d)\}\)

Two vertices are adjacent (or neighbors) iff an edge exists between them. In the example, \(a\) and \(b\) are adjacent; \(a\) and \(d\) are not adjacent.
Graphs (undirected)

An **undirected graph** is a graph \((V, E)\) such that \((a, b) \in E\) if and only if \((b, a) \in E\).

**Example graph** \((V, E)\):
- \(V = \{a, b, c, d, e\}\)
- \(E = \{(a, b), (b, a), (b, c), (c, b), (b, e), (e, b), (e, d), (d, e)\}\)

We use undirected graphs to model networks of processes:
- Each vertex represents a process
- Two vertices are neighbors iff the corresponding processes can directly exchange messages.
Paths

A **path** is a sequence of distinct vertices \((v_1, \ldots, v_N)\) such that, for all \(i \in [1, N - 1]\), \(v_i\) and \(v_{i+1}\) are adjacent.

Some paths in \((V, E)\):

- \((a, b)\)
- \((a, b, c)\)
- \((a, b, e, d)\)

While

- \((a, c, e)\) is **not** a path: a and c are not adjacent!
Connectivity

Two distinct vertices $a$ and $z$ are **connected** if and only if at least one path $(a, …, z)$ exists in the graph. A graph is connected if any two distinct vertices are connected.

A connected graph

A disconnected graph
Exercise 1 (connectivity)

Prove that connectivity is a symmetric property on an undirected graph: let $a, b$ be vertices such that $a$ is connected with $b$. Prove that $b$ is connected with $a$.

Hint: you can do it constructively.
Exercise 1 (solution)

- If $a$ is connected to $b$, then a path $p$ exists from $a$ to $b$. Let $p = (a, v_1, \ldots, v_N, b)$.

- Since the graph is undirected, if $v$ is adjacent to $w$, then $w$ is adjacent to $v$.

- Therefore, the sequence $p' = (b, v_N, \ldots, v_1, a)$ is also a path.

- Since $p'$ begins in $b$ and ends in $a$, a path exists between $b$ and $a$. Consequently, $b$ is connected to $a$. 
Exercise 2 (connectivity)

Prove that connectivity is a transitive property on an undirected graph: let $a$, $b$, $c$ be vertices such that $a$ is connected with $b$ and $b$ is connected with $c$. Prove that $a$ is connected with $c$.

*Hint: double-check the definition of a path.*
Exercise 2 (solution)

- Let $p = (v_1, \ldots, v_N)$ and $q = (w_1, \ldots, w_M)$ be the paths from $a$ to $b$ and from $b$ to $c$, respectively. We have $v_1 = a$, $v_N = w_1 = b$, $w_M = c$.
- We note that $(v_1, \ldots, v_N, w_2, \ldots, w_M)$ is in general not a path, as the vertices are not guaranteed to be disjoint.
- If $a \in q$, then $a$ and $c$ are trivially connected. Indeed, a subpath $q' = (w_K, \ldots, w_M)$ already exists in $q$ such that $w_K = a$ and $w_M = c$.
- If $\neg(a \in q)$, then let $v_K = w_H$ be the first element of $p$ that is also in $q$. Since $v_N = w_1 = b$, $v_K$ is guaranteed to exist.
- By definition, $v_1, \ldots, v_{K-1}$ are not in $q$. Therefore, $r = (v_1, \ldots, v_K, w_{H+1}, \ldots, w_M)$ is a path.
- Since $r$ begins in $a$ and ends in $c$, $a$ and $c$ are connected.
Exercise 3 (connectivity)

Write a procedure (pseudocode or any programming language) that inputs an undirected graph $G = (V, E)$ and outputs $true$ if and only if the $G$ is connected.

*Hint: use the results from Exercises 1 and 2.*
Exercise 3 (solution)

We start by noting that, since connectivity is symmetric and transitive, we only need to check if any node is connected to every other. We can implement the following algorithm:

● Pick any vertex \( v \) from \( V \). Initialize a frontier set \( F = \{v\} \). Initialize an interior set \( I = \emptyset \).
● Until \( F \) is empty:
  ○ Pick an element \( f \) from \( F \). Remove \( f \) from \( F \), add \( f \) to \( I \).
  ○ For every neighbor \( n \) of \( f \):
    ■ If \( \neg(n \in F \cup I) \), add \( n \) to \( F \).
● If \( I = V \), then \( G \) is connected.

Let \( w \) be any vertex, if and only if path exists between \( v \) and \( w \), then \( w \) is eventually added to \( F \), then removed from \( F \) and added to \( I \). If eventually \( I = V \), then every vertex is connected to \( v \), and consequently \( G \) is connected.
Gossip

We use an undirected graph to represent which processes can communicate. Upon receiving a new message $m$, a process forwards $m$ to all its neighbors.

Example: diffusion of a message $m$ from process $e$.

- $e$ issues $m$
Gossip

We use an undirected graph to represent which processes can communicate. Upon receiving a new message $m$, a process forwards $m$ to all its neighbors.

Example: diffusion of a message $m$ from process $e$.

- $e$ issues $m$.
- $b$ and $d$ receive $m$. 
Gossip

We use an undirected graph to represent which processes can communicate. Upon receiving a new message $m$, a process forwards $m$ to all its neighbors.

Example: diffusion of a message $m$ from process $e$.

- $e$ issues $m$.
- $b$ and $d$ receive $m$.
- $a$ and $c$ receive $m$.

Gossip is correct if and only if, if the sender is correct, every correct process eventually receives the message.
Exercise 4 (gossip)

Prove that gossip is correct if and only if the subgraph of correct processes is connected.

Note: prove both directions of the implication!

Hint: induction is your friend.
Exercise 4 (solution)

*If the subgraph of correct processes is connected, then gossip is correct.*

Let $G = (V, E)$ be the gossip network, let $N = |V|$, let $s$ be the sender. By induction:

- Let $s$ be the sender. We obviously have that $s$ eventually delivers the message $m$.
- Let $V_L$ denote the set of vertices that are connected to $s$ by a path no longer than $L$. We have $V_0 = \{s\}$.
- Let $N_L$ denote the set of vertices that have at least one neighbor in $V_L$. If every process in $V_L$ eventually delivers $m$, then also every process in $N_L$ delivers $m$ (as $m$ is sent to every neighbor).
- Since $N_L \cup V_L = V_{L+1}$, if every process in $V_L$ eventually delivers $m$ then every process in $V_{L+1}$ eventually delivers $m$.
- Since all the vertices in a path are distinct, no path longer than $N$ can exist on the gossip path. Therefore, $V = V_N$. Consequently, every node in $V$ eventually delivers $m$. 
Exercise 4 (solution)

If gossip is correct, then the subgraph of correct processes is connected.

Let $G = (V, E)$ be the gossip network, let $N = |V|$, let $s$ be the sender.

- Let $v \neq s$ be a correct process. Regardless of the crashes, $v$ eventually delivers $m$. Therefore, $v$ eventually receives $m$ from a correct process.
- We use induction similarly to the previous slide, defining $W_L$ as the set of processes that are connected to $v$ by a path not longer than $L$.
- Let $i \in [0, N]$. If $W_i$ includes $s$, then $v$ is connected to $s$.
- If $W_i$ does not include $s$, then at least one process in $W_i$ eventually receives $m$ from one of its neighbors, and that neighbor is not in $W_i$.
- Since the size of $W_i$ is strictly increasing until $W_i$ includes $s$, we have that $W_N$ must include $s$.
- Since this holds true for every $v$, every process is connected to $s$, making the subgraph of correct processes connected.
Exercise 5 (gossip)

In the following system, exactly one process crashes. What is the minimum number of edges we need to add so that gossip is always correct?
Exercise 5 (solution)

In the following system, exactly one process crashes. What is the minimum number of edges we need to add so that gossip is always correct?
**k-connectivity**

Two paths $p, p'$ connecting two vertices $a$ and $z$ are **disjoint** if they have no vertex in common, except $a$ and $z$:

\[ p = (a, b, ..., y, z) \]

\[ p' = (a, b', ..., y', z) \]

\[ \{a, b, ..., y, z\} \cap \{a, b', ..., y', z\} = \{a, z\} \]

A graph is **$k$-connected** if and only if $k$ disjoint paths exist between any two vertices of the graph.
Robustness

Gossip is **robust** to \( k \) failures if and only if it is always correct, as long as no more than \( k \) nodes are crashed.

A fully connected gossip graph is robust to \( N \) failures, where \( N \) is the number of processes.
Exercise 6 (robustness)

Prove that, if the gossip graph is \((k+1)\)-connected, then gossip is \(k\)-robust.

Is the converse also true? Find a counterexample if not.

*Hint: contradiction is your friend.*
Exercise 6 (solution)

- By contradiction, let us assume that gossip is \((k + 1)\)-connected, but \(k\) processes exist such that, if they all crash, then two correct processes \(a\) and \(b\) are no longer connected.
- By hypothesis, \((k + 1)\) distinct paths \(p_1, \ldots, p_{k+1}\) exist between \(a\) and \(b\).
- If some \(i\) exists such that no process crashes in \(p_i\), then \(a\) and \(b\) are still connected by correct processes, and (as we proved in Exercise 4) they can gossip with each other.
- Since \(p_1, \ldots, p_{k+1}\) are all distinct, at least one distinct process must crash in each \(p_i\) for \(a\) and \(b\) to be disconnected. But at most \(k\) processes can crash!
Exercise 6 (solution)

Technically:

But does it still work for $N > 2$?
Random failures

Suppose that processes can fail independently with probability $f$.

What is the probability that two *correct* processes can communicate in the presence of failures?

*It depends on their connectivity!*

e.g. $\alpha$, $\beta$ can communicate iff $x$ has not failed $\Rightarrow$

$\alpha$, $\beta$ communicate with probability $1-f$. 
Exercise 7 (random failures on series topology)

Suppose that processes $x_i, i=1, \ldots, n$ can fail independently with probability $f$.

What is the probability that $a$ and $b$ can communicate?
Exercise 7 (solution)

- Each process survives (i.e., it does not fail) with independent probability $(1 - f)$.
- Therefore, all processes survive with probability $(1 - f)^n$. 
Exercise 8 (random failures on parallel topology)

Suppose that processes $x_i$, $i=1, \ldots, n$ can fail independently with probability $f$.

What is the probability that $a$ and $b$ can communicate?
Exercise 8 (solution)

- Each process fails with independent probability $f$.
- Therefore, all processes fail with probability $f^n$.
- Finally, at least one process survives with probability $(1 - f^n)$. 
Exercise 9 (random failures on series/parallel topology)

Suppose that processes $x_{ij}$, $i=1, \ldots, n$, $j=1, \ldots, m$ can fail independently with probability $f$.

Prove that $a$ and $b$ can communicate with probability $1 - [1 - (1-f)^m]^n$. 

Exercise 9 (solution)

- As we proved in Exercise 7, every *branch* fails with independent probability $g = 1 - (1 - f)^m$.
- We can now consider each *branch* as if it was one of the processes in Exercise 8. The probability that no branch fails is $1 - g^n = 1 - [1 - (1-f)^m]^n$. 
Erdös-Renyi graphs

An Erdös-Renyi graph $G(N, p)$ is a random undirected graph with $N$ vertices, such that any two distinct vertices have an independent probability $p$ of being adjacent.

An Erdös-Renyi graph is defined by the values of $N(N - 1)/2$ independent Bernoulli random variables:

$$E_{ij} \sim \text{Bernoulli}(p)$$

$$E_{ij} = E_{ji}$$

with $i, j \in V$. Vertices $i$ and $j$ are adjacent iff $E_{ij} = 1$. 
What distribution underlies the number of edges in an Erdös-Renyi $G(N, p)$? What distribution underlies the degree (i.e., number of links) of any vertex? Are the degrees of any two vertices independently distributed?

*Hint: how is the sum of Bernoulli variables distributed?*
Connectivity of $G(N, p)$

Let $C(N, p)$ denote the probability of a random graph $G(N, p)$ being connected. It is possible to prove that:

\[
\lim_{N \to \infty} G(N, p) = 0 \quad \text{iff} \quad p < \ln(N) / N
\]

\[
\lim_{N \to \infty} G(N, p) = 1 \quad \text{iff} \quad p > \ln(N) / N
\]

A large Erdös-Renyi graph is almost surely connected, as long as each vertex has an expected degree larger than $\ln(N)$.

We can use Erdös-Renyi graphs to build probabilistic gossip with logarithmic communication complexity!
Bonus Exercise 11 (Erdös-Renyi graphs)

Write a distributed procedure that runs on \( N \) processes to build an Erdös-Renyi graph \( G(N, \ln(N)/N) \). We assume no failures. Each process can invoke:

- A procedure \( \text{rand}(x) \) that returns a real number between 0 and \( x \), independently picked with uniform probability.
- A procedure \( \text{connect}(i) \) to connect to the \( i \)-th process.

Is it possible for the procedure to have \( O(\ln(N)) \) computation complexity?