SAPIENZA
UNIVERSITÀ DI ROMA

Networking for big data
Data centers

Lecturer: Andrea Baiocchi

email: andrea.baiocchi@uniroma1.it

Department of Information Engineering, Electronics, and Telecommunications
SAPIENZA University of Rome

a.y. 2015/2016

Definitions

- A graph is a couple $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is a finite, nonempty set of nodes and $\mathcal{A}$ is a subset of $\mathcal{N} \times \mathcal{N}$.
    - If $(i, j) \in \mathcal{A} \Rightarrow (j, i) \in \mathcal{A}$, we say the graph is undirected.
    - A very useful model for representing network connectivity as seen at a given architectural level.
    - The interpretation of nodes and arcs depends on the considered architectural level, e.g., nodes can be routers and arcs subnets connecting them at IP level or switches connected with physical links at layer 2.
- If a function $w : \mathcal{A} \mapsto \mathbb{R}$ is defined, we say the graph is weighted. The weight of arc (i, j) is denoted with w_{ij}.
- We define:

n	the number of nodes;
a	the number of arcs.

Laplacian

- The laplacian of a graph in an $n \times n$ matrix $\mathbf{L}$ defined as follows:
  - $L_{ii} = d_i$, where d_i is the degree of node i;
  - $L_{ij} = 1$ if and only if $(i, j) \in \mathcal{A}$.
- We can write:
$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$
- $\mathbf{L}$ is a symmetric matrix.
- The sum of each row and each column of $\mathbf{L}$ is 0. Therefore 0 is always an eigenvalue of $\mathbf{L}$

Incidence matrix

- The incidence matrix of a graph in an $a \times n$ matrix $\mathbf{M}$ defined as follows:
  - $M_{\ell i} = 1$, if $\ell = (i, j) \in \mathcal{A}$;
  - $M_{\ell j} = -1$, if $\ell = (i, j) \in \mathcal{A}$;
  - $M_{\ell j} = 0$, otherwise.
- It can be verified that:

$$\mathbf{L} = \frac{1}{2}\mathbf{M}^T\mathbf{M}$$

- Hence $\mathbf{L}$ is a positive definite matrix, i.e., $\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0$ for any vector $\mathbf{x}$.
- Positive definite and symmetric $\rightarrow$ the eigenvalues of $\mathbf{L}$ are real and non-negative.

Random graphs

- **Erdos-Renyi random graphs**: two models are possible.
  - $G(n, m)$ model: a graph is chosen uniformly at random from the collection of all graphs which have n nodes and m edges.
  - $G(n, p)$ model: a graph is constructed by connecting nodes randomly. Each edge is included in the graph with probability p independent from every other edge. ;
- ***r*-regular random graph**: a graph selected from $\mathcal{G}_{n,r}$, which denotes the probability space of all r-regular graphs on n vertices, where $3 \leq r < n$ and nr is even.
  - An r regular graph is a graph where each vertex has the same number r of neighbors.
  - r regular graphs for $r = 0, 1, 2$ are trivial.

Connectivity

- A path between nodes s and t is an ordered sequence of nodes and arcs connecting the two nodes, i.e., $(s, k_1), (k_1, k_2), \ldots, (k_{r-1}, t)$ is a path of length r.
- A graph is said to be connected if there exists a path connecting any two nodes.
- A graph is connected if one of the following statements is true
  - The adjacency matrix $\mathbf{A}$ is irreducible;
  - The second largest eigenvalue of $\mathbf{L}$ is positive;
- An $n \times n$ matrix $\mathbf{A}$ is irreducible if

$$\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^{n-1} > \mathbf{0}$$

Breadth-First-Search algorithm

Finds all paths form a given node.

Given a graph 'Graph' and a node 'root' in it:

```
for each node n in Graph:
  n.distance = INFINITY
  n.parent = NIL
endfor
create empty queue Q
root.distance = 0
Q.enqueue(root)
while Q is not empty:
  current = Q.dequeue()
  for each node n that is adjacent to current:
  if n.distance == INFINITY:
   n.distance = current.distance + 1
   n.parent = current
   Q.enqueue(n)
  endif
endwhile
```

Assignments

- Write a script to generate a random graph with all models defined above; display some results.
- Write a script to implement the breadth-first search for all nodes of the graph.
- Write a script to check the connectivity of a given graph.
  - algebraic method 1 (irreducibility);
  - algebraic method 2 (eigenvalue of the Laplacianmatrix);
  - breadth-first based algorithm.
- Write a script that generates a fat tree topology graph and finds all paths connecting a given source ToR and a given destination ToR. Find also all *disjoint* paths.