Cloud Computing — Project Series 1: Fat-Tree Topology

Instructor: Dr. Alireza Shamali
Submission: Upload through the course LMS only.

Submission format

- Submit a single code file named `STDNUM_FAMILY_NAME.<ext>` (replace placeholders with your own info).
- Include all generated data/figures as separate files alongside your code.
- Use English for all filenames, comments, and outputs.

Topic

Implement and explore a classic **Fat-Tree** data center topology. The reference figure in class uses **k = 4** and a fixed node-numbering scheme. Keep your node IDs consistent with that scheme so your results are checkable.

This series has **two phases**.

Phase 1 — Build and Inspect the Topology (k = 4)

Task 1. After the user enters `k`, construct the k-ary Fat-Tree graph (core, aggregation, edge, and host layers).
2. Model the network so your program **shows only links that actually exist** between the numbered nodes. If a link does **not** exist, nothing should be shown (e.g., there is no direct link between nodes `0` and `1`, or between `0` and `15`).

Output - A table/CSV that lists existing links, one row per edge. Example columns: `Connection`, `Node 1`, `Node 2` (you may add an `Arc Exists` boolean column if you prefer).
- A visualization of the fat-tree layered by node type.

Notes - Keep node numbering exactly consistent with the class reference figure.
- Use clear, English column names in your CSV.

Phase 2 — Enumerate All Shortest Paths Between Two Nodes

Task 1. After the user enters two node IDs (source and destination), compute the **subgraph containing all shortest paths** between them.
2. Display those paths. If the two nodes are in **different pods**, there will be **(k/2)^2** equal-cost shortest paths.

Example - With the reference numbering for `k = 4`, nodes `0` and `8` lie in different pods; there are **4** distinct shortest paths between them, each via a different core switch.

Output - A CSV listing every shortest path as a sequence of node IDs from source to destination (one row per path).
- A printed list and/or a visualization that highlights exactly those shortest paths.

What to turn in

- Your code file(s) named per the submission rule.
- `fat_tree.csv` (Phase 1 link table).
- `all_shortest_paths.csv` (Phase 2 results).
- Any figures you generated (e.g., PNG of the topology or highlighted paths).
- A brief `README` describing how to run your code and how the outputs are structured.

Technical guidance (non-binding)

- Any language is acceptable; **Python + NetworkX** is recommended for speed of implementation.
- Ensure your script accepts inputs from the user (e.g., prompt for `k`, source node, destination node).
- Validate inputs (e.g., reject invalid node IDs).
- Keep code readable and modular.

Academic integrity

Submit your own original work. Cite any external libraries or references you use.

Grading (indicative)

- Correct topology construction and numbering: **35%**
- Phase 1 link table and visualization quality: **25%**
- Phase 2 shortest-paths correctness and CSV output: **30%**
- Code clarity and documentation: **10%**