NFV VNF Placement on a Fat-Tree (Gusek/GLPK)

Toolchain: Gusek (GLPK IDE) + GNU MathProg (GMPL)

Learning goals

- Formulate an **NFV placement + routing** optimization for multiple flows on a **fat-tree** data-center topology.
- Encode objectives and constraints in **GMPL** and solve with **GLPK** via Gusek or `glpsol`.
- Support **policy toggles** (e.g., VNF colocation on/off) without editing model logic.

Inputs (what the model must accept)

- **Topology:** a k-ary fat-tree (start with **k = 4**, then extend to **k = 8** for final submission).
- **Flows:** each flow `f` has a source, a destination, a demand, and an **ordered list of required VNFs** (order may be significant).
- **Node capacities:** CPU and memory per node.
- **Link attributes:** bandwidth and traversal **cost** per directed link.
- **Policy switches (parameters):**
- `COLOCATE ∈ {0,1}` — 0 = distribute identical VNFs; 1 = allow colocating them on one node
- `AVOID_LOOPS ∈ {0,1}` — enable/disable anti-cycle enforcement
- `PATHLEN_CAP ≥ 0` — 0 = unlimited; else maximum hop count

Decision variables (suggested)

- `x[i,j,f] ∈ {0,1}` — flow `f` uses directed link `(i,j)`
- `y[n,f,m] ∈ {0,1}` — VNF module `m` of flow `f` is placed on node `n`
- Optional ordering auxiliaries to respect the VNF chain order along the chosen path

Objective (suggested)

Minimize a weighted sum of:

1. **Path cost**: sum of link costs × demand over links used by each flow
2. **Resource cost**: CPU and memory usage from placed VNFs

Core constraints (must-have)

1. **Flow conservation** for every flow (source/sink exceptions).
2. **Link capacity**: total routed demand on a link $\leq$ its bandwidth.
3. **Node capacity**: total CPU/MEM consumed by placed VNFs $\leq$ node limits.
4. **Placement on path**: every required VNF of a flow is placed on a node the flow's path actually visits.
5. **Chain feasibility**: if VNF order matters, enforce ordering along the path.

6. **No spurious functions**: place only required VNFs per flow.
7. **No cycles**: via explicit anti-cycle constraints or a hop cap.
8. **Policy toggles**: the model must run with `COLOCATE` on/off (and other toggles) **without code edits**; only parameter values change.

Topology requirement

- Demonstrate first on **k = 4**; finalize and submit results for **k = 8**.

What to turn in

- GMPL **model** (`.mod`) file (and optionally a separate `.dat` file if you externalize data).
- Any helper **scripts** used to generate node/link data.
- A short **run log** (solver output) and a brief textual summary of the solution: objective value, where VNFs placed, and a per-flow path summary.

Hints

- You may start from the in-class **TSP** template in Gusek and adapt it to this problem structure.
- Keep node/link/flow data in clearly named tables (either in the `.dat` file or inline in the model's `data; ... end;`).
- Make policy switches true parameters so the grader can flip scenarios in seconds without touching constraints.

Deliverable language: English.
File naming: follow the course's convention when submitting to the LMS; the public repo can use descriptive names.