# Lam Nguyen Network

Tasks:

- Read input → data struct. $\Theta(n)$ ↳ read every entry
  (pref. hash table/dict.) → Python?
  - easy access
  - fast for large network
- Use Dij.'s algorithm to creat rout. table
  $\Theta(V^2)$
- Output results

$\boxed{\text{Most time consuming}}$

Input: (format)

| | | |
|---|---|---|
| • node1 | node2 | link-cost |
| node3 | node4 | ___ |
| ___ 2 | ___ 4 | ___ |

$\boxed{\text{Size = ?}}$

Out put:

• Node A:
  Node_B: ...
  ___ C: ...

• Node B:
  Node_C: ...
  ___ D: ...

...

(☆) Assign weight to params:

- Simple algorithm:

$$\text{link\_cost} = \sum_{i=0}^{n} \text{Elem}_i \cdot \text{Sig}_i$$

• n = # of factors (params)
• $\text{Elem}_i$ : Value of $\text{param}_i$
  $\text{Sig}_i$ : How important the param is

(☆) If there is two equal routes
⇒ use counter (=0 or 1)
to prioratize & spread the traffic out.

# Dijkstra's Algorithm

. Tasks :

⊕ Create class Vertex $\Longrightarrow$ add neighbors, get info ( weight, id, ...), set info

⊕ Create class Graph
- ↳ add vertex, edge
- ↳ get infos

⊕ Use Dijkstra's algorithm to find the shortest path

↳ heapq ?