南方科技大學
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Algorithm Design and Analysis (H)

CS216

Prof. Shiqi Yu (于仕琪)

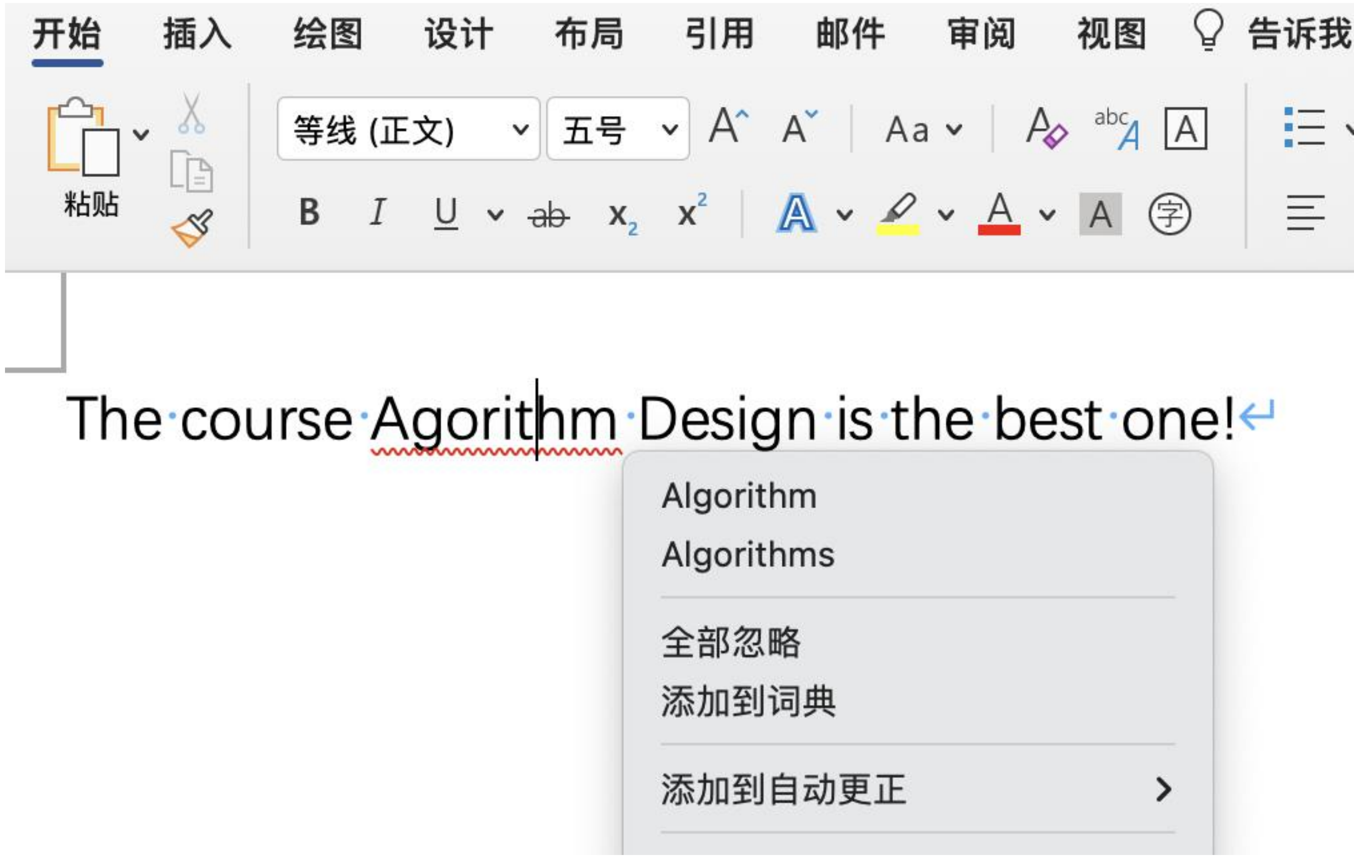
yusq@sustech.edu.cn

http://faculty.sustech.edu.cn/yusq/

Dynamic Programming

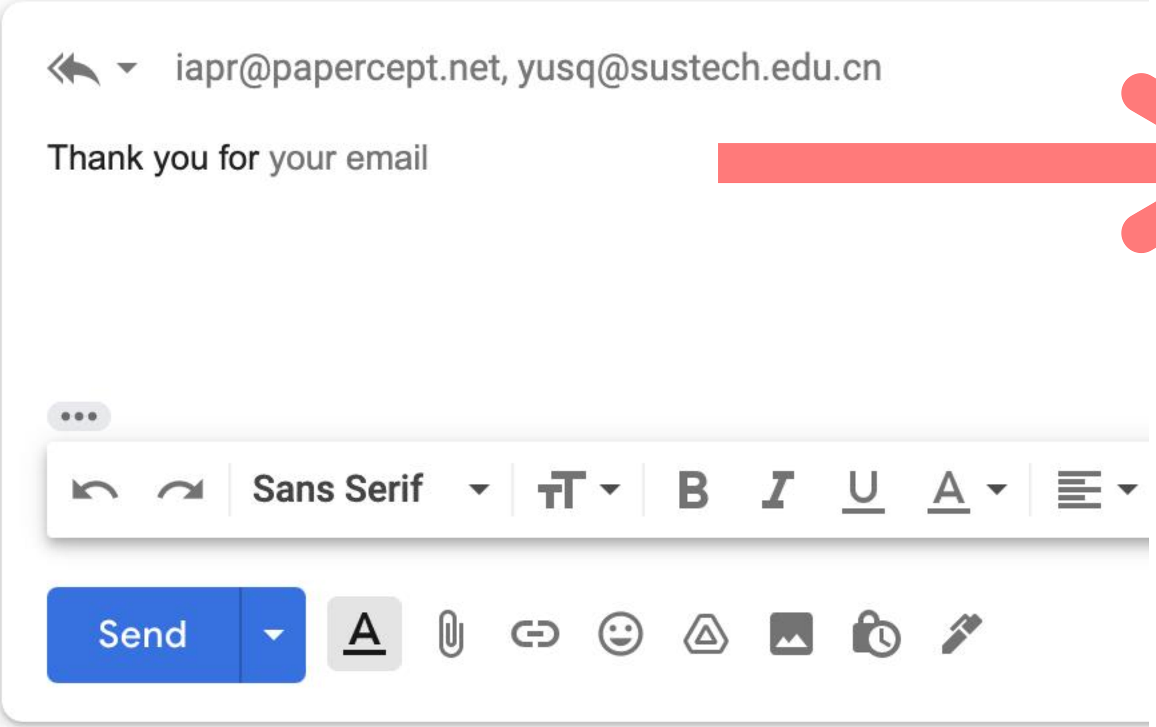
5. Sequence Alignment

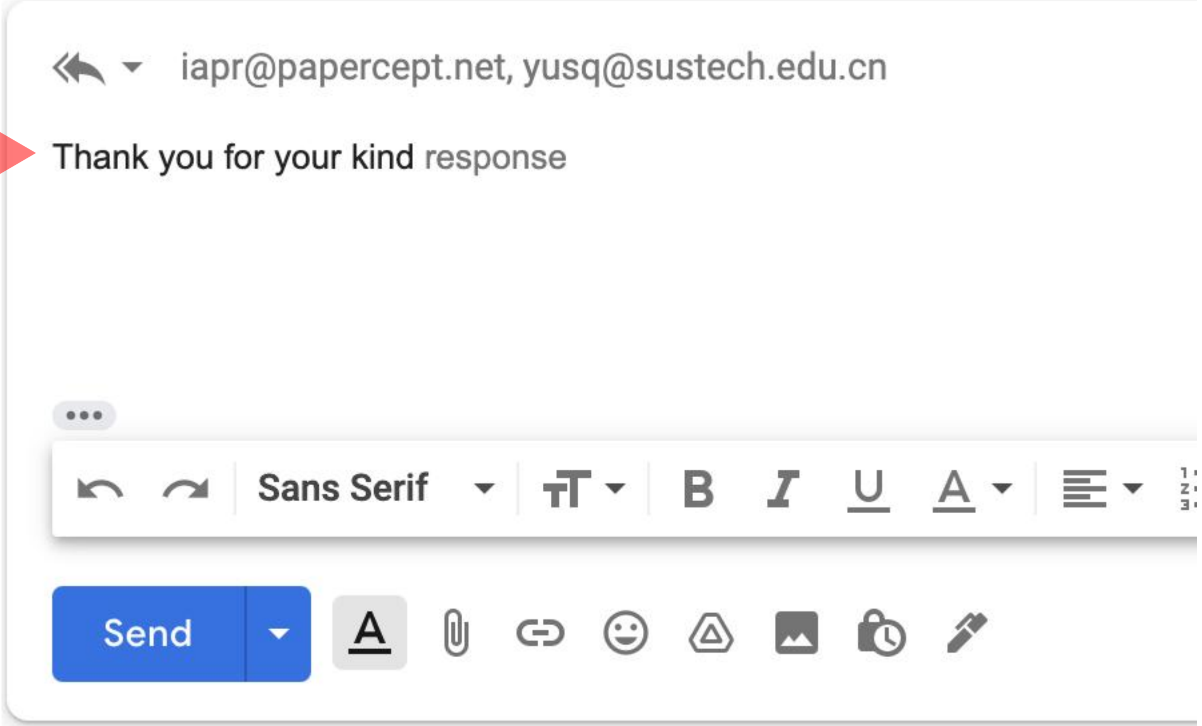
Correct Typos Automatically



It can do more …

- Gmail





String Similarity

- How similar are two strings?
  - ocurrance
  - occurrence

o	c	u	r	r	a	n	c	e	–
o	c	c	u	r	r	e	n	c	e

6 mismatches, 1 gap

o	c	–	u	r	r	a	n	c	e
o	c	c	u	r	r	e	n	c	e

1 mismatch, 1 gap

o	c	–	u	r	r	–	a	n	c	e
o	c	c	u	r	r	e	–	n	c	e

0 mismatches, 3 gaps

Edit Distance

- Applications.
  - Basis for Unix diff.
  - Speech recognition.
  - Computational biology.

- Edit distance. [Levenshtein 1966, Needleman-Wunsch 1970]
  - Gap penalty δ; mismatch penalty α_{pq}.
  - Cost = sum of gap and mismatch penalties.

C	T	G	A	C	C	T	A	C	C	T
C	C	T	G	A	C	T	A	C	A	T

$\alpha_{TC} + \alpha_{GT} + \alpha_{AG} + 2\alpha_{CA}$

-	C	T	G	A	C	C	T	A	C	C	T
C	C	T	G	A	C	-	T	A	C	A	T

$2\delta + \alpha_{CA}$

Sequence Alignment

- Goal: Given two strings $X = x_1\, x_2\, . . .\, x_m$ and $Y = y_1\, y_2\, . . .\, y_n$ find alignment of minimum cost.

- Def. An alignment M is a set of ordered pairs x_i-y_j such that each item occurs in at most one pair and no crossings.

- Def. The pair x_i-y_j and $x_{i'}$-$y_{j'}$ cross if i < i', but j > j'.

- Ex: CTACCG vs. TACATG.

Sol: $M = x_2\text{-}y_1,\ x_3\text{-}y_2,\ x_4\text{-}y_3,\ x_5\text{-}y_4,\ x_6\text{-}y_6$.

x_1	x_2	x_3	x_4	x_5		x_6
C	T	A	C	C	–	G
–	T	A	C	A	T	G
	y_1	y_2	y_3	y_4	y_5	y_6

Sequence Alignment: Problem Structure

- Def. OPT(i, j) = min cost of aligning strings $x_1\ x_2 \ldots x_i$ and $y_1\ y_2 \ldots y_j$.
  - Case 1: OPT matches x_i-y_j.
    - ✓ pay mismatch for x_i-y_j + min cost of aligning two strings $x_1\ x_2 \ldots x_{i-1}$ and $y_1\ y_2 \ldots y_{j-1}$
  - Case 2a: OPT leaves x_i unmatched.
    - ✓ pay gap for x_i and min cost of aligning $x_1\ x_2 \ldots x_{i-1}$ and $y_1\ y_2 \ldots y_j$
  - Case 2b: OPT leaves y_j unmatched.
    - ✓ pay gap for y_j and min cost of aligning $x_1\ x_2 \ldots x_i$ and $y_1\ y_2 \ldots y_{j-1}$

$$
\mathrm{OPT}(i,j) = \begin{cases} j\delta, & \text{if } i = 0 \\ \min\begin{cases} \alpha_{x_i y_j} + \mathrm{OPT}(i-1, j-1) \\ \delta + \mathrm{OPT}(i-1, j) \\ \delta + \mathrm{OPT}(i, j-1) \end{cases}, & \text{otherwise} \\ i\delta, & \text{if } j = 0 \end{cases}
$$

Sequence Alignment: Algorithm

```
Sequence-Alignment(m, n, x1x2...xm, y1y2...yn, δ, α) {
   for i = 0 to m
      M[i, 0] = iδ
   for j = 0 to n
      M[0, j] = jδ

   for i = 1 to m
      for j = 1 to n
         M[i, j] = min(α[xi, yj] + M[i-1, j-1],
                       δ + M[i-1, j],
                       δ + M[i, j-1])
   return M[m, n]
}
```

$$\mathrm{OPT}(i,j) = \begin{cases} j\delta, & \text{if } i = 0 \\ \min \begin{cases} \alpha_{x_i y_j} + \mathrm{OPT}(i-1, j-1) \\ \delta + \mathrm{OPT}(i-1, j) \\ \delta + \mathrm{OPT}(i, j-1) \end{cases}, & \text{otherwise} \\ i\delta, & \text{if } j = 0 \end{cases}$$

- Analysis. $\Theta(mn)$ time and space.
- English words or sentences: m, n $\leq$ 10.
- Computational biology: m = n = 100,000. 10 billions ops OK, but 10GB array?

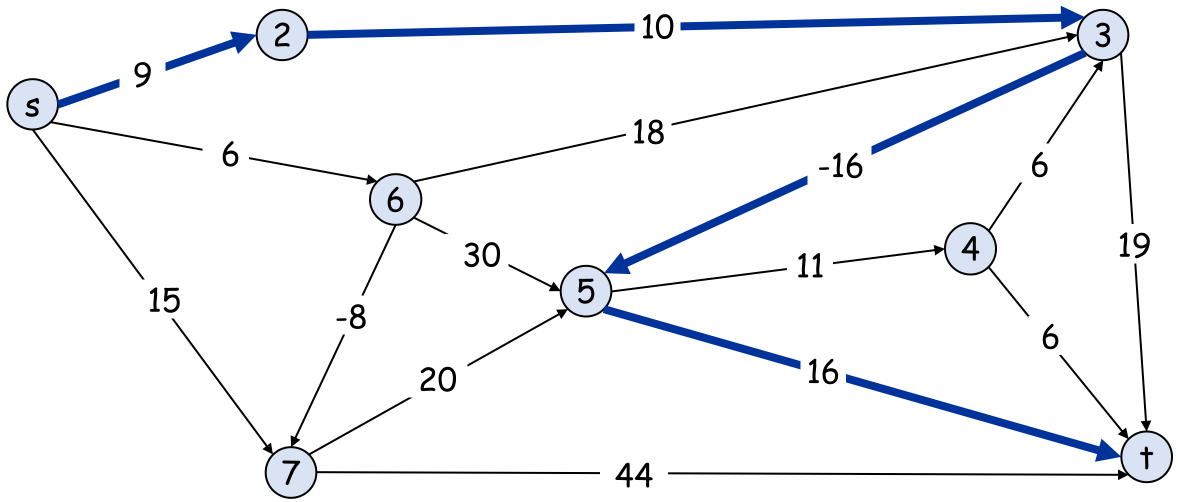
6. Shortest Paths

Shortest Paths

- Shortest path problem. Given a directed graph G = (V, E), with edge weights c_{vw}, find shortest path from node s to node t.

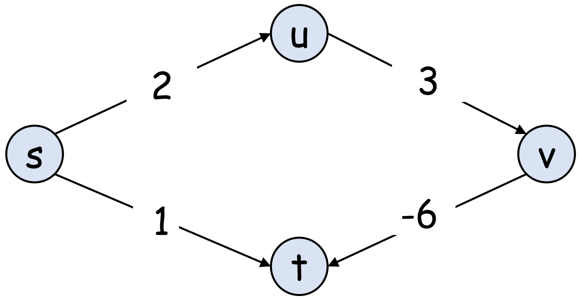
  allow negative weights

- Ex. Nodes represent agents in a financial setting and c_{vw} is cost of transaction in which we buy from agent v and sell immediately to w.

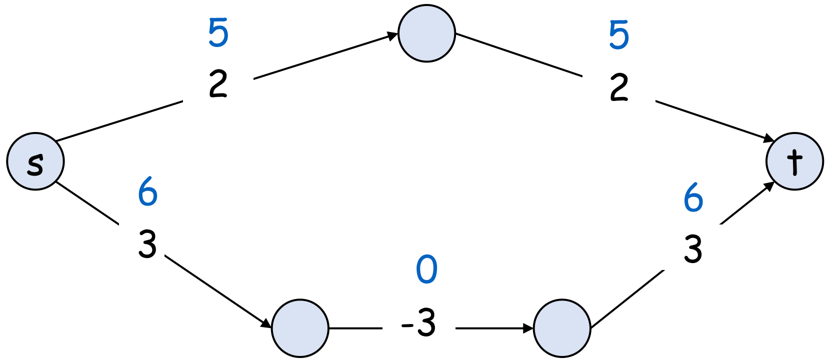


Shortest Paths: Failed Attempts

- Dijkstra. Can fail if negative edge costs.

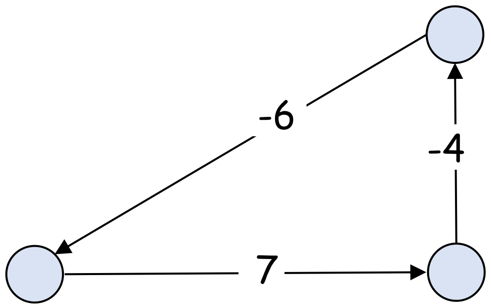


- Re-weighting. Adding a constant to every edge weight can fail.

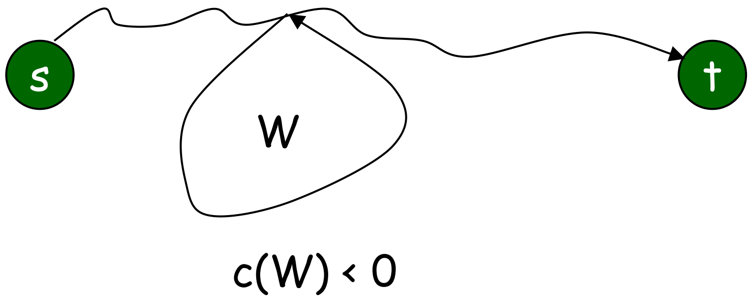


Shortest Paths: Negative Cost Cycles

- Negative cost cycle.



- Observation. If some path from s to t contains a negative cost cycle, there does not exist a shortest s-t path; otherwise, there exists one that is simple.



Shortest Paths: Dynamic Programming

- Def. OPT(i, v) = length of shortest v-t path P using at most i edges.

  - Case 1: P uses at most i-1 edges.
    - ✓ OPT(i, v) = OPT(i-1, v)

  - Case 2: P uses exactly i edges.
    - ✓ if (v, w) is first edge, then OPT uses (v, w), and then selects best w-t path using at most i-1 edges

$$OPT(i, v) = \min(OPT(i-1, v), \min_{w \in V} (OPT(i-1, w) + C_{vw}))$$

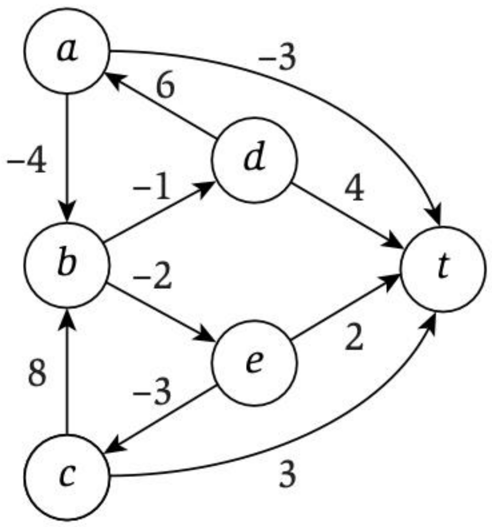
- Remark. By previous observation, if no negative cycles, then OPT(n-1, v) = length of shortest v-t path.

Shortest Paths: Implementation

```
Shortest-Path(G, t) {
   foreach node v ∈ V
      M[0, v] ← ∞
   M[0, t] ← 0

   for i = 1 to n-1
      foreach node v ∈ V
         M[i, v] ← M[i-1, v]
      foreach edge (v, w) ∈ E
         M[i, v] ← min { M[i, v], M[i-1, w] + c_vw }
}
```

- Analysis. $\Theta(mn)$ time, $\Theta(n^2)$ space.
- Finding the shortest paths. Maintain a "successor" for each table entry.

Shortest Paths: Example



(a)

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	−3	−3	−4	−6	−6
b	∞	∞	0	−2	−2	−2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

(b)

$$OPT(i,v) = \min(OPT(i-1,v), \min_{w \in V} (OPT(i-1,w) + C_{vw}))$$

Shortest Paths: Practical Improvements

- Practical improvements.
  - Maintain only one array M[v] = shortest v-t path that we have found so far.
  - No need to check edges of the form (v, w) unless M[w] changed in previous iteration.

- Theorem. Throughout the algorithm, M[v] is length of some v-t path, and after i rounds of updates, the value M[v] is no larger than the length of shortest v-t path using $\leq$ i edges.

- Overall impact.
  - Memory: O(m + n).
  - Running time: O(mn) worst case, but substantially faster in practice.

Bellman-Ford: Efficient Implementation

```
Push-Based-Shortest-Path(G, s, t) {
   foreach node v ∈ V {
      M[v] ← ∞
      successor[v] ← ϕ
   }

   M[t] = 0
   for i = 1 to n-1 {
      foreach node w ∈ V {
      if (M[w] has been updated in previous iteration) {
         foreach node v such that (v, w) ∈ E {
            if (M[v] > M[w] + c_vw) {
               M[v] ← M[w] + c_vw
               successor[v] ← w
            }
         }
      }
      If no M[w] value changed in iteration i, stop.
   }
}
```

7. Distance Vector Protocol

Distance Vector Protocol

- Communication network.
  - Node ≈ router.
  - Edge ≈ direct communication link. ← naturally nonnegative, but Bellman-Ford used anyway!
  - Cost of edge ≈ delay on link.

- Dijkstra's algorithm. Requires global information of network.

- Bellman-Ford. Uses only local knowledge of neighboring nodes.

- Synchronization. We don't expect routers to run in lockstep. The order in which each `foreach` loop executes is not important. Moreover, algorithm still converges even if updates are asynchronous.

Push-based shortest path

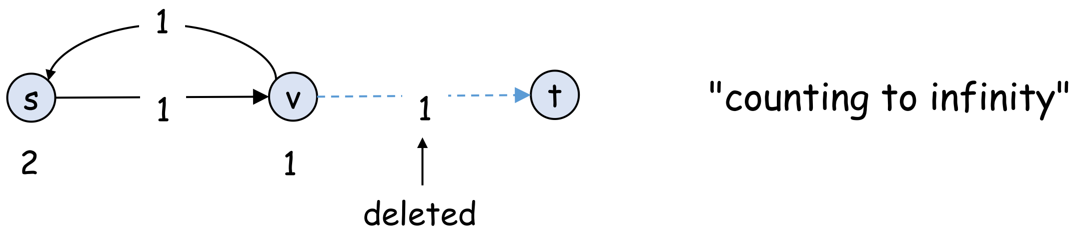
```
Push-Based-Shortest-Path(G, s, t)
  n = number of nodes in G
  Array M[V]
  Initialize M[t] = 0 and M[v] = ∞ for all other v ∈ V
  For i = 1, . . . , n − 1
    For w ∈ V in any order
      If M[w] has been updated in the previous iteration then

        For all edges (v, w) in any order
          M[v] = min(M[v], c_vw + M[w])
          If this changes the value of M[v], then first[v] = w
        Endfor
    Endfor
    If no value changed in this iteration, then end the algorithm
  Endfor
  Return M[s]
```

Asynchronous shortest path

```
Asynchronous-Shortest-Path(G, s, t)
  n = number of nodes in G
  Array M[V]
  Initialize M[t] = 0 and M[v] = ∞ for all other v ∈ V
  Declare t to be active and all other nodes inactive
  While there exists an active node
    Choose an active node w
       For all edges (v, w) in any order
         M[v] = min(M[v], c_vw + M[w])
         If this changes the value of M[v], then
           first[v] = w
           v becomes active
       Endfor
       w becomes inactive
  EndWhile
```

Distance Vector Protocol

- Distance vector protocol.
  - Each router maintains a vector of shortest path lengths to every other node (distances) and the first hop on each path (directions).
  - Algorithm: each router performs n separate computations, one for each potential destination node.
  - "Routing by rumor."
- Ex. RIP, Xerox XNS RIP, Novell's IPX RIP, Cisco's IGRP, DEC's DNA Phase IV, AppleTalk's RTMP.
- Caveat. Edge costs may change during algorithm (or fail completely).



Path Vector Protocols

- Link state routing.

  not just the distance and first hop

  - Each router also stores the entire path.
  - Based on Dijkstra's algorithm.
  - Avoids "counting-to-infinity" problem and related difficulties.
  - Requires significantly more storage.

- Ex. Border Gateway Protocol (BGP), Open Shortest Path First (OSPF).