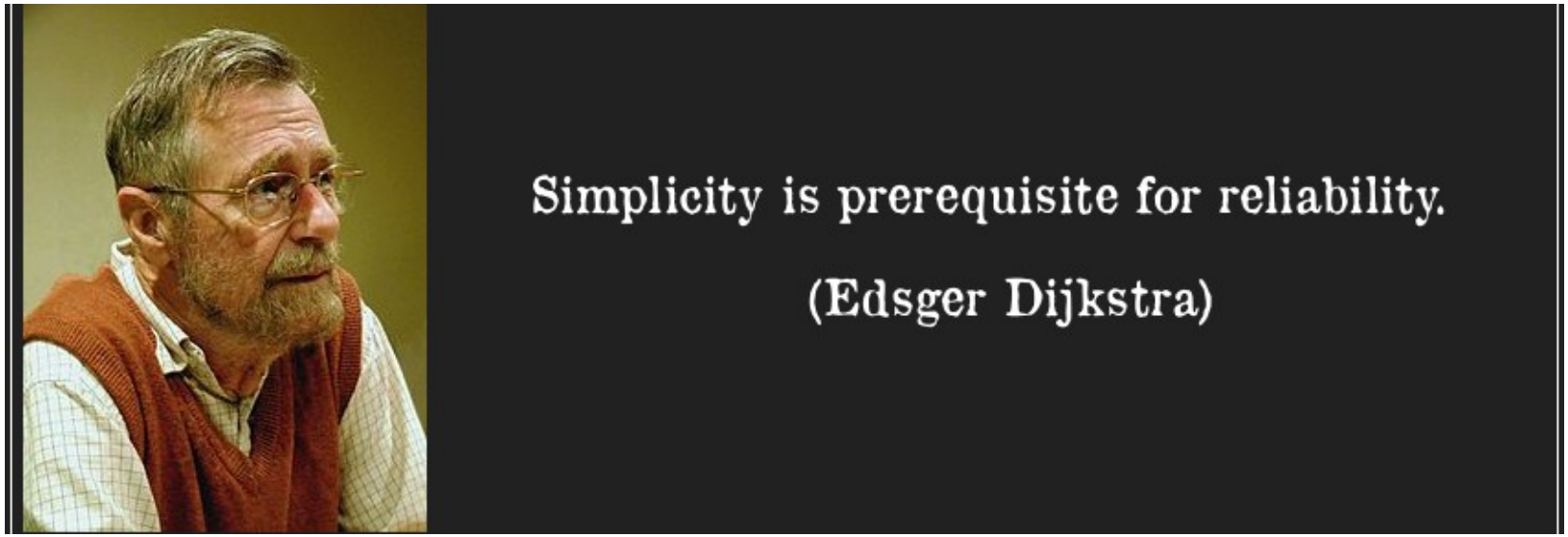
DIJKSTRA'S ALGORITHM

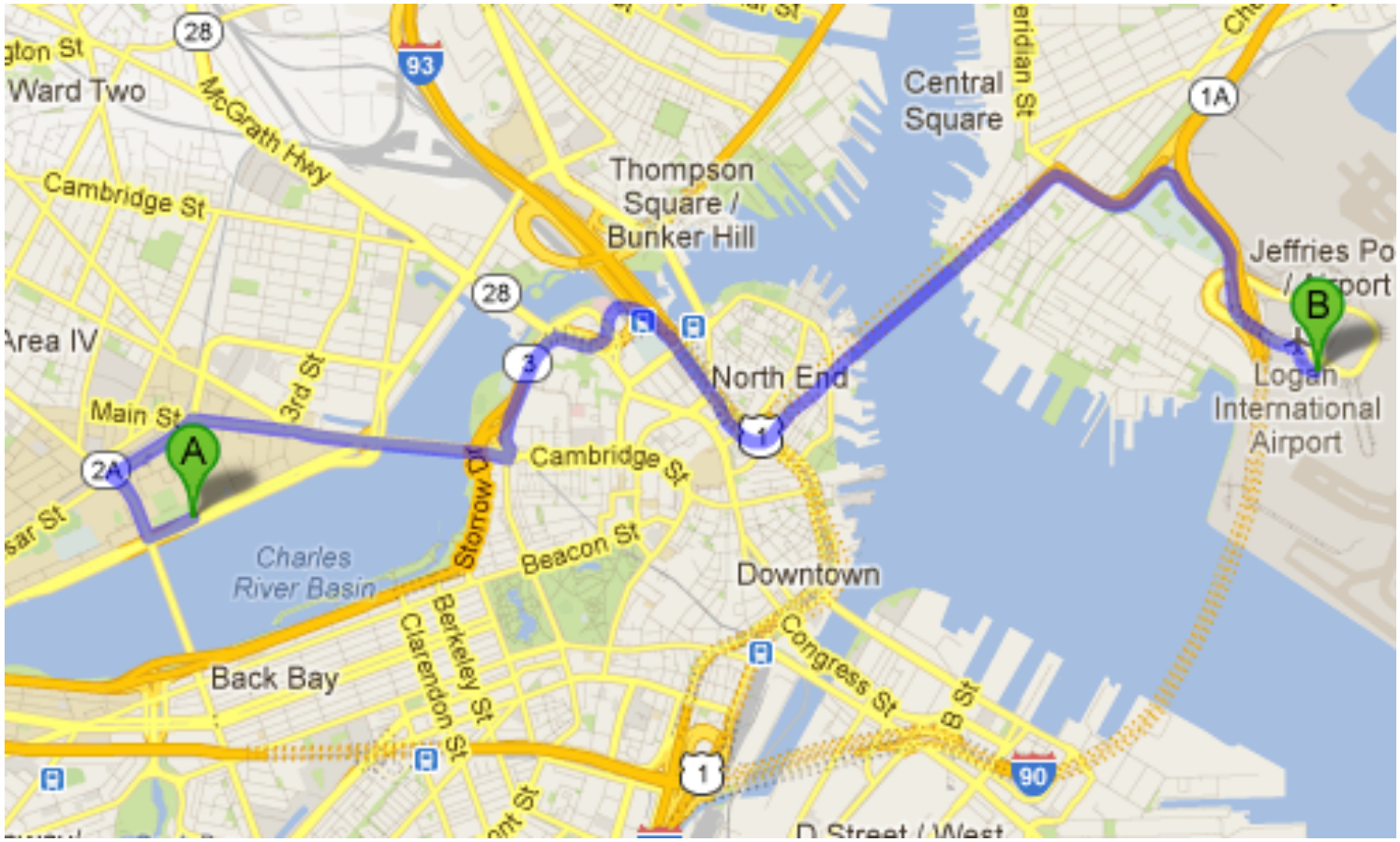
Melissa Yan

Edsger Wybe Dijkstra

- May 11, 1930 – August 6, 2002
- Dutch computer scientist from Netherlands
- Received the 1972 A. M. Turing Award, widely considered the most prestigious award in computer science
- Known for his many essays on programming

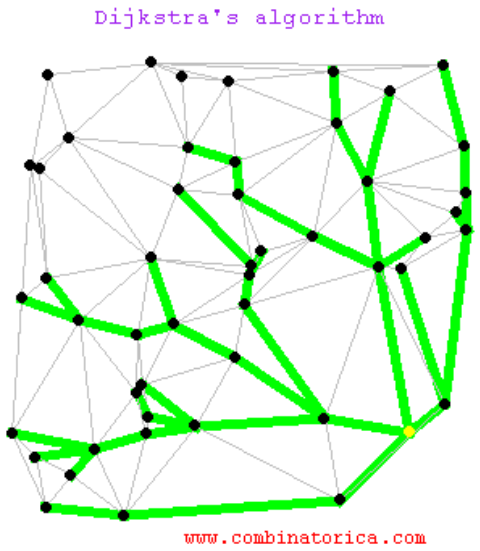


How do we get from 2-151 to Logan?



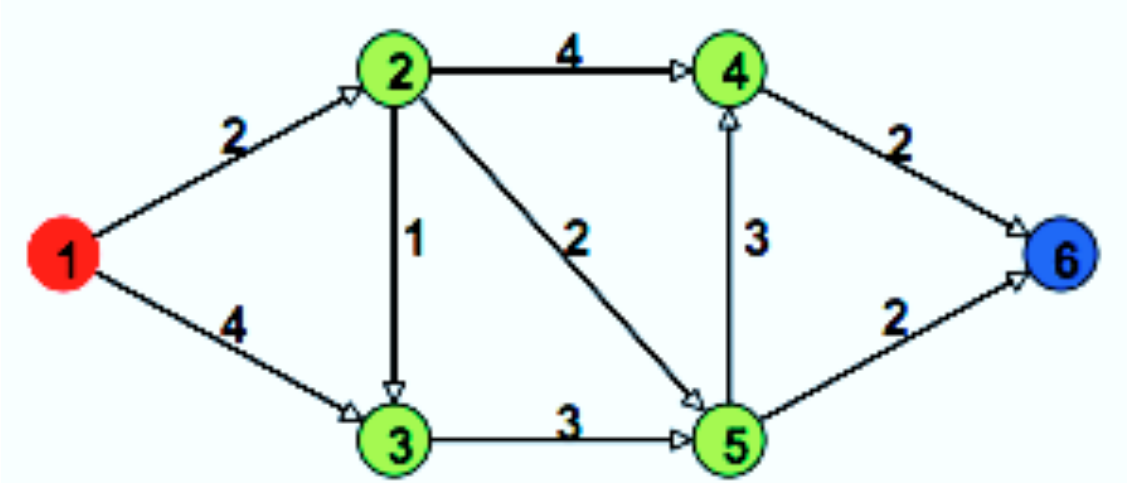
Single-Source Shortest Path Problem

- The problem of finding shortest paths from a source vertex v to all other vertices in the graph.
- Weighted graph $G = (E,V)$
- Source vertex $s \in V$ to all vertices $v \in V$



Dijkstra's Algorithm

- Solution to the single-source shortest path problem in graph theory
  - Both directed and undirected graphs
  - All edges must have nonnegative weights
  - Graph must be connected



Pseudocode

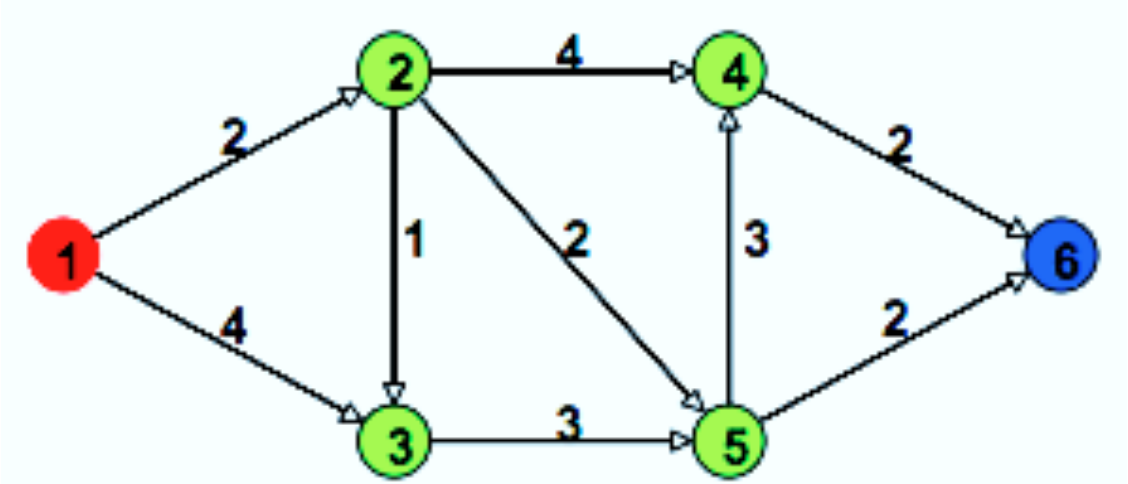
```
dist[s] ←o                          (distance to source vertex is zero)
for  all v ∈ V–{s}
      do  dist[v] ←∞                (set all other distances to infinity)
S←∅                                 (S, the set of visited vertices is initially empty)
Q←V                                 (Q, the queue initially contains all vertices)
while Q ≠∅                          (while the queue is not empty)
do   u ← mindistance(Q,dist)        (select the element of Q with the min. distance)
     S←S∪{u}                        (add u to list of visited vertices)
      for all v ∈ neighbors[u]
           do  if   dist[v] > dist[u] + w(u, v)        (if new shortest path found)
                    then     d[v] ←d[u] + w(u, v)     (set new value of shortest path)
                                                      (if desired, add traceback code)

return dist
```

Output of Dijkstra's Algorithm

- Original algorithm outputs value of shortest path not the path itself
- With slight modification we can obtain the path

Value: $\delta(1,6) = 6$

Path: {1,2,5,6}



Why It Works Intuitively

- **Lemma 1: Optimal Substructure**
  The subpath of any shortest path is itself a shortest path.
- **Lemma 2: Triangle inequality**
  If $\delta(u,v)$ is the shortest path length between u and v, $\delta(u,v) \leq \delta(u,x) + \delta(x,v)$

Proof of Correctness 1

- Invariant: $d[v] \geq \delta(s,v)$ for all $v \in V$
  - Holds after initialization and any sequence of relaxation steps

Proof

Hint: We will need the relaxation part of the pseudocode.

$$d[v] \leftarrow d[u] + w(u,v)$$

Proof of Correctness 1.5

- Relaxation step not only maintains the invariant but allows us to find next shortest path

- Suppose s → ... → u → v is a shortest path
  - If d[u] = δ (s,u)
  - And we relax edge (u,v)
  - Then d[v] = δ (s,v) after relaxation

Proof

Proof of Correctness 2

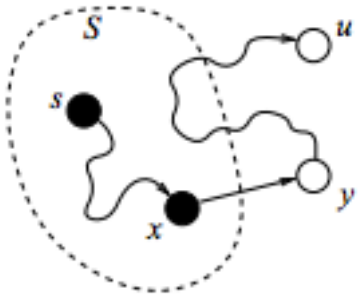
- When Dijkstra terminates,

$$d[v] = \delta(s,v) \text{ for all } v \in V$$

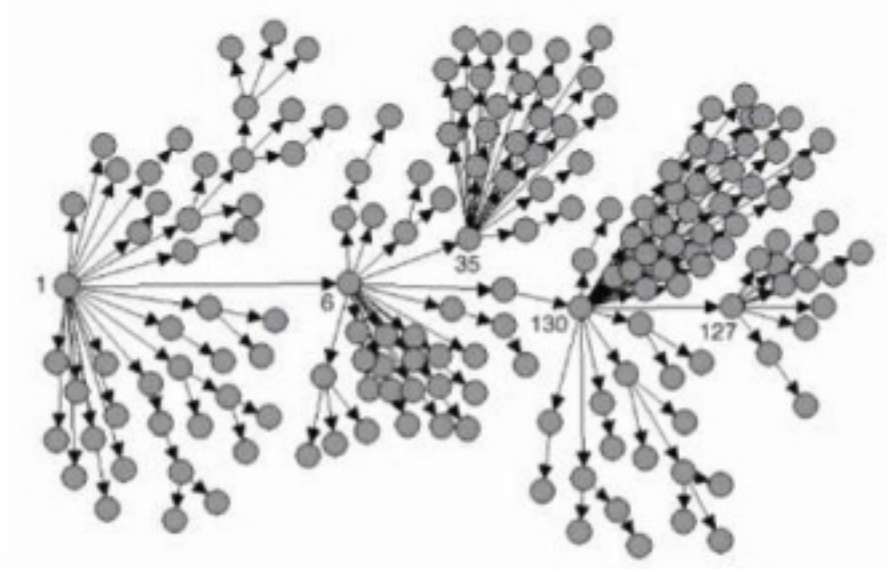
- Proof

Hint: We will need min-extraction part of the pseudo-code.

$$u \leftarrow \text{mindistance}(Q,dist)$$



Applications of Dijkstra's Algorithm



LaTeX is a document preparation system for the TeX typesetting program. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout, bibliographies, and much more. LaTeX was originally written in 1984 by Leslie Lamport and has become the dominant method for using TeX; few people write in plain TeX anymore. The current version is LaTeX 2_ε.

$$E = mc^2 \tag{1}$$

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{2}$$

Why do we use Dijkstra's Algorithm?

- Algorithms for calculating shortest path from source to sink about as computationally expensive as calculating shortest paths from source to any vertex

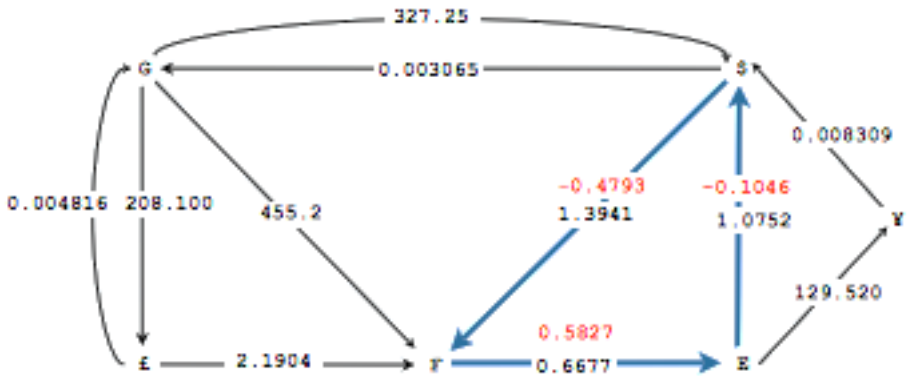
Currency Exchange Problem

- Is there an arbitrage opportunity?
- Ex. $1 → 1.3941 Francs → 0.9308 Euros → $1.00084

Currency	£	Euro	¥	Franc	$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3941	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold (oz.)	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

Currency Exchange Problem (con.)

- Vertex = currency, edge = transaction
- Weight = exchange rate
- Find path that maximizes product of weights → reduce to sum of weights
- Is there a negative cost cycle?



Bellman-Ford Algorithm

- Works for negative weights
  - Detects a negative cycle if any exist
  - Finds shortest simple path if no negative cycle exists

If graph $G = (V,E)$ contains negative-weight cycle, then some shortest paths may not exist.