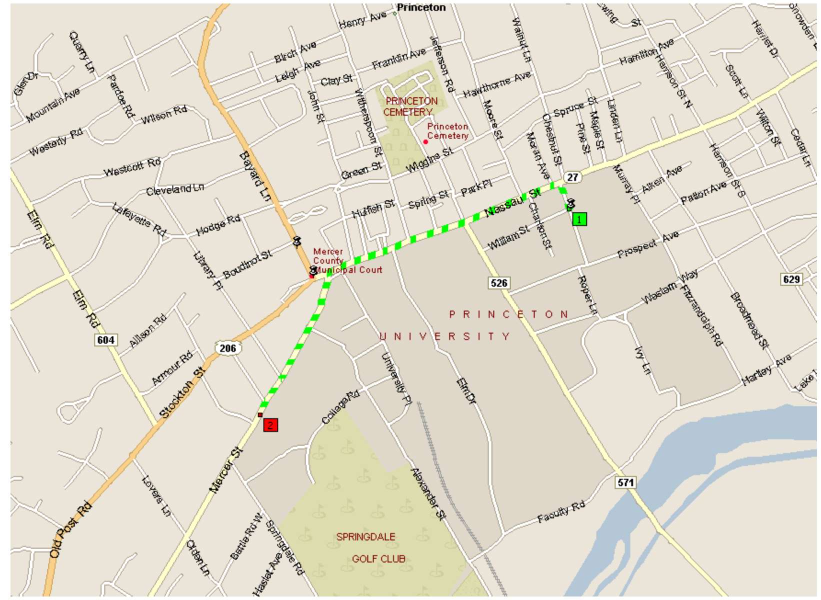
Shortest Paths

Dijkstra's algorithm

Bellman-Ford algorithm

Fastest Route from CS Dept to Einstein's House



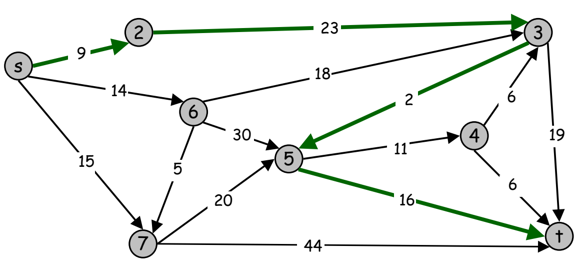
Shortest Path Problem

Shortest path network.

- Directed graph.
- Source s, destination t.
- cost(v-w) = cost of using edge from v to w.

Shortest path problem: find shortest directed path from s to t.

- Cost of path = sum of arc costs in path.



Cost of path s-2-3-5-t
= 9 + 23 + 2 + 16
= 48.

Brief History

Shimbel (1955). Information networks, min-sum algebra.

Ford (1956). RAND, economics of transportation.

Leyzorek, Gray, Johnson, Ladew, Meaker, Petry, Seitz (1957).
Combat Development Dept. of the Army Electronic Proving Ground.

Dantzig (1958). Simplex method for linear programming.

Bellman (1958). Dynamic programming.

Moore (1959). Routing long-distance telephone calls for Bell Labs.

Dijkstra (1959). Simpler and faster version of Ford's algorithm.

Applications

More applications.

- Robot navigation.
- Typesetting in TeX.
- Urban traffic planning.
- Tramp steamer problem.
- Optimal pipelining of VLSI chip.
- Telemarketer operator scheduling.
- Subroutine in higher level algorithms.
- Routing of telecommunications messages.
- Approximating piecewise linear functions.
- Exploiting arbitrage opportunities in currency exchange.
- Open Shortest Path First (OSPF) routing protocol for IP.
- Optimal truck routing through given traffic congestion pattern.

Reference: *Network Flows: Theory, Algorithms, and Applications*, R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Prentice Hall, 1993.

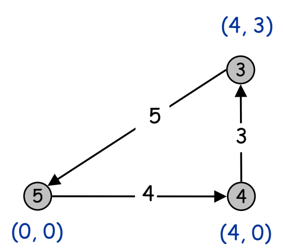
Graphs

Graph	Vertices	Edges
communication	telephones, computers	fiber optic cables
circuits	gates, registers, processors	wires
mechanical	joints	rods, beams, springs
hydraulic	reservoirs, pumping stations	pipelines
financial	stocks, currency	transactions
transportation	street intersections, airports	highways, airway routes
scheduling	tasks	precedence constraints
software systems	functions	function calls
internet	web pages	hyperlinks
games	board positions	legal moves
social relationship	people, actors	friendships, movie casts
neural networks	neurons	synapses
protein networks	proteins	protein-protein interactions
chemical compounds	molecules	bonds

Shortest Path

Some versions of the problem that we consider.

- Undirected.
- Directed.
- Single source.
- All-pairs. next programming assignment
- Arc costs are ≥ 0.
- Points in plane with Euclidean distances.



Euclidean

Shortest Path: Edge Relaxation

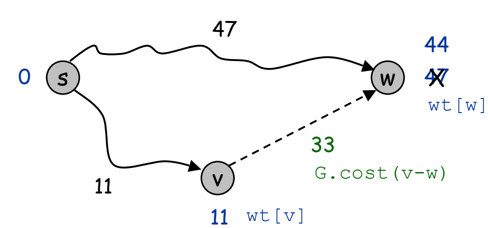
Valid weights. For all `v`, `wt[v]` is length of some path from `s` to `v`.

Edge relaxation.

- Consider edge `v-w` with `G.cost(v, w)`.
- If path from `s` to `v` plus edge `v-w` is better than current path to `w`, then update.

```
if (wt[w] > wt[v] + G.cost(v, w)) {
    wt[w] = wt[v] + G.cost(v, w);
    pred[w] = v;
}
```

edge relaxation



Edsger W. Dijkstra

The question of whether computers can think is like the question of whether submarines can swim.

Do only what only you can do.

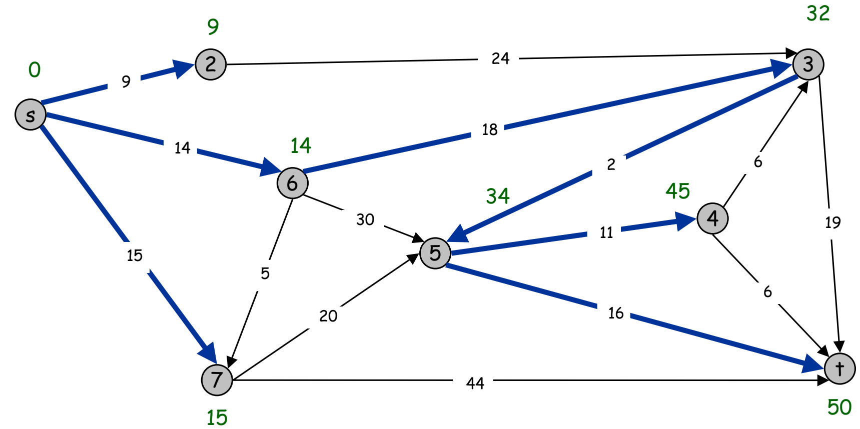
In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.

The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.

APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.

Dijkstra's Algorithm

Dijkstra's algorithm.
- Finds the shortest path from `s` to all other vertices.
- Shortest paths from `s` form a tree.



Dijkstra's Algorithm: Implementation

Dijkstra's algorithm.
- Initialize `wt[v] = ∞` and `wt[s] = 0`.
- Insert all vertices `v` onto PQ with priorities `wt[v]`.
- Repeatedly delete node `v` from PQ that has min `wt[v]`.
  - add `v` to S
  - for each `v-w`, relax `v-w`

```
while (!pq.isEmpty()) {
   int v = pq.delMin();
   IntIterator i = G.neighbors(v);
   while (i.hasNext()) {
      int w = i.next();
      if (wt[w] > wt[v] + G.cost(v, w)) {
          wt[w] = wt[v] + G.cost(v, w);
          pq.decrease(w, wt[w]);
          pred[w] = v;
      }
   }
}
```
Main Loop

Dijkstra's Algorithm: Proof of Correctness

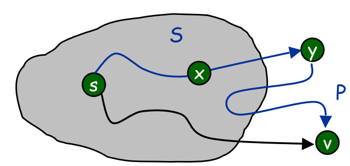
Invariant. For each vertex v, wt[v] is length of shortest s-v path whose internal vertices are in S; for each vertex v in S, wt[v] = wt*[v].

length of shortest s-v path

Proof: by induction on |S|.

Base case: |S| = 0 is trivial.

Induction step:
- Let v be next vertex added to S by Dijkstra's algorithm.
- Let P be a shortest s-v path, and let x-y be first edge leaving S.



- We show wt[v] = wt*[v].

$$wt[v] \;\geq\; wt^*[v] \;\geq\; wt^*[y] \;=\; wt[y] \;\geq\; wt[v]$$

wt[v] length of some path — nonnegative weights — induction — Dijkstra chose v before y

Dijkstra's Algorithm: Implementation Cost Summary

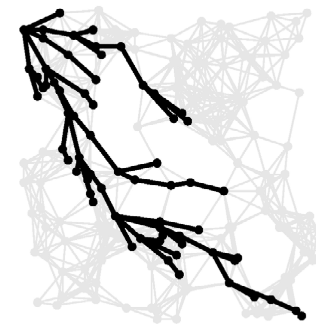
		Priority Queue			
Operation	Dijkstra	Array	Binary heap	d-way Heap	Fib heap †
insert	V	V	log V	d $\log_d$ V	1
delete-min	V	V	log V	d $\log_d$ V	log V
decrease-key	E	1	log V	$\log_d$ V	1
is-empty	V	1	1	1	1
total		V^2	E log V	E $\log_{E/V}$ V	E + V log V

† Individual ops are amortized bounds

Observation: algorithm is almost identical to Prim's MST algorithm!

Priority first search: variations on a theme.

Shortest Path in Euclidean Graphs



Euclidean graph (map).

- Vertices are points in the plane.
- Edges weights are Euclidean distances.

Sublinear algorithm.

- Assume graph is already in memory.
- Start Dijkstra at `s`.
- Stop as soon as you reach `t`.

Exploit geometry. (A* algorithm)

- For edge `v-w`, use weight `d(v, w) + d(w, t) – d(v, t)`.
- Dijkstra's proof of correctness still applies.
- In practice only $O(V^{1/2})$ vertices examined.

Euclidean distance

Shortest Path Application: Currency Conversion

Given currencies and exchange rates, what is best way to convert one ounce of gold to US dollars?

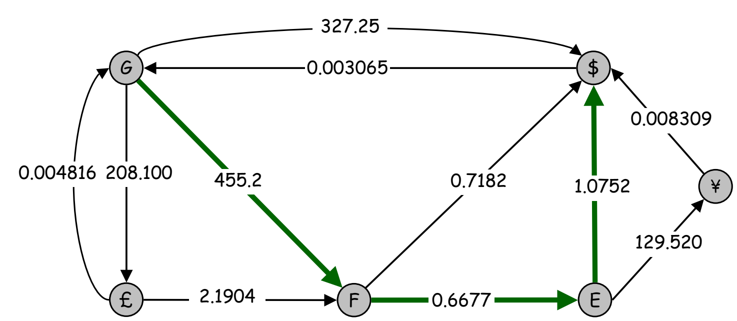
- 1 oz. gold ⇒ \$327.25.
- 1 oz. gold ⇒ £208.10 ⇒ 208.10 (1.5714) ⇒ \$327.00.
- 1 oz. gold ⇒ 455.2 Francs ⇒ 304.39 Euros ⇒ \$327.28.

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	1.3929	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

Shortest Path Application: Currency Conversion

Graph formulation.

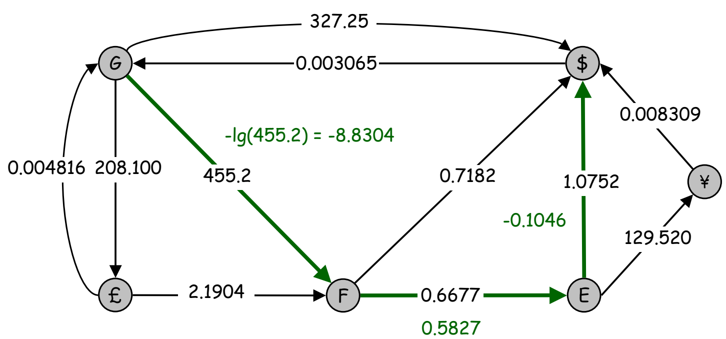
- Create a vertex for each currency.
- Create a directed edge for each possible transaction, with weight equal to the exchange rate.
- Find path that maximizes product of weights.



Shortest Path Application: Currency Conversion

Reduction to shortest path problem.

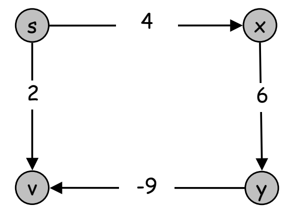
- Let γ_{vw} be exchange rate from currency v to w.
- Let $c_{vw} = - \lg \gamma_{vw}$.
- Shortest path with costs c corresponds to best exchange sequence.

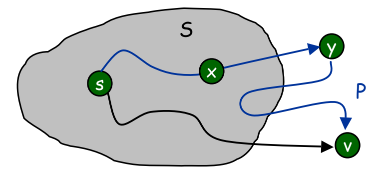


Dijkstra's Algorithm With Negative Costs

Dijkstra's algorithm fails if there are negative weights.

- Ex: Selects vertex `v` immediately after `s`.
  But shortest path from `s` to `v` is `s-x-y-v`.





Dijkstra proof of correctness breaks down since it assumes cost of P is nonnegative.

Challenge: shortest path algorithm that works with negative costs.

Dynamic Programming

Dynamic programming.

- Initialize `wt[v]` = ∞, `wt[s]` = 0.
- Repeat `V` times: relax each edge `v-w`.

```
for (int i = 1; i <= V; i++) {     ⇦ phase i
   for (int v = 0; v < V; v++) {
      IntIterator i = G.neighbors(v);
      while (i.hasNext()) {
         int w = i.next();
         if (wt[w] > wt[v] + G.cost(v, w)) {
            wt[w] = wt[v] + G.cost(v, w);
            pred[w] = v;             relax v-w
         }
      }
   }
}
```

Invariant. At end of phase `i`, `wt[v]` ≤ length of any path from `s` to `v` using at most `i` edges.

Running time. Θ(E V).

Bellman-Ford-Moore Algorithm

Practical improvement.

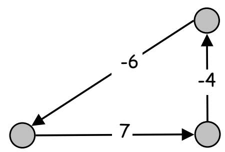
- If `wt[v]` doesn't change during phase `i`, don't relax any edges of the form `v-w` in phase `i+1`.
- Programming solution: maintain queue of nodes that have changed.

```
while (!q.isEmpty()) {
   int v = q.dequeue();
   IntIterator i = G.neighbors(v);
   while (i.hasNext()) {
      int w = i.next();
      if (wt[w] > wt[v] + G.cost(v, w)) {
         wt[w] = wt[v] + G.cost(v, w);          relax v-w
         pred[w] = v;
         q.enqueue(w);   ⇦ discard duplicates
      }
   }
}
```

Running time. Still Ω(E V) in worst case, but linear in practice!

Negative Cycles

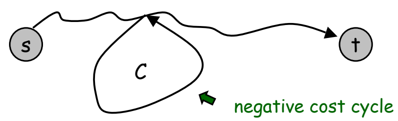
Negative cycle. Directed cycle whose sum of edge costs is negative.



-6 + 7 - 4 = -5

Caveat. Bellman-Ford terminates and finds shortest (simple) path after at most V phases if and only if no negative cycles.

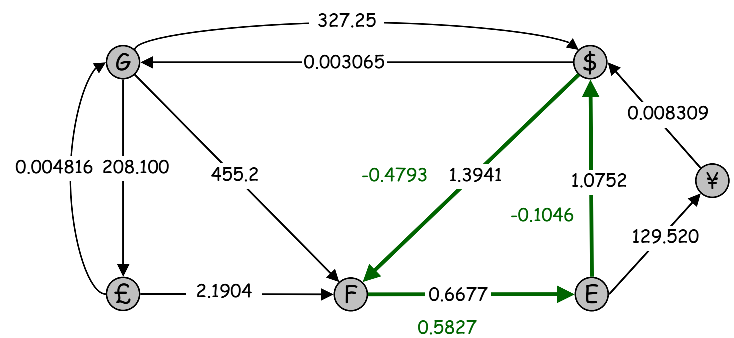
Observation. If negative cycle on path from s to t, then shortest path can be made arbitrarily negative by spinning around cycle.



Shortest Path Application: Arbitrage

Arbitrage.
- Is there an arbitrage opportunity in currency graph?
- Ex: \$1 $\Rightarrow$ 1.3941 Francs $\Rightarrow$ 0.9308 Euros $\Rightarrow$ \$1.00084.
- Is there a negative cost cycle?
- Fastest algorithm very valuable!

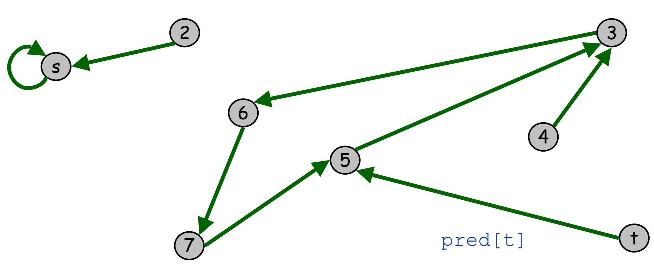
-0.4793 + 0.5827 - 0.1046 < 0



Bellman-Ford-Moore Algorithm

Finding the shortest path itself.
- Trace back `pred[v]` as in Dijkstra's algorithm.

Finding a negative cycle.
- If any node `v` is enqueued `V` times, there must be a negative cycle.
- Fact: can trace back `pred[v]` to find cycle.



Single Source Shortest Paths Implementation: Cost Summary

Algorithm	Worst Case	Best Case	Space
Dijkstra (classic) †	V^2	V^2	linear
Dijkstra (heap) †	E log V	linear	linear
Dynamic Programming ‡	E V	E V	linear
Bellman-Ford ‡	E V	linear	linear

† nonnegative costs
‡ no negative cycles or negative cycle detection

Remark 1: negative weights makes the problem harder.
Remark 2: negative cycles makes the problem intractable.