


Shortest Path First

Dijkstra's Famous Algorithm



*“The question of whether
computers can think is
like the question of whether
submarines can swim”*



Edsger Wybe Dijkstra

Dijkstra's SP Algorithm



- Famous paper "A note on two problems in connection with graphs" (1959)
- Single source SP problem in a directed graph
- Important applications include
 - ♦ Network routing protocols (OSPF, IS-IS)
 - ♦ Traveller's route planner

Single source SP algorithms find all shortest paths to all vertices at once. The only difference to single-pair SP algorithms is the termination condition.

Terms



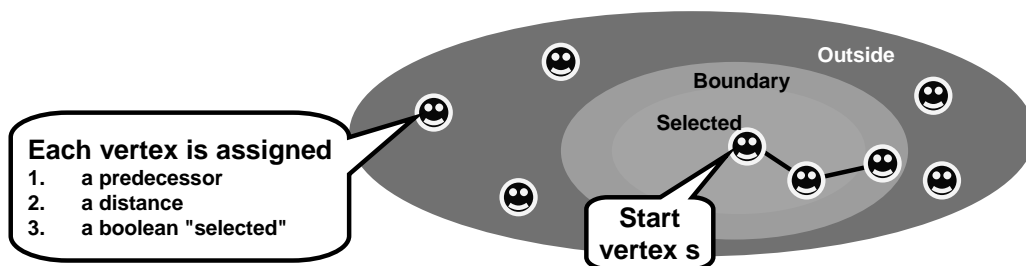
- Graph $G(V,E)$ consists of vertices V and edges E
- Edges are assigned costs c
- "Length" of graph $c(G) = \text{sum of all costs}$
 - ♦ Assumed to be positive ("Distance Graph")
- "Distance" between two vertices $d(v,v') = \min\{c(p)\}$, $p \dots \text{path}$
 - ♦ Can be infinite
- p with $c(p) = d(v,v')$ is called shortest path $sp(v,v')$

SPs are easier to calculate for distance graphs where the costs are only positive.

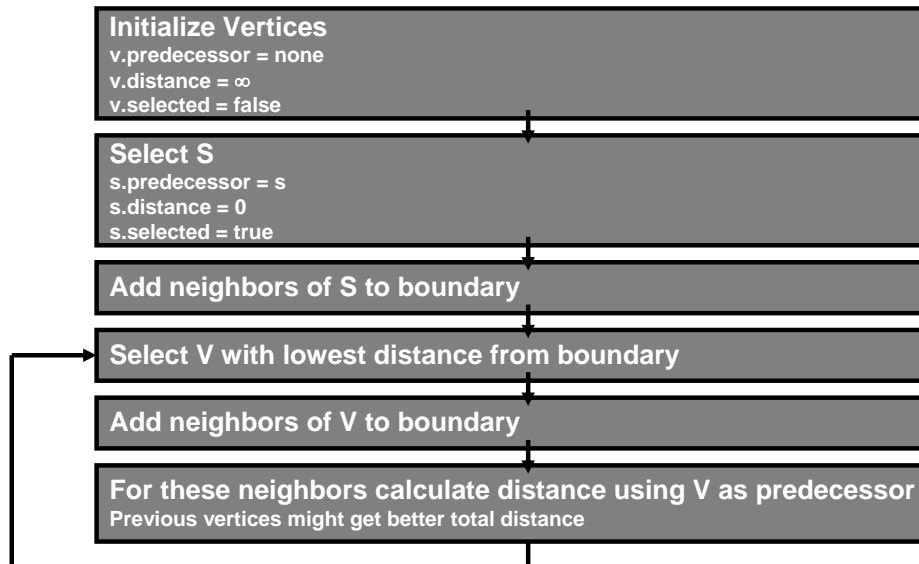
Definitions



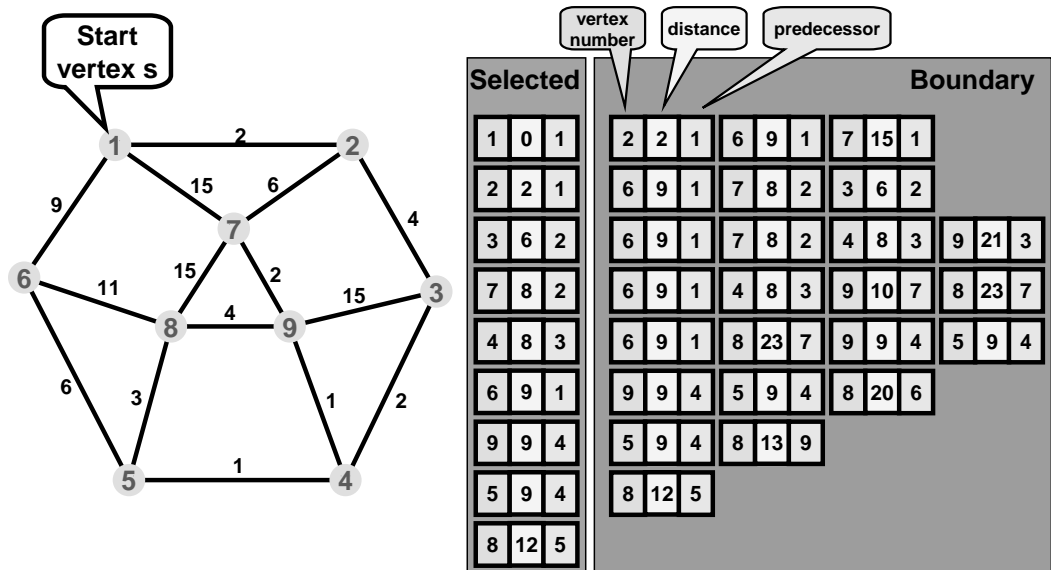
- **Select start vertex s**
- **Three sets of vertices:**
 - ♦ **Selected** (sp already calculated)
 - ♦ **Boundary** (currently subject of calculation)
 - ♦ **Outside** (not yet examined)



The Algorithm



Example

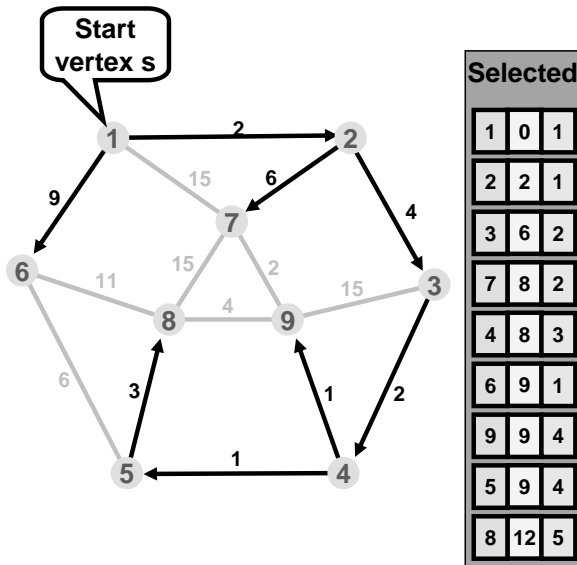


(C) Herbert Haas 2005/03/11

7

Note that the left list ("Selected") is sometimes called the PATH list, and the right list ("Boundary") is sometimes called the TENT list (from tentative). It's got nothing to do with a beer tent.

Result



- **Single source SP**
- **Minimal length**
- **Complete**

Performance



- **Greedy algorithm**
- **Most critical: Implementation of boundary data structure**
 - ♦ No explicit structure: $O(|V|^2)$
 - ♦ Fibonacci heap: $O(|E| + |V| \log |V|)$
- **Alternatives**
 - ♦ Bellman-Ford (RIP) algorithm
 - ♦ Floyd-Warshall algorithm
 - ♦ A* algorithm
 - Extends SPF with a estimation function to enhance performance in certain situations

The SPF algorithm is of “greedy” type. Dijkstra originally proposed to treat the boundary vertices like outside vertices, therefore no explicit data structure is needed for the boundary vertices. This implementation is efficient for graphs with lots of edges but not efficient with so-called “thin” graphs. One of the best implementations use Fibonacci heaps for boundary representation.

Alternative algorithms are for example the Bellman-Ford or the Floyd-Warshall algorithm, which bases on Belman’s optimization principle (“if the shortest path from A to C runs over B, then the partial path AB must also be the shortest possible”).

About E. W. Dijkstra



- **Born in 1930 in Rotterdam**
- **Degrees in mathematics and theoretical physics from the University of Leyden and a Ph.D. in computing science from the University of Amsterdam**
 - ♦ **Programmer at the Mathematisch Centrum, Amsterdam, 1952-62**
 - ♦ **Professor of mathematics, Eindhoven University of Technology, 1962-1984**
 - ♦ **Burroughs Corporation research fellow, 1973-1984**
 - ♦ **Schlumberger Centennial Chair in Computing Sciences at the University of Texas at Austin, 1984-1999**
 - ♦ **Retired as Professor Emeritus in 1999**
 - ♦ **1972 recipient of the ACM Turing Award, often viewed as the Nobel Prize for computing**
- **Died 6 August 2002**



Edsger W. Dijkstra
(1930-2002)

(C) Herbert Haas 2005/03/11

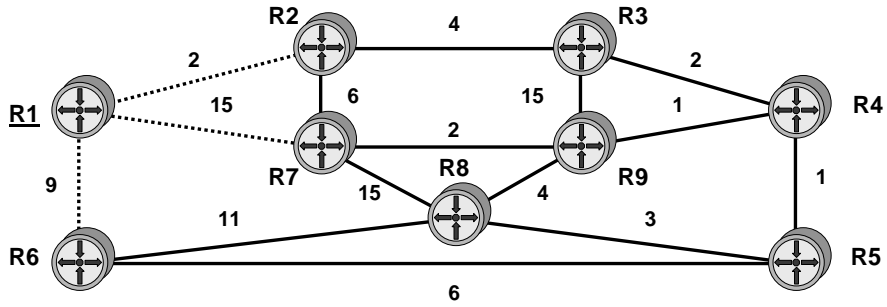
10

Member of the Netherlands Royal Academy of Arts and Sciences, a member of the American Academy of Arts and Sciences, and a Distinguished Fellow of the British Computer Society. He received the 1974 AFIPS Harry Goode Award, the 1982 IEEE Computer Pioneer Award, and the 1989 ACM SIGCSE Award for Outstanding Contributions to Computer Science Education. Athens University of Economics awarded him an honorary doctorate in 2001. In 2002, the C&C Foundation of Japan recognized Dijkstra "for his pioneering contributions to the establishment of the scientific basis for computer software through creative research in basic software theory, algorithm theory, structured programming, and semaphores".

Dijkstra enriched the language of computing with many concepts and phrases, such as structured programming, separation of concerns, synchronization, deadly embrace, dining philosophers, weakest precondition, guarded command, the excluded miracle, and the famous "semaphores" for controlling computer processes. The Oxford English Dictionary cites his use of the words "vector" and "stack" in a computing context.

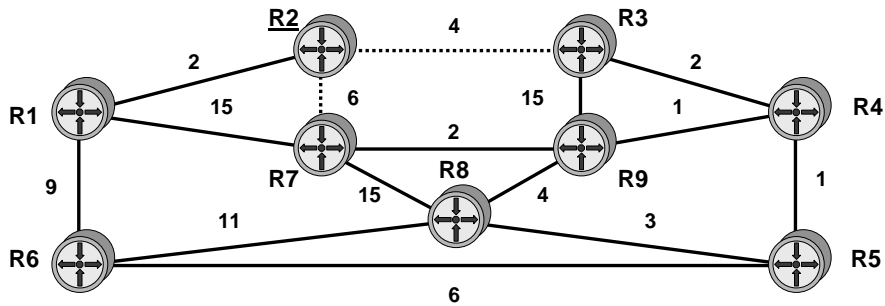
(Source: <http://www.cs.utexas.edu>)

Select root (R1)



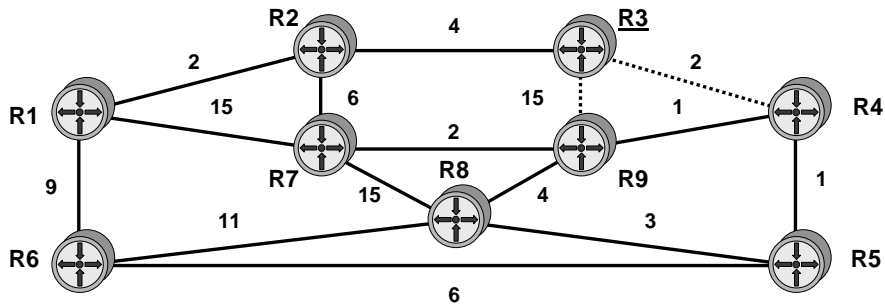
Selected			Boundary								
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1

Select router with lowest cost in boundary (R2), calculate cost for neighbours R3, R7



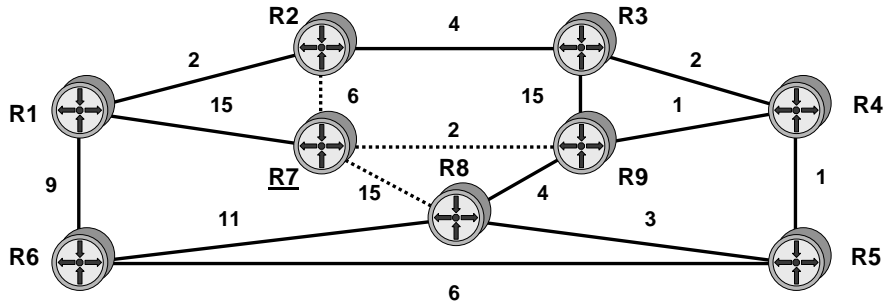
Selected			Boundary								
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1
R2	2	R1	R6	9	R1	R7	8	R2	R3	6	R2

Select router with lowest cost in boundary (R3), calculate cost for neighbours R9, R4



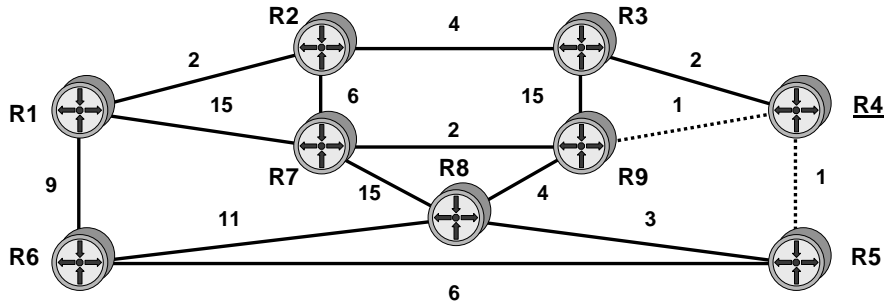
Selected			Boundary											
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1			
R2	2	R1	R6	9	R1	R7	8	R2	R3	6	R2			
R3	6	R2	R6	9	R1	R7	8	R2	R9	21	R3	R4	8	R3

Select one router with lowest cost in boundary (R7), calculate cost for neighbours R8, R9



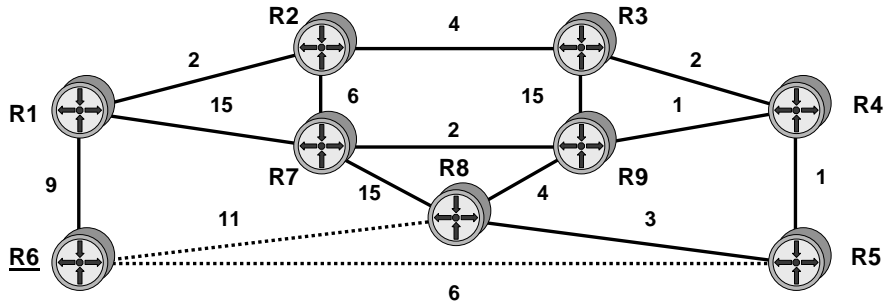
Selected			Boundary											
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1			
R2	2	R1	R6	9	R1	R7	8	R2	R3	6	R2			
R3	6	R2	R6	9	R1	R7	8	R2	R9	21	R3	R4	8	R3
R7	8	R2	R6	9	R1	R4	8	R3	R9	10	R7	R8	23	R7

Select router with lowest cost in boundary (R4), calculate cost for neighbours R9, R5



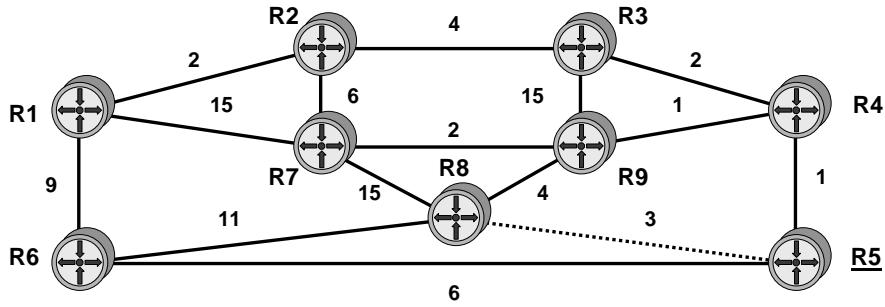
Selected			Boundary											
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1			
R2	2	R1	R6	9	R1	R7	8	R2	R3	6	R2			
R3	6	R2	R6	9	R1	R7	8	R2	R9	21	R3	R4	8	R3
R7	8	R2	R6	9	R1	R4	8	R3	R9	10	R7	R8	23	R7
R4	8	R3	R6	9	R1	R8	23	R7	R9	9	R4	R5	9	R4

Select one router with lowest cost in boundary (R6), calculate cost for neighbours R5 and R8



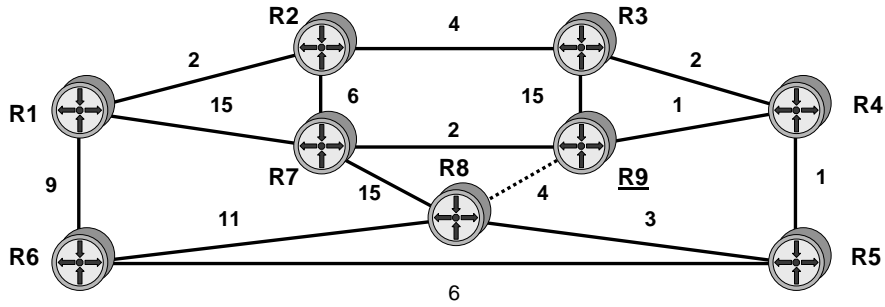
Selected			Boundary														
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1						
R2	2	R1	R6	9	R1	R7	8	R2	R3	6	R2						
R3	6	R2	R6	9	R1	R7	8	R2	R9	21	R3	R4	8	R3			
R7	8	R2	R6	9	R1	R4	8	R3	R9	10	R7	R8	23	R7			
R4	8	R3	R6	9	R1	R8	23	R7	R9	9	R4	R5	9	R4			
R6	9	R1	R9	9	R4	R8	20	R6	R5	9	R4						

Select one neighbour with lowest cost in boundary (R5), calculate cost for neighbour R8



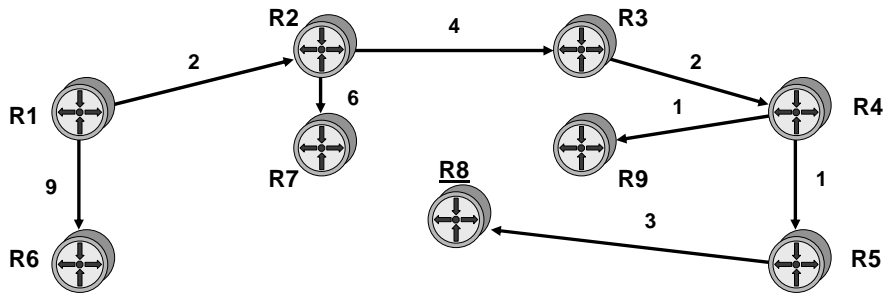
Selected			Boundary											
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1			
R2	2	R1	R6	9	R1	R7	8	R2	R3	6	R2			
R3	6	R2	R6	9	R1	R7	8	R2	R9	21	R3	R4	8	R3
R7	8	R2	R6	9	R1	R4	8	R3	R9	10	R7	R8	23	R7
R4	8	R3	R6	9	R1	R8	23	R7	R9	9	R4	R5	9	R4
R6	9	R1	R9	9	R4	R8	20	R6	R5	9	R4			
R5	9	R4	R9	9	R4	R8	12	R5						

Select router with lowest cost in boundary (R9), calculate cost for neighbours R8



Selected			Boundary											
R1	0	R1	R2	2	R1	R6	9	R1	R7	15	R1			
R2	2	R1	R6	9	R1	R7	8	R2	R3	6	R2			
R3	6	R2	R6	9	R1	R7	8	R2	R9	21	R3	R4	8	R3
R7	8	R2	R6	9	R1	R4	8	R3	R9	10	R7	R8	23	R7
R4	8	R3	R6	9	R1	R8	23	R7	R9	9	R4	R5	9	R4
R6	9	R1	R9	9	R4	R8	20	R6	R5	9	R4			
R5	9	R4	R9	9	R4	R8	12	R5						
R9	9	R4	R8	12	R5									

Select last router in boundary (R8), algorithm terminated, all shortest paths found



Selected		
R1	0	R1
R2	2	R1
R3	6	R2
R7	8	R2
R4	8	R3
R6	9	R1
R5	9	R4
R9	9	R4
R8	12	R5

Boundary								
R2	2	R1	R6	9	R1	R7	15	R1
R6	9	R1	R7	8	R2	R3	6	R2
R6	9	R1	R7	8	R2	R9	21	R3
R6	9	R1	R4	8	R3	R9	10	R7
R6	9	R1	R8	23	R7	R9	9	R4
R9	9	R4	R8	20	R6	R5	9	R4
R9	9	R4	R8	12	R5			
R8	12	R5						