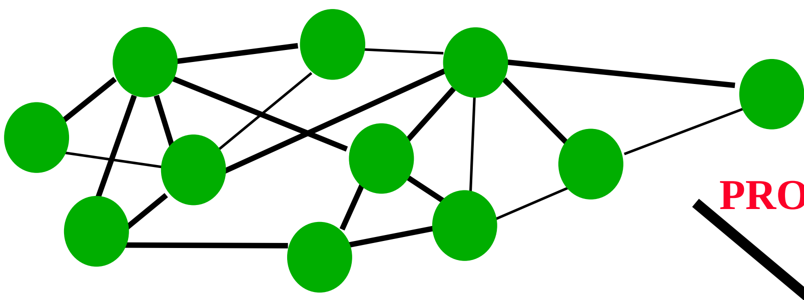
Telecommunications Networking

1. Introduction
2. Local Area Networking
3. Error Control and Retransmission Protocols
4. Architectural Principles of the Internet
5. Flow Control in Internet: TCP
6. **Routing Algorithms**
7. Routing Protocols
8. The principles of ATM
9. Traffic Management in ATM
10. Scheduling
11. Quality of Service
12. Quality of Service routing
13. Peer-to-peer networks

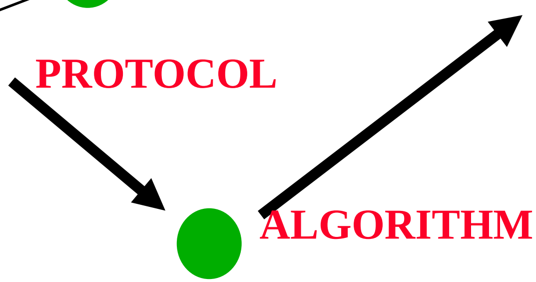
What is Routing?

- Finding one (or more) paths between two (or more) nodes in the network
  - often there is an optimality criterion (e.g. shortest path)

- Basic building blocks



network topology

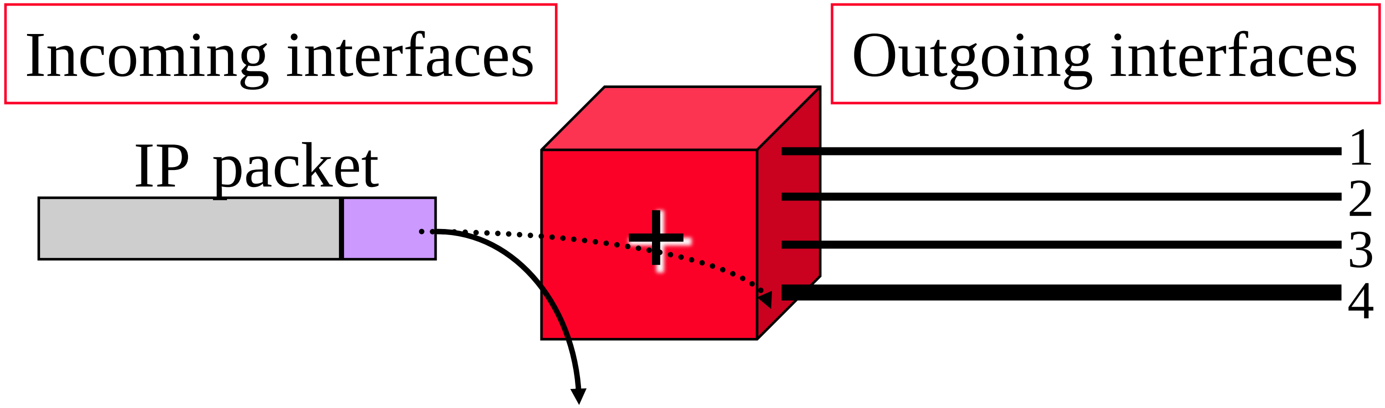


consistent network view

To	Link	Cost
A	local	0
B	1	1
C	3	1
E	3	6

forwarding table

Forwarding in Internet

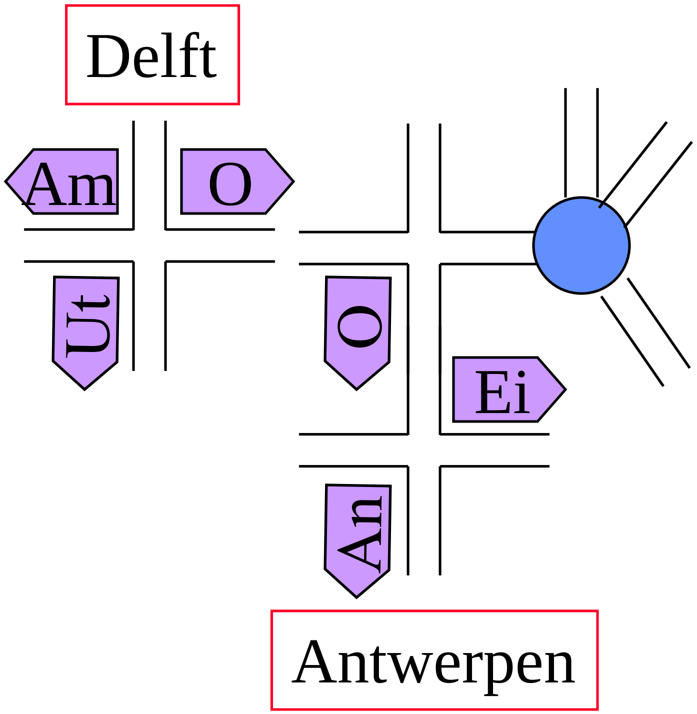


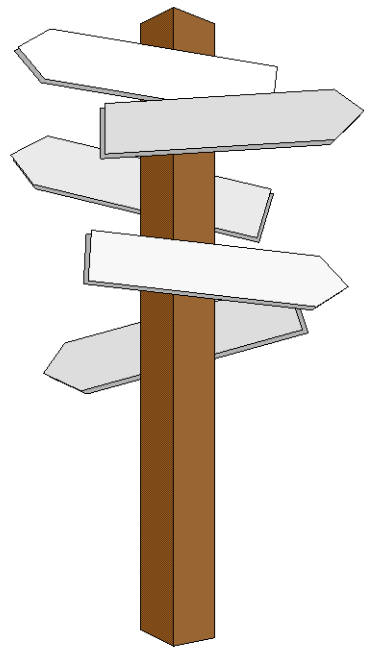
ROUTING TABLE

IP address destination	Link	Cost
IP_1	1	5
IP_2	4	2
IP_3	1	7
...	...	...

Routing Information

- minimum information (next hop info)

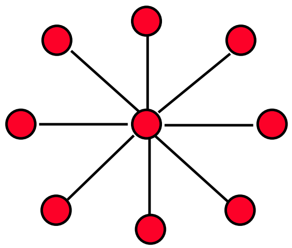




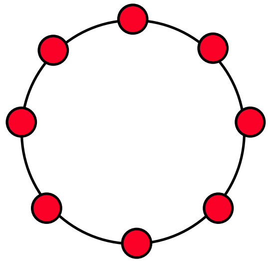
To	Link	Cost
A	local	0
B	1	1
C	3	1
E	3	6

- complete topology information

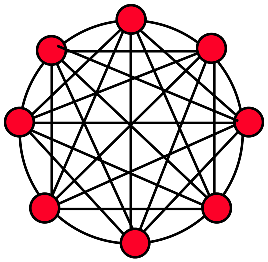
Network Topologies



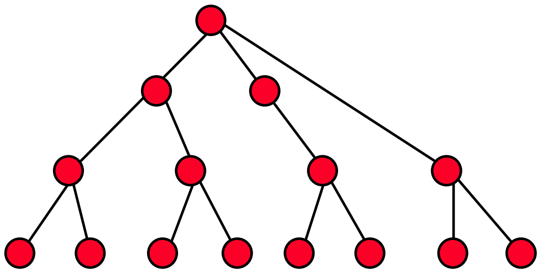
star



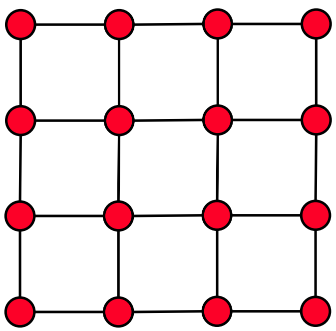
ring



full mesh
(complete graph)



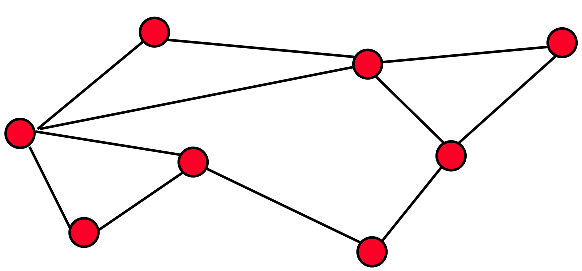
Tree (connected,
loopless graph)



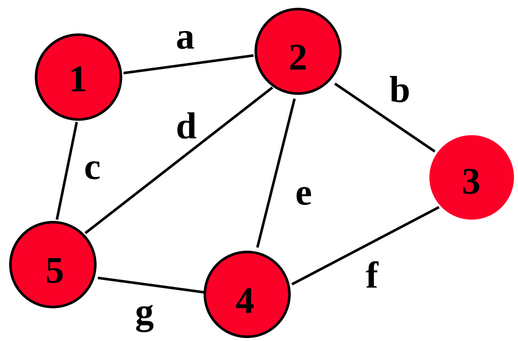
2D (square) lattice

arbitrary topology:
Graph G(N,L)

N : set of nodes
L : set of links



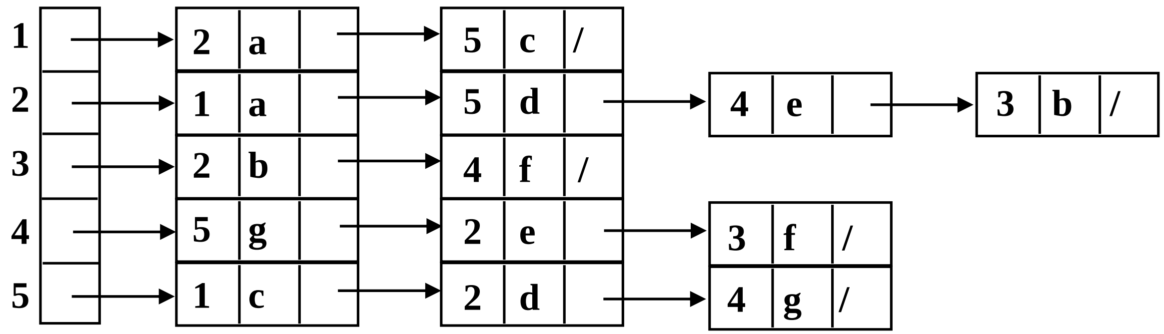
Representations of Graphs



undirected graph G

$$\begin{vmatrix} 0 & a & 0 & 0 & c \\ a & 0 & b & e & d \\ 0 & b & 0 & f & 0 \\ 0 & e & f & 0 & g \\ c & d & 0 & g & 0 \end{vmatrix}$$

adjacency matrix



adjacency-list representation

Graph characteristics

- Number of nodes N=|N|
- Number of edges L=|L| (max: $N(N-1)/2$)

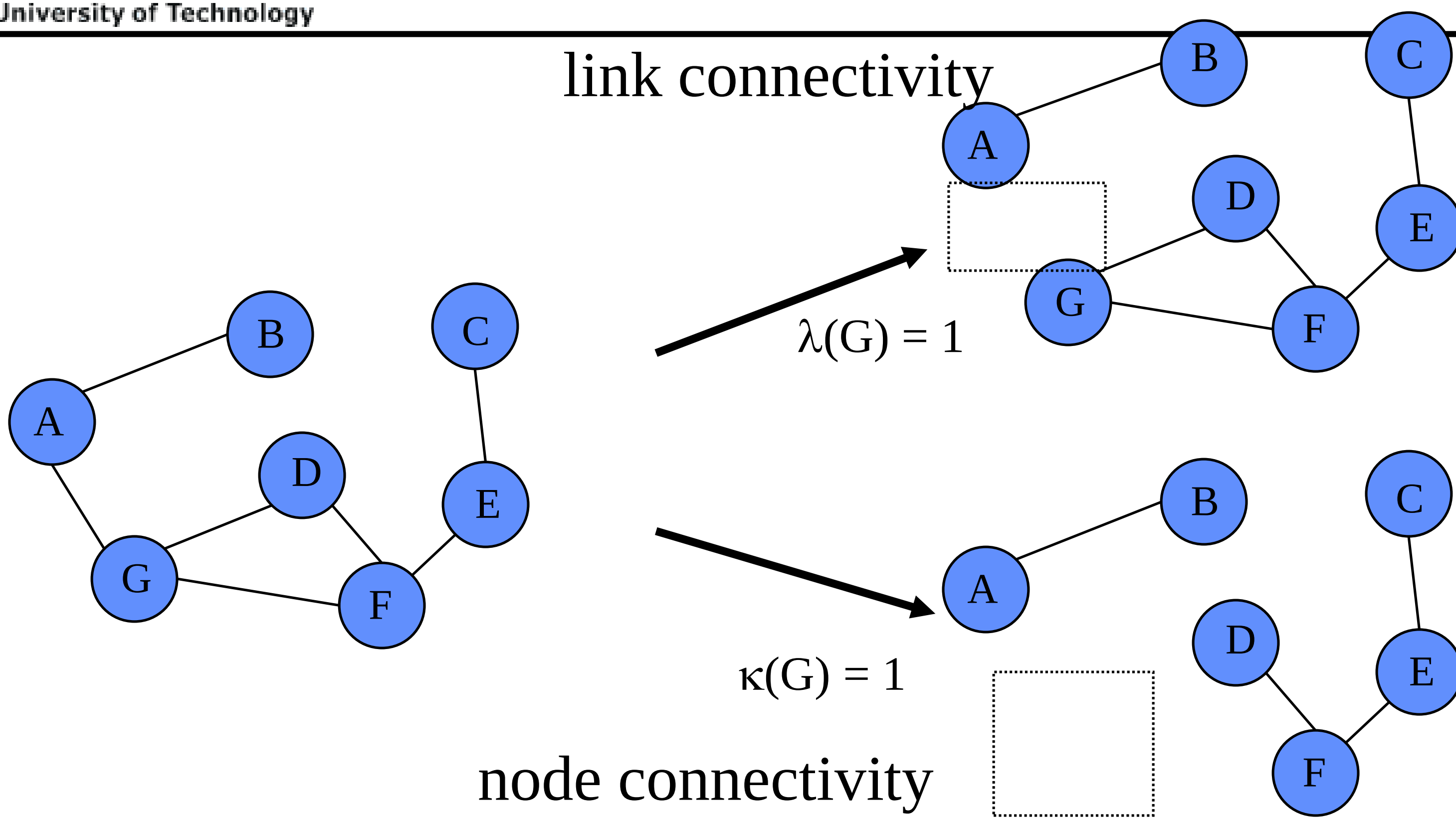
- Degree d_j of node j: number of incident links
- Note that:

$$0 \leq d_j \leq N - 1$$

$$\sum_{j=1}^{N} d_j = 2L$$

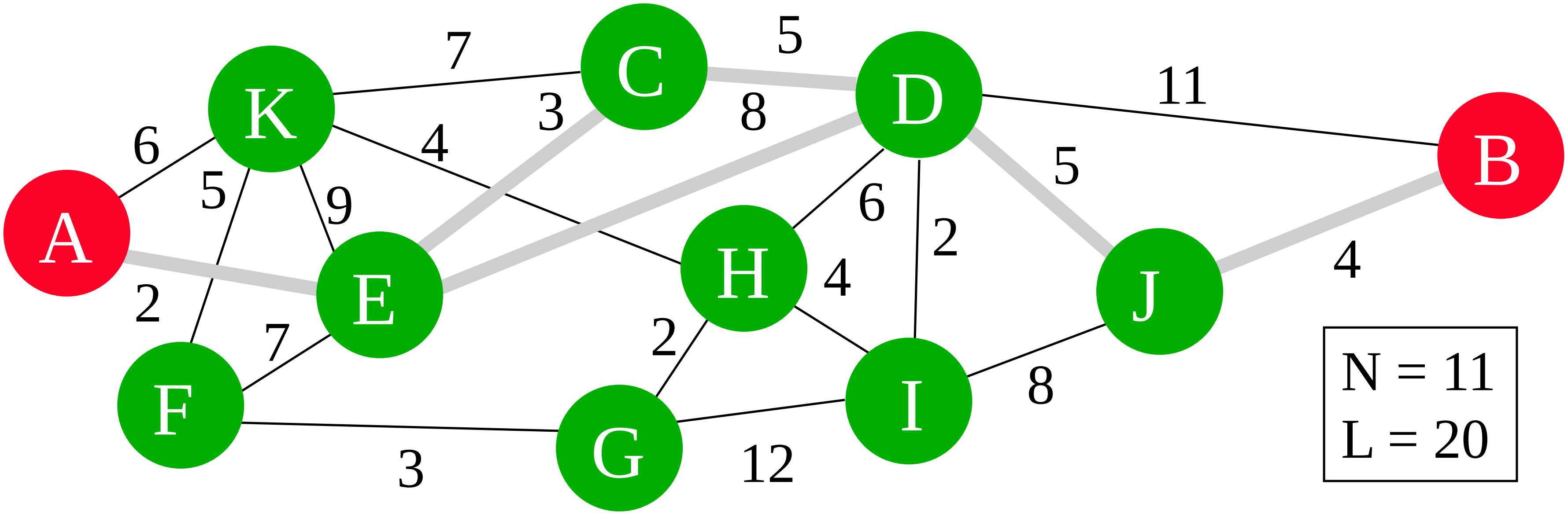
- Minimum degree $\delta(G) \equiv \min_{j \in G} d_j$
- Average degree $d_a \equiv \frac{1}{N} \sum_{j=1}^{N} d_j = \frac{2L}{N}$

Connectivity of a Graph



$\lambda(G)$ (or $\kappa(G)$) : the minimum number of links (nodes) whose removal disconnects G

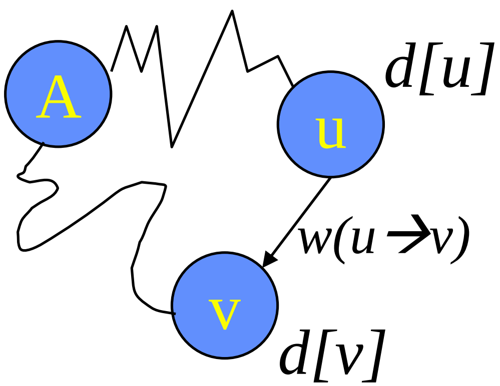
Shortest Path Routing



- Link metric is either *additive* (e.g. delay, cost,...) or *min-max* (e.g. available capacity,...)
- Shortest path P(A,B) is minimizer of $\sum_{i \to j \in P(A,B)} w(i \to j)$

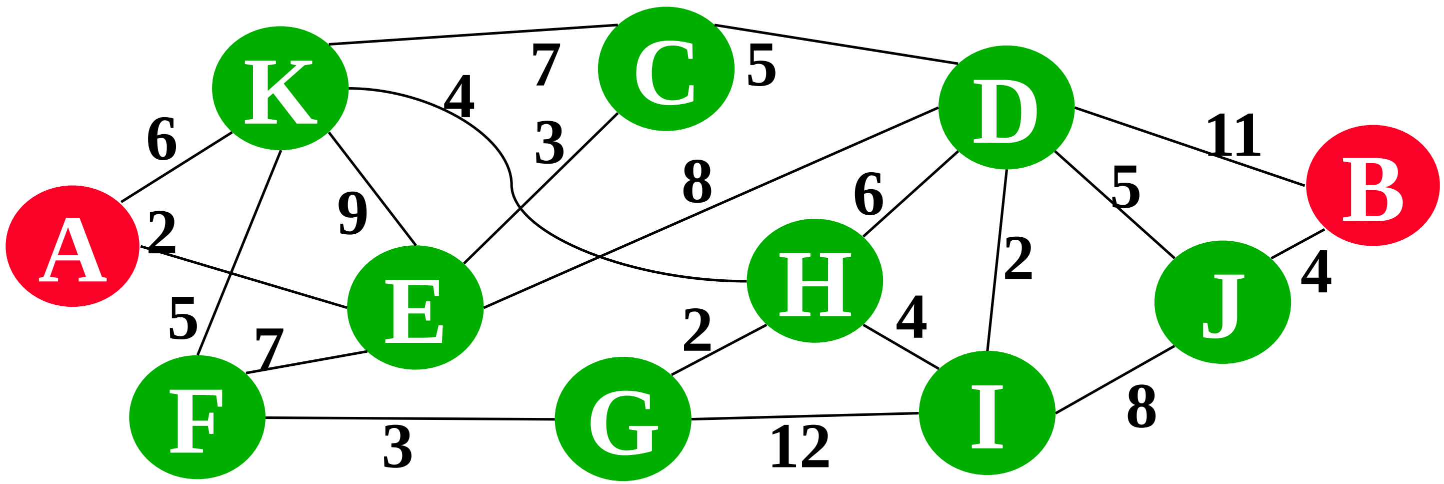
Principles of Shortest Path Routing

- Algorithms:
  - Dijkstra's shortest path algorithm
  - The Bellman-Ford shortest path algorithm
- Subsections of shortest paths are again shortest paths
  - only valid in single parameter routing
- Relaxation:
  - Let $d[n] = w(P_{A\rightarrow n})$. Can we lower $d[n]$ by finding another path from A $\rightarrow$n that has smaller weight
  - estimates of $d[n]$ descend monotonously towards the shortest path



RELAX(u,v,w,d,π)
1. if $d[v] > d[u] + w(u\rightarrow v)$
2. then $d[v] \leftarrow d[u] + w(u\rightarrow v)$
3. $\pi[v] \leftarrow u$

Dijkstra’s Algorithm



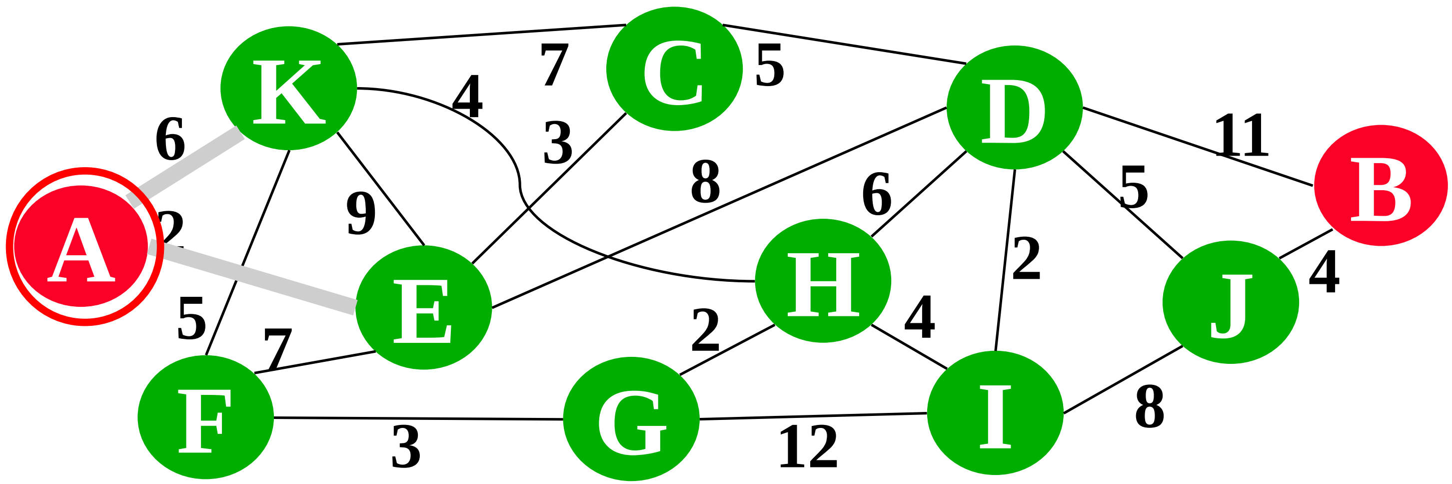
INITIALIZE

Node	Dist	Prev	In-Q
A	0	Local	Y
B	∞	None	Y
C	∞	None	Y
D	∞	None	Y
E	∞	None	Y
F	∞	None	Y

Node	Dist	Prev	In-Q
G	∞	None	Y
H	∞	None	Y
I	∞	None	Y
J	∞	None	Y
K	∞	None	Y

Is $d[u] + w(u \rightarrow v) < d[v]$?

Dijkstra's Algorithm



Extract A, Relax E and K

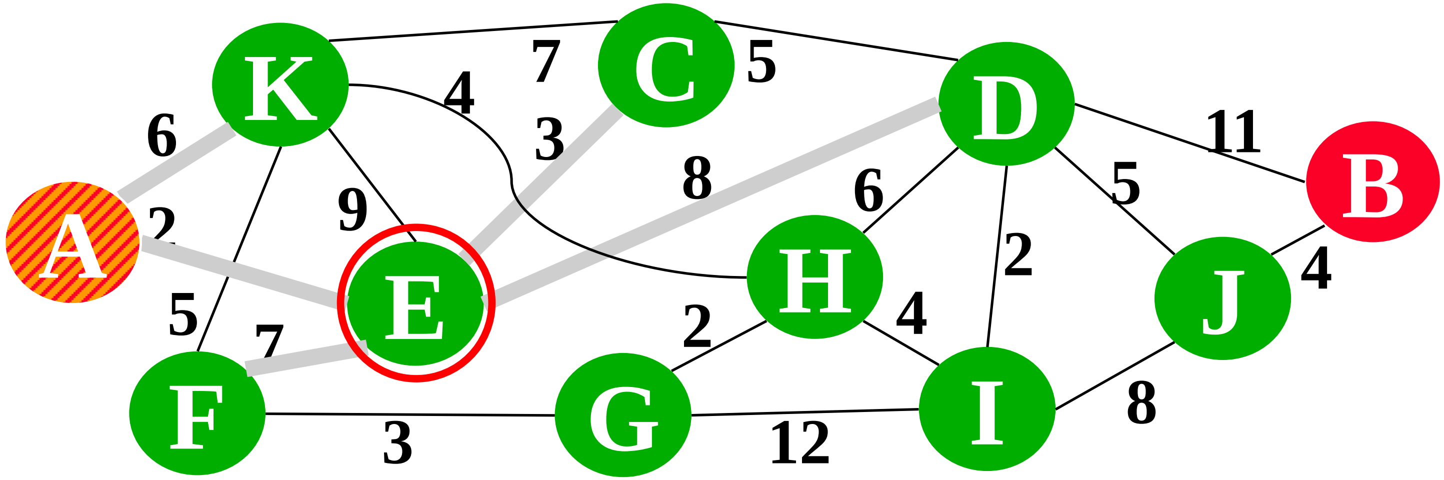
*Queue reorders with updates

Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	Y
K	6	A	Y
B	∞	None	Y
C	∞	None	Y
D	∞	None	Y

Node	Dist	Prev	In-Q
F	∞	None	Y
G	∞	None	Y
H	∞	None	Y
I	∞	None	Y
J	∞	None	Y

Is d[u] + w(u→v) < d[v] ?

Dijkstra's Algorithm



Extract E, Relax C, D, F and K

Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	Y
K	6	A	Y
F	9	E	Y
D	10	E	Y

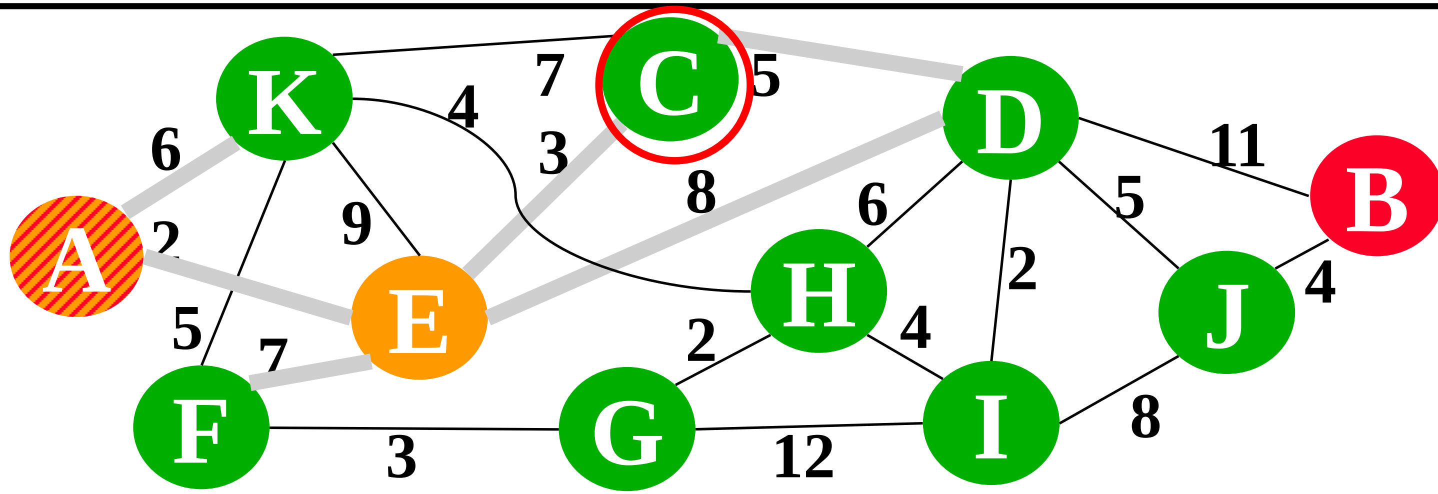
*K doesn't decrease,
C, D and F do

Node	Dist	Prev	In-Q
B	∞	None	Y
G	∞	None	Y
H	∞	None	Y
I	∞	None	Y
J	∞	None	Y

Is $d[u] + w(u \rightarrow v) < d[v]$?

Delft University of Technology

Dijkstra's Algorithm



Extract C, Relax D and K

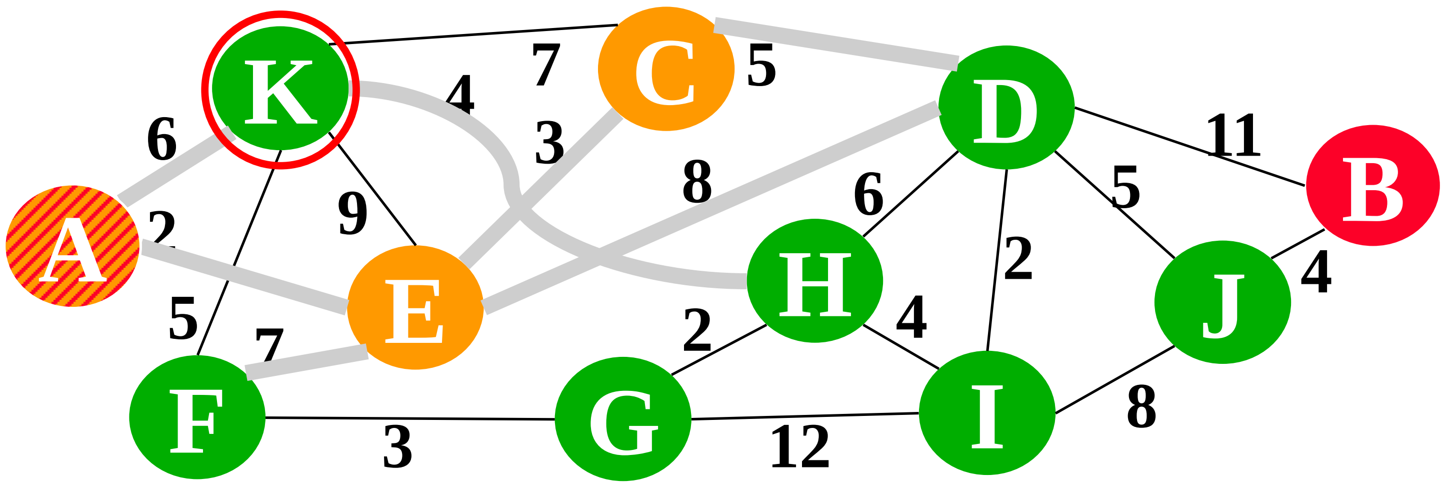
Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	Y
F	9	E	Y
D	10	E/C	Y

Node	Dist	Prev	In-Q
B	∞	None	Y
G	∞	None	Y
H	∞	None	Y
I	∞	None	Y
J	∞	None	Y

*2nd Path Found

Is $d[u] + w(u \rightarrow v) < d[v]$ *?*

Dijkstra’s Algorithm



Extract K, Relax F, H (others already permanently labeled)

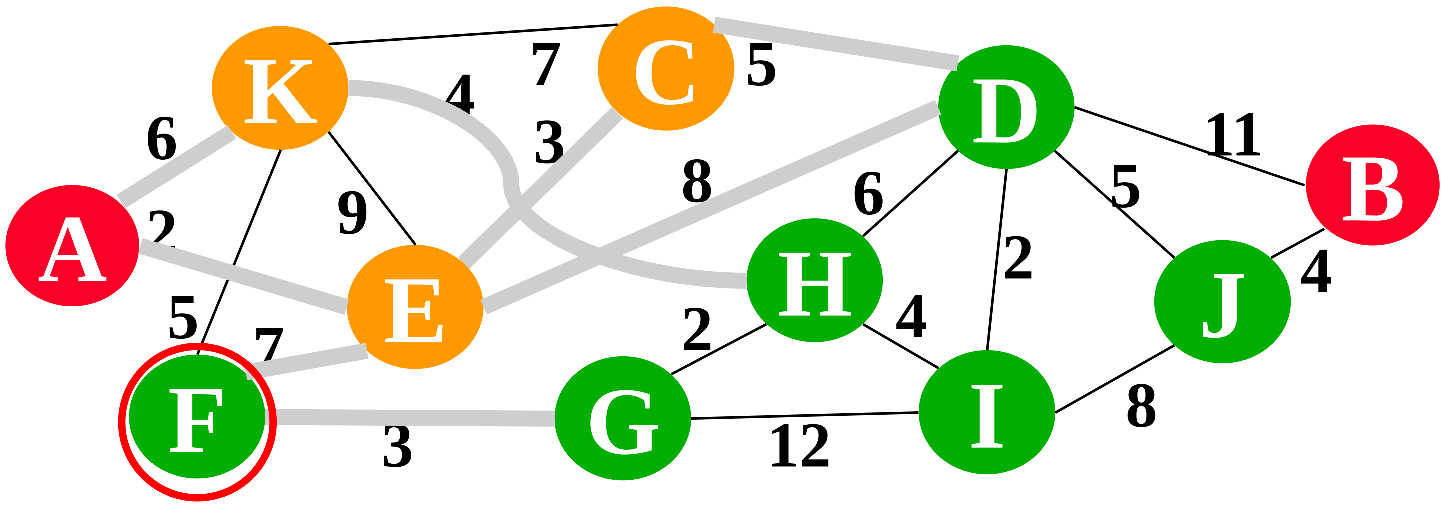
Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	N
F	9	E	Y
D	10	E/C	Y

Node	Dist	Prev	In-Q
H	10	K	Y
B	∞	None	Y
G	∞	None	Y
I	∞	None	Y
J	∞	None	Y

Is $d[u] + w(u \rightarrow v) < d[v]$ *?*

Dijkstra's Algorithm

TU Delft
Delft University of Technology



Extract F, Relax G

Need to toss for next extract

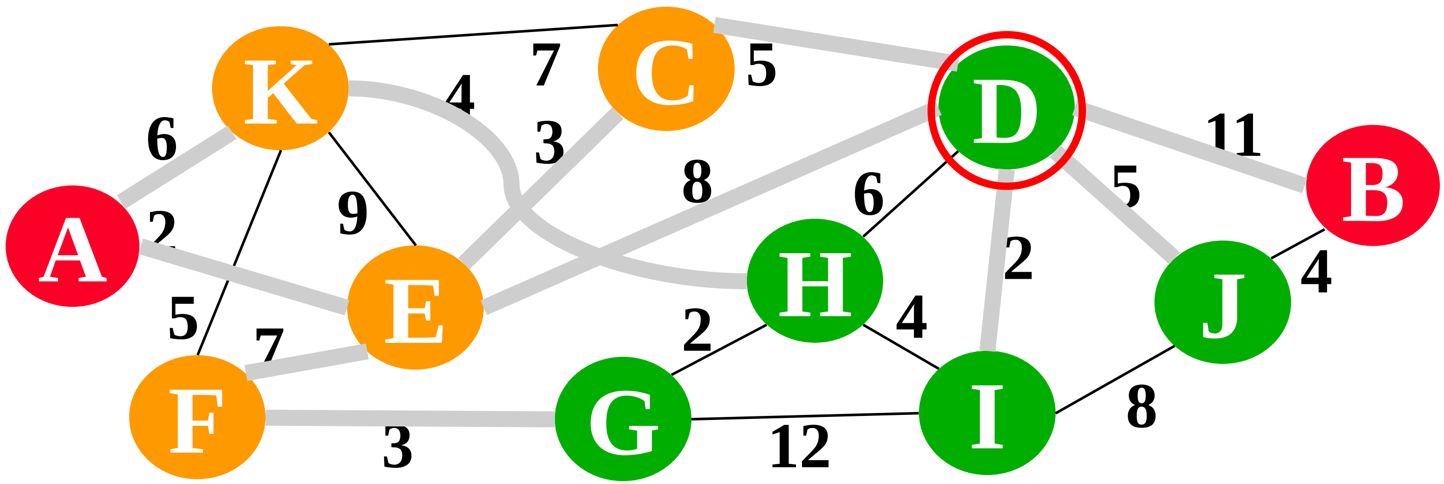
Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	N
F	9	E	N
D	10	E/C	Y

Node	Dist	Prev	In-Q
H	10	K	Y
G	12	F	Y
B	∞	None	Y
I	∞	None	Y
J	∞	None	Y

Is d[u] + w(u→v) < d[v] ?

Delft University of Technology

Dijkstra’s Algorithm



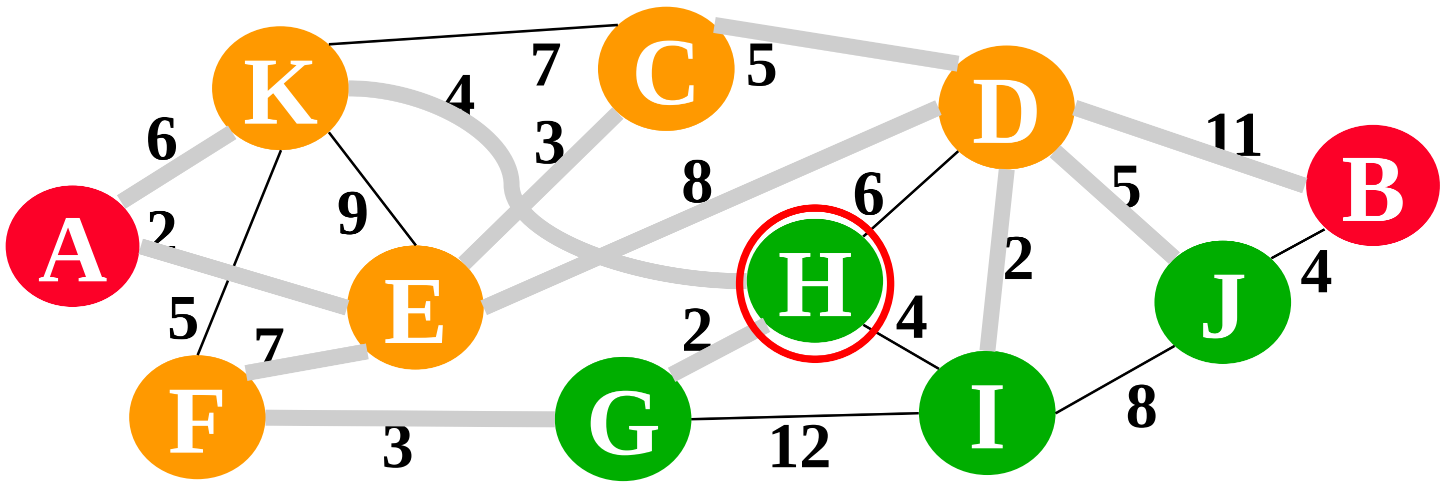
Extract D, Relax B, H, I, J

Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	N
F	9	E	N
D	10	E/C	N

Node	Dist	Prev	In-Q
H	10	K	Y
G	12	F	Y
I	12	D	Y
J	15	D	Y
B	21	D	Y

Is d[u] + w(u→v) < d[v] ?

Dijkstra's Algorithm



Extract H, Relax G and I

Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	N
F	9	E	N
D	10	E/C	N

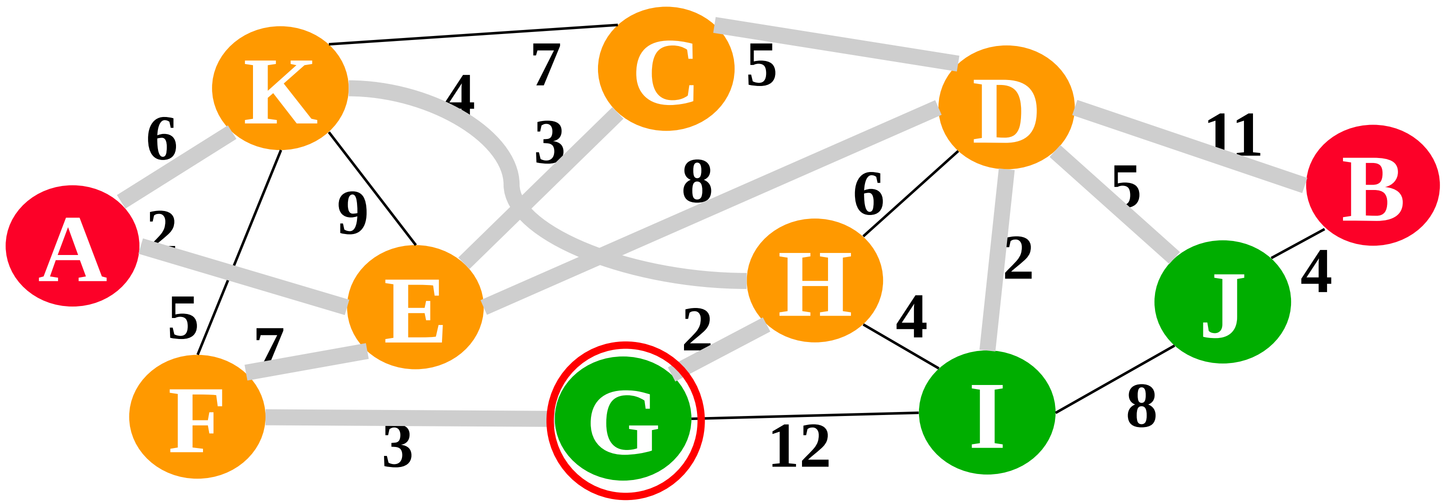
Node	Dist	Prev	In-Q
H	10	K	N
G	12	F/H	Y
I	12	D	Y
J	15	D	Y
B	21	D	Y

*2nd Path Found

Is $d[u] + w(u \rightarrow v) < d[v]$ *?*

Dijkstra's Algorithm

Delft University of Technology



Extract G (tossed in favor of I), Relax I

Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	N
F	9	E	N
D	10	E/C	N

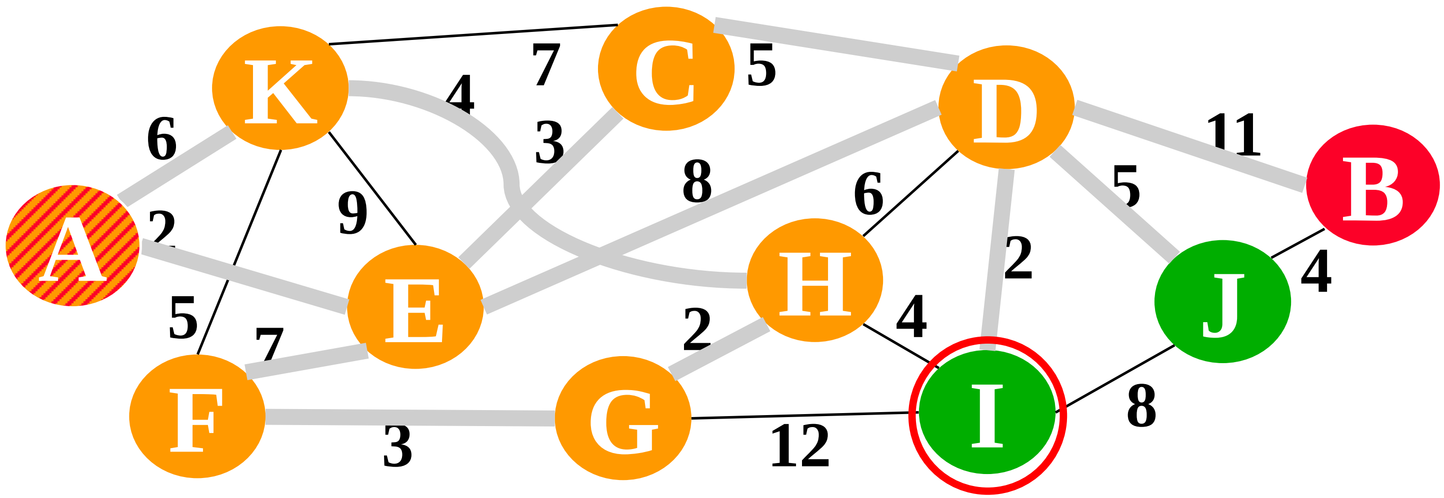
Node	Dist	Prev	In-Q
H	10	K	N
G	12	F/H	N
I	12	D	Y
J	15	D	Y
B	21	D	Y

Is $d[u] + w(u \rightarrow v) < d[v]$?

TU Delft

Delft University of Technology

Dijkstra’s Algorithm



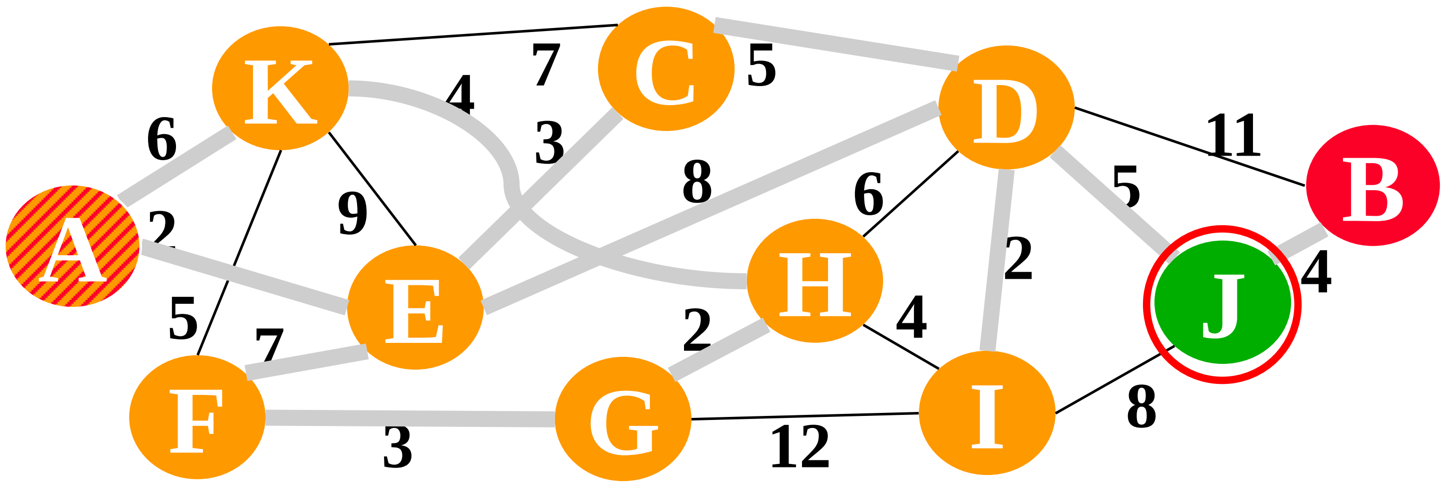
Extract I, Relax J

Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	N
F	9	E	N
D	10	E/C	N

Node	Dist	Prev	In-Q
H	10	K	N
G	12	F/H	N
I	12	D	N
J	15	D	Y
B	21	D	Y

Is $d[u] + w(u \rightarrow v) < d[v]$ *?*

Dijkstra's Algorithm



Extract J, Relax B

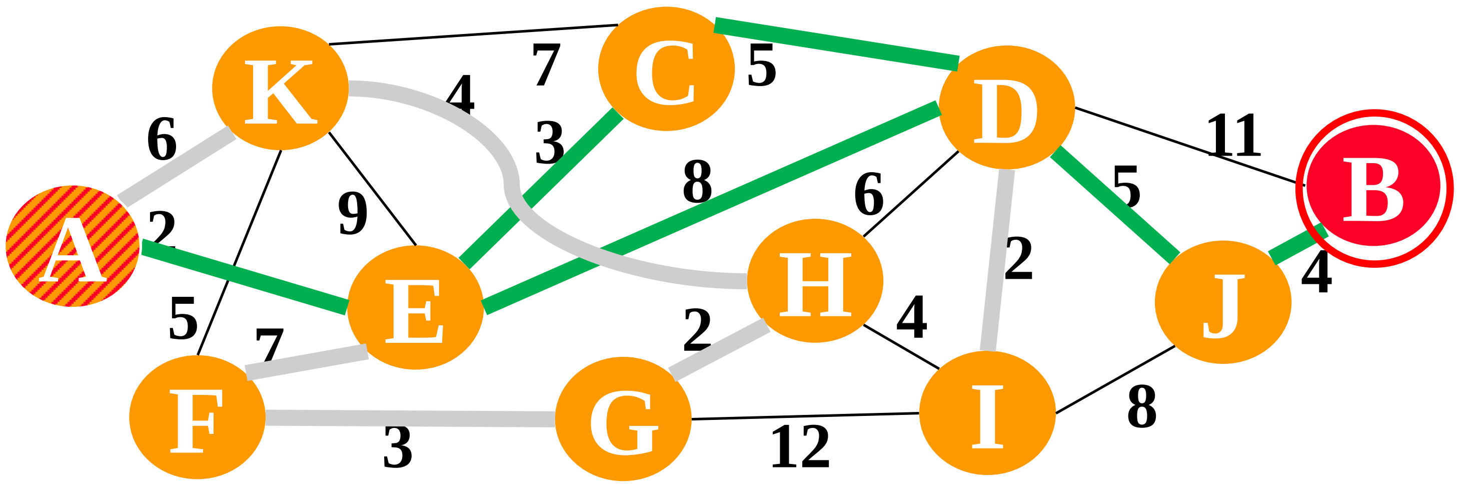
Node	Dist	Prev	In-Q
A	0	Local	N
E	2	A	N
C	5	E	N
K	6	A	N
F	9	E	N
D	10	E/C	N

Node	Dist	Prev	In-Q
G	12	F/H	N
I	12	D	N
H	14	G	N
J	15	D	N
B	19	J	Y

Is $d[u] + w(u \rightarrow v) < d[v]$ *?*

Delft University of Technology

Dijkstra's Algorithm

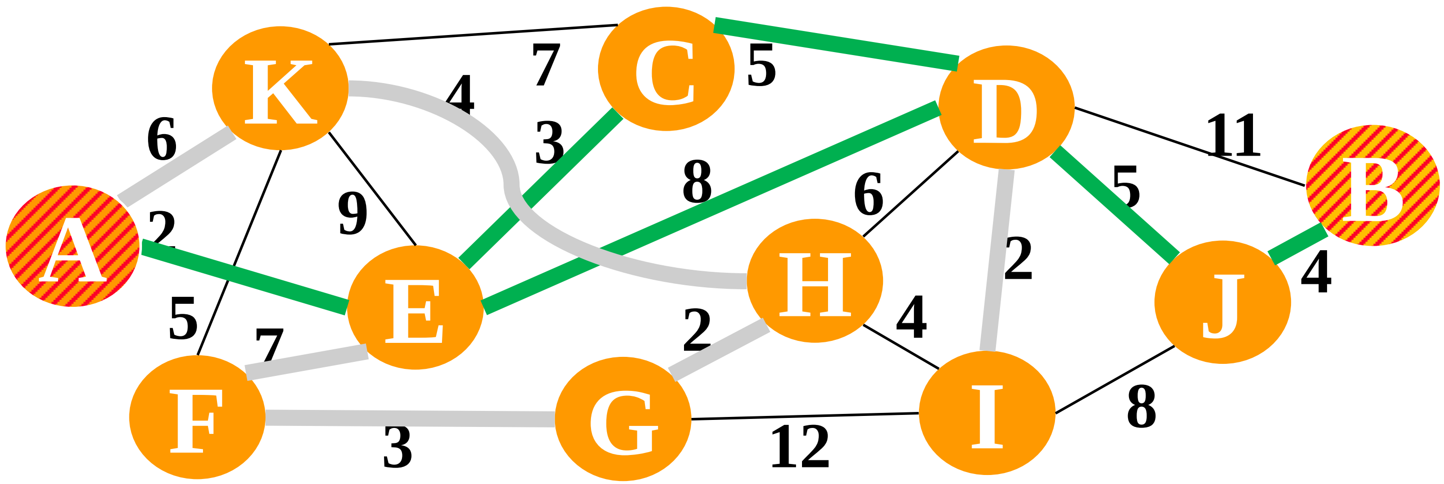


Extract B, Return-path Reverse of Prev values from B to A

Node	Dist	Prev	In-Q
A	0	**Local**	N
E	2	**A**	N
C	5	**E**	N
K	6	A	N
F	9	E	N
D	10	**E/C**	N

Node	Dist	Prev	In-Q
G	12	F/H	N
I	12	D	N
H	14	G	N
J	15	**D**	N
B	19	**J**	N

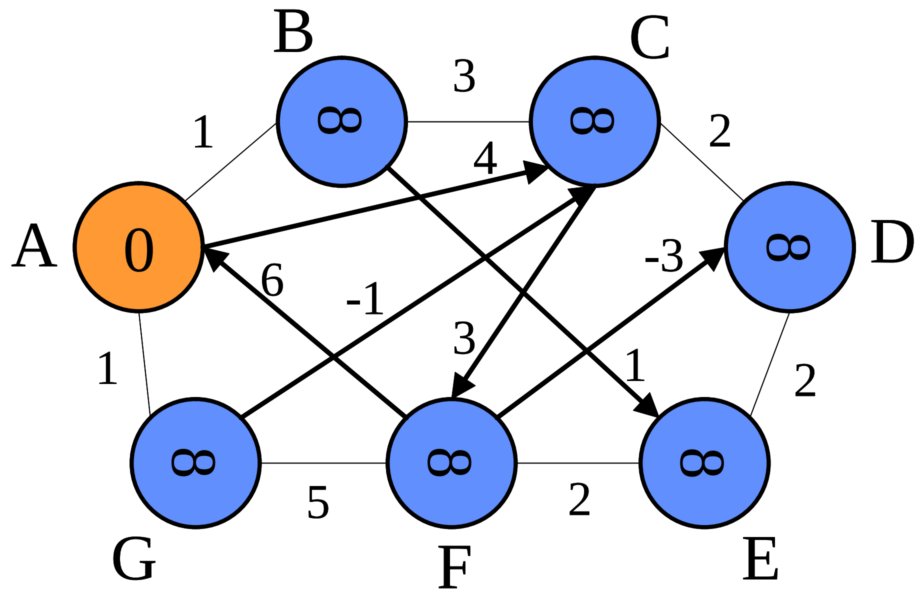
Dijkstra's Algorithm



- Other paths also found
- Once extracted, node distance is minimal and definite (label-setting)
- If B wasn't furthest path, early stop could have occurred

- Based on queue implementation, computation complexity varies: O(|N|+|L|.log|L|) using Fibonacci heap

Bellman-Ford



Bellman-Ford (*G*,*A*,*B*)

1. Initialize (*G*,*A*,*d*,*π*)
2. for *h* = *1* to *N*-*1*
3. do for each link $u \rightarrow v$ of *L*
4. Relax(*u*,*v*,*w*,*d*,*π*)

Relax every link per hopcount in each iteration

O(N.L) complexity

Bellman-Ford

Adjacency list

u	v	w
A	B	1
A	C	1
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Initialize vector of distances and previous nodes

Node	Distance	Prev
A	0	Local
B	∞	None
C	∞	None
D	∞	None
E	∞	None
F	∞	None
G	∞	None

Is $d[u] + w(u \rightarrow v) < d[v]$ *?*

Bellman-Ford

Adjacency list

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

h=1

Node	Distance	Prev
A	0	Local
B	∞ -> 1	None->A
C	∞	None
D	∞	None
E	∞	None
F	∞	None
G	∞	None

Is $d[u] + w(u \rightarrow v) < d[v]$?

$d[A] + w(A, B) < d[B]$

$0 \quad + 1 \quad < \infty$ True

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	∞ -> 4	None->A
D	∞	None
E	∞	None
F	∞	None
G	∞	None

Is d[u] + w(u→v) < d[v] ?

d[A] + w(A, C) < d[C]

0 + 4 < ∞ True

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	4	A
D	∞	None
E	∞	None
F	∞	None
G	∞ -> 1	None->A

Is $d[u] + w(u \rightarrow v) < d[v]$?

$d[A] + w(A, G) < d[G]$

$0 \quad + 1 \quad < \infty$ True

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	4	A
D	∞	None
E	∞	None
F	∞	None
G	1	A

Is $d[u] + w(u \rightarrow v) < d[v]$?

$d[B] + w(B, A) < d[A]$

$1 \quad + 1 \qquad < 0$ False

Bellman-Ford

Delft University of Technology

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	4	A(/B)
D	∞	None
E	∞	None
F	∞	None
G	1	A

Is d[u] + w(u→v) < d[v] ?

d[B] + w(B, C) < d[C]

1 + 3 < 4

2nd path found!

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	4	A(/B)
D	∞	None
E	∞ -> B	None->2
F	∞	None
G	1	A

Is $d[u] + w(u \rightarrow v) < d[v]$?

$d[B] + w(B, E) < d[E]$

$1 \quad + 1 \quad < \infty$ True

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	4	A(/B)
D	∞	None
E	B	2
F	∞	None
G	1	A

Is d[u] + w(u→v) < d[v] ?

$d[C] + w(C, B) < d[B]$

$4 + 3 < 1$ False

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	4	A(/B)
D	∞ -> 6	None->C
E	B	2
F	∞	None
G	1	A

Is d[u] + w(u→v) < d[v] ?

$d[C] + w(C, D) < d[D]$

$4 \quad + 2 \qquad < \infty$ True

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	Distance	Prev
A	0	Local
B	1	A
C	4	A(/B)
D	6	C
E	2	B
F	∞ -> 7	None->C
G	1	A

Is $d[u] + w(u \rightarrow v) < d[v]$?

$d[C] + w(C, F) < d[F]$

$4 \quad + 3 \qquad < \infty$ True

Bellman-Ford

Adjacency list

h=1

u	v	w
A	B	1
A	C	4
A	G	1
B	A	1
B	C	3
B	E	1
C	B	3
C	D	2
C	F	3
D	C	2

u	v	w
D	E	2
E	D	2
E	F	2
F	A	6
F	D	-3
F	E	2
F	G	5
G	A	1
G	C	-1
G	F	5

Node	**Distance**	**Prev**
A	0	Local
B	1	A
C	4	A(/B)
D	6 -> 4	C -> E
E	2	B
F	7	C
G	1	A

Is $d[u] + w(u \rightarrow v) < d[v]$ *?*

$d[E] + w(E, D) < d[D]$

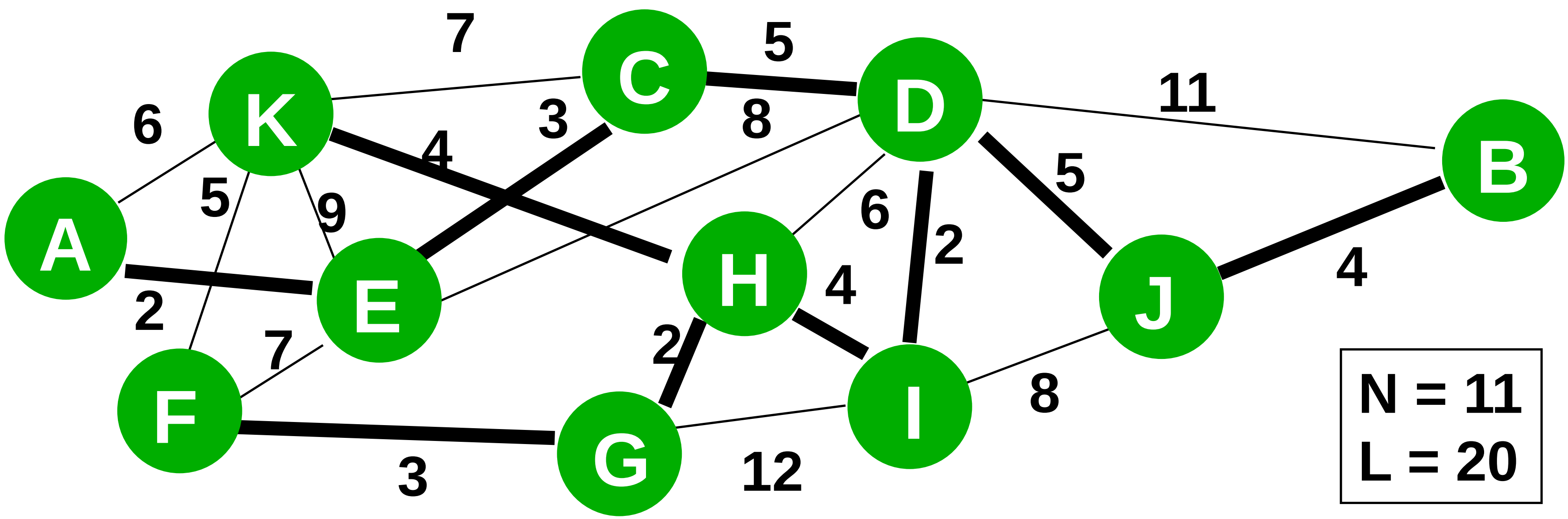
$2 \quad + 2 \qquad < 6$ True, REPLACE!!!

Bellman-Ford

- Finish for all L
  - Repeat procedure until no changes
  - At most N-1 times

- Compared to Dijkstra:
  - O(N.L) complexity, theoretically longer in execution
  - Much easier to program (no min-queues)
  - Handles negative weights correctly (given no loops)
- Optional Nth iteration detects loops
  - Shouldn’t change

Minimum Spanning Trees

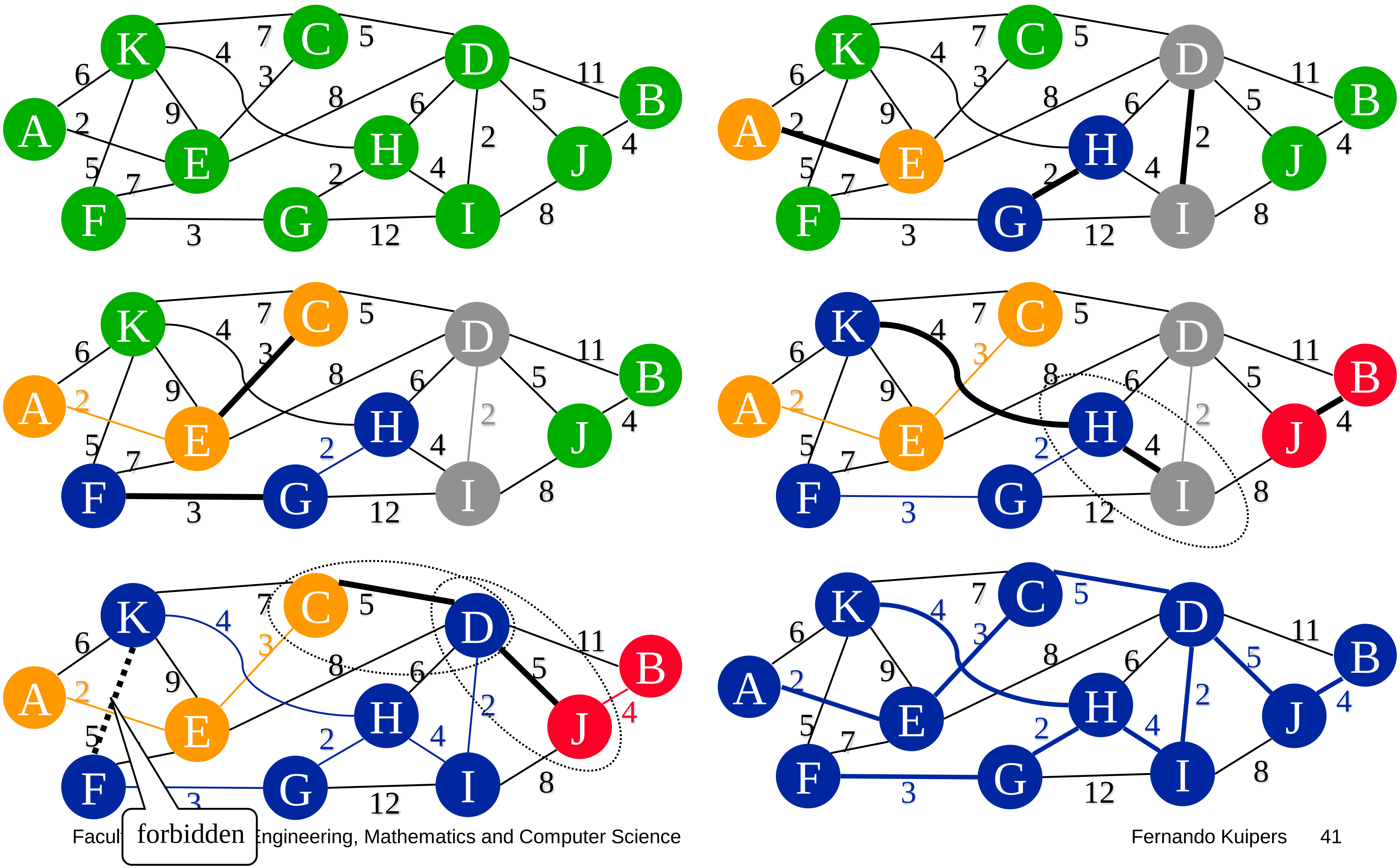


- Best (all terminal) multicast interconnection
- Prim's and Kruskal's algorithm: greedy
- Intelligent flooding?

TU Delft

Delft University of Technology

Kruskal’s Algorithm



Kruskal's Algorithm

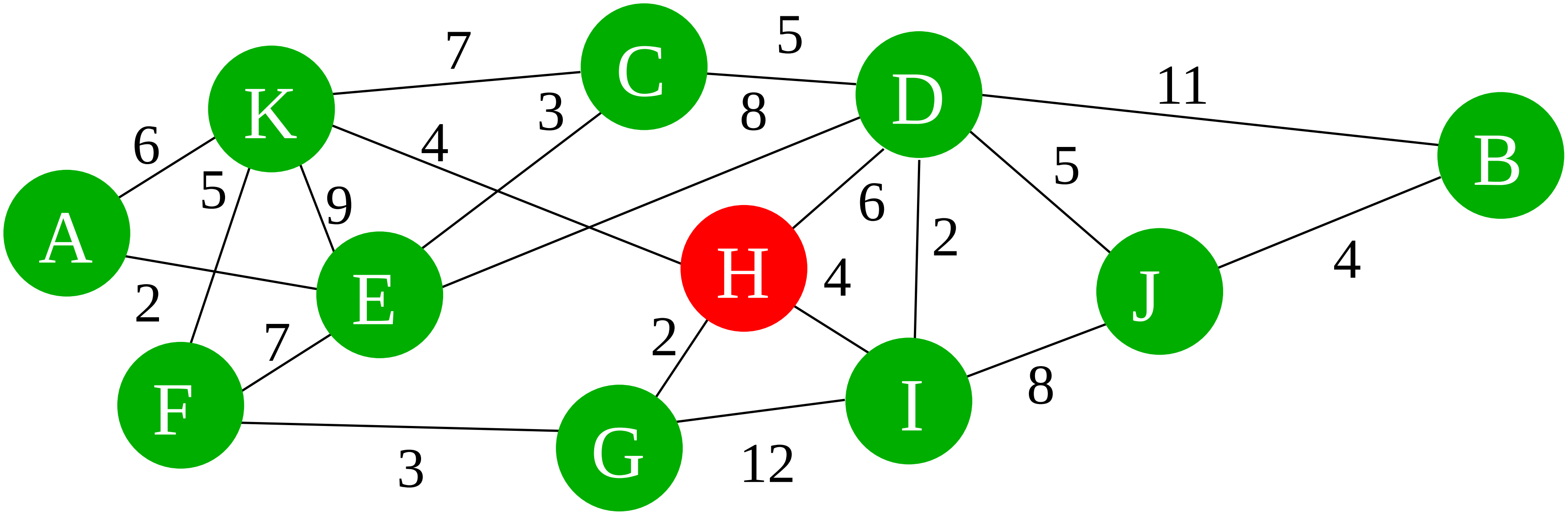
- Select by independent link weight
- Keep administration of trees in memory

- Easy to program (sort links by weight)
- Low computation complexity $O(|L|.\log|L|)$

TU Delft

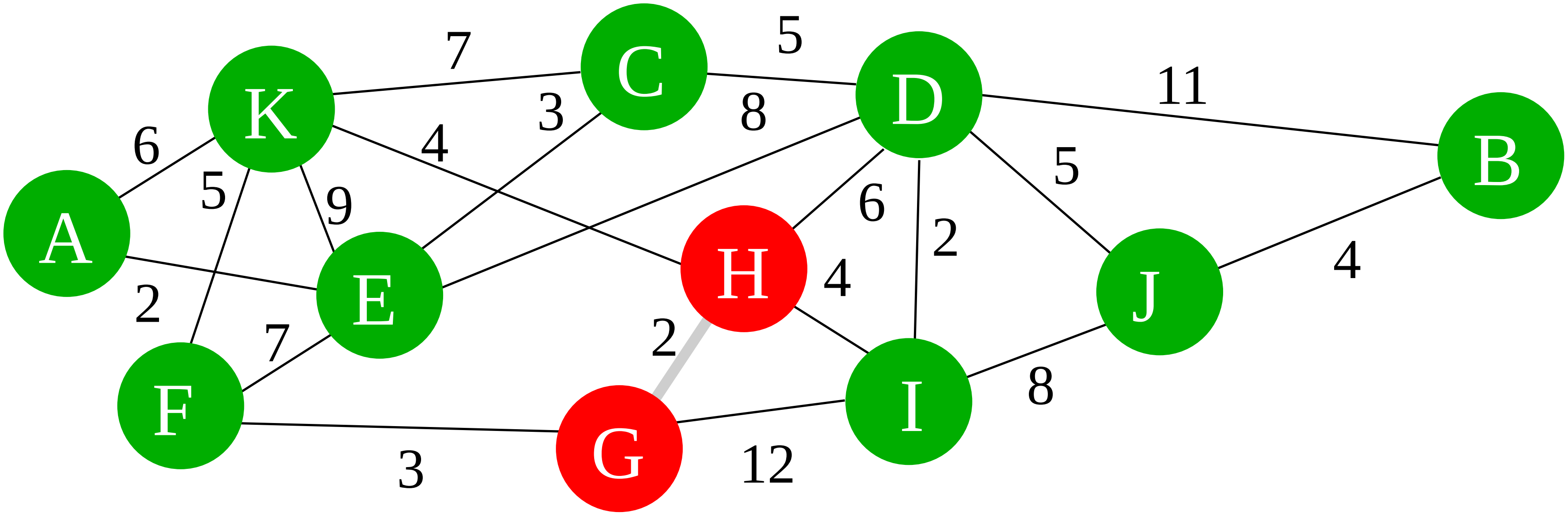
Delft University of Technology

Prim's Algorithm



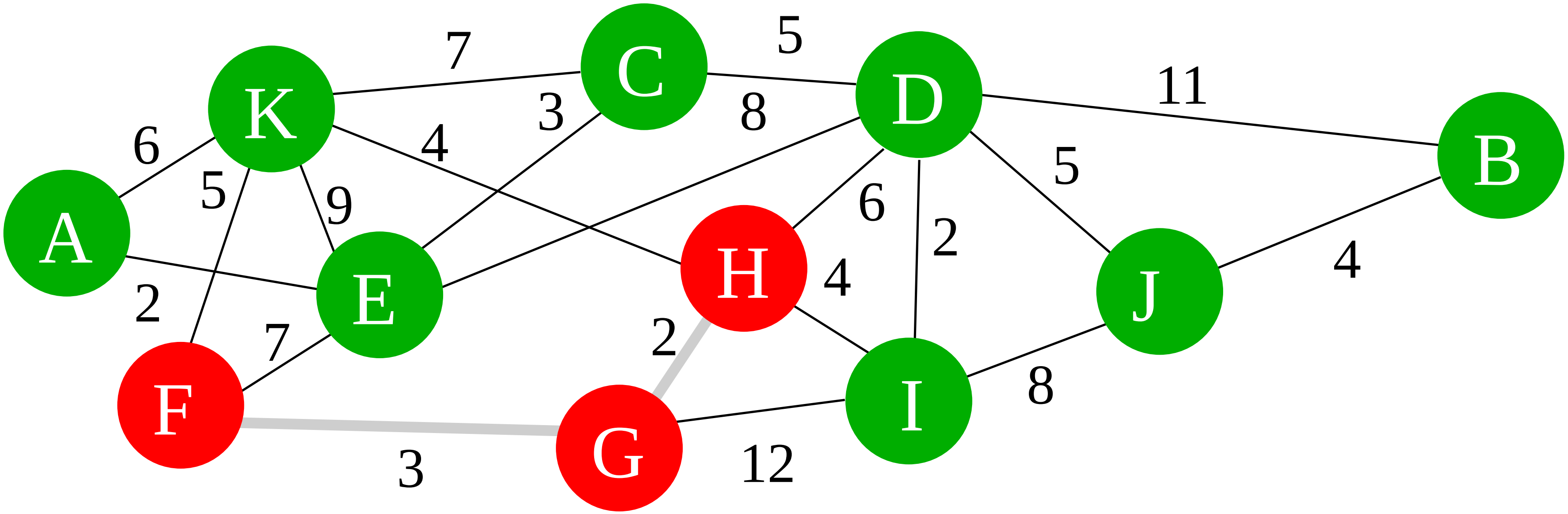
- Select arbitrary starting node as start tree

Prim's Algorithm



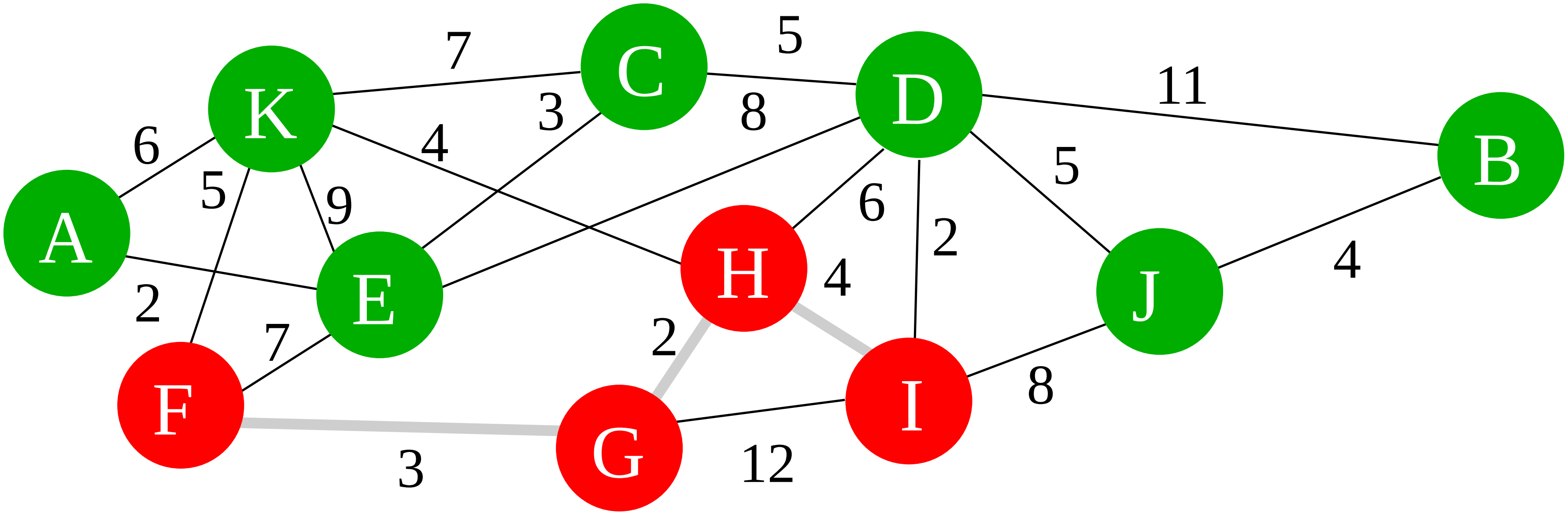
- Select smallest link attached to current tree
- Expand to connected node

Prim's Algorithm



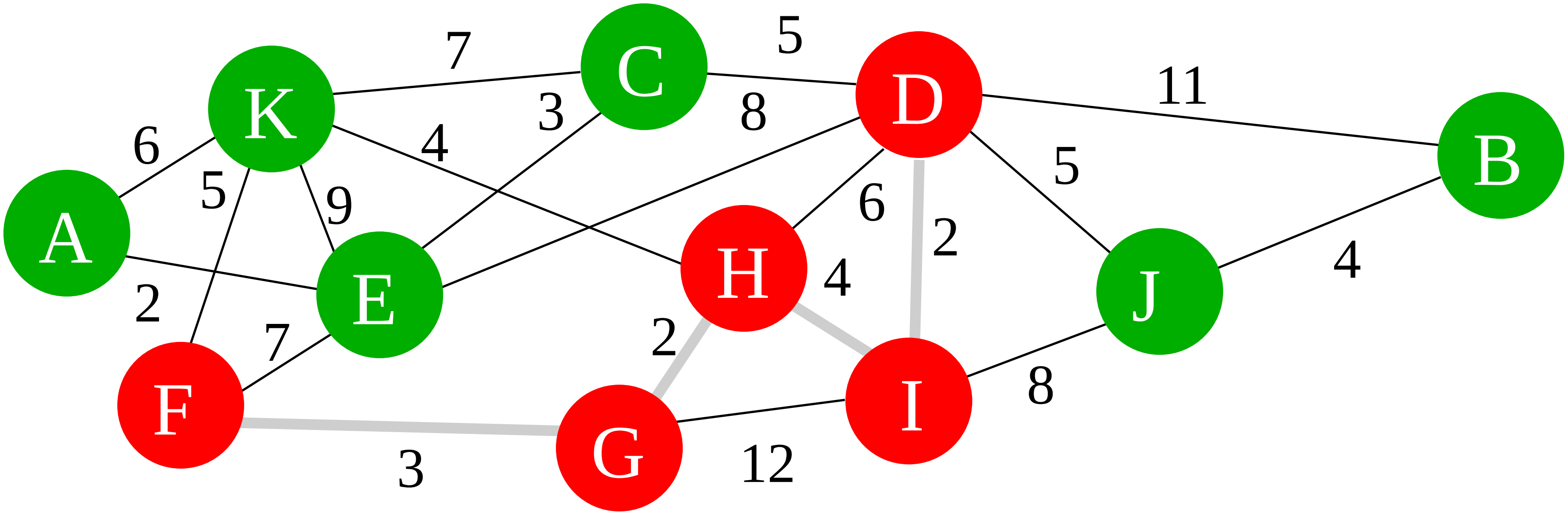
- Repeat …

Prim's Algorithm



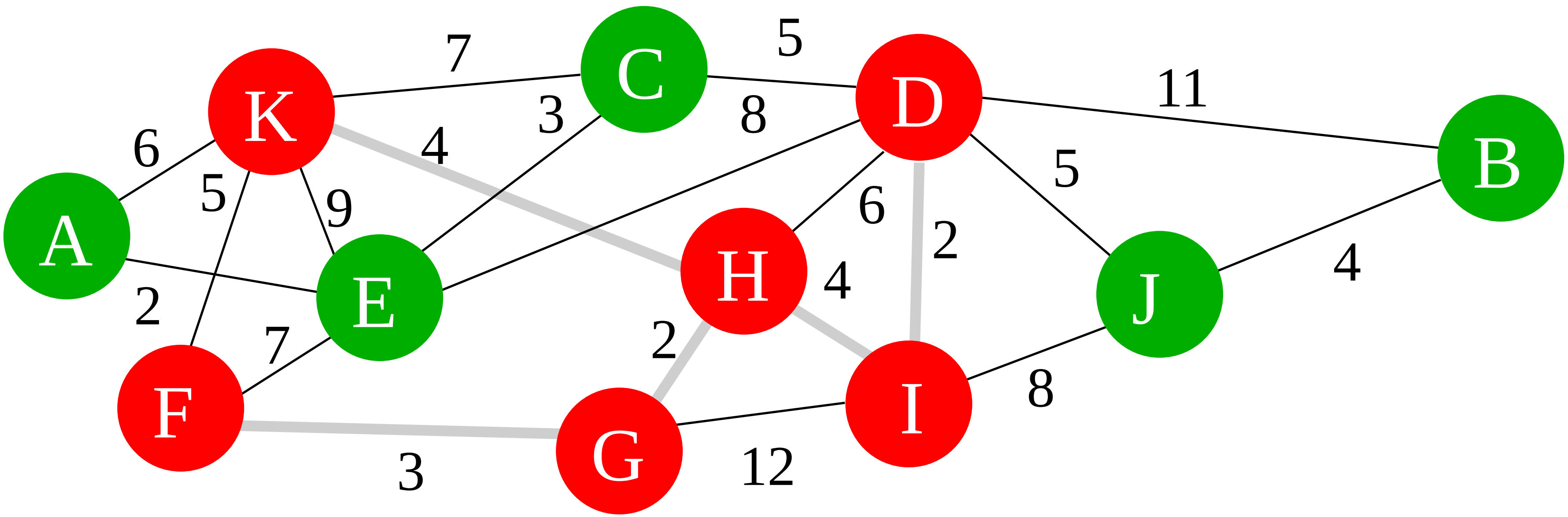
- Repeat until …

Prim's Algorithm



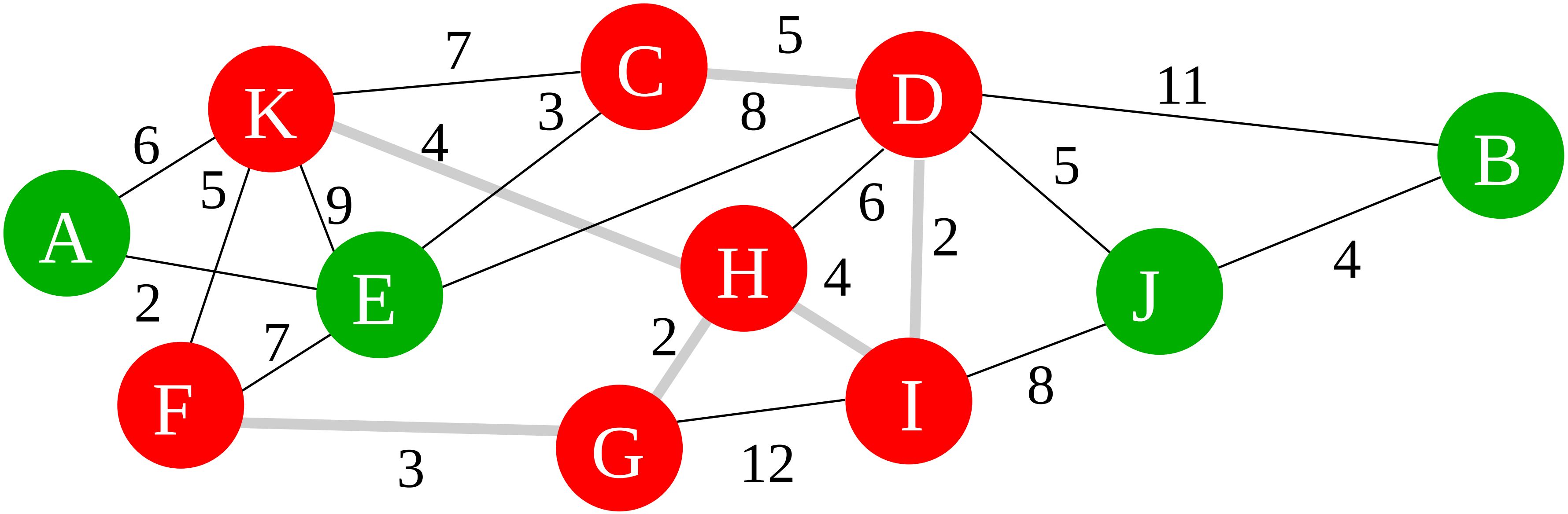
- Repeat until all …

Prim's Algorithm



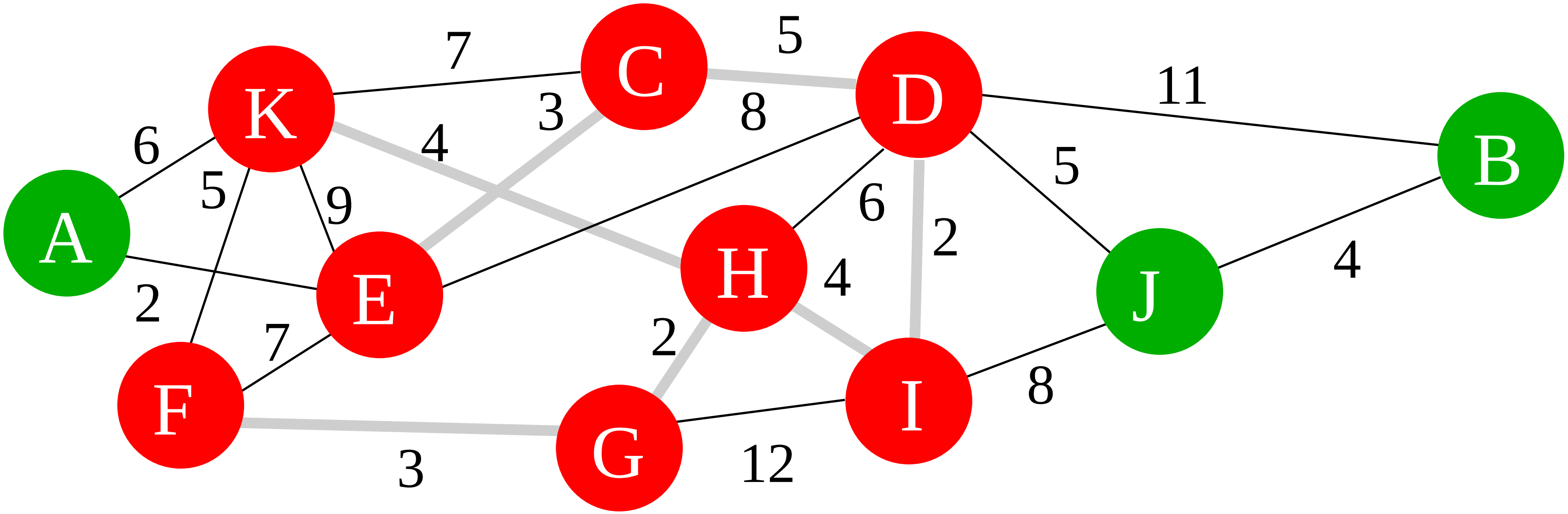
- Repeat until all nodes …

Prim's Algorithm



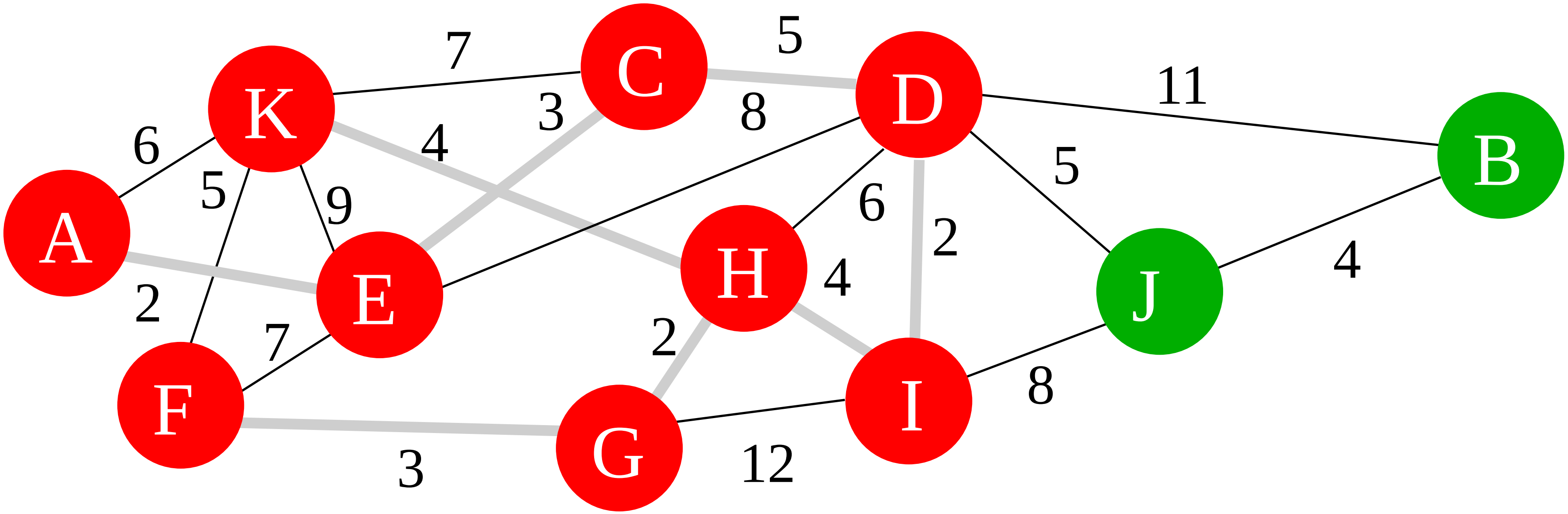
- Repeat until all nodes …

Prim’s Algorithm



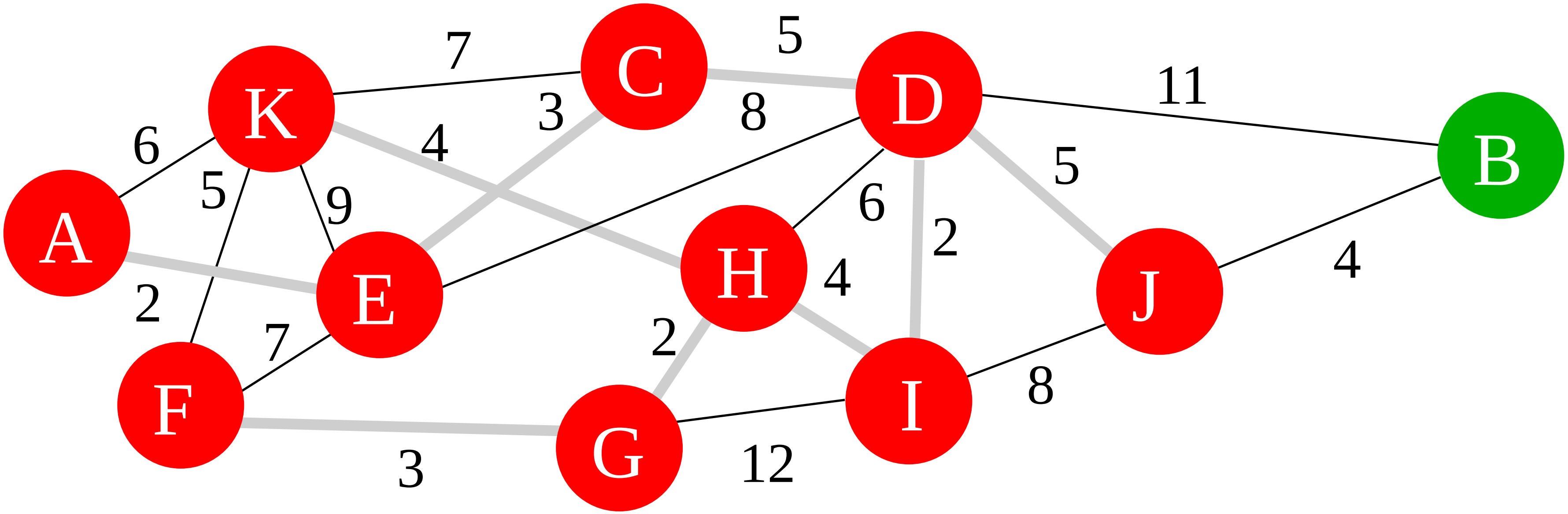
- Repeat until all nodes …

Prim's Algorithm



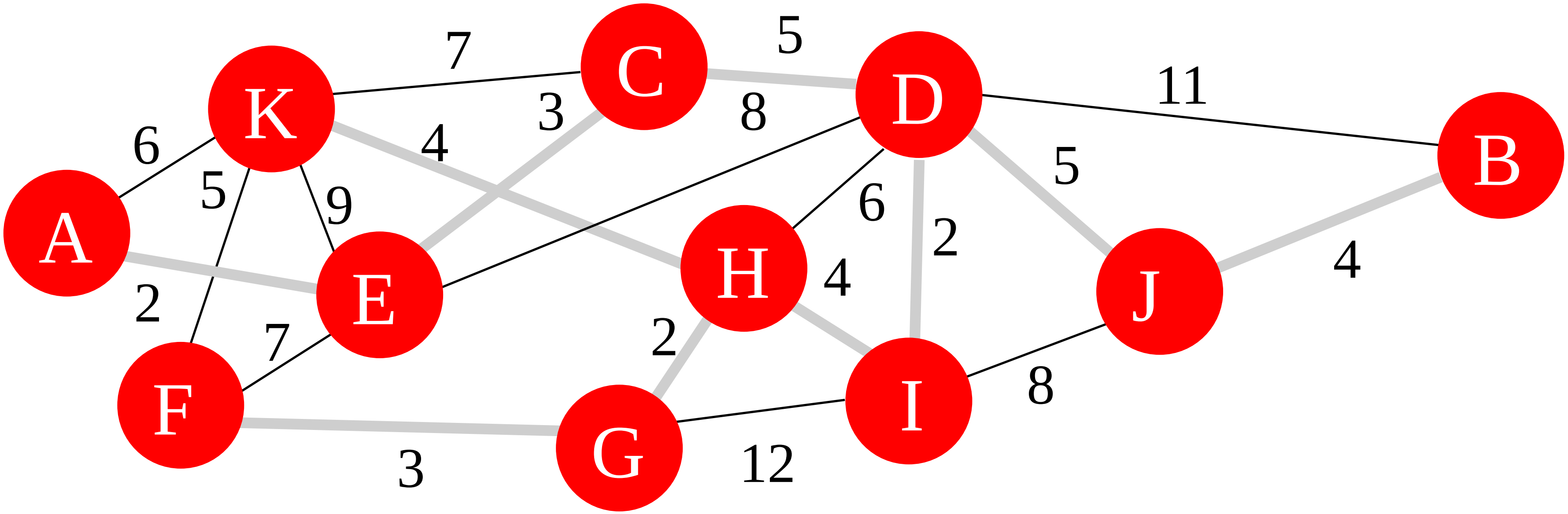
- Repeat until all nodes have …

Prim's Algorithm



- Repeat until all nodes have been …

Prim's Algorithm



- Repeat until all nodes have been found
- Easy to remember, complex to program
- Looks like Dijkstra's algorithm, O(|N|+|L|.log|L|)

Questions Ch. 6

- On network X, compute with algorithm Y the Z (shortest path or minimum spanning tree) from A (to B). Give the solution and an activity table of your calculations.