Introduction
Optimization on graphs

Leo Liberti

LIX, École Polytechnique

`liberti@lix.polytechnique.fr`

Teachers

- Leo Liberti (LL)
  `http://www.lix.polytechnique.fr/~liberti`
- Ruslan Sadykov (RS)
  `http://www.lix.polytechnique.fr/~sadykov`
- Giacomo Nannicini (GN)
  `http://www.lix.polytechnique.fr/~giacomon`

Course website:

```
http://www.lix.polytechnique.fr/˜liberti/
       teaching/isic/isc612-07/
```

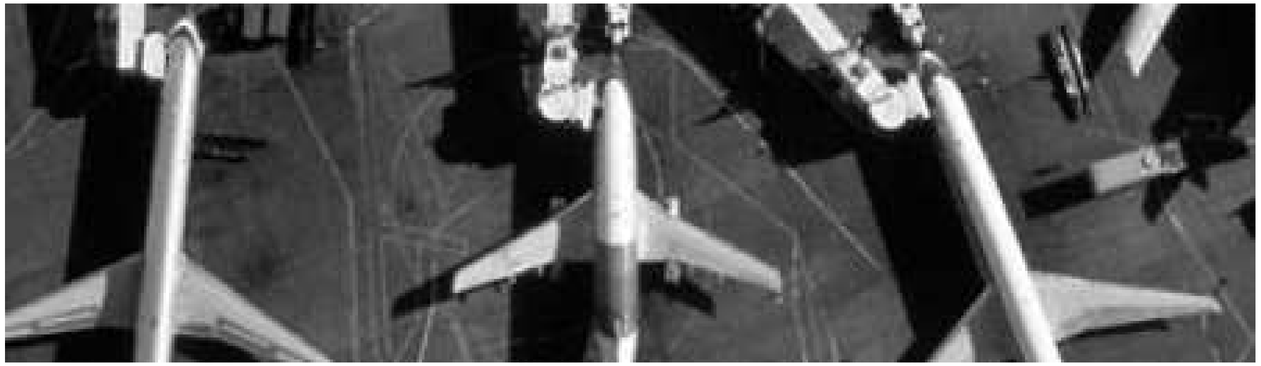
Timetable

```
071025 thu 08:30-12:30 Optimization on graphs    LL/RS
071031 wed 08:30-12:30 Linear programming I      LL/LL
071108 thu 08:30-12:30 Mixed integer programming RS/RS
071115 thu 08:30-12:30 Shortest paths algorithms GN/GN
071122 thu 08:30-12:30 Linear programming II     LL/RS
071129 thu 08:30-12:30 Nonlinear modelling I     LL/LL
071206 thu 08:30-12:30 Constraint programming    RS/RS
071213 thu 08:30-12:30 Nonlinear modelling II    LL/LL
```

- Amphis: S.72 (bat Lévy)
- TDs: SI33
- Exam: team (2/3 students) project on an OR subject to be submitted by 20/12/2007.

Motivations for OR I

CONTINENTAL AIRLINES SPEEDS RECOVERY AFTER 9/11



Operations research firm CALEB Technologies worked with Continental to develop a decision support system to generate near-optimal crew recovery solutions for responding to emergencies. Continental estimates that the system helped it save $40 million in 2001. And thanks to the system, Continental led the American airline industry in recovering operations after September 11, 2001.

Motivations for OR II

PSA PEUGEOT CITROËN SPEEDS UP PRODUCTION

To meet the CEO’s ambitious new targets for growth, innovation, and profitability, the O.R. team at PSA Peugeot Citroën focused on improving production line efficiency in its plants’ car body shops. The O.R. tools they developed improved throughput with minimal capital investment and no compromise in quality — contributing $130 million to the bottom line in 2001 alone.

Motivations for OR III

BRITISH TELECOM SAVES $150 MILLION A YEAR

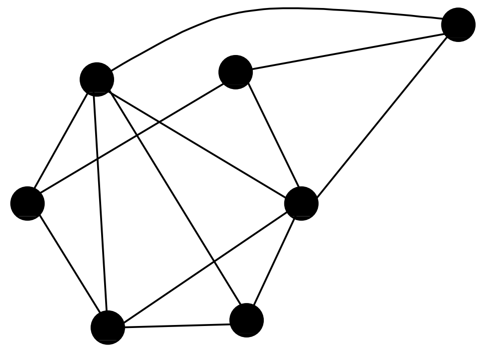
To improve workforce scheduling for thousands of field engineers, the O.R. department at BT developed information systems with “O.R. inside" to automate work management and field communications. Rolled out in 1997, the system was saving BT $150 million a year on operational costs by 2000. When deployed over the targeted workforce of 40,000 people, the system is projected to save $250 million a year.

Motivations for OR IV

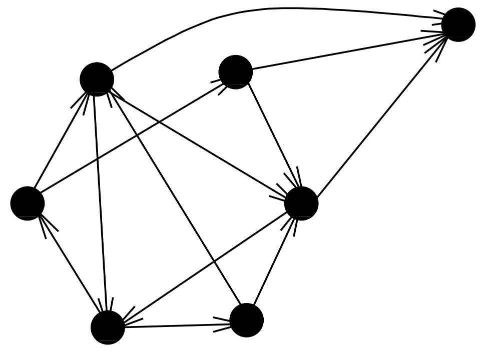
- Many more industrial success cases in `www.scienceofbetter.org`
- Course will focus on some theoretical and algorithmic aspects
- Will also include an applied section on *modelling* of optimization problems

Definitions I

- An *undirected graph* (or simply *graph*) $G = (V, E)$ is a pair where V is a set of *vertices* (or nodes) and E is a set of *edges*; each edge is a set $\{u, v\}$ of 2 vertices $u, v \in V$

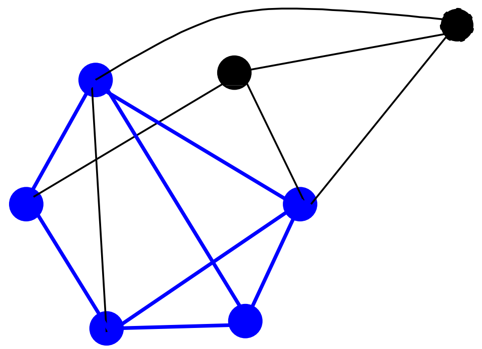


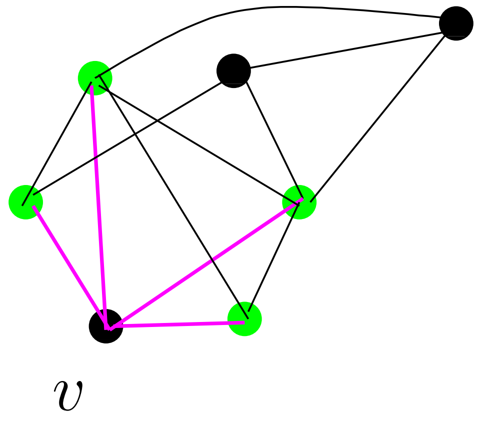
- A *directed graph* (or *digraph*) $G = (V, A)$ is a couple where A is a set of *arcs*; each arc is an ordered pair (u, v) of vertices $u, v \in V$



Definitions II

- A *subgraph* $H=(U,F)$ of a graph (digraph) $G=(V,E)$ is a graph such that $U \subseteq V$, $F \subseteq E$ and for all $\{u,v\} \in F$ we have $u,v \in U$

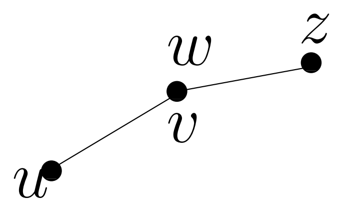




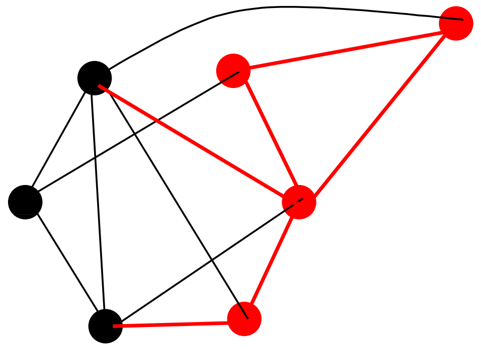
- For each vertex $v \in V$ of a graph $G=(V,E)$ we define the *star* $\delta(v)=\{u \in V \mid \{v,u\} \in E\}$ and the *edge star* $\bar{\delta}(v)=\{\{w,z\} \in E \mid w=v\}$
- For digraphs: *outgoing* ($\delta^+(v)$) and *incoming* stars ($\delta^-(v)$)
- For $F \subseteq E, v \in V$, $\delta_F(v)=\{u \in V \mid \{v,u\} \in F\}$ (similar definitions for $\delta_F^+, \delta_F^-, \bar{\delta}$)

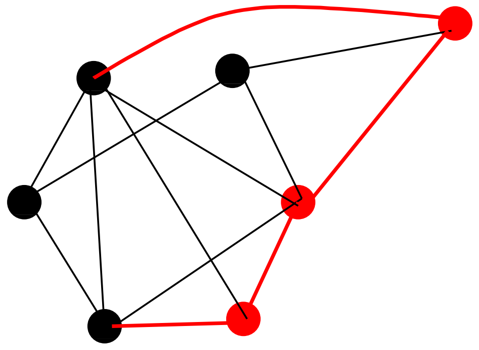
Definitions III

- Two edges (arcs) $\{u, v\}, \{w, z\}$ are *consecutive* in $G = (V, E)$ if $v = w$



- A (directed) *path* p in a graph (digraph) G is a sequence of consecutive edges (arcs) in G

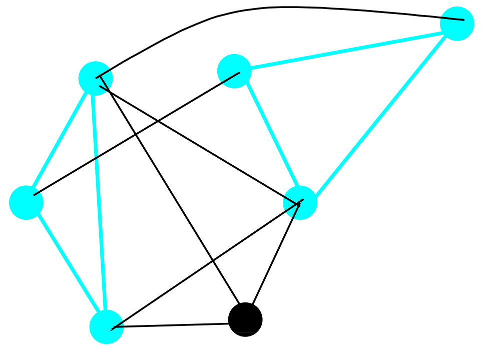
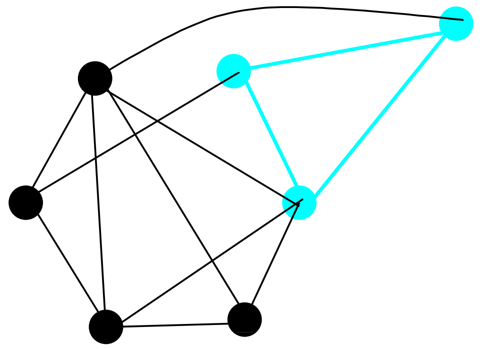




- A path is *simple* if every vertex in the path is incident to at most two edges of the path and if each edge is unique in the sequence

Definitions III

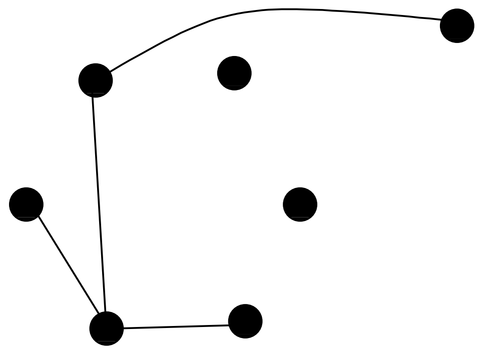
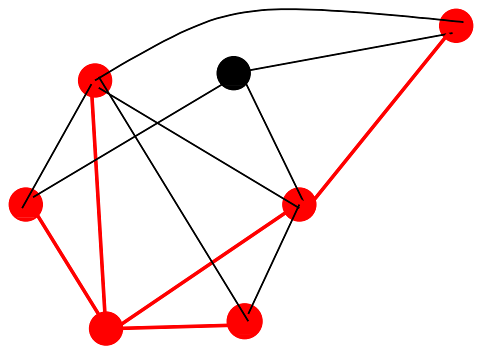
- A graph $G = (V, E)$ is *connected* if there is a path in between each pair of vertices in V
- A digraph is *strongly connected* if there is a directed path between each pair of vertices in V
- A *cycle* C in the graph $G = (V, E)$ is a subgraph (U, F) of G such that for every $u \in U$ the cardinality of $\delta(u)$ in C is even

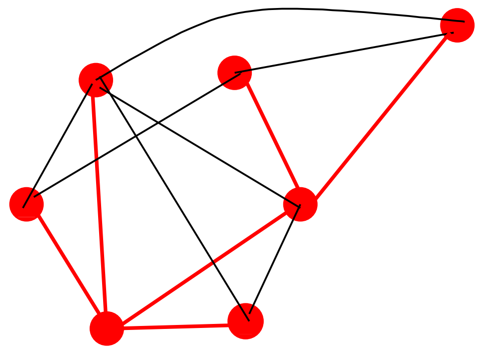
- A cycle is *simple* (or *elementary*) if it is connected and the cardinality of each vertex is exactly 2, *Eulerian* if it passes through each edge once, *Hamiltonian* if it passes through each vertex once

Definitions IV

- A graph G is *acyclic* if no subgraph of G is a cycle

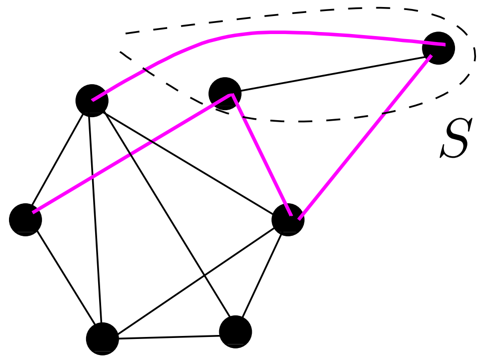



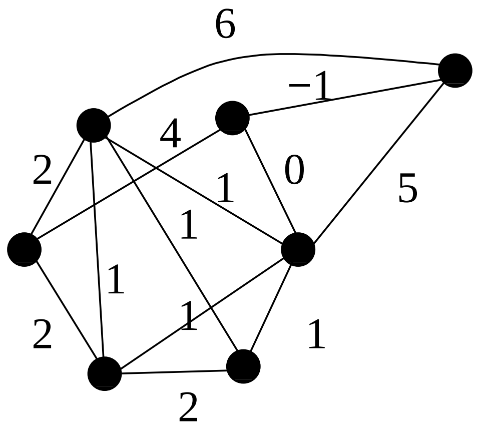
- A *tree* T in a graph $G = (V, E)$ is a connected acyclic subgraph of G
- A tree $T = (U, F)$ in a graph $G = (V, E)$ is *spanning* if $U = V$



Definitions V

- Given a graph $G = (V, E)$ and a subset of vertices $S \subseteq V$, the *cut* $\bar{\delta}(S)$ in G w.r.t S is the set of edges $\{u, v\}$ where $u \in S$ and $v \in V \smallsetminus S$ (can use notation $\bar{\delta}^{+}(S)$ and $\bar{\delta}^{-}(S)$ for digraphs, and $\delta(S)$ to mean the vertices in $V \smallsetminus S$ adjacent to the edges in $\bar{\delta}(S)$)

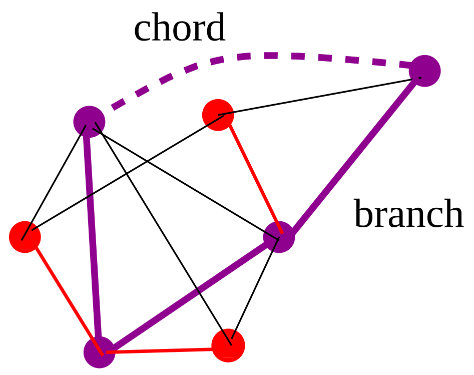


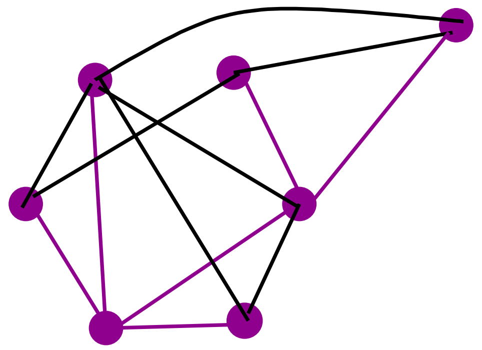


- Sometimes a cut intended as a set of edges (as opposed to a set of vertices adjacent to the edges) is called a *cutset*
- A graph (digraph) $G = (V, E)$ is *weighted* if there is a cost function $c : E \to \mathbb{R}$ on the edges (arcs)

Cycle bases

- The set of all cycles of a graph $G = (V, E)$ forms a vector space over $\mathbb{F}_2$ (i.e. $\{0.1\}$ with XOR addition and usual multiplication)
- Given a graph $G = (V, E)$ and a spanning tree T of G, the edges of the tree are called *branches* and the edges in $G \setminus T$ are called *chords*
- To each chord $e \in G \setminus T$ there corresponds a cycle γ_e consisting of e and the unique path on T joining the endpoints of e





- Such cycles are called *fundamental cycles*; $\{\gamma_e \mid e \in G \setminus T\}$ is a cycle basis called *fundamental cycle basis*

Some optimization problems on graphs

MINIMUM SPANNING TREE: given a weighted graph $G = (V, E, c)$, find the spanning tree of minimum cost in G

SHORTEST PATH: given a weighted graph $G = (V, E, c)$ and a vertex $v \in V$, find the paths of minimum cost from v to all other vertices of V

MINIMUM CYCLE BASIS: find a basis of minimum cost of the vector space (over $\mathbb{F}_2$) of all cycles

MINIMUM FUNDAMENTAL CYCLE BASIS: find a fundamental cycle basis of minimum cost

MINIMUM CUT: find a cut of minimum cost

MAXIMUM CUT: find a cut of maximum cost

TRAVELLING SALESMAN PROBLEM: find a Hamiltonian cycle of minimum cost

Minimum Spanning Tree

- Several applications
  - Telecom: building a network backbone connecting given set of points at minimum cost
  - Economical storing scheme for situations where vertices are costly to store
  - Used as a step of many complex algorithms
- Can be solved in polynomial time w.r.t. number of vertices n of the graph
- Constructive greedy algorithms: Prim (best next vertex), Kruskal (best next edge)

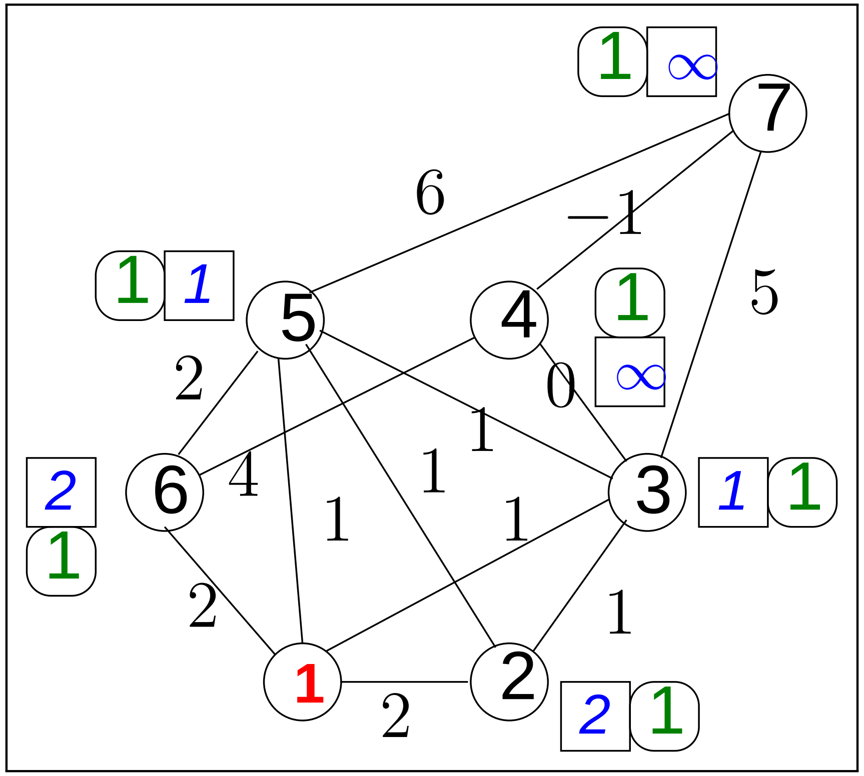
Prim's algorithm

- **Input**: a weighted graph $G = (V, E, c)$
- **Output**: a spanning tree of G of minimum cost
- **Data structures**: $\boxed{T\text{: set of edges of the spanning tree}}$, $\boxed{r \in V\text{: root}}$, $\boxed{S \subseteq V\text{: currently reached vertices}}$, $\boxed{h \in V\text{: current best choice}}$, $\boxed{\zeta : V \smallsetminus S \to \mathbb{R}\text{: cost of edges from } S \text{ (assoc. to adj. vertices)}}$, $\boxed{\pi : V \smallsetminus S \to S\text{: vertex predecessor in } S}$
- **Algorithm**:

```
1. Initialize $T = \emptyset$, $S = \{r\}$, $\zeta(j) = c(\{r, j\})$ if $\{r, j\} \in E$ and $+\infty$
   otherwise and $\pi(j) = r$ for $j \in V \smallsetminus S$
2. While $|T| \leq n - 1$:
   (a) Choose $h \in V \smallsetminus S$ such that $\zeta(h) = \min\{\zeta(j) \mid j \in V \smallsetminus S\}$
   (b) Update tree $T \leftarrow T \cup \{\pi(h), h\}$, $S \leftarrow S \cup \{h\}$
   (c) Update data structures: $\forall\, j \in \delta(h) \smallsetminus S : \zeta(j) > c(\{h, j\})$
       let $\zeta(j) = c(\{h, j\})$ and $\pi(j) = h$
```

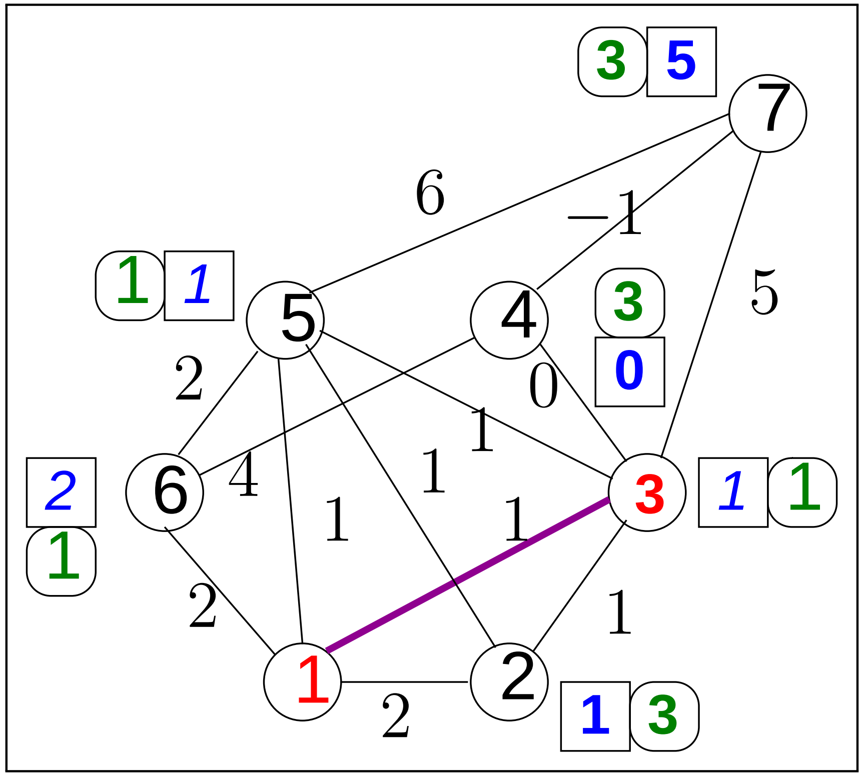
Prim's algorithm: Example I

1. Initialization $r = 1$



$S = \{1\}$ π, ζ

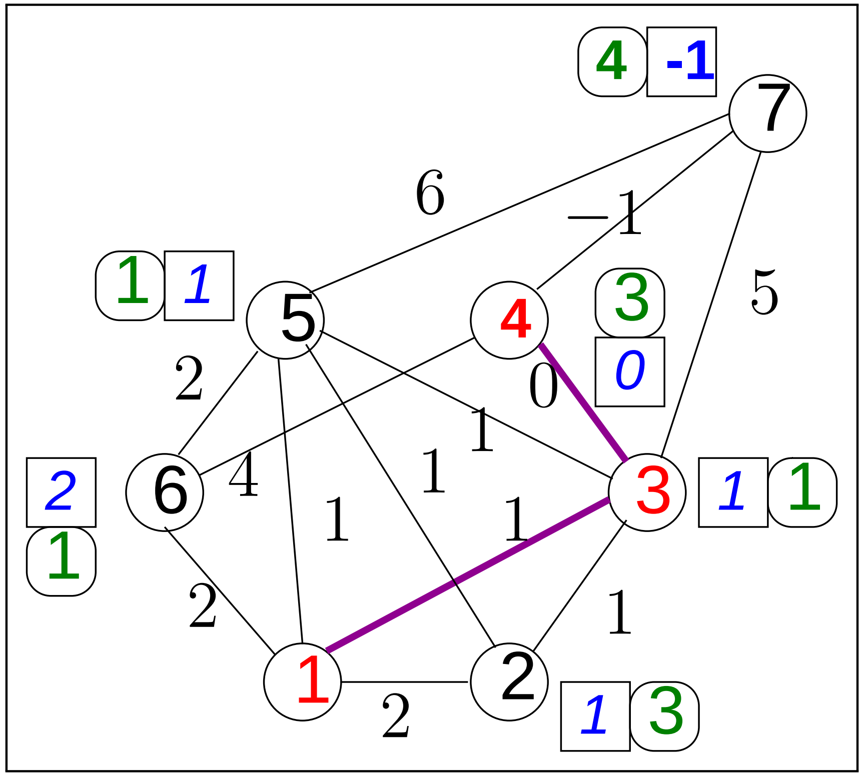
2. Choose $h = 3$



$S = \{1, 3\}$

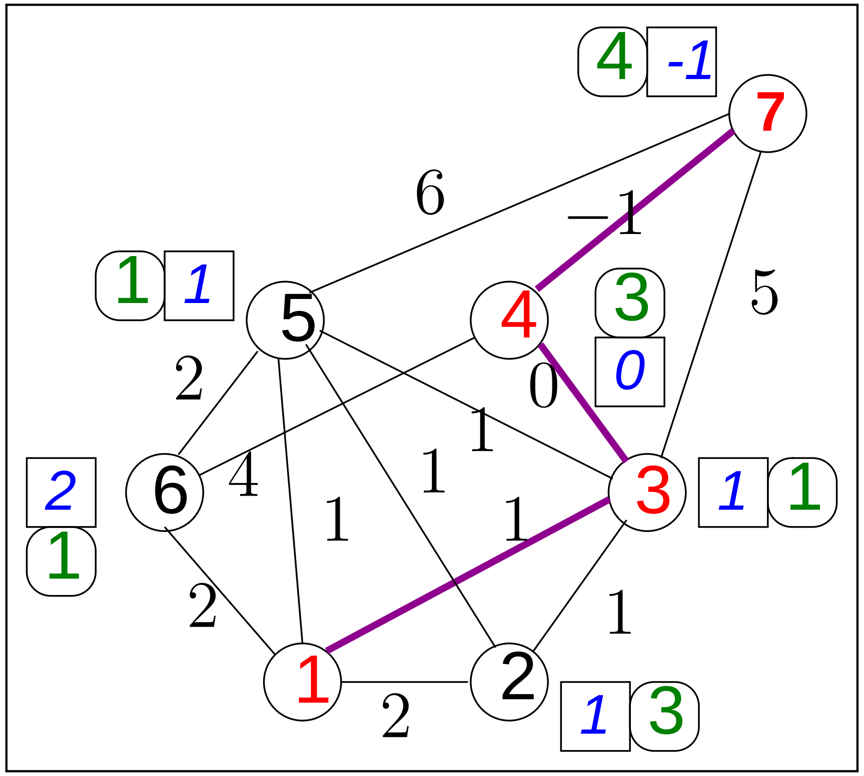
Prim's algorithm: Example II

3. Choose $h = 4$



$S = \{1, 3, 4\}$

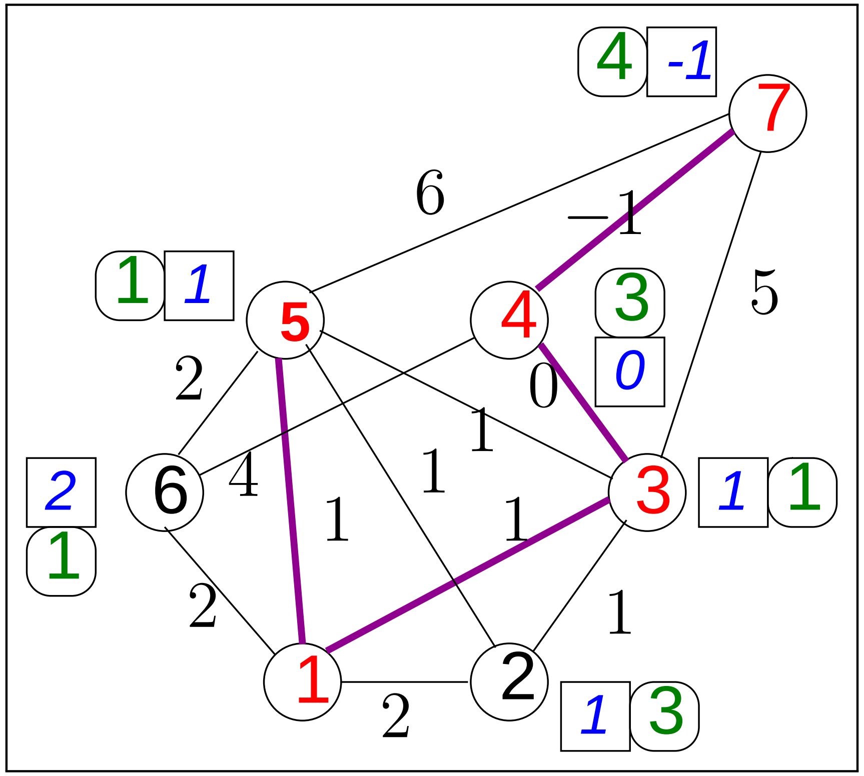
4. Choose $h = 7$



$S = \{1, 3, 4, 7\}$

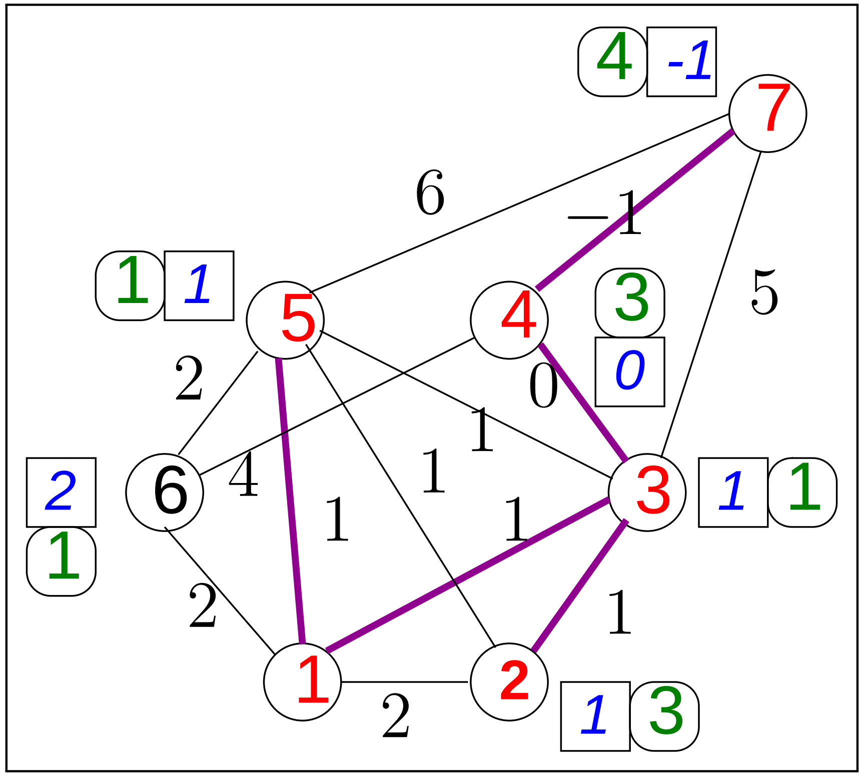
Prim’s algorithm: Example III

5. Choose $h = 5$



$S = \{1, 3, 4, 5, 7\}$

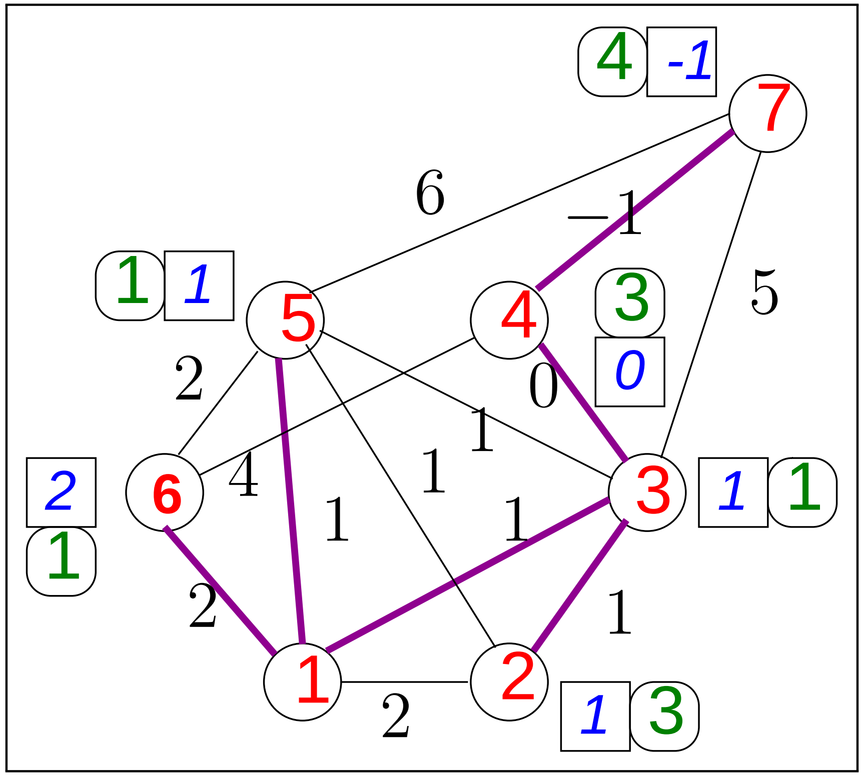
6. Choose $h = 2$



$S = \{1, 2, 3, 4, 5, 7\}$

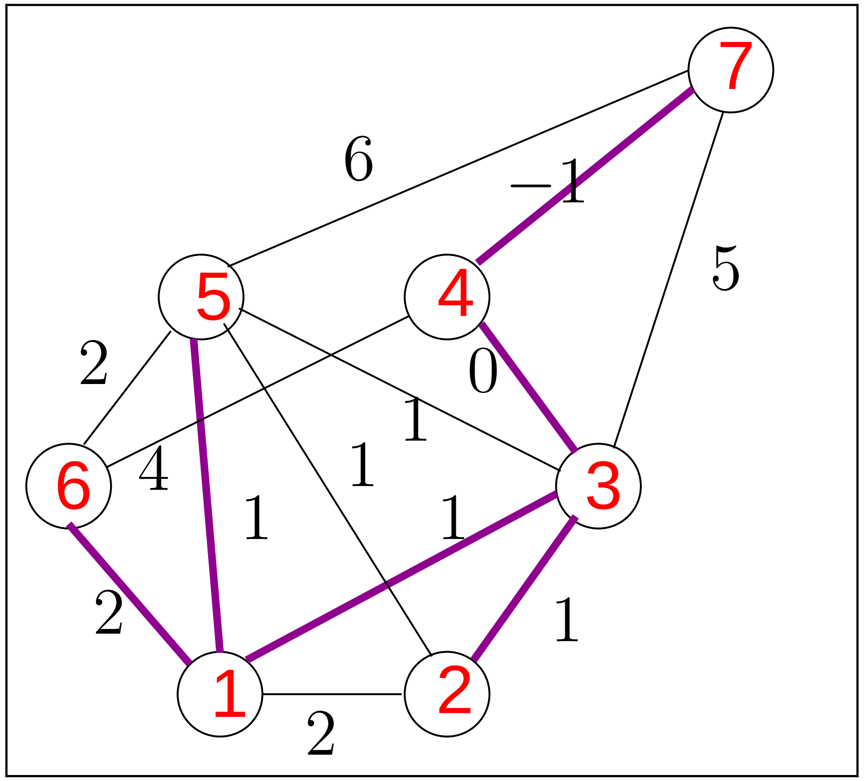
Prim's algorithm: Example IV

7. Choose $h = 6$



$S = \{1, \ldots, 7\}$

End. Opt. tree T, $c(T) = 4$



$S = V$

Prim's algorithm: Complexity

Let $n = |V|$

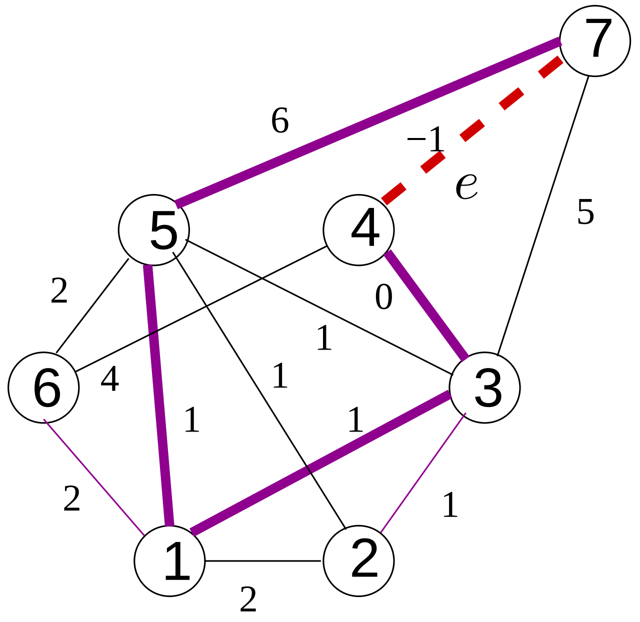
1. Initialization: $O(n)$
2. Loop: $O(n)$
   (a) Choice: $O(n)$
   (b) Tree update: $O(1)$
   (c) Data structures update: $O(n)$

$$\begin{aligned}
&\Rightarrow && O(n + n(n + 1 + n)) \\
&= && O(n + n^2 + n + n^2) \\
&= && O(2(n + n^2)) \\
&= && O(n + n^2) = O(n^2)
\end{aligned}$$

Prim's algorithm: Correctness I

Defn.

Let T be the set of edges of a spanning tree in $G = (V, E, c)$. Let $e \in E \smallsetminus T$ be a chord and $C(e)$ the set of edges of the cycle in G consisting of e and the unique path in T between the vertices adjacent to e. e is a *decreasing edge* w.r.t. T if for all $f \in C(e)$, $c(e) < c(f)$.



Thm.

Let T be the set of edges of a minimum spanning tree in $G = (V, E, c)$. Then there are no decreasing edges in $E \smallsetminus T$.

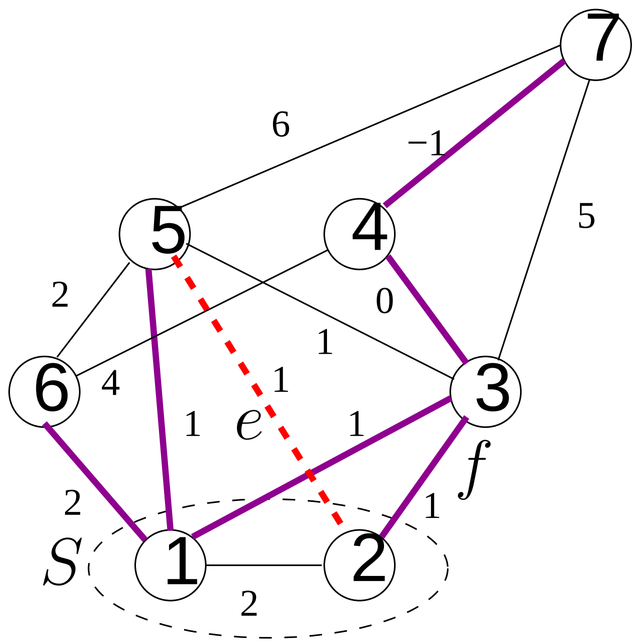
Proof

Suppose e is a decreasing edge and $f \in C(e)$: $c(e) < c(f)$. Let π be the permutation (e, f). Then πT is a sp. tree (why?) and has a lower cost than T, contradicting optimality.

Prim's algorithm: Correctness II

Thm.

Consider $G = (V, E, c)$, let $S \subseteq V$ and e an edge in $\bar{\delta}(S)$ of minimum cost $c(e)$. Then there is a minimum spanning tree containing e.



Proof

Consider a minimum spanning tree with set of edges T, and suppose $e \not\in T$. Let $f \in \bar{\delta}(S) \cap C(e)$ and $f \neq e$. Since T is optimal, e cannot be a decreasing edge $\Rightarrow c(f) \leq c(e)$. Since e is of minimum cost in $\bar{\delta}(S)$, $c(f) \geq c(e)$. Hence, $c(f) = c(e)$. Apply perm. $\pi = (e, f)$ to T: πT is a minimum cost sp. tree.

Shortest paths

- Several applications
  - Network routing
  - Transportation and logistics
  - Used as a step of many complex algorithms
- Several variants
  - Point-to-point shortest path
  - Shortest path from one source vertex to all others
  - Shortest paths between all pairs of vertices
  - Non-negative edge (arc) weights
  - Negative edge (arc) weights
  - Acyclic graph
  - Shortest paths with constraints on the number of edges (arcs)

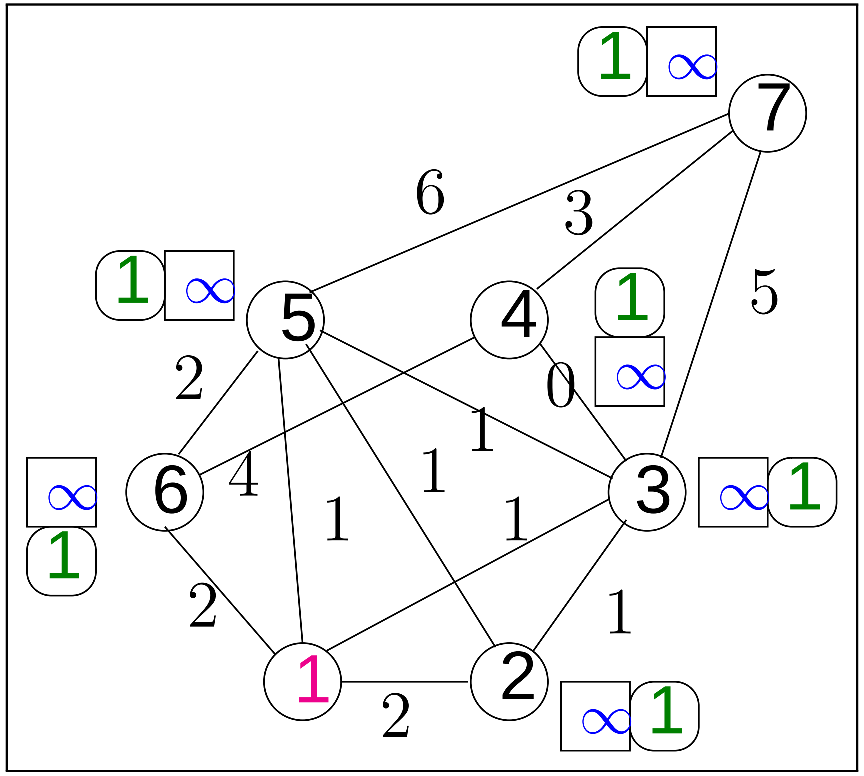
Dijkstra's algorithm

- **Input**: a non-negatively weighted (di)graph $G = (V, E, c)$ and a source vertex $s \in V$
- **Output**: all shortest paths from s to $t \in V \setminus \{s\}$
- **Data structures**: $\boxed{S \subseteq V\text{: *reached* vertices}}$, $\boxed{h \in V\text{: *settled* vertex}}$, $\boxed{\zeta : V \to \mathbb{R}\text{: cost of shortest path from source}}$, $\boxed{\pi : V \setminus S \to S\text{: vertex predecessor in shortest path from source}}$
- **Algorithm**:

```
1. Initialize $S = \{s\}$, $\zeta(s) = 0$; $\zeta(j) = +\infty$, $\pi(j) = s$ for $j \in V \setminus S$
2. While $S \neq \emptyset$:
    (a) Settle vertex $h \in S$ such that $\zeta(h) = \min\{\zeta(j) \mid j \in S\}$
    (b) Update $S \leftarrow S \setminus \{h\}$
    (c) Update data structures: $\forall\, j \in \delta(h) : \zeta(j) > \zeta(h) + c(\{h, j\})$
        let $\zeta(j) = \zeta(h) + c(\{h, j\})$, $\pi(j) = h$, $S \leftarrow S \cup \{h\}$
```

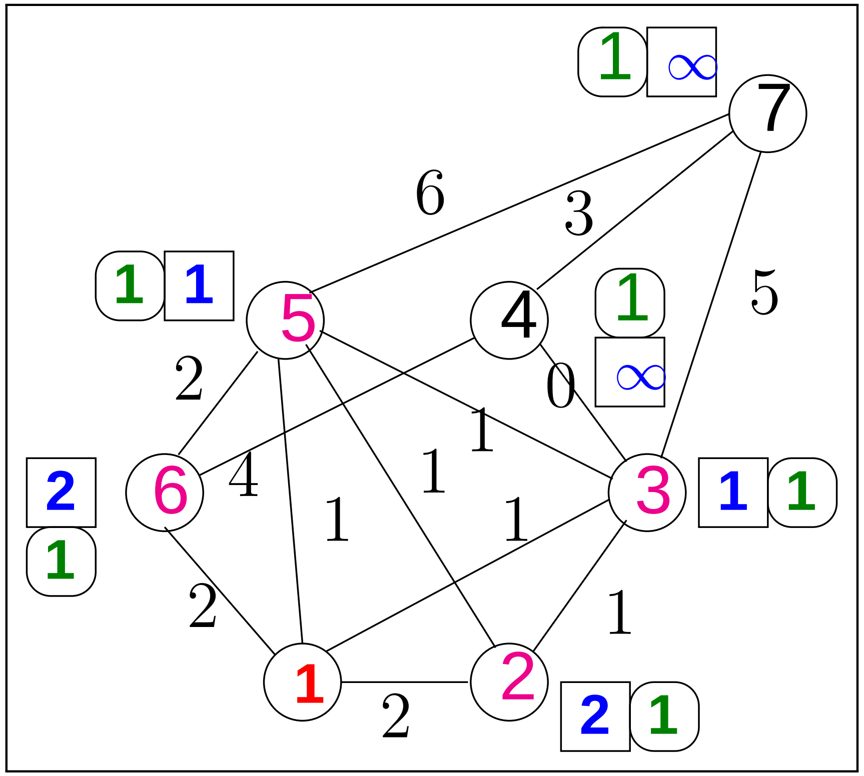
Dijkstra's algorithm: Example I

1. Initialisation



$S = (1)$ π, ζ

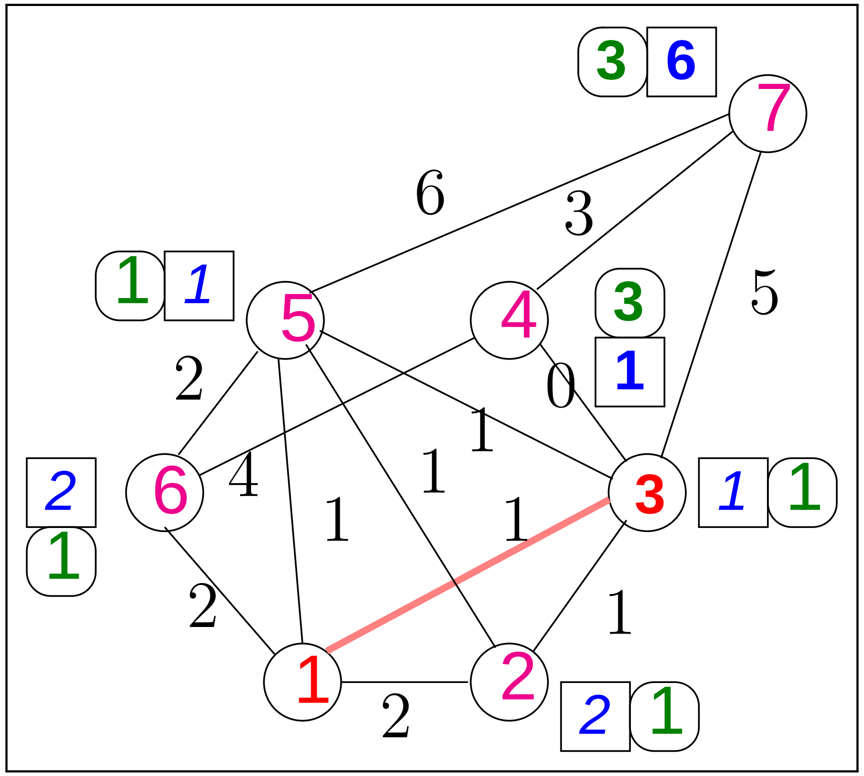
2. Settle $h = s = 1$, $\zeta(1) = 0$



$S = (3, 5, 2, 6)$

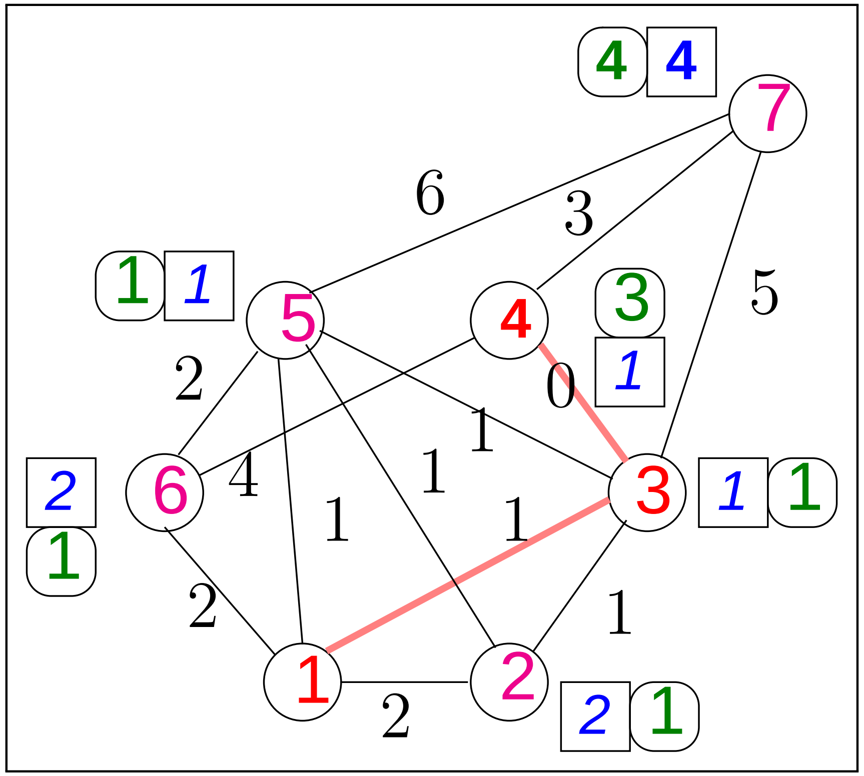
Dijkstra's algorithm: Example II

3. Choose $h = 3$, $\zeta(3) = 1$



$S = (4, 5, 2, 6, 7)$

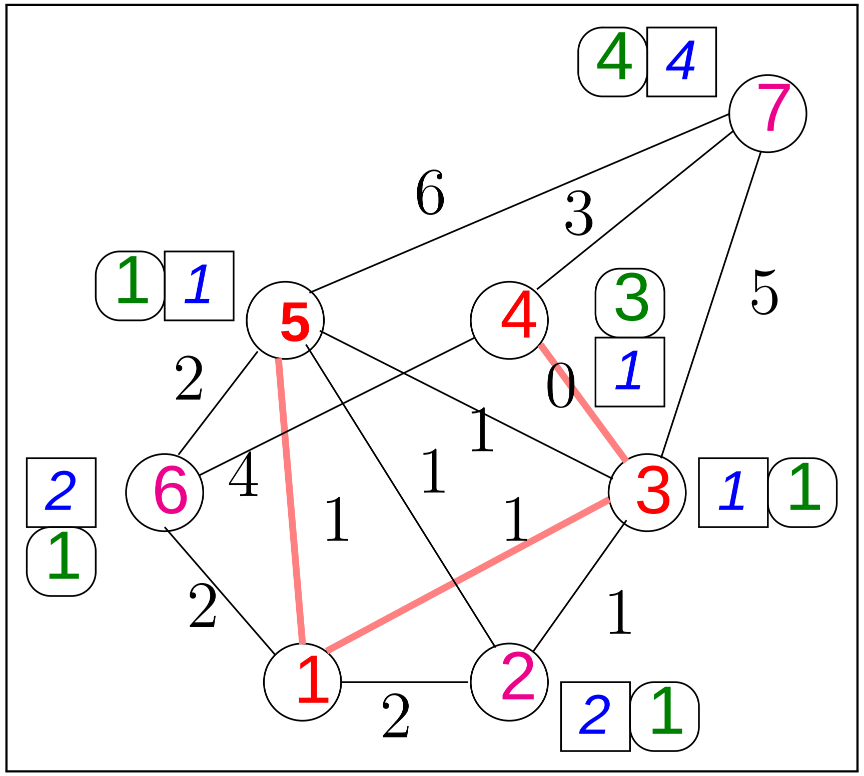
3. Choose $h = 4$, $\zeta(4) = 1$



$S = (5, 2, 6, 7)$

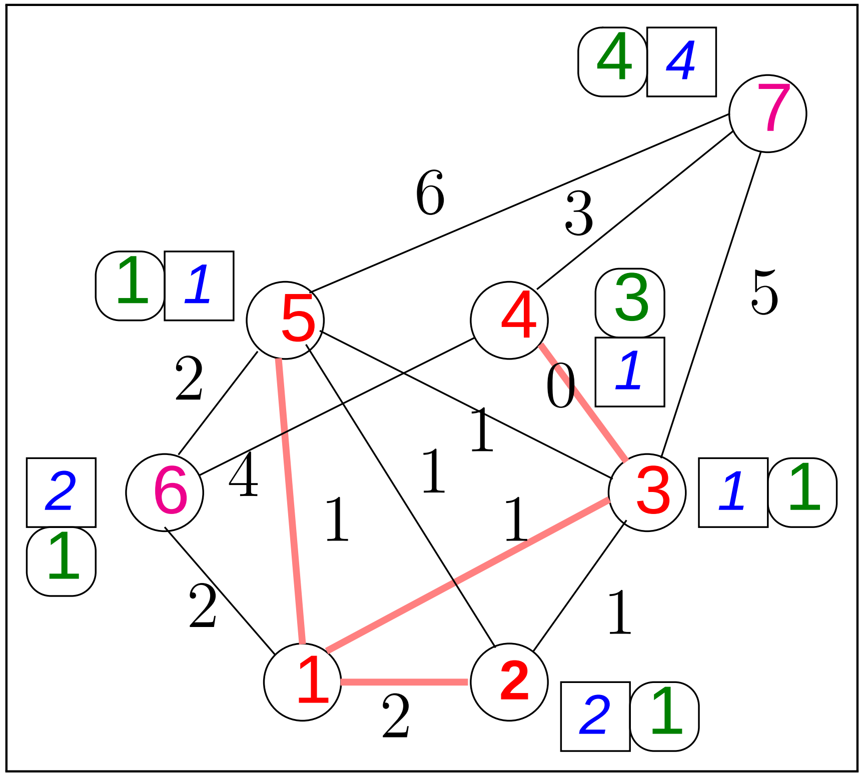
Dijkstra’s algorithm: Example III

3. Choose $h = 5$, $\zeta(5) = 1$



$S = (2, 6, 7)$

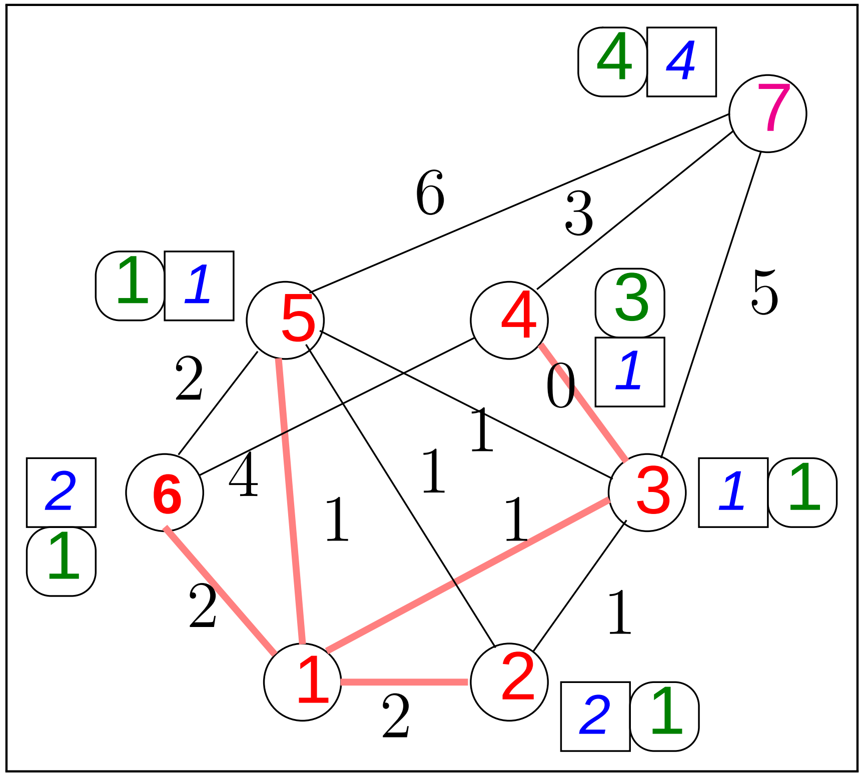
3. Choose $h = 2$, $\zeta(2) = 2$



$S = (6, 7)$

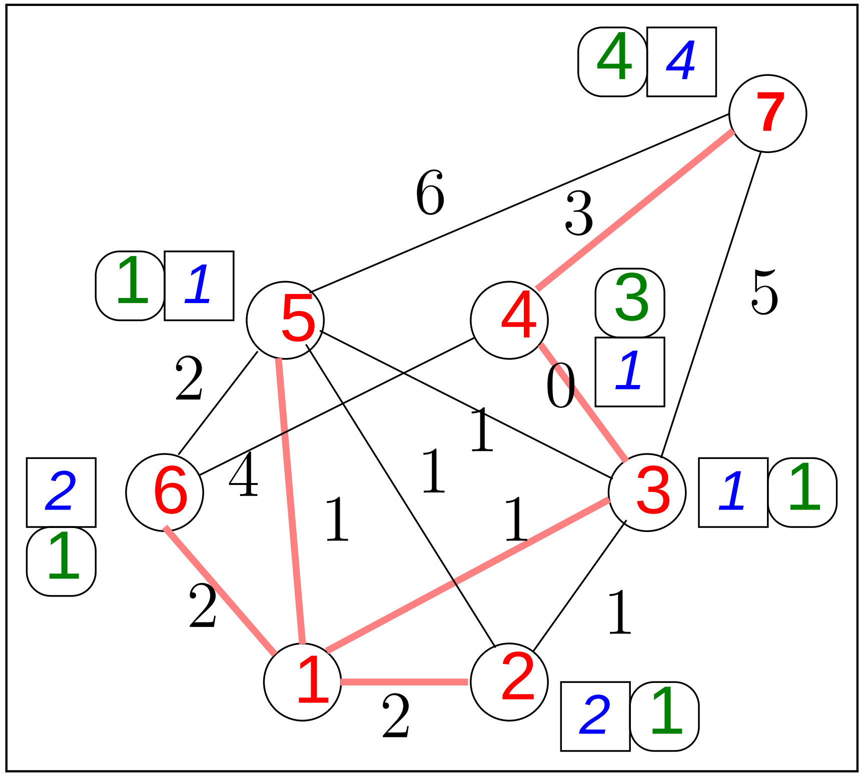
Dijkstra's algorithm: Example IV

3. Choose $h = 6$, $\zeta(6) = 2$



$S = (7)$

3. Choose $h = 7$, $\zeta(7) = 4$



$S = \emptyset$

Dijkstra’s algorithm: Complexity I

Thm.

Each vertex enters S at most once

Proof

Suppose false, then $\exists z \in S$ s.t. $\zeta(z) = \min\{\zeta(j) \mid j \in S\}$ at iteration k and such that at iteration $l > k$ we have $z \in S$ again. Then $\exists w \in S : z \in \delta(w)$ and $\zeta(w) + c(\{w, z\}) < \zeta(z)$. Since $c : V \to \mathbb{R}$ is non-negative, $\zeta(w) < \zeta(z)$ contradicting the minimality of $\zeta(z)$.

Dijkstra's algorithm: Complexity II

Let $n = |V|$

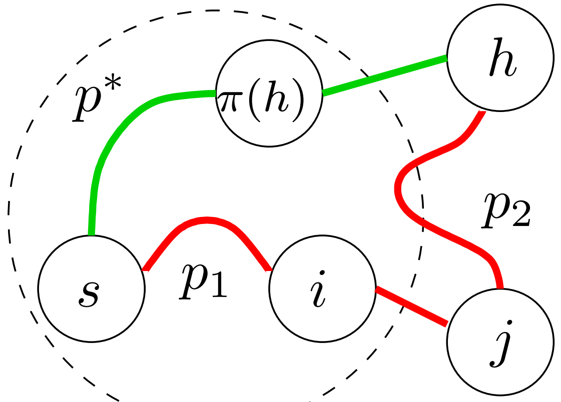
1. Initialization: $O(n)$
2. Loop: $O(n)$ by previous theorem
   (a) Choice: $O(n)$
   (b) Update S: $O(1)$
   (c) Data structures update: $O(n)$

$$\begin{aligned}
\Rightarrow \quad & O(n + n(n + 1 + n)) \\
= \quad & O(n + n^2 + n + n^2) \\
= \quad & O(2(n + n^2)) \\
= \quad & O(n + n^2) = O(n^2)
\end{aligned}$$

Dijsktra's algorithm: Correctness

Thm.

Dijkstra's algorithm produces a shortest path tree T rooted on s



S at itn. k

Proof

We show by induction at iteration k that: $\forall\ j \in V \smallsetminus S$, $\zeta(j) =$ cost of a s.p. from s to j *with all intermediate vertices in* S (trivial for $k=1$). Assume true for $k-1$, let h be the settled vertex at iteration k and let p^* be the s.p. to h given by the algorithm. Consider a path p from s to h with cost $c(p)$. Since $s \in S$ and $h \notin S$, $\exists \{i,j\} \in \bar{\delta}(S)$ such that $p = p_1 \cup \{i,j\} \cup p_2$ where $p_1 \subseteq S$. Then $c(p) = c(p_1) + c(\{i,j\}) + c(p_2) \geq c(p_1) + c(\{i,j\}) = \zeta(i) + c(\{i,j\}) \geq \zeta(\pi(h)) + c(\{\pi(h), h\}) = c(p^*)$.

Exercises

Exercise

Perform Prim's algorithm on the same graph but root it at vertex 5. Do you get a different output? If so, why?

Exercise

Give a counterexample showing that Dijkstra's algorithm may fail if $\exists\{i,j\} \in E$ such that $c(\{i,j\}) < 0$.