Portland State University

W'21 CS 584/684

Algorithm Design & Analysis

Fang Song

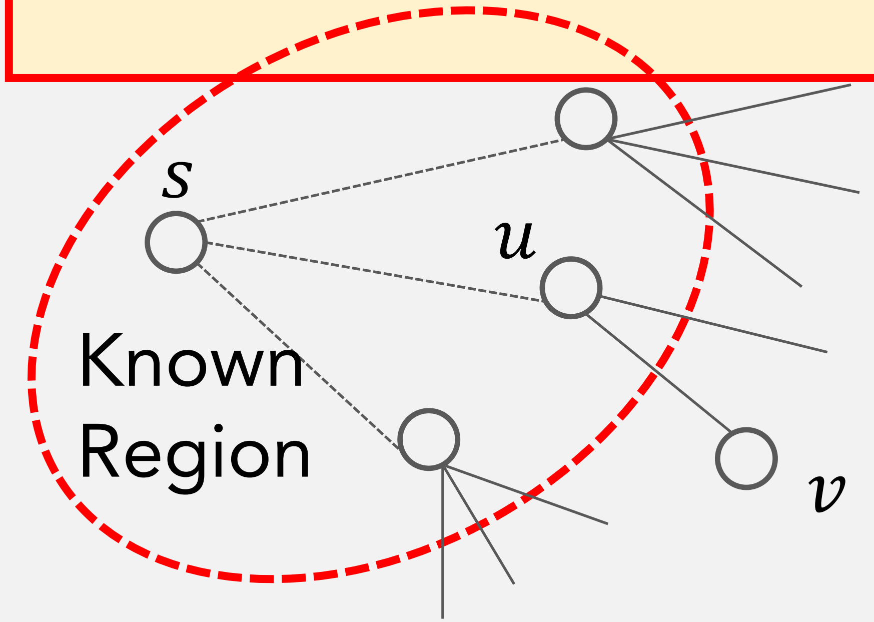
Lecture 11

- Dijkstra's algorithm cont'd
- Interval scheduling
- Minimum spanning tree

Reflection on Dijkstra: greedy stays ahead

- Known region R: in which the shortest distance to s is known
- Growing R: adding v that has the shortest distance to s
- How to Identify v? The one that minimizes $d(u) + l(u, v)$ for $u \in R$

Shortest path to some u in known region, followed by a single edge (u, v)



Dijk(G, s) // initialize $d(s) = 0$, $d(u) = \infty$, R=∅
While R $\neq V$
    Pick $v \notin R$ w. smallest $d(v)$ // by Priority Q
    Add v to R
    For all edges $(v, w) \in E$
    If $d(v) > d(u) + l(u, v)$
        $d(v) \leftarrow d(u) + l(u, v)$

Contrast with Bellman-Ford

- Dijkstra (Greedy) $O((m+n)\log n)$

$$d(v) = \min_{u \in R} d(u) + l(u,v)$$

  - Positive weight: no need to wait; additional edges in a path do not help.

- Bellman-Ford (Dynamic programming) $O(mn)$

$$\mathrm{OPT}(i,v) = \min\left\{\mathrm{OPT}(i-1,v),\ \min_{v\to w \in E}\{\mathrm{OPT}(i-1,w) + l_{v\to w}\}\right\}$$

❖Global vs. Local

- Dijkstra's requires global information: known region & which to add
- Bellman-Ford uses only local knowledge of neighbors, suits distributed setting

Network routing: distance-vector protocol

- Communication network
  - Nodes: routers
  - Edges: direct communication links
  - Cost of edge: delay on link.

naturally nonnegative, but Bellman-Ford used anyway!

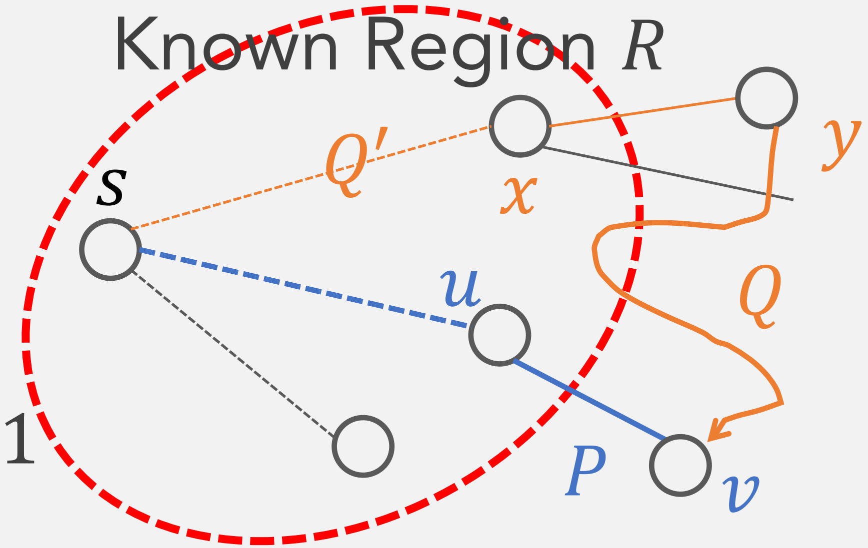
- Distance-vector protocol ["routing by rumor"]
  - Each router maintains a vector of shortest-path lengths to every other node (distances) and the first hop on each path (directions).
  - Algorithm: each router performs separate computations for each potential destination node.
- Path-vector protocol: coping with dynamic costs.
  - Border Gateway Protocol (BGP).

Correctness of Dijkstra's algorithm

Invariant. For each node $u \in R$, $d(u)$ is the length of a shortest $s - u$ path

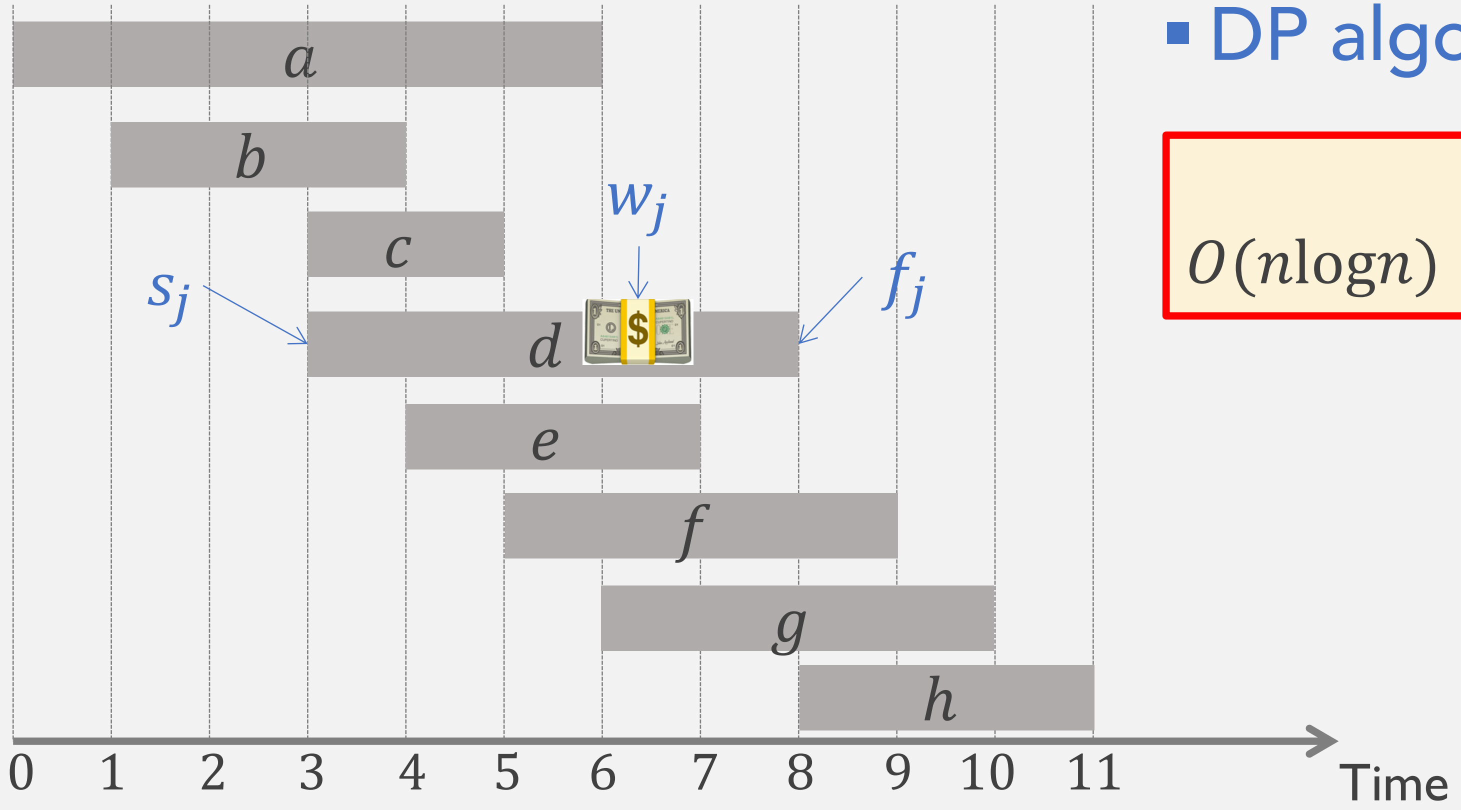
Proof. (By induction on size of R)

- Base case: $|R| = 1$ trivial
- Induction hypothesis: true for $|R| = k \geq 1$
- Show $|R| = k + 1$.
  - Let v be the next node added to R and (u, v) be the chosen edge. Call this $s - u - v$ path P.
  - Consider any $s - v$ path Q. [Next show it's no shorter than P]
  - Let (x, y) be the first edge in Q leaving R; let Q' be the $s - x$ segment.
  - $l(Q) \geq l(Q') + l(x, y) \geq d(x) + l(x, y) \geq l(P)$; because Dijkstra's picked v in this iteration (node outside R with shortest distance to s)



Recall: weighted interval scheduling

- **Input.** n jobs; job j starts at s_j, finishes at f_j, weight w_j
- **Output.** Subset of mutually compatible jobs of maximum weight



- DP algorithm $O(n \log n)$

Today

$O(n\log n)$ Greedy algorithm for $w_j = 1$.

Greedy strategies

Recall. DP recurrence. OPT(j) = value of optimal solution to jobs $1,2,\ldots,j$

$$\text{OPT}(j)=\begin{cases}0 & \text{if } j=0\\ \max\{\text{OPT}(j-1), w_j+\text{OPT}(pre(j))\} & \text{otherwise}\end{cases}$$

- Greedy: be lazy & pick the next compatible job that "looks nice"
  - Earliest start time: ascending order of s_j.
  - Earliest finish time: ascending order of f_j.
  - Shortest interval: ascending order of $f_j - s_j$.
  - Fewest conflicts: the one that conflicts the least number of jobs runs first.
- Exercise. Find counterexamples for each strategy (if possible)

Greedy: counterexamples

☹ Earliest start time:

☹ Shortest interval:

☹ Fewest conflicts:

☺ Earliest finishing time

Greedy Algorithm: earliest finishing time

IntScheduling $(\{s_j, f_j\})$

1. Sort by finishing time so that $f_1 \le f_2 \le \cdots \le f_n$ $\rightarrow$ $O(n\log n)$
2. $A \leftarrow \emptyset$ // set of selected jobs
3. For $j = 1, \dots, n$
   If j compatible with A
   $A \leftarrow A \cup \{j\}$

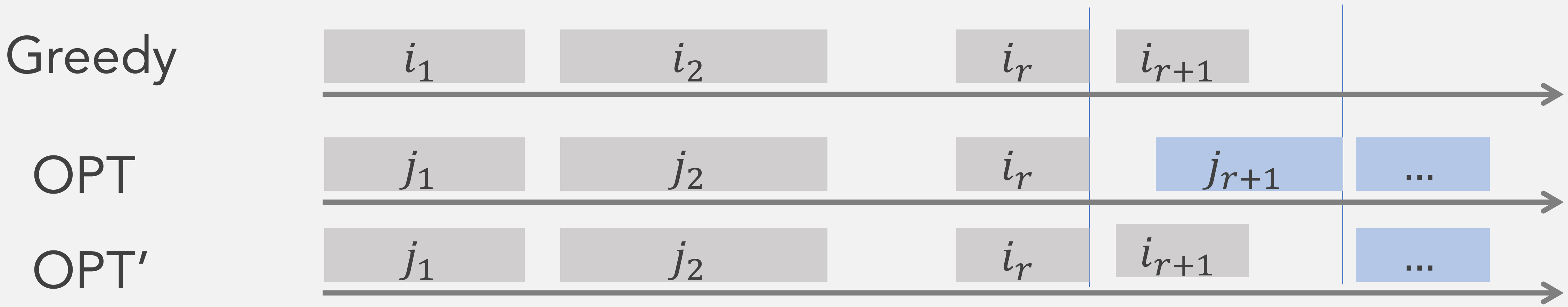
$n \times O(1)$

- Running time: $O(n \log n)$
- Correctness: proof by contradiction
  - Suppose greedy is not optimal.
  - Consider an optimal strategy: one that agrees with Greedy for as many initial jobs as possible.
  - Look at the first place that they differ: show a new optimal that agrees with greedy more.

Greedy Algorithm: correctness

Proof (by contradiction): Suppose greedy is not optimal

- Let $i_1, i_2, \dots, i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \dots, j_m$ be set of jobs in the optimal solution OPT where $i_1 = j_1, i_2 = j_2, \dots, i_r = j_r$ for the largest possible value of r.
- Sub i_{r+1} for j_{r+1} in OPT: still feasible and optimal (OPT'); but agrees with Greedy at $r + 1$ positions; contradicts the maximality of r.



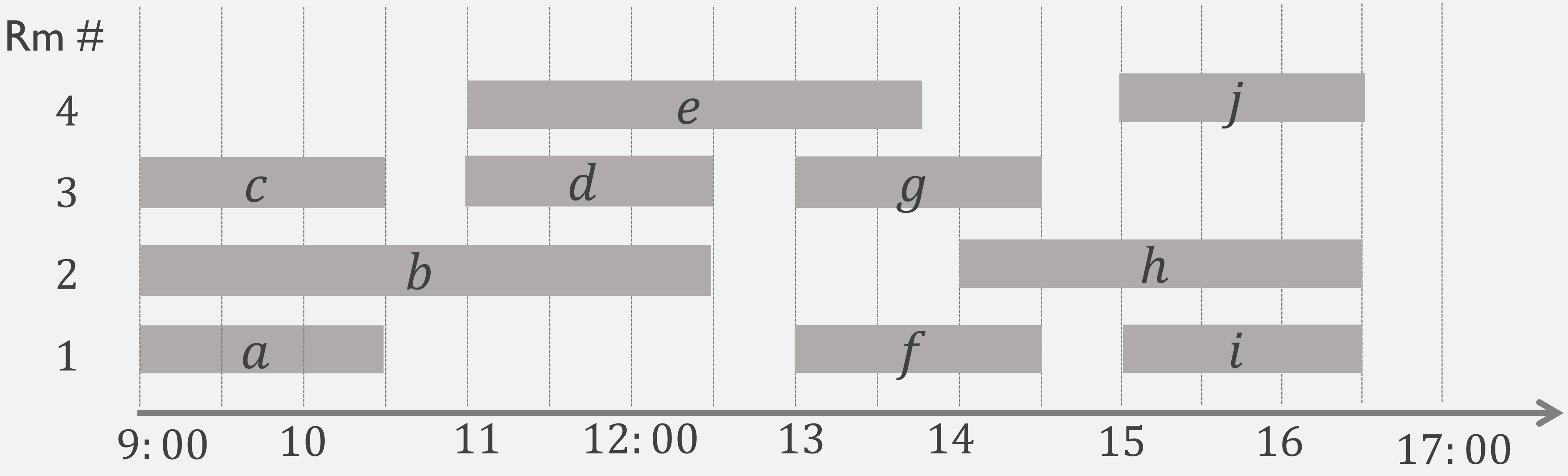
Interval Partitioning Problem

Scheduling classes

- Input. Lectures $\{s_j, f_j\}$
- Output. Minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.

Can you do better?

10 lectures scheduled in 4 classrooms

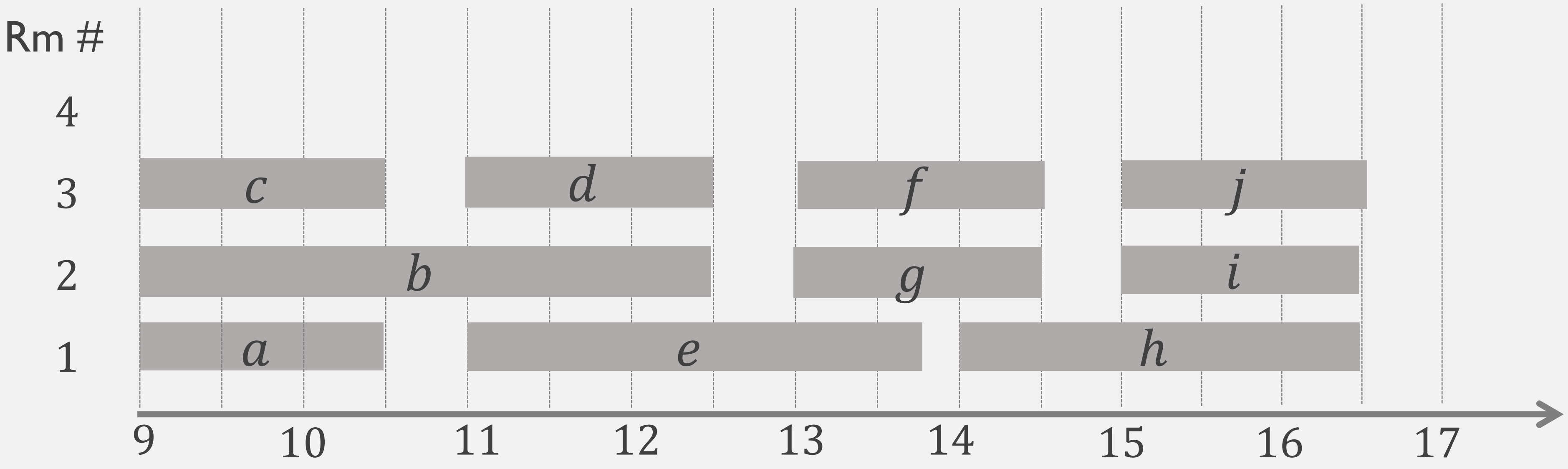


Interval Partitioning Problem

Scheduling classes

- Input. Lectures $\{s_j, f_j\}$
- Output. Minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.

YES! 10 lectures scheduled in 3 classrooms



Greedy algorithm

- Idea. Sort lectures in increasing order of start time: assign lecture to any compatible classroom.

IntPartition($\{s_j, f_j\}$) // $r \leftarrow 0$ # of allocated rooms
1. Sort by starting time so that $s_1 \leq s_2 \leq \cdots \leq s_n$
2. For $j = 1, \ldots, n$
   - If j compatible with some classroom k
     - Schedule j in room k
   - Else allocate new classroom $r + 1$
     - Schedule j in room $r + 1$
     - $r \leftarrow r + 1$

How to do it in $O(\log r)$

OBS. # rm needed $\geq$ depth of input intervals (i.e., Max. number of lectures that overlap)

- Running time. $O(n \log n)$
- Optimality. #Rm allocated = depth of input intervals

Minimum spanning tree (MST)

- Input. A connected undirected graph $G = (V, E)$.
  - Weight function $w: E \to \mathbb{R}$.
  - For now, assume all edge weights are distinct.

A tree that connects all vertices

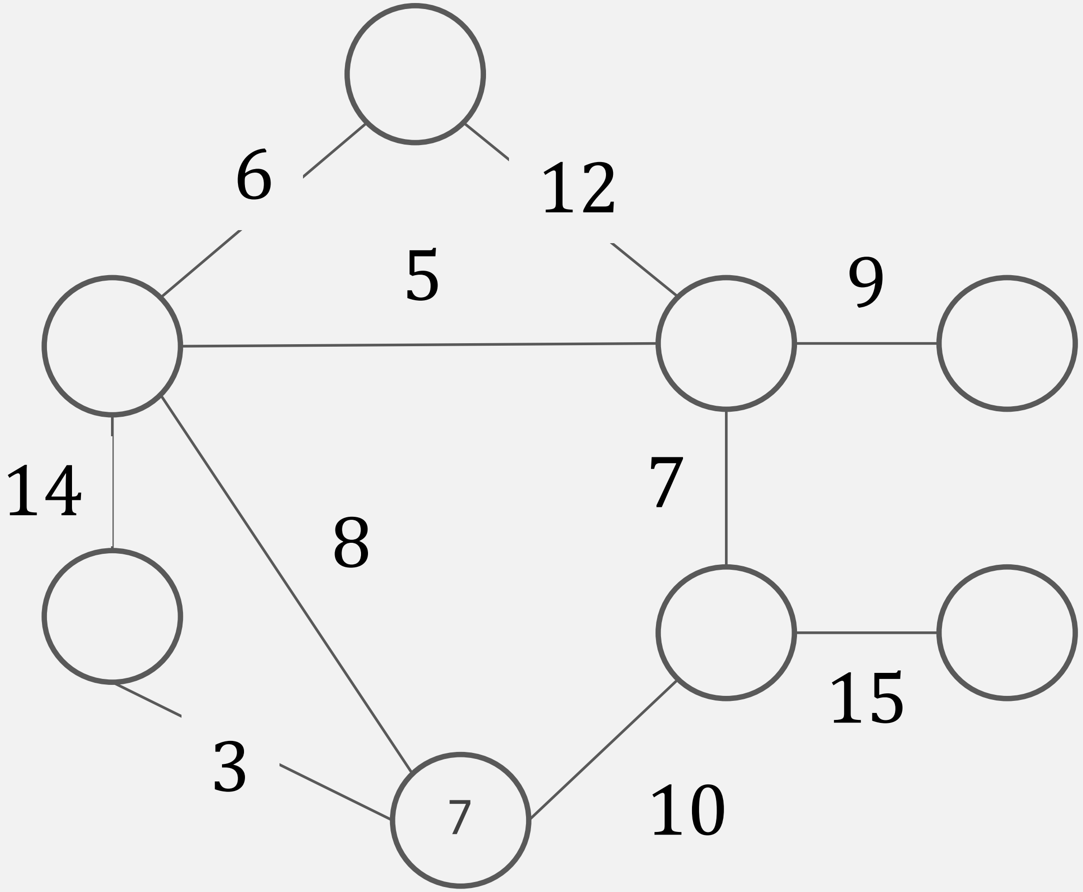
- Output. A spanning tree T of minimum weight.

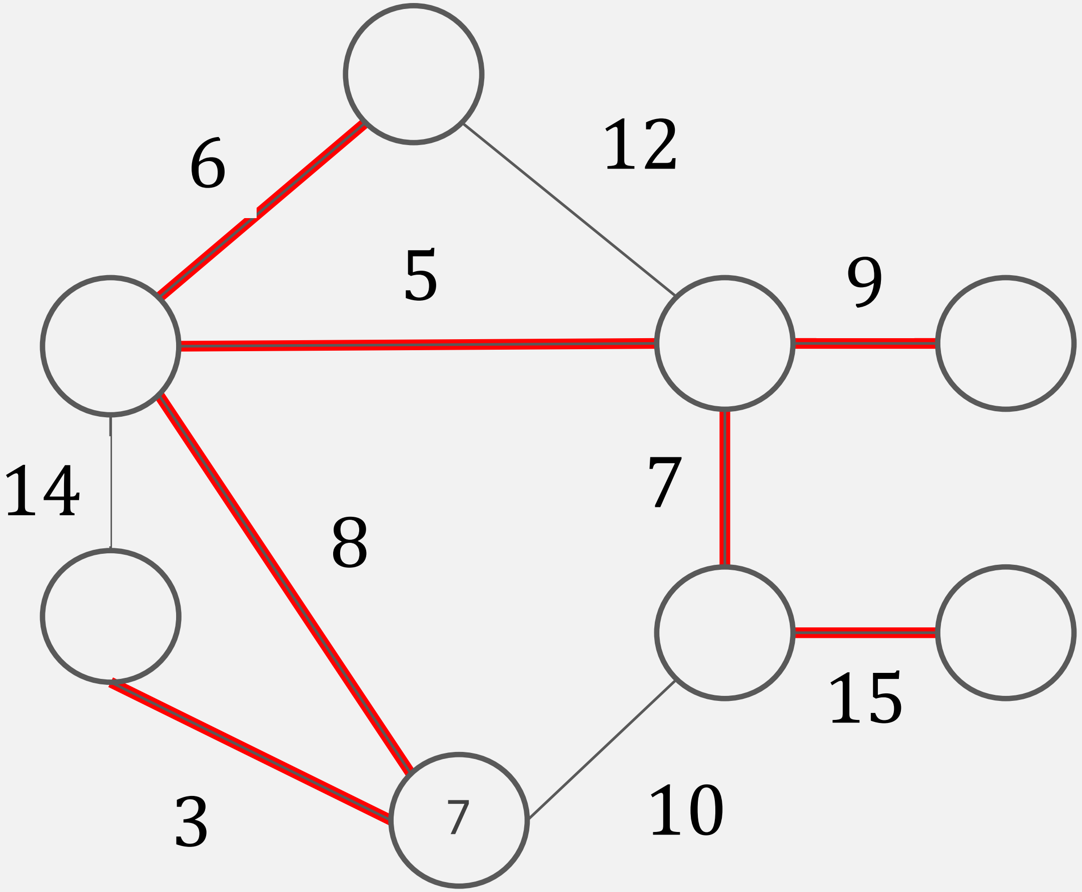
$$w(T) \coloneqq \sum_{(u,v) \in T} w(u, v)$$

Applications

- Cluster, Real-time face verification.
- Network design (communication, electrical, computer, road).
- ….

Example of MST





Pop quiz 1

Which of the following are true for all spanning trees?

A. Contains exactly $|V| - 1$ edges
B. The removal of any edge disconnects it
C. The addition of any edge creates a cycle
D. All of the above

Cayley's theorem.

The complete graph on n nodes has n^{n-2} spanning trees.

[Brute-force forbidden]

Brainstorming

Greedy strategies for computing an MST?

Greedy algorithms for MST

- Kruskal's. Start with $T = \emptyset$. Insert edges in ascending order of weights, unless it creates a cycle.
- Reverse-Delete. Start with $T = E$. Remove edges in descending order of weights, unless it disconnects T.

Edge-driven

- Prim's. Start with some node s. Grow a tree T from s outward. Add v to T such that $w(u, v)$ cheapest and $u \in T$.

Node-driven

Sounds familiar? Dijkstra's?

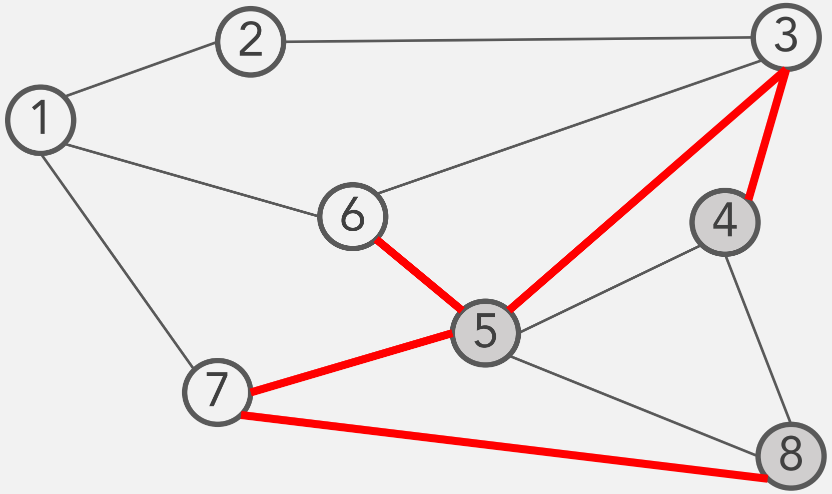
☺ In this extremely lucky case, all of them work! But correctness proofs are non-trivial. We need the following tools to prove them.

Cycles and cuts

- Cycle: set of edges of form $(a, b), (b, c), \ldots, (z, a)$.

- Cut: a subset of nodes $S \subseteq V$.

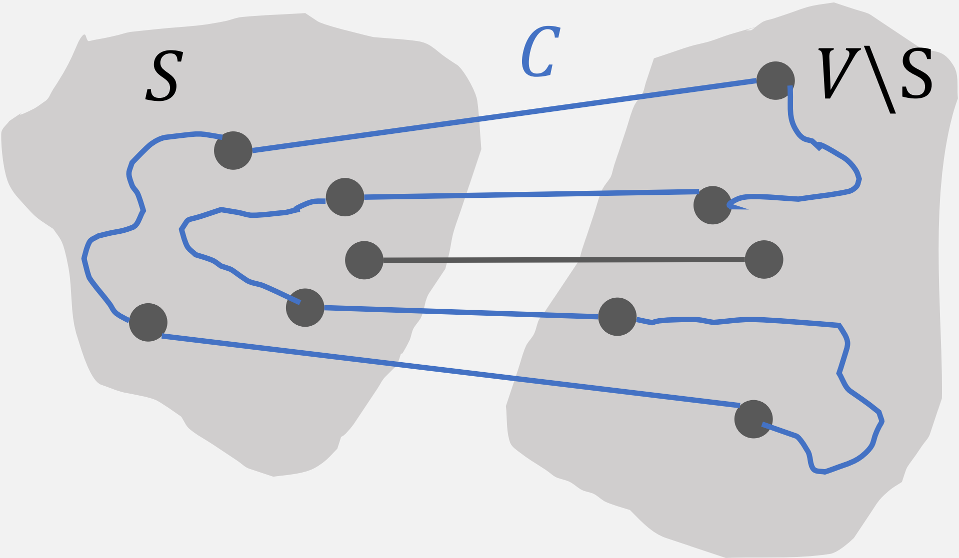
- Cutset $D(S)$: subset of edges with exactly one endpoint in S.



Ex. Cut $S = \{4,5,8\}$
Cutset $D(S) = \{(4,3), (5,7), (5,6), (7,8)\}$

Observation: cycle-cut intersection

Claim*. A *cycle* & a *cutset* intersect in an even number of edges.



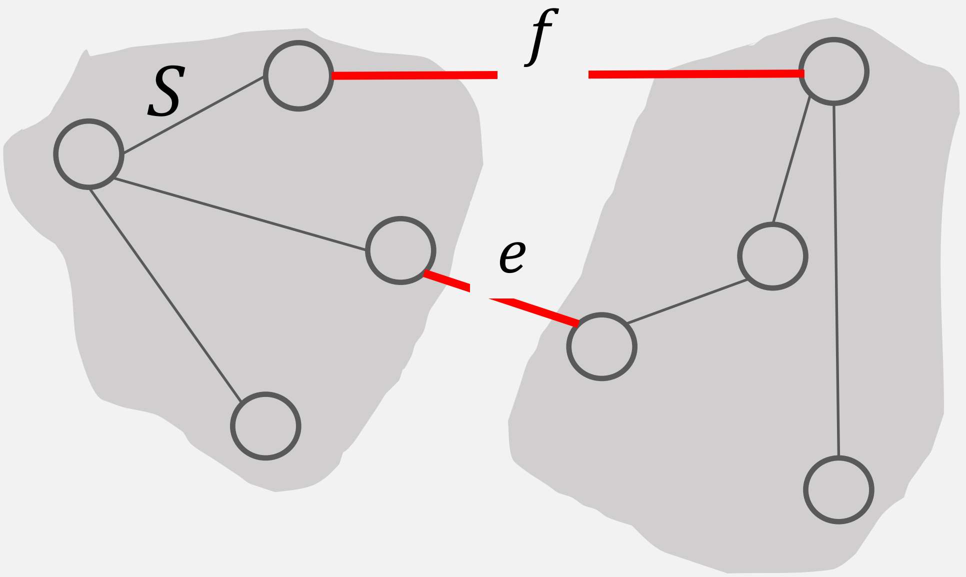
- Proof. A cycle has to leave & enter the cut the same number of times.

Cut Property

Cut property. Let S be a subset of nodes. Let e be the min weight edge with exactly one endpoint in S. Then any MST T contains e.

- Proof. (exchange argument)
  - Suppose e does not belong to T
  - Adding e to T creates a cycle C
  - Edge e is both in C and in the cutset $D(S)$

  ➔ there exists another edge, say f, that is in both C and D. [Claim*]
  - $T' := T \cup \{e\} - \{f\}$ is also a spanning tree
  - $w_e < w_f$ ➔ $w(T') < w(T)$. Contradiction!



Cycle property

Cycle property. Let C be a cycle, and let f be the max weight edge in C. Then any MST T does not contain f.

- Proof. (exchange argument)
  - Suppose f belongs to T
  - Deleting f creates a cut S
  - Edge f is both in C and in the cutset $D(S)$

  ➔ there exists another edge, say e, that is in both C and D.
  - $T' \coloneqq T \cup \{e\} - \{f\}$ is also a spanning tree
  - $w_e < w_f$ ➔ $w(T') < w(T)$. Contradiction!