Program Structure and Algorithms

Sid Nath

Lecture 10

Agenda

- Administrative
  - HW4 will be released this week
- Activity selection
- Interval scheduling
- Union-Find
- Minimum Spanning Trees (MSTs)

Problem 3: Activity Selection

Given: n activities that require *exclusive use of a common resource* (e.g., scheduling use of GSB-125)

$S = \{a_1, a_2, \dots, a_n\}$; a_i needs resource during $[s_i, f_i)$

s_i: *start time*; f_i: *finish time*; $s_i < f_i$

Goal: Select the ***largest*** possible set of non-overlapping (i.e., mutually exclusive) activities

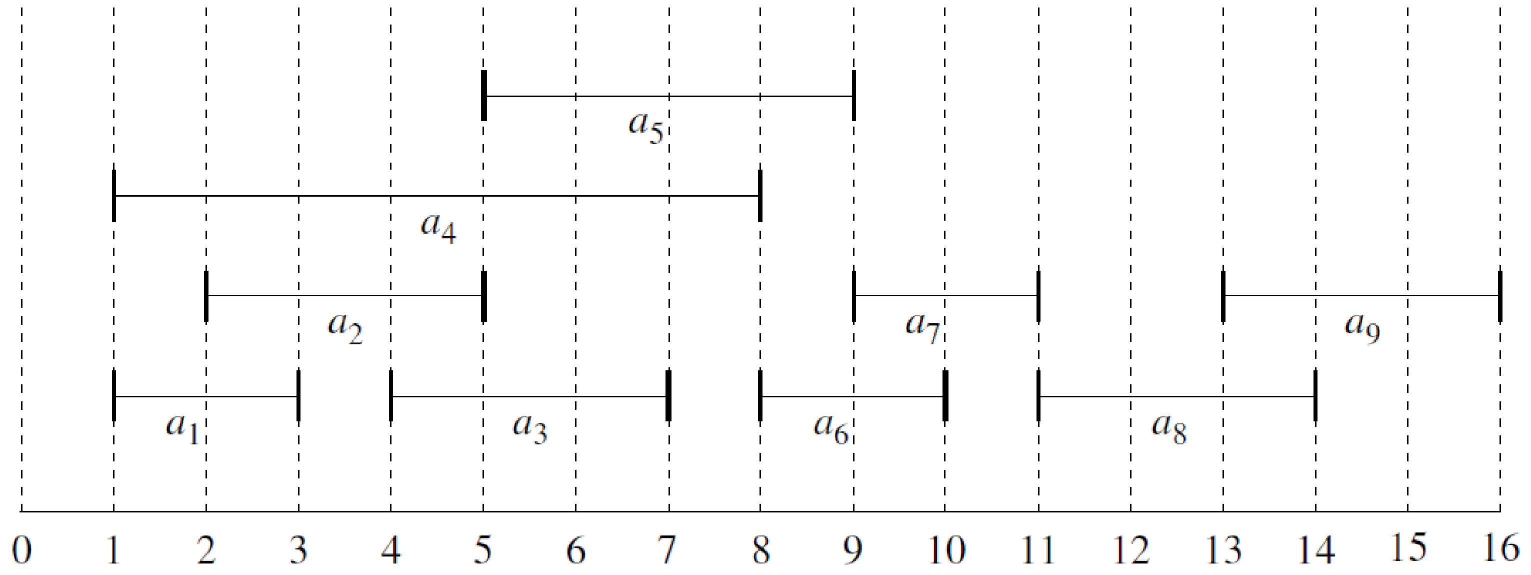
May also be:

Schedule room for the longest time

Maximize rental income fees

Example

i	1	2	3	4	5	6	7	8	9
s_i	1	2	4	1	5	8	9	11	13
f_i	3	5	7	8	9	10	11	14	16



Mutually compatible sets:

$\{a_1, a_3, a_6, a_8\}, \{a_1, a_3, a_6, a_9\}, \{a_2, a_5, a_7, a_8\}$ ...

Max size: 4

Solution Construction

How to develop a greedy choice?

- Which activity leaves the resource available for the most other activities?
- First activity to finish? Let's use this.
    - Sort activities by finish times

$$f_1 \leq f_2 \leq \cdots \leq f_{n-1} \leq f_n$$

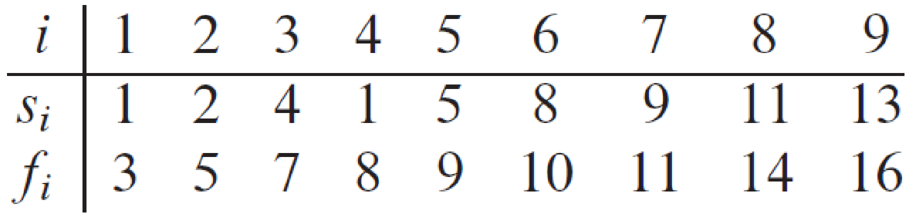
    - We begin by selecting a_1 (earliest finish time)
    - Reduces to only one subproblem: find a max size set of m.c. activities that start after a_1 finishes.

Pseudocode

```
Recursive-Activity-Selector(s, f, k, n)
// k: index of current subproblem
// n: number of activities in S
   m ← k + 1
   while(m≤ n and s[m]<f[k]) // find $a_m$ compatible with $a_k$ ($s_m \geq f_k$)
      m ← m + 1
   if m ≤ n
      return $\{a_m\} \cup$ Recursive-Activity-Selector(s, f, m, n)
   else
      return $\phi$
```

Runtime: $\theta(n \log n)$

Example



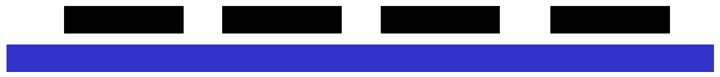
i	1	2	3	4	5	6	7	8	9
s_i	1	2	4	1	5	8	9	11	13
f_i	3	5	7	8	9	10	11	14	16

Also Called

- Interval scheduling
  - Activities ≡ jobs

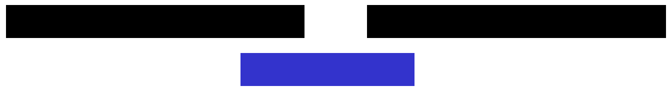
Other Strategies

- Earliest start time: order activities in ascending order of start times s_j



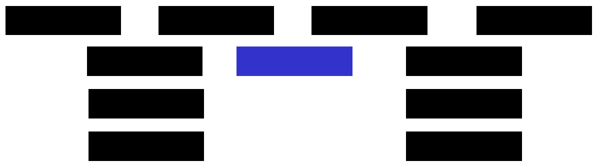
Counterexample for earliest start time

- Shortest interval: order activities in ascending order of $f_j - s_j$



Counterexample for shortest interval

- Fewest conflicts: For each activity, count #conflicting activities c_j, schedule in ascending order of c_j

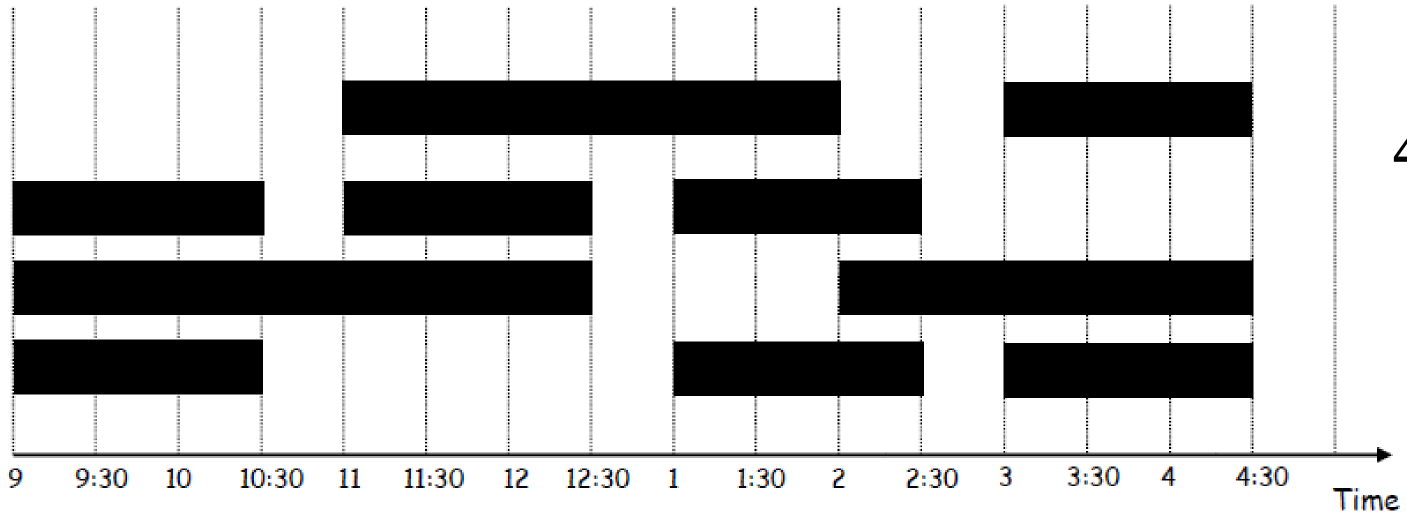


Counterexample for fewest conflicts

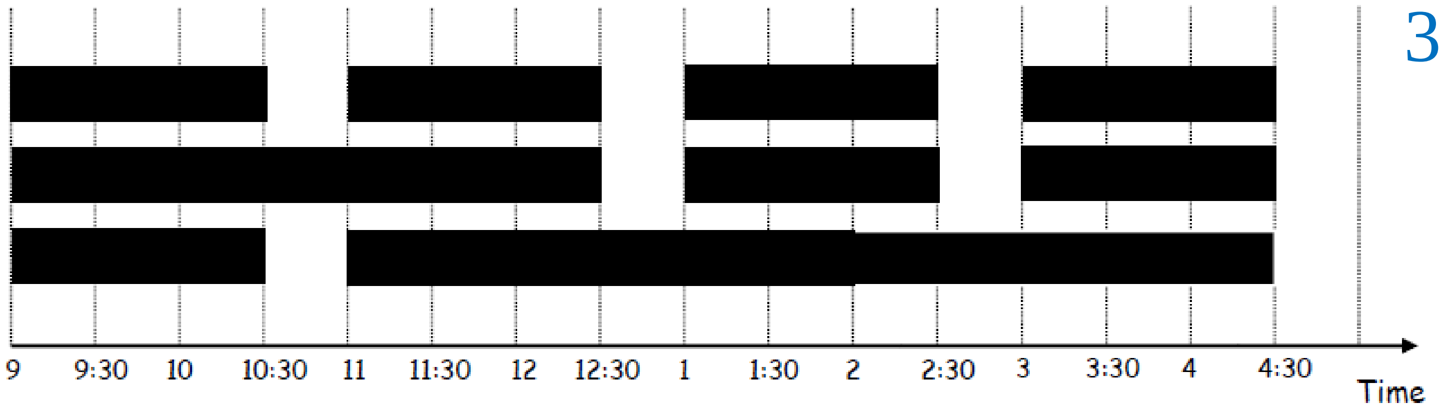
Problem 4: Interval Partitioning

- Lecture j starts at s_j and finishes at f_j
- Goal: Find minimum #classrooms to schedule all lectures so that no two occur at the same time in the same classroom
- This schedule uses 4 classrooms for 10 lectures

Examples of 10 lectures



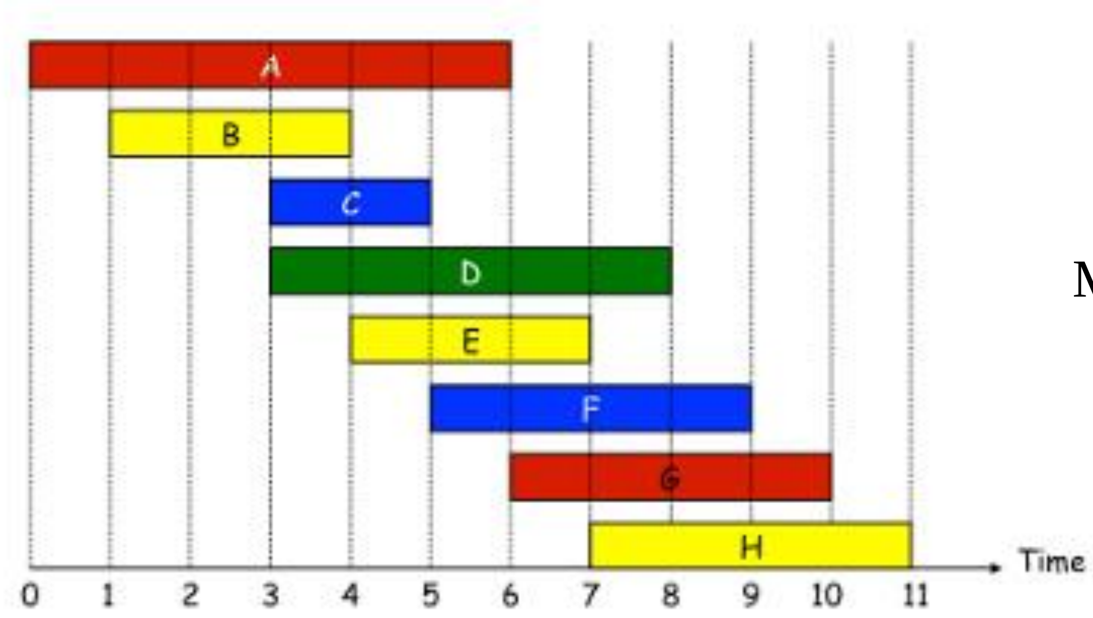
4 classrooms



3 classrooms

Similar Problem: Interval Coloring

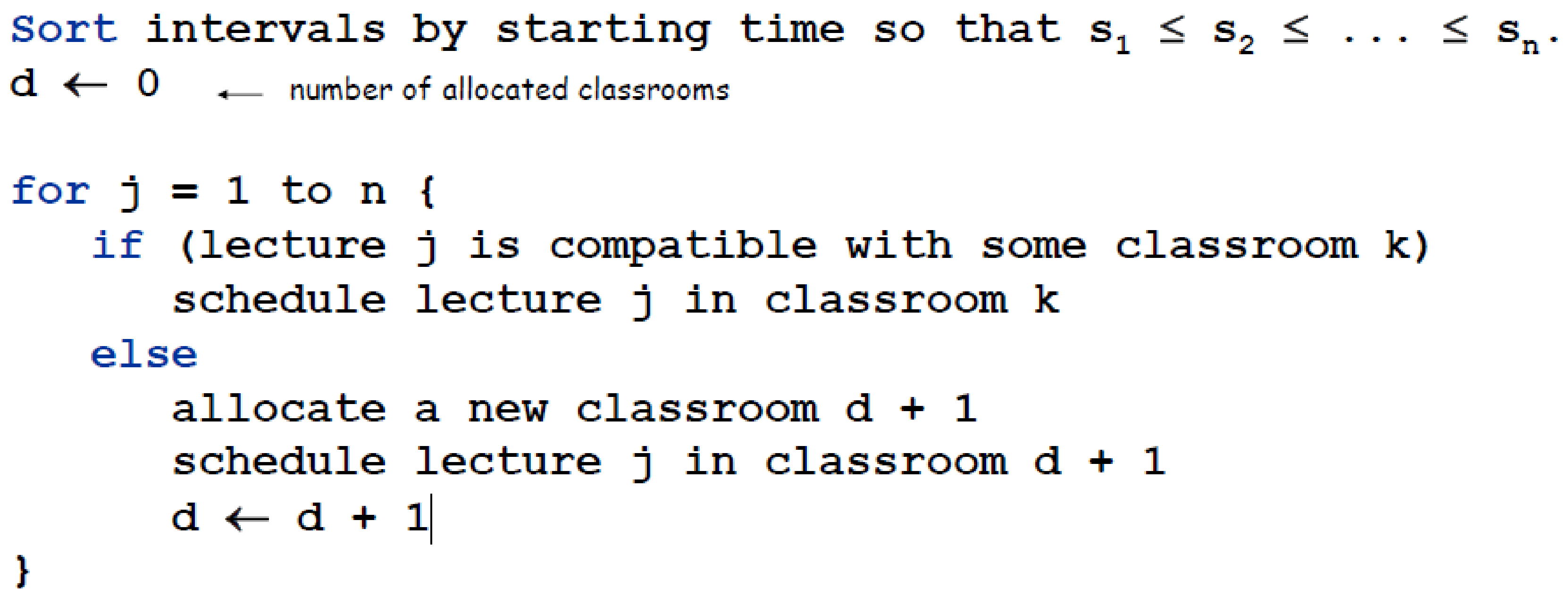
- Given n intervals $(s_1, f_1), (s_2, f_2), \dots (s_n, f_n)$, colors all the intervals with the fewest possible colors such that intervals having the same color do not intersect (aka mutually compatible)
- Lectures have intervals
  - Each color: each unique classroom/resource



Min #colors: 4

Greedy Algorithm for Interval Partitioning

- Greedy choice: Consider lectures in increasing order of start time: assign lecture to any compatible classroom



```
Sort intervals by starting time so that $s_1 \leq s_2 \leq ... \leq s_n$.
d ← 0    ← number of allocated classrooms

for j = 1 to n {
   if (lecture j is compatible with some classroom k)
      schedule lecture j in classroom k
   else
      allocate a new classroom d + 1
      schedule lecture j in classroom d + 1
      d ← d + 1
}
```

Problem 5: Minimum Spanning Trees

Given a connected graph G = (V, E) with real-valued edge weights, an MST is a subset of the edges $T \subseteq E$ such that T is a spanning tree whose sum of edge weights is minimized

- A spanning tree whose cost is minimum over all spanning trees is a MST

Example problem: A town has a set of houses and a set of roads. A road connects 2 and only 2 houses. A road connecting houses A and B has a repair cost w(A, B)

Goal: Repair enough (and no more) roads such that

- everyone stays connected (i.e., can reach every house from all other houses), and total repair cost is minimum

Example #1

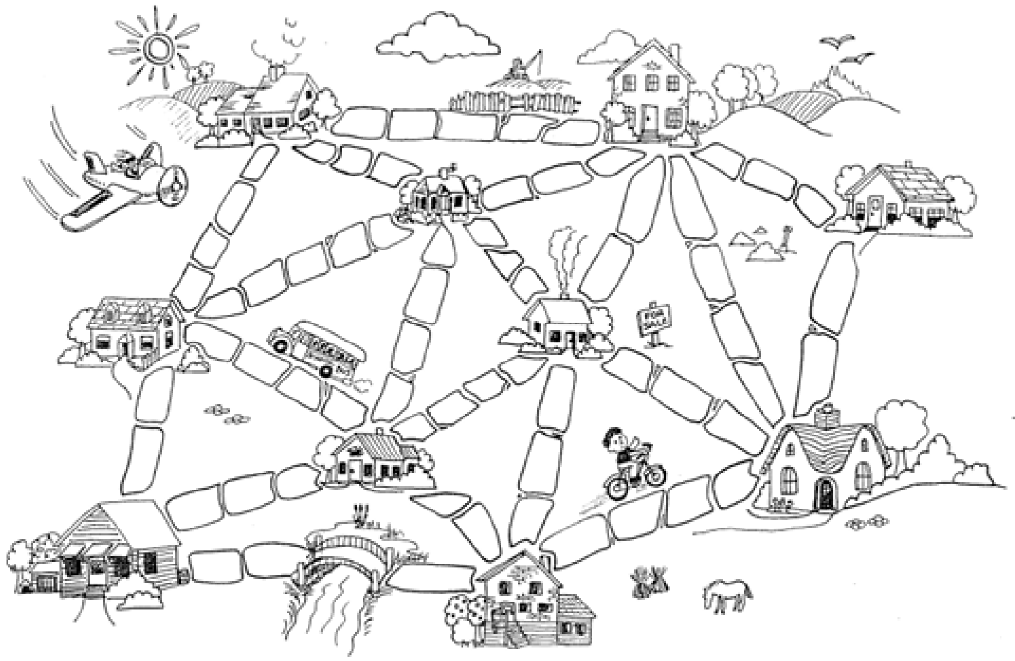
A city has no paved roads, only muddy roads. So, getting around after rainstorms is very difficult. The city mayor decides to pave “some” roads to minimize costs.

Which roads to pave such that it is possible for everyone to travel from their house to anyone else’s house only along the paved roads with minimum cost?

Example #1

A city has no paved roads, only muddy roads. So, getting around after rainstorms is very difficult. The city mayor decides to pave “some” roads to minimize costs.

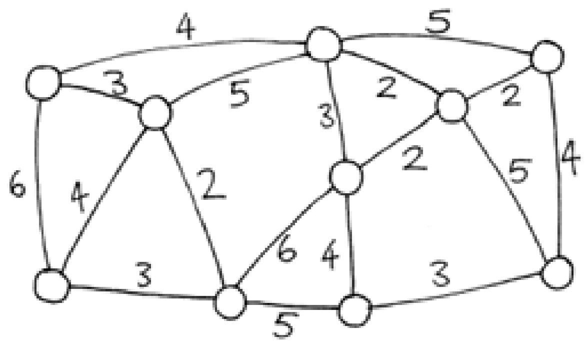
Which roads to pave such that it is possible for everyone to travel from their house to anyone else’s house only along the paved roads with minimum cost?

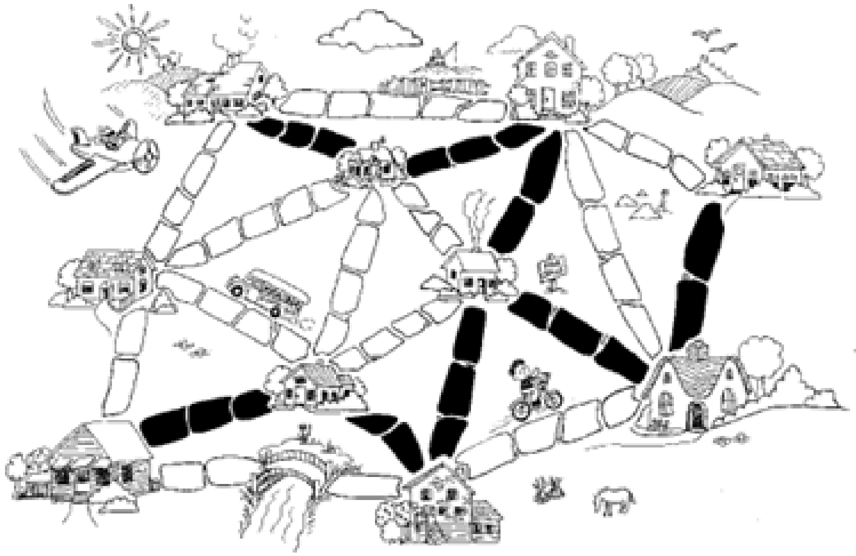


Example #1

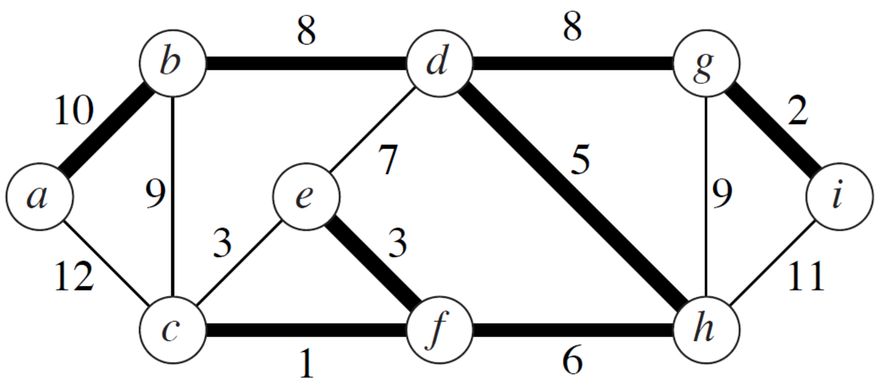
A city has no paved roads, only muddy roads. So, getting around after rainstorms is very difficult. The city mayor decides to pave "some" roads to minimize costs.

Which roads to pave such that it is possible for everyone to travel from their house to anyone else's house only along the paved roads with minimum cost?





Example



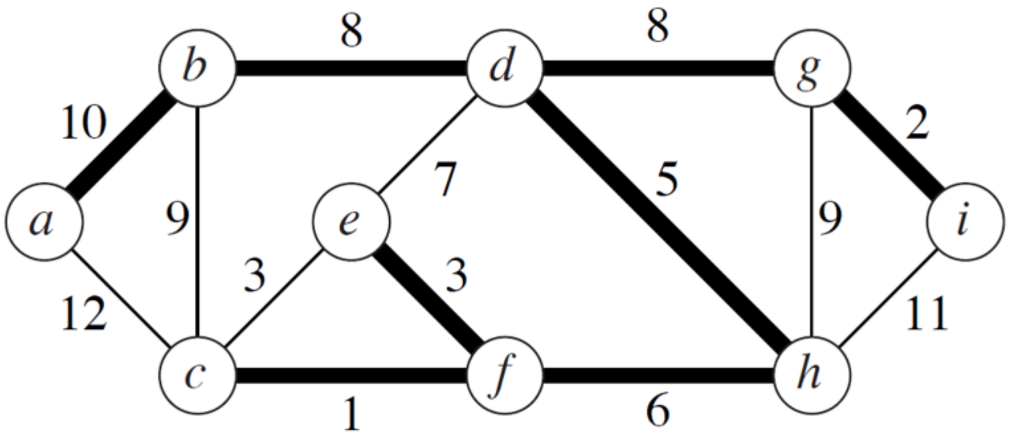
In this example, there is more than one MST. Replace edge (e, f) in the MST by (c, e). Get a different spanning tree with the same weight.

Towards a Solution

- Build a set A of edges
  - Initially |A| = 0 (i.e., no edges)
- As edges are added to A, maintain a loop invariant
  - **Loop invariant**: A is a subset of some MST
- Only add edges that maintain the invariant
  - If A is a subset of some MST, an edge (u, v) is ***safe*** for A if and only if $A \cup \{(u, v)\}$ is also a subset of some MST
  - So, only add ***safe edges***

```
GENERIC-MST(G, w)
A = ∅
while A does not form a spanning tree
    find an edge (u, v) that is safe for A
    A = A ∪ {(u, v)}
return A
```

How to Find a Safe Edge?



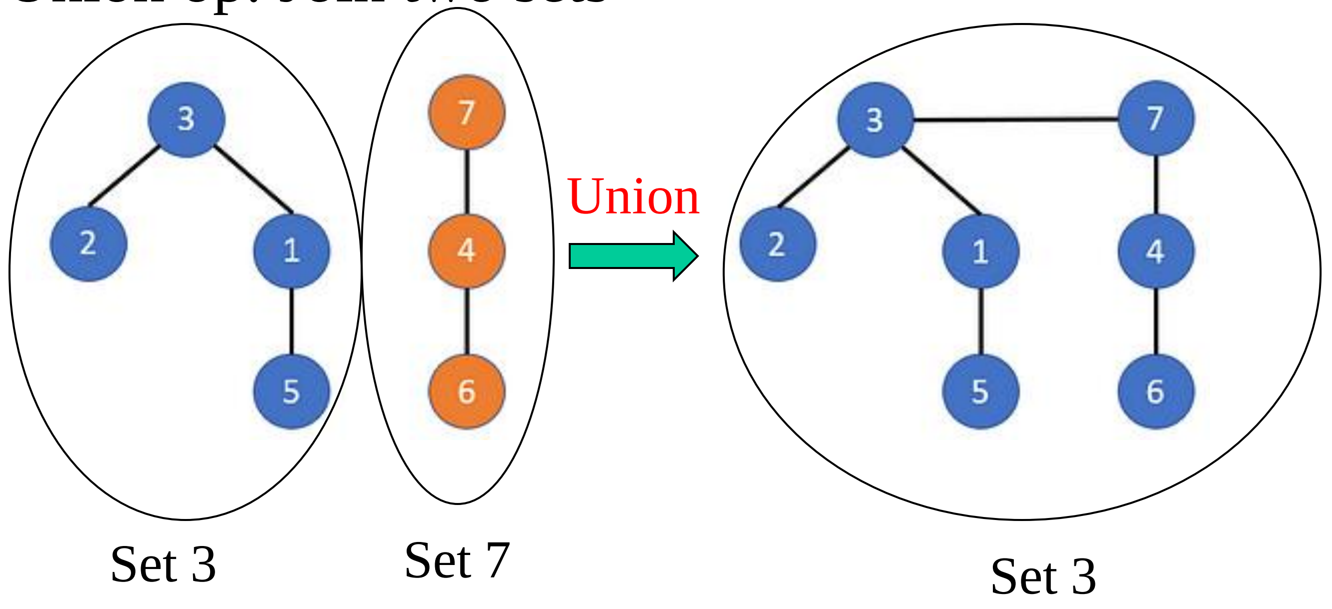
- Edge $(c, f)$is the smallest weight edge in the graph
  - Is it safe for $A = \phi$?
- Intuitively: Let $S \subset V$ be any proper subset of vertices that include c but not f (so, f is in $V - S$)
- In any MST, there *must be at least one edge* that connects S with $V - S$ $\rightarrow$ pick the edge with minimum weight !!!

Definitions

- A ***cut*** $(S, V - S)$ is a partition of vertices into disjoint sets S and $\mathrm{V} - \mathrm{S}$
- Edge $(u, v) \in E$ ***crosses*** cut $(S, V - S)$ if one endpoint is in S and the other is in $\mathrm{V} - \mathrm{S}$
- A cut ***respects*** A if and only if no edge in A crosses the cut
- An edge is a ***light edge*** crossing a cut if and only if its weight is minimum over all edges crossing the cut.
  - For a given cut, there can be > 1 light edge crossing it

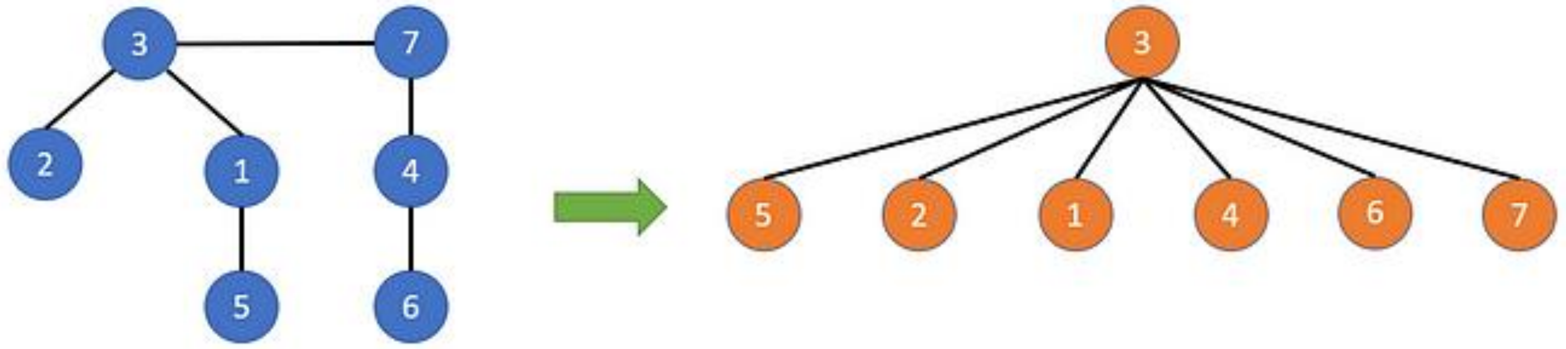
Disjoint Set (Union Find)

- Maintain non-overlapping sets of elements
  - No element can be in more than one set
  - Clustering: “similar” elements are in the same set
- Find op: True if element belongs to a set
- Union op: Join two sets



Disjoint Set (Union Find)

- Union by rank: Attach shorter tree to the root of the taller tree
- Path compression: Every node in path points to grandparent's node; flatten tree by changing parent to root node
- Makes both find and union operations log(n) (n: # elements)



Kruskal's Algorithm

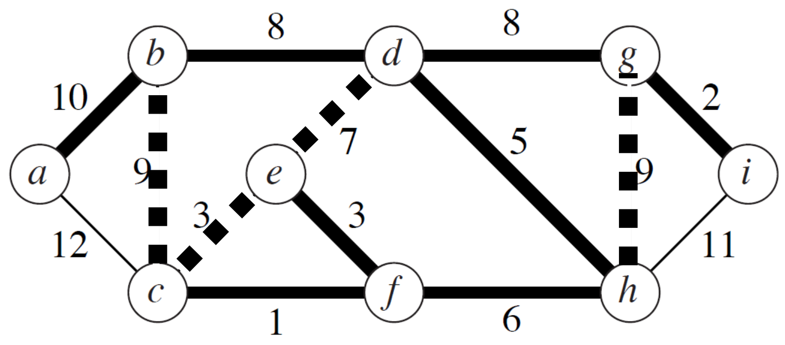
$G = (V, E)$ is a connected, undirected, weighted graph. $w : E \to \mathbb{R}$.

- Starts with each vertex being its own component.
- Repeatedly merges two components into one by choosing the light edge that connects them (i.e., the light edge crossing the cut between them).
- Scans the set of edges in monotonically increasing order by weight.
- Uses a disjoint-set data structure to determine whether an edge connects vertices in different components.

```
MST-KRUSKAL($G, w$)
1   $A = \emptyset$
2   for each vertex $v \in G.V$
3       MAKE-SET($v$)
4   create a single list of the edges in $G.E$
5   sort the list of edges into monotonically increasing order by weight $w$        O(|E| log |E|)
6   for each edge $(u, v)$ taken from the sorted list in order
7       if FIND-SET($u$) $\neq$ FIND-SET($v$)                                        O(|E|)
8           $A = A \cup \{(u, v)\}$
9           UNION($u, v$)
10  return $A$
```

Running time: O(|E| log |E| + |V|) by using union-find and path compression

Example of Kruskal



(c, f)	:	safe
(g, i)	:	safe
(e, f)	:	safe
(c, e)	:	reject
(d, h)	:	safe
(f, h)	:	safe
(e, d)	:	reject
(b, d)	:	safe
(d, g)	:	safe
(b, c)	:	reject
(g, h)	:	reject
(a, b)	:	safe

Prim's Algorithm

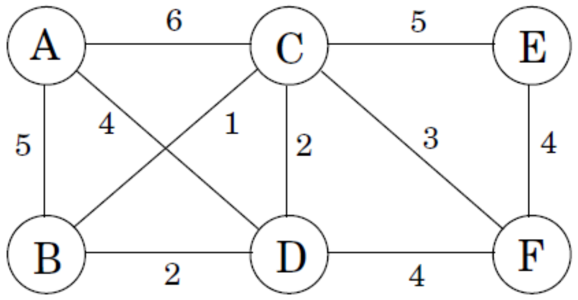
- Builds one tree, so A is always a tree.
- Starts from an arbitrary "root" r.
- At each step, find a light edge connecting A to an isolated vertex. Such an edge must be safe for A. Add this edge to A.

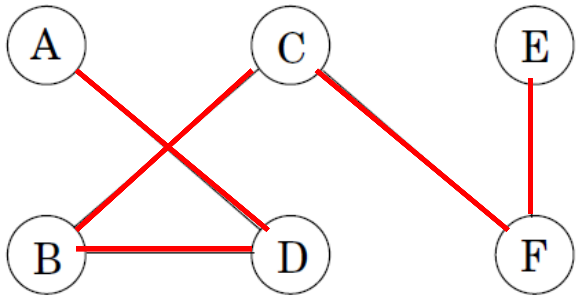
```
procedure prim$(G, w)$
Input:    A connected undirected graph $G = (V, E)$ with edge weights $w_e$
Output:   A minimum spanning tree defined by the array prev

for all $u \in V$:
   cost$(u) = \infty$
   prev$(u) =$ nil
Pick any initial node $u_0$
cost$(u_0) = 0$

$H =$ makequeue$(V)$     (priority queue, using cost-values as keys)
while $H$ is not empty:
   $v =$ deletemin$(H)$
   for each $\{v, z\} \in E$:
      if cost$(z) > w(v, z)$:
         cost$(z) = w(v, z)$
         prev$(z) = v$
         decreasekey$(H, z)$
```

$$\texttt{cost}(v) \quad = \quad \min_{u \in S} \; w(u, v).$$

Running time: O(|V|log|V|+|E|log|V|)

Example of Prim's





Cost = 14

Set S	A	B	C	D	E	F
$\{\}$	$0/\text{nil}$	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
A		$5/A$	$6/A$	$\boxed{4/A}$	∞/nil	∞/nil
A, D		$\boxed{2/D}$	$2/D$		∞/nil	$4/D$
A, D, B			$\boxed{1/B}$		∞/nil	$4/D$
A, D, B, C					$5/C$	$\boxed{3/C}$
A, D, B, C, F					$\boxed{4/F}$	

Example #2

Distribute message to N agents. Each agent can communicate with some of the other agents, but their communication is (independently) detected with probability p_{ij}.

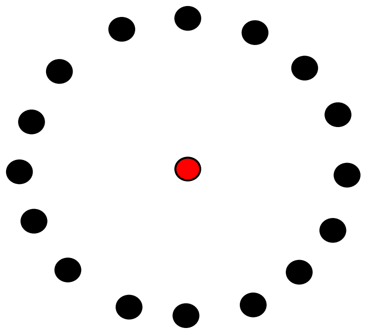
Group leader wants to transmit message to all agents to minimize the total probability that message is detected.
How will you solve this?

MST Applications

- Communication. transportation networks
- Clustering
- TSP Christofides’ heuristic
- Image segmentation, recognition, handwriting recognition
- Circuit design
- ….

Prim vs. Dijkstra

- Both use a PQ to remove a vertex with the smallest cost/dist in each iteration
- In Prim's, we track to make sure we don't revisit a vertex that has been removed from the PQ
- How can we combine?
  - $(1-\alpha)\cdot Prim's + \alpha \cdot$ Dijkstra $(0 \leq \alpha \leq 1)$
  - $\alpha = \{0, 0.5, 1\}..$what types of trees do we get (assume all edges from the red vertex to black vertices have weight "r")

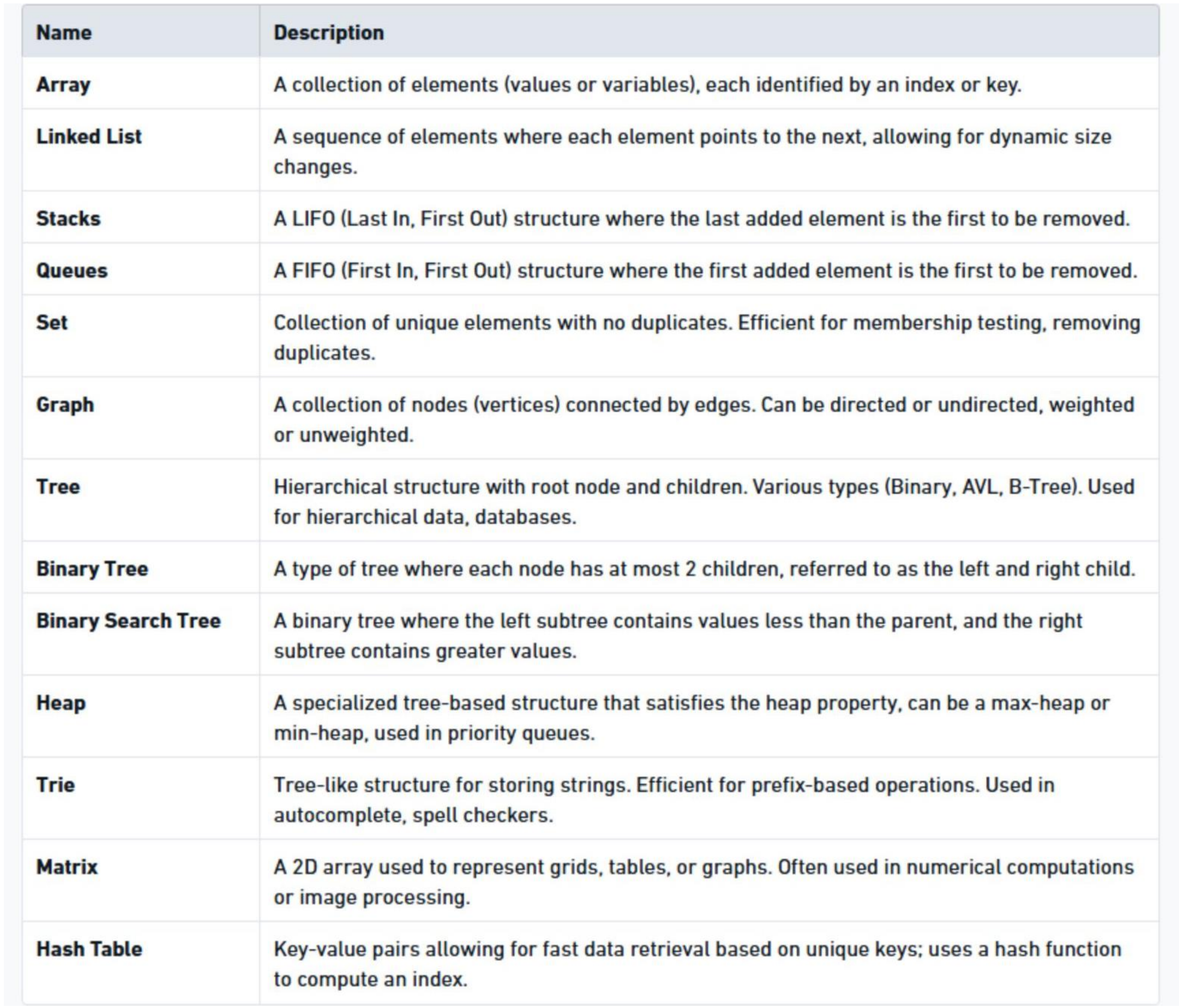


Greedy Summary

- Important: Pick a greedy choice!!!

Problem	**Type: min/max**	**Greedy choice**
Dijkstra’s	Min path length	Shortest distance from every visited edge
Huffman’s coding	Min symbols	Smallest freq character first
Interval scheduling / activity selection	Max #activities for a resource	Earliest finish times
Interval partitioning	Min #classrooms / resources	Earliest start times
Minimum spanning trees	Min cost tree	Lightest edge first
Coin change (US currency)	Min #coins	Highest denomination first
Fractional knapsack	Max value in knapsack	Highest value/weight ratio first

Important Data Structures



Name	Description
Array	A collection of elements (values or variables), each identified by an index or key.
Linked List	A sequence of elements where each element points to the next, allowing for dynamic size changes.
Stacks	A LIFO (Last In, First Out) structure where the last added element is the first to be removed.
Queues	A FIFO (First In, First Out) structure where the first added element is the first to be removed.
Set	Collection of unique elements with no duplicates. Efficient for membership testing, removing duplicates.
Graph	A collection of nodes (vertices) connected by edges. Can be directed or undirected, weighted or unweighted.
Tree	Hierarchical structure with root node and children. Various types (Binary, AVL, B-Tree). Used for hierarchical data, databases.
Binary Tree	A type of tree where each node has at most 2 children, referred to as the left and right child.
Binary Search Tree	A binary tree where the left subtree contains values less than the parent, and the right subtree contains greater values.
Heap	A specialized tree-based structure that satisfies the heap property, can be a max-heap or min-heap, used in priority queues.
Trie	Tree-like structure for storing strings. Efficient for prefix-based operations. Used in autocomplete, spell checkers.
Matrix	A 2D array used to represent grids, tables, or graphs. Often used in numerical computations or image processing.
Hash Table	Key-value pairs allowing for fast data retrieval based on unique keys; uses a hash function to compute an index.

Lecture 10 summary

- Activity selection
- Interval scheduling
- MSTs