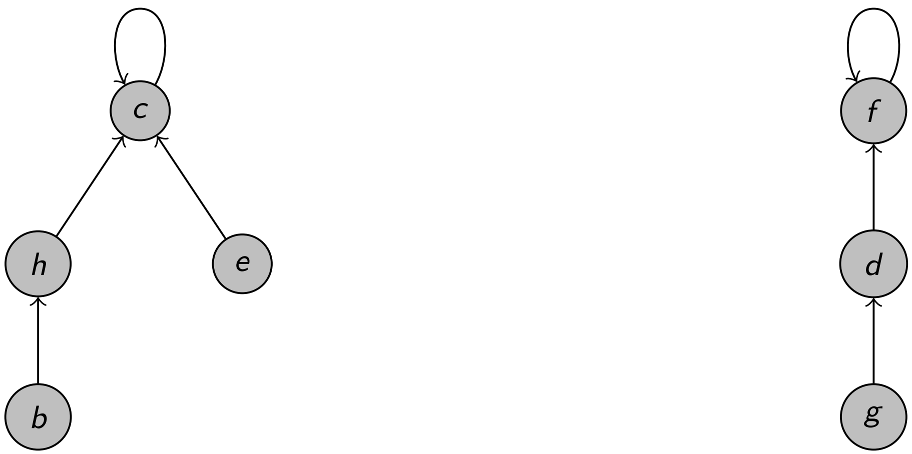
Minimum Spanning Trees: Krushkal's Algorithm and Prim's Algorithm

Subhabrata Samajder

IIIT, Delhi
Winter Semester,
31th May, 2023

Disjoint-set Forests (Cont.)

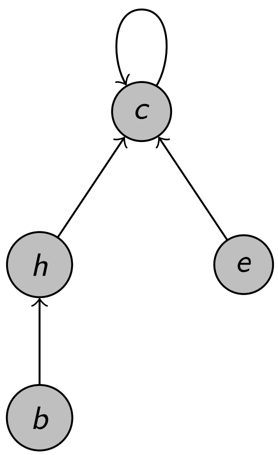
An Example: UNION(e, g) (Recap)



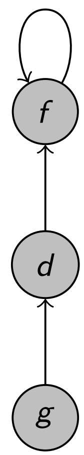
$S_1 = \{b, c, e, h\}$

$S_2 = \{d, f, g\}$

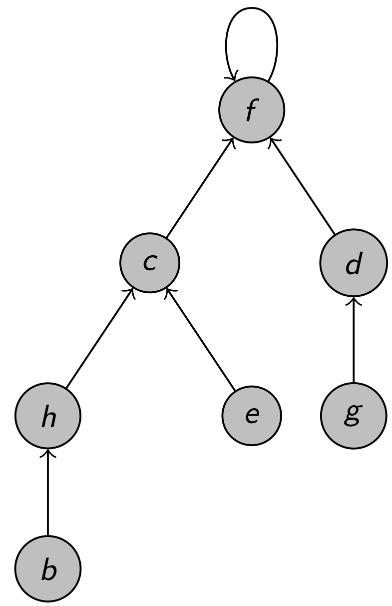
An Example: $\textsc{Union}(e, g)$ (Recap)



$S_1 = \{b, c, e, h\}$



$S_2 = \{d, f, g\}$



$\textsc{Union}(e, g) = S_1 \cup S_2$

Heuristics 1: Union by Rank (Recap)

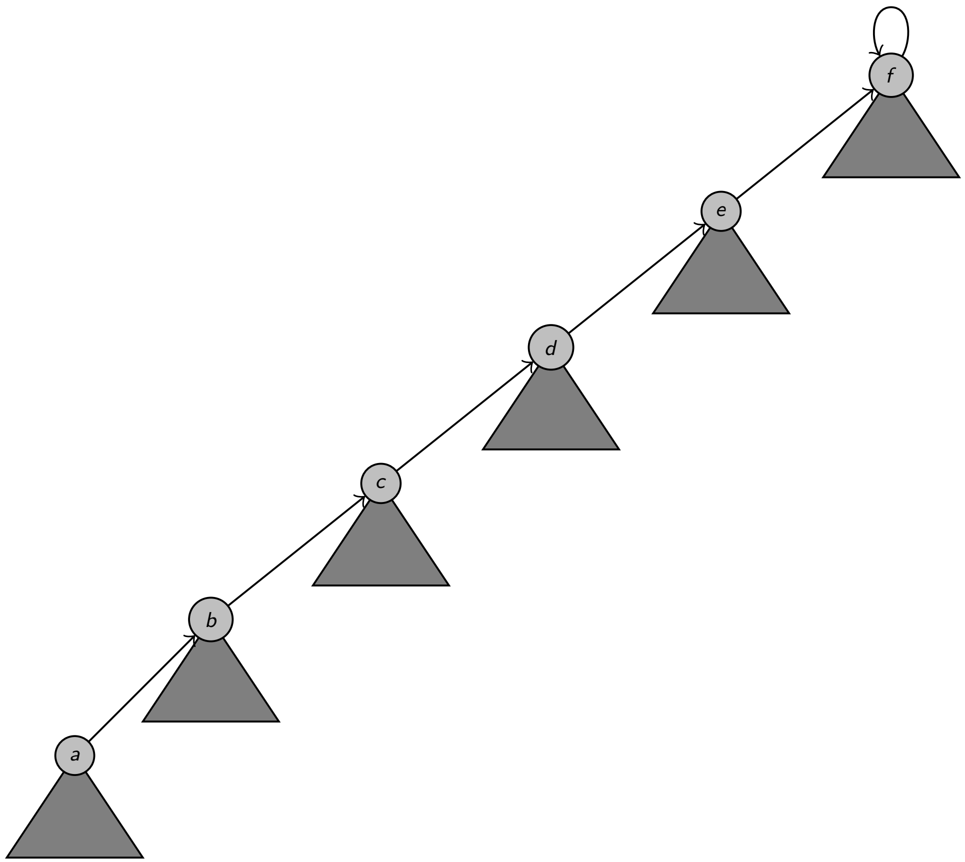
- **Idea:** During UNION operation, make the root of the tree with fewer nodes point to the root of the tree with more nodes.
- Rather than explicitly keeping track of the size of the subtree rooted at each node, we shall use an approach that eases the analysis.
- For each node, we maintain a rank that is an upper bound on the height of the node.
- UNION: Point the root with smaller rank to the root with larger rank.

Heuristics 2: Path Compression (Recap)

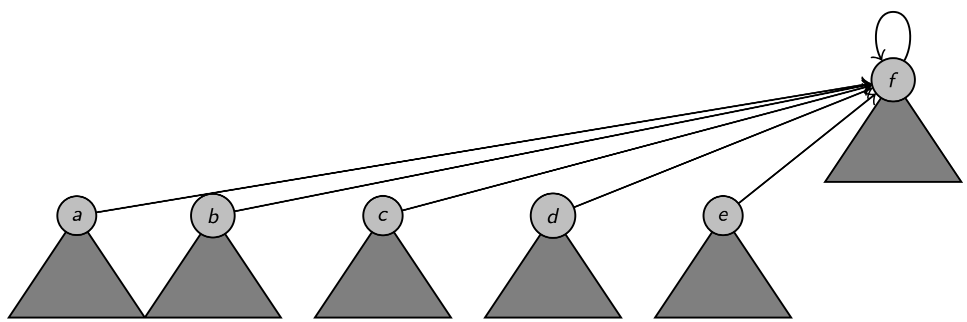
- FIND-SET: Make each node on the FIND-PATH to point directly to the root.

- Path compression does not change any ranks.

An Example: Before FIND-SET(a) (Recap)



An Example: After FIND-SET(a) (Recap)



Pseudocode for Disjoint-set Forests

- Each node x maintains an integer value $rank[x]$.
- $rank[x]$: Is an upper bound on the height (the number of edges in the longest path between x and a descendant leaf) of x.
- After MAKE-SET the initial rank of the single node in the corresponding tree is 0.
- FIND-SET operation leaves all ranks unchanged.
- When applying Union to two trees, there are two cases, depending on whether the roots have equal rank.
  - Unequal rank: Make the root of higher rank the parent of the root of lower rank, but the ranks themselves remain unchanged.
  - Equal ranks: Arbitrarily choose one of the roots as the parent and increment its rank by 1.

MAKE-SET, UNION and LINK

MAKE-SET(x)
1. $p[x] \leftarrow x$; // $p[x]$ denotes the parent of x
2. $rank[x] \leftarrow 0$;

MAKE-SET, UNION and LINK

MAKE-SET(x)
1. $p[x] \leftarrow x$; // $p[x]$ denotes the parent of x
2. $rank[x] \leftarrow 0$;

UNION(x, y)
1. LINK(FIND-SET(x), FIND-SET(y));

MAKE-SET, UNION and LINK

MAKE-SET(x)
$p[x] \leftarrow x$; // $p[x]$ denotes the parent of x
$rank[x] \leftarrow 0$;

UNION(x, y)
LINK(FIND-SET(x), FIND-SET(y));

LINK(x, y) // Takes pointers to the roots of x and y as inputs
if ($rank[x] > rank[y]$)
    $p[y] \leftarrow x$;
else
    $p[x] \leftarrow y$;
    if ($rank[x] = rank[y]$)
        $rank[y] = rank[y] + 1$;

Find-Set

Find-Set(x)
1. if ($x \neq p[x]$)
2. $p[x] \leftarrow$ Find-Set($p[x]$);
3. return $p[x]$;

It is a two-pass method:

- Takes one pass up the find path to find the root.
- Second pass back down the find path to update each node so that it points directly to the root.
- **Note:** Line 2 updates node x to point directly to the root, and this pointer is returned in line 3.

Effect of the Heuristics on the Running Time

- Separately, either union by rank or path compression improves the running time of the operations on disjoint-set forests.
- The improvement is even greater when the two heuristics are used together.
- Alone, union by rank yields a running time of $\mathcal{O}(m \log n)$ and this bound is tight.
- If there are n MAKE-SET operations (and hence at most $n-1$ UNION operations) and f FIND-SET operations, the path-compression heuristic alone gives a worst-case running time of $\mathcal{O}(n + f \cdot \log_{2+f/n} n)$.
- **Worst-case Complexity (Together):** $\mathcal{O}(m \cdot \alpha(n))$, where $\alpha(n)$ is a *very* slowly growing function.
- In any conceivable application $\alpha(n) \leq 4$.
- $\therefore$ in most practical situations the running time is linear in m.

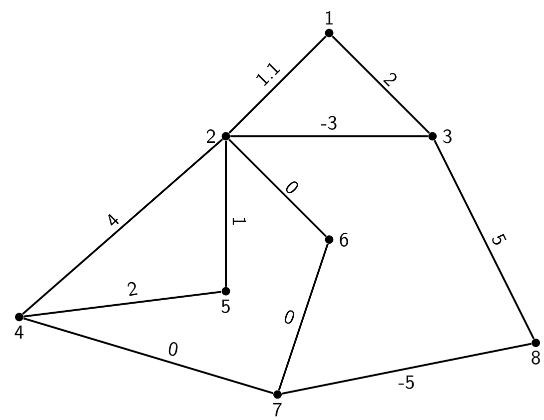
Minimum Spanning Treee (MST)

Some Definitions

Definition (Weighted Graph)

A **weighted graph** $G = (V, E, w)$ is a graph $G = (V, E)$ with an associated function $w : E \to \mathbb{R}$, called the **weight function**.

Example: Let H be a subgraph of G.



Then w is easily extended to subgraphs in a natural way,

$$w(H) = \sum_{e \in H} w(e).$$

Some Definitions (Cont.)

Definition (Spanning Tree)

A **spanning tree** for $G = (V, E)$ is a tree T whose vertex set is V.

Some Definitions (Cont.)

Definition (Spanning Tree)

A **spanning tree** for $G = (V, E)$ is a tree T whose vertex set is V.

Definition (Minimum Spanning Tree (MST))

A **minimum spanning tree** for a weighted graph $G = (V, E)$ is a spanning tree T, such that $w(T)$ is the minimum among the weights of all spanning trees of G.

Note: There may be more than one MST's for G.

Motivation

Problem: Suppose there are n cities in a country. For each pair of cities c_1 and c_2 one can estimate the cost of connecting c_1 and c_2 by a direct highway. What is the minimum cost of connecting all the cities?

Motivation

Problem: Suppose there are n cities in a country. For each pair of cities c_1 and c_2 one can estimate the cost of connecting c_1 and c_2 by a direct highway. What is the minimum cost of connecting all the cities?

Solution: A MST for K_n with a suitable weight function.

Kruskal's Algorithm

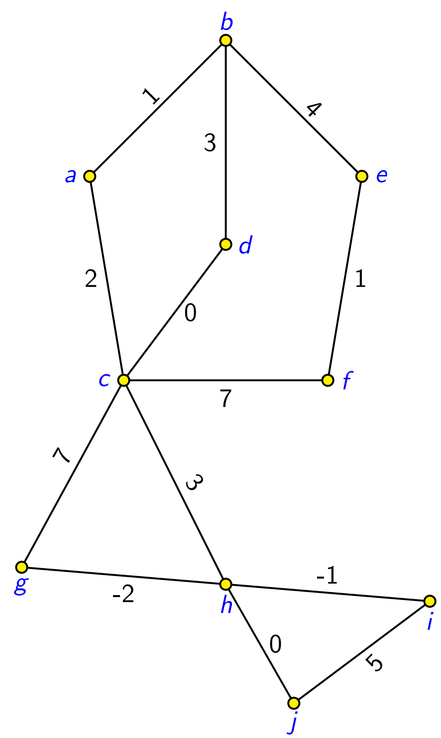
Introduction

- Joseph Kruskal (1956). "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". *Proceedings of the American Mathematical Society*.
- Uses a Greedy strategy.
- **Idea:** Finds an edge of the least possible weight connecting any two trees in the forest.
- Connected graphs: It finds a subset of the edges that forms a tree that includes every vertex, such that the total weight of all the edges in the tree is minimized.
- Disconnected graphs: It finds a minimum spanning forest.
- **Other MST Algorithms:** Prim's algorithm, Reverse-delete algorithm, and Boråvka's algorithm.

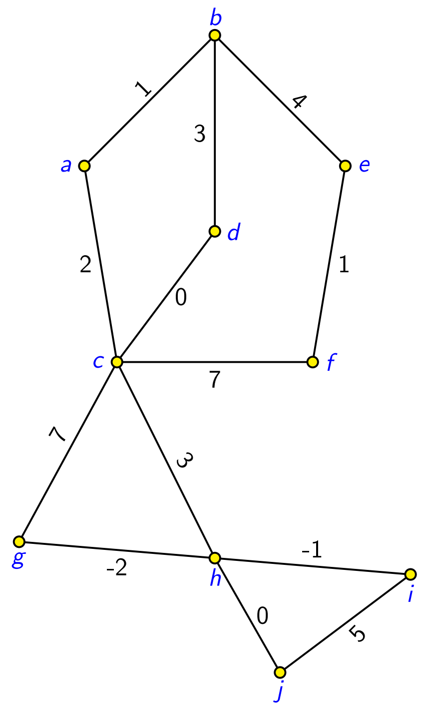
Introduction

- Joseph Kruskal (1956). "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". *Proceedings of the American Mathematical Society*.
- Uses a Greedy strategy.
- **Idea:** Finds an edge of the least possible weight connecting any two trees in the forest.
- Connected graphs: It finds a subset of the edges that forms a tree that includes every vertex, such that the total weight of all the edges in the tree is minimized.
- Disconnected graphs: It finds a minimum spanning forest.
- **Other MST Algorithms:** Prim's algorithm, Reverse-delete algorithm, and Boråvka's algorithm.

KruskalAlgorithm(G)



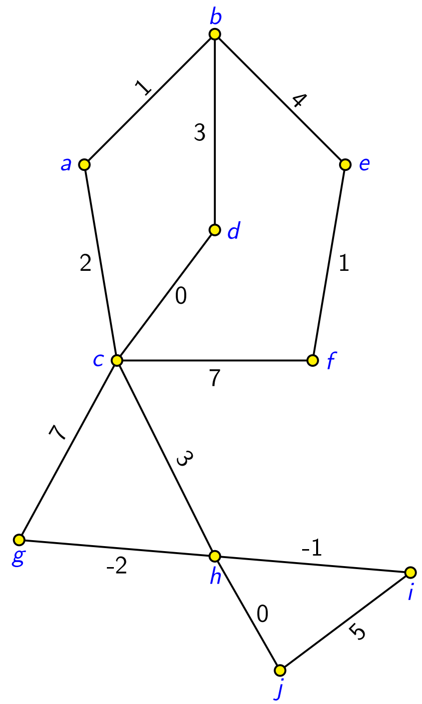
KruskalAlgorithm(G)



I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

KruskalAlgorithm(G)



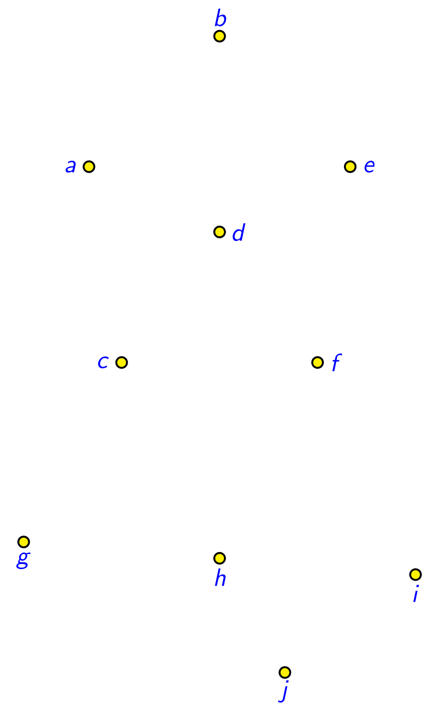
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



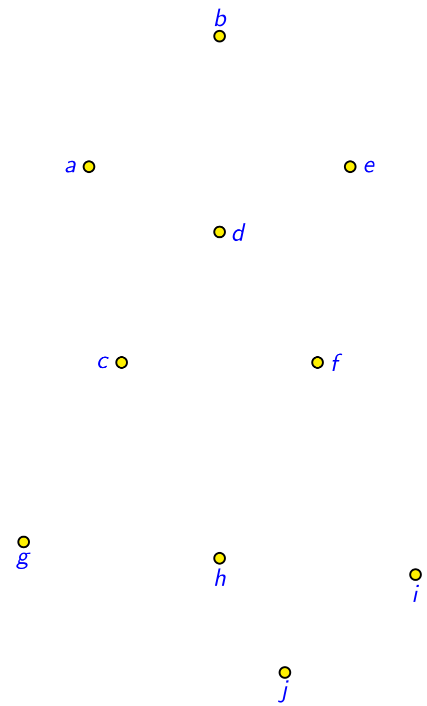
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



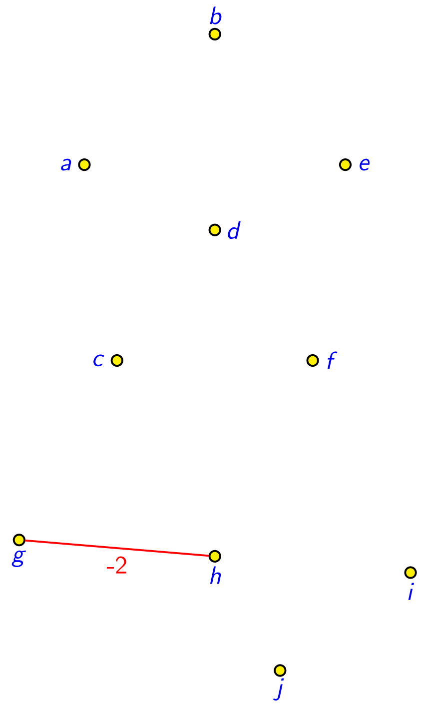
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KRUSKALALGORITHM(G)



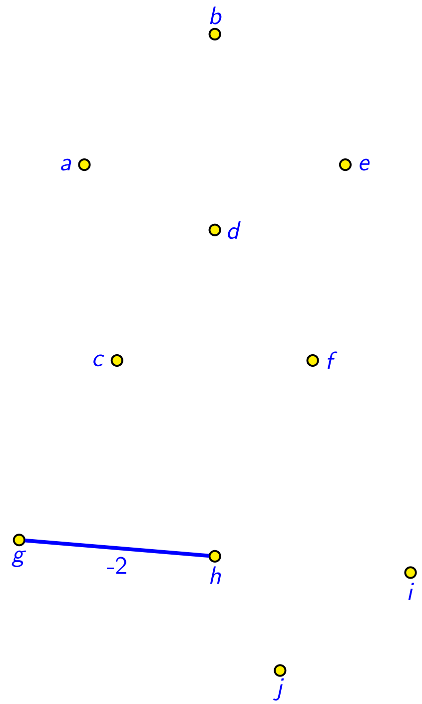
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** *$T \cup \{e\}$ creates a cycle*

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



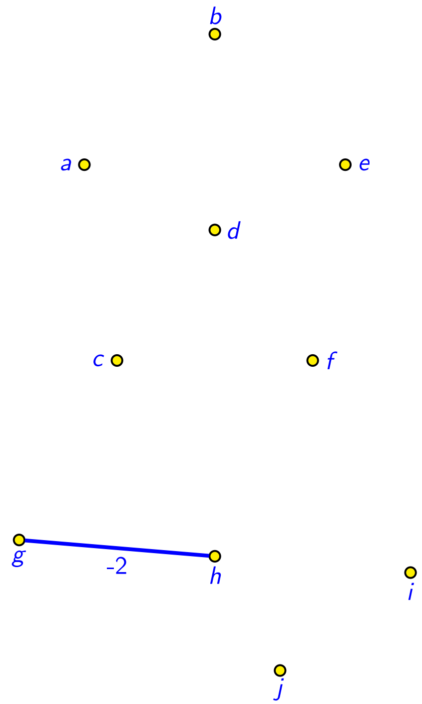
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



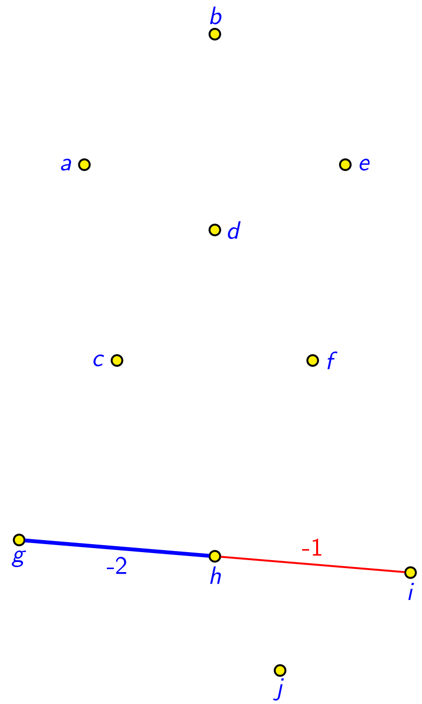
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KRUSKALALGORITHM(G)



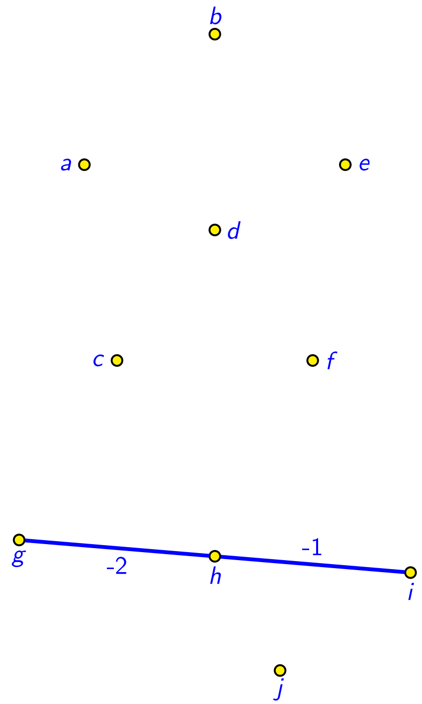
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



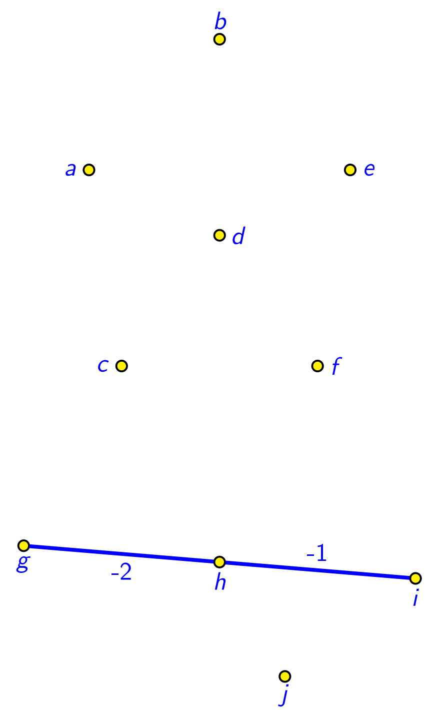
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



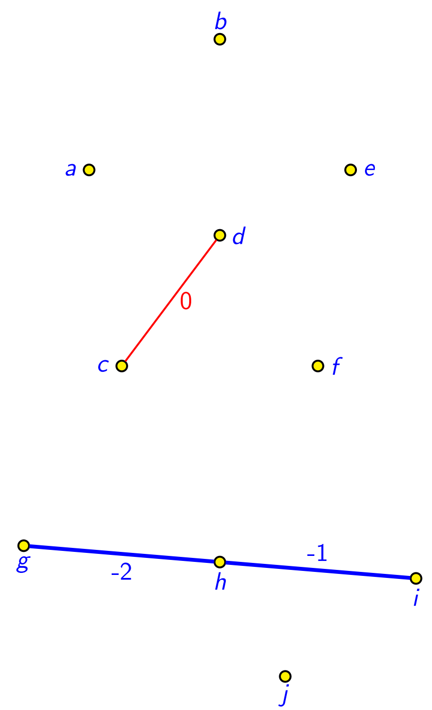
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



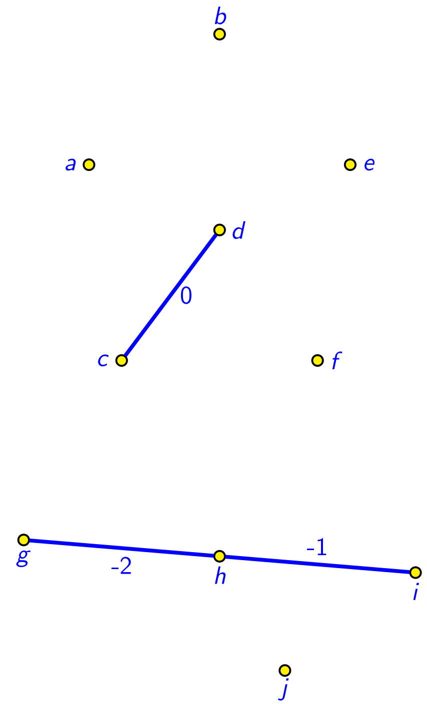
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** *$T \cup \{e\}$ creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



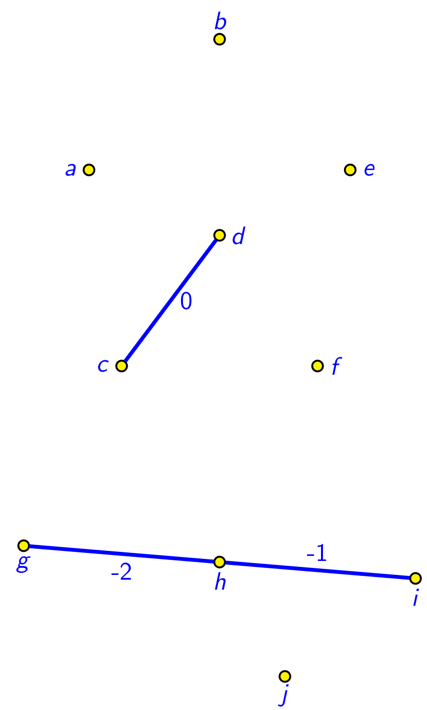
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KRUSKALALGORITHM(G)



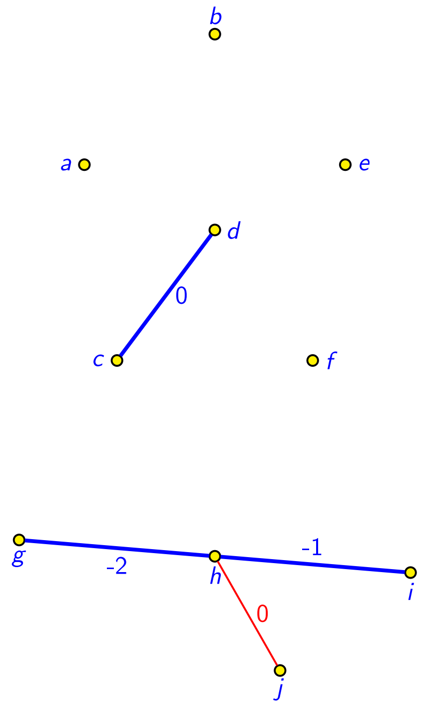
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



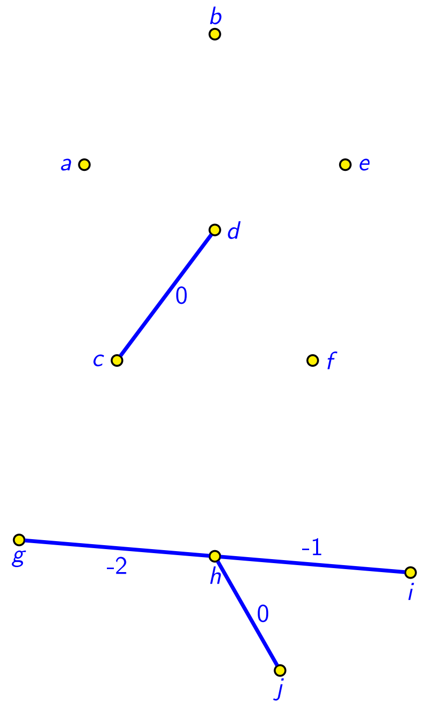
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** *$T \cup \{e\}$ creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



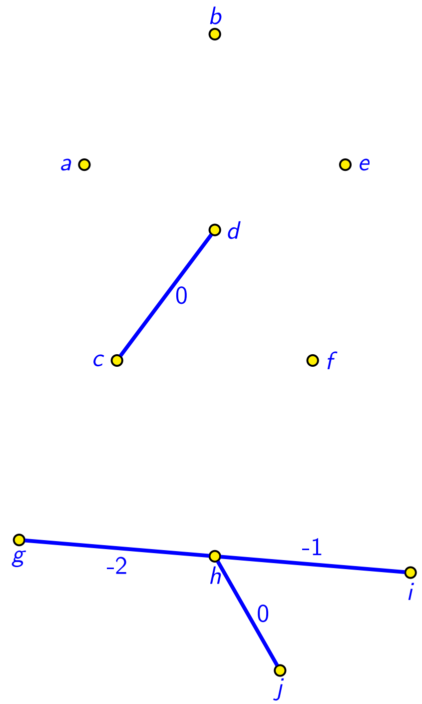
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KRUSKALALGORITHM(G)



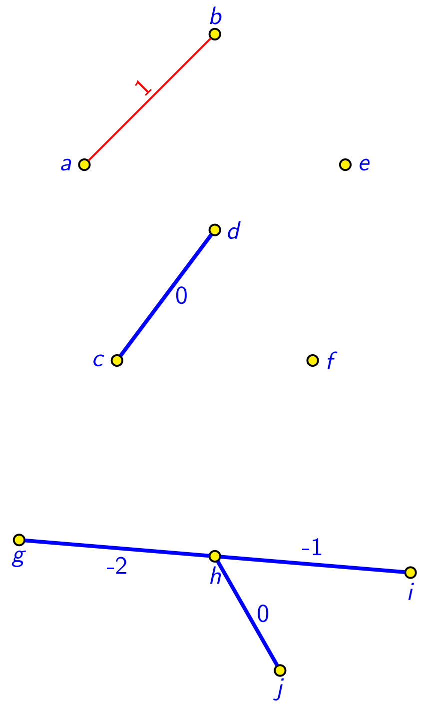
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



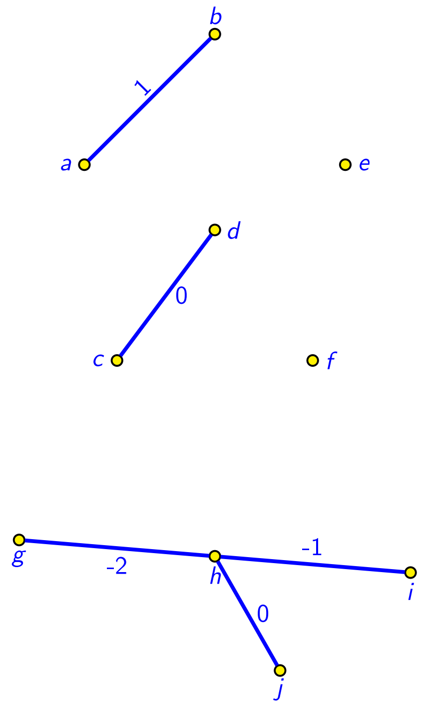
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



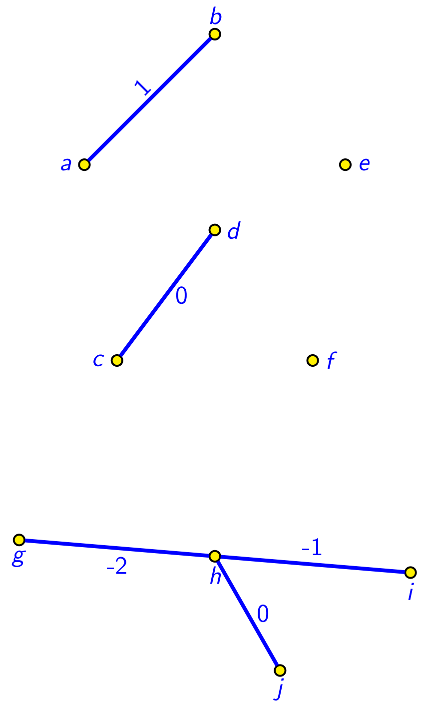
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



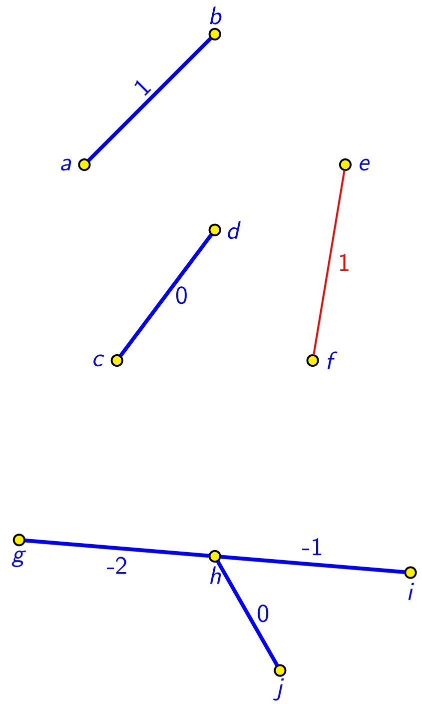
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

$\textsc{KruskalAlgorithm}(G)$



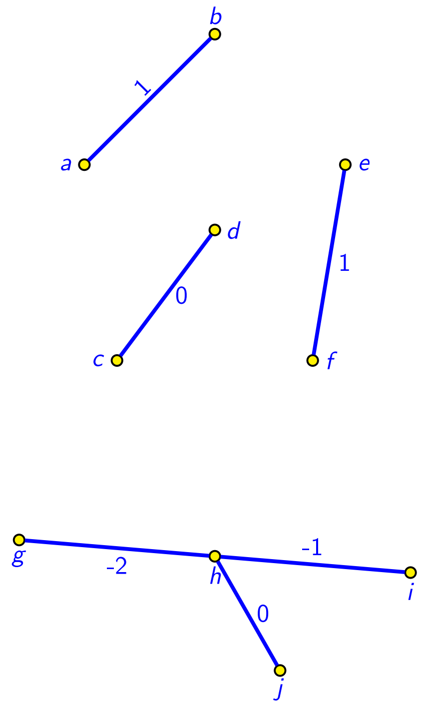
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



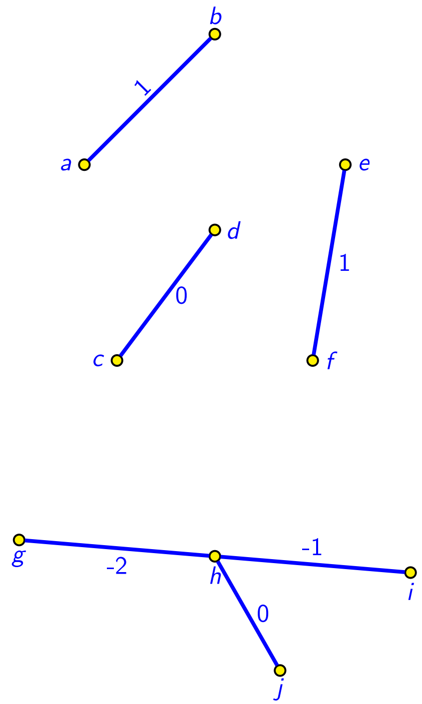
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



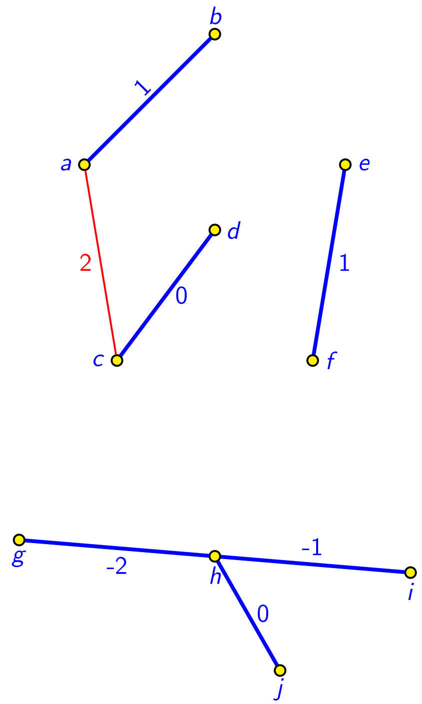
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



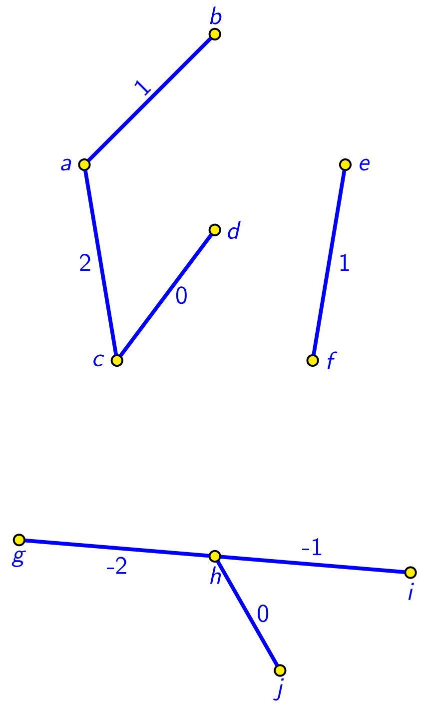
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



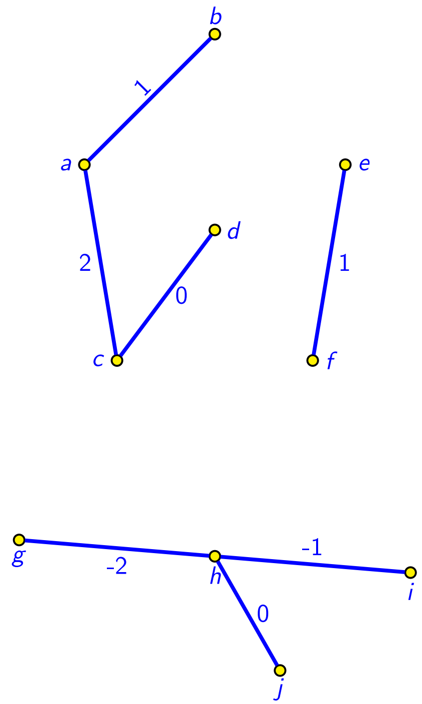
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



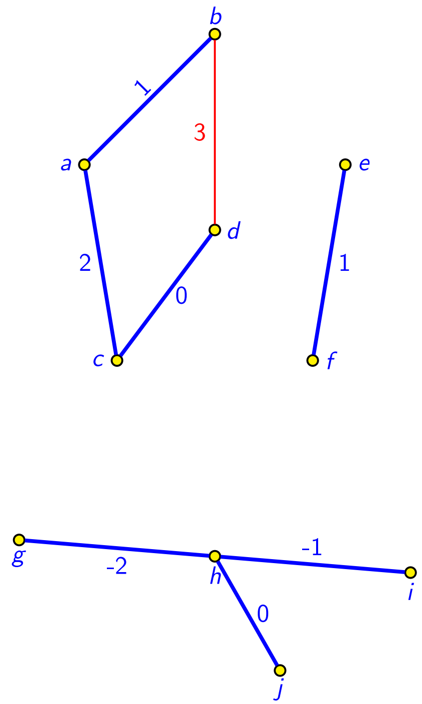
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KRUSKALALGORITHM(G)



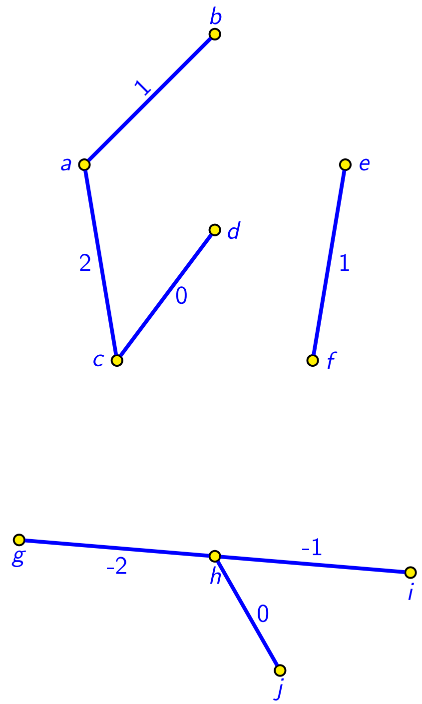
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



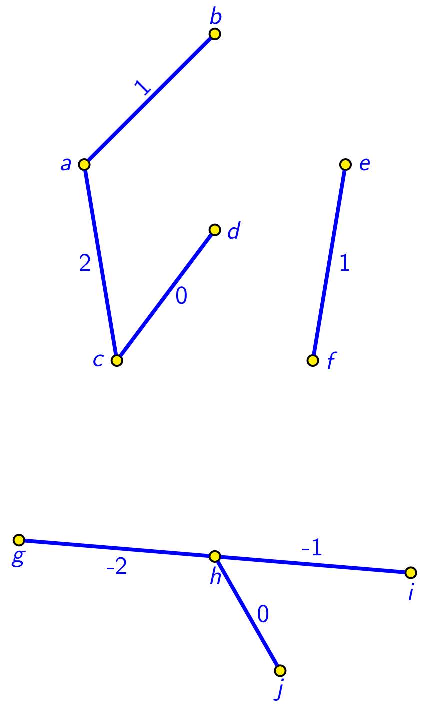
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



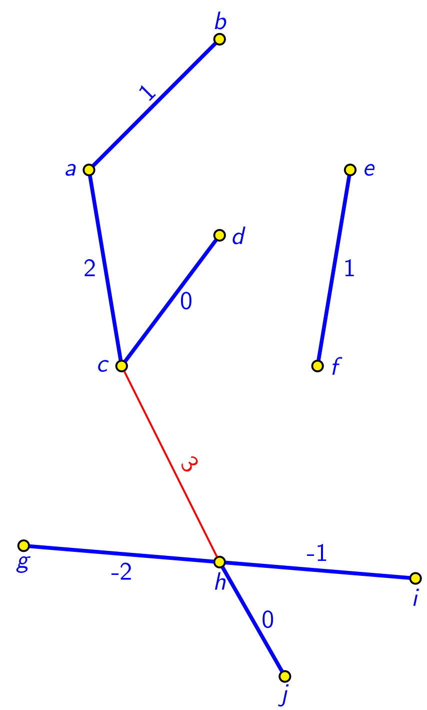
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



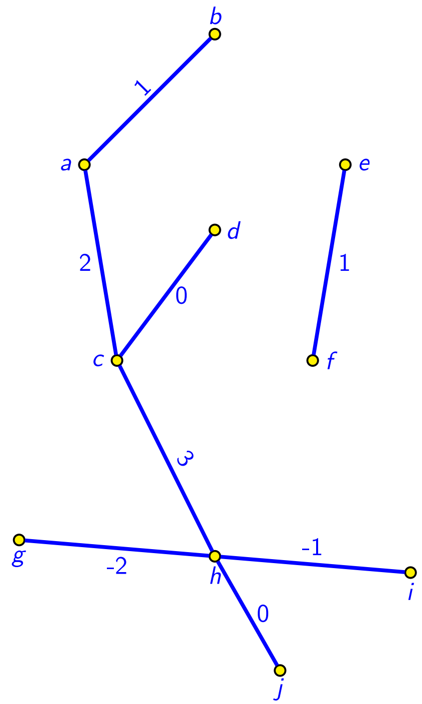
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** *$T \cup \{e\}$ creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



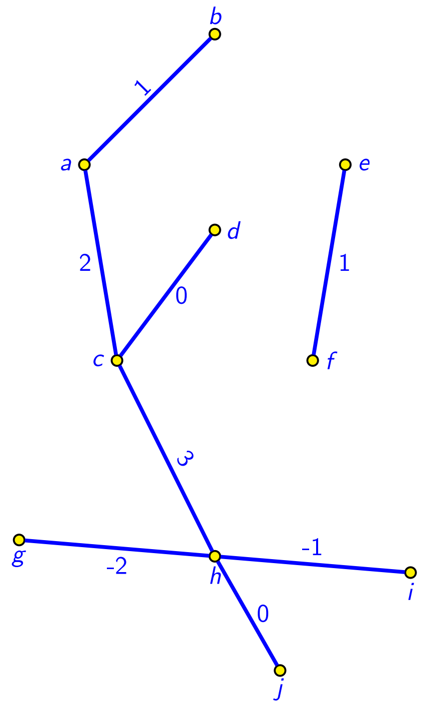
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



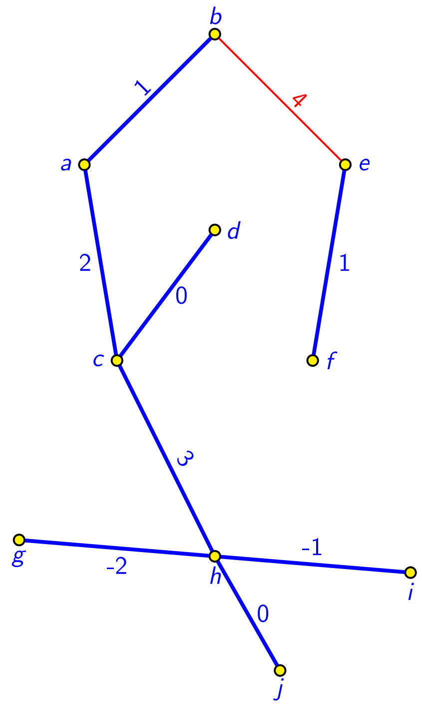
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g,h)	-2	(a,b)	1	(c,h)	3	(c,g)	7
(h,i)	-1	(e,f)	1	(b,e)	4		
(c,d)	0	(a,c)	2	(j,i)	5		
(h,j)	0	(b,d)	3	(c,f)	6		

KruskalAlgorithm(G)



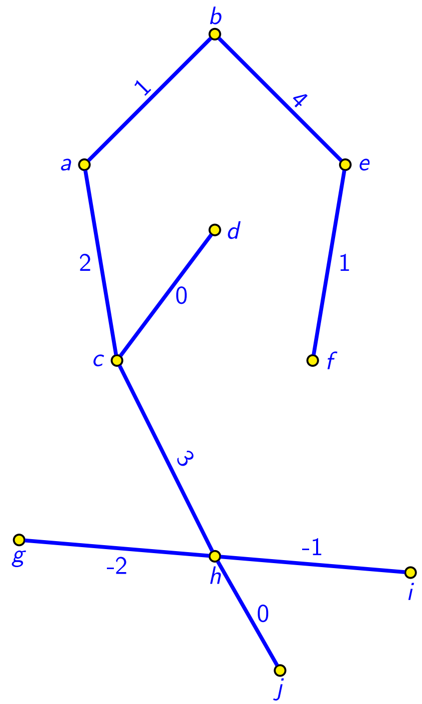
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** *$T \cup \{e\}$ creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



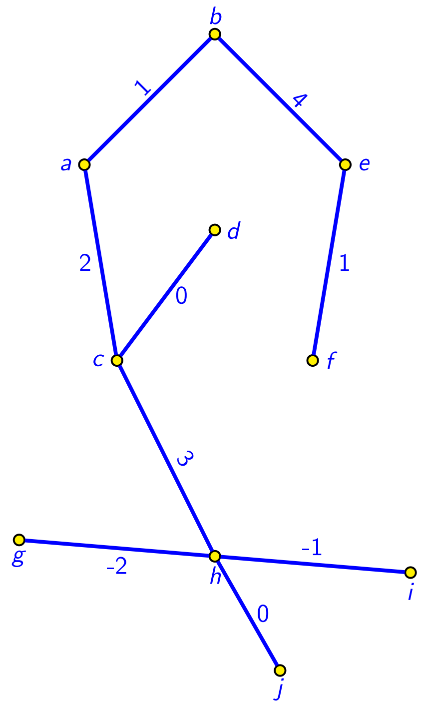
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



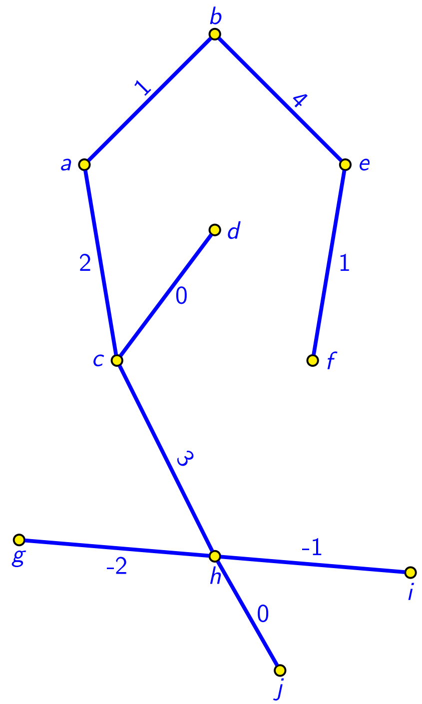
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KRUSKALALGORITHM(G)



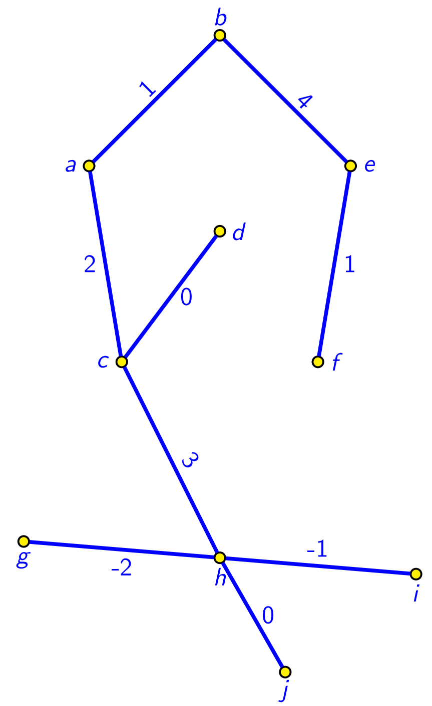
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

Edges	Weights	Edges	Weights	Edges	Weights	Edges	Weights
(g, h)	-2	(a, b)	1	(c, h)	3	(c, g)	7
(h, i)	-1	(e, f)	1	(b, e)	4		
(c, d)	0	(a, c)	2	(j, i)	5		
(h, j)	0	(b, d)	3	(c, f)	6		

KruskalAlgorithm(G)



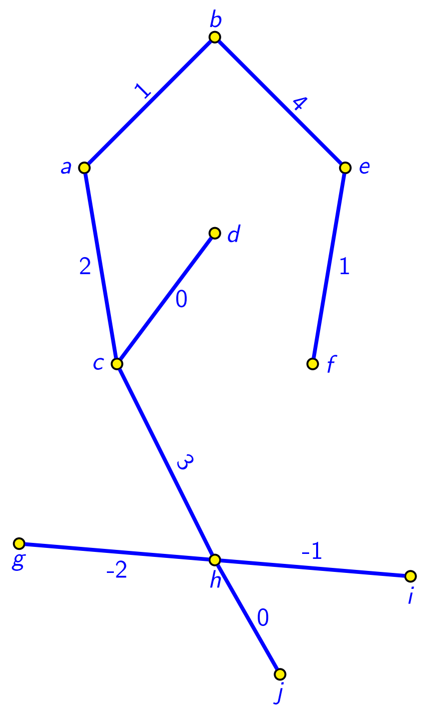
I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

$$w(G) = \sum_{e \in E'} w(e) = 8.$$

KruskalAlgorithm(G)



I/P: A weighted undirected $G = (V, E, w)$.

O/P: An MST $T = (V, E')$ of G.

1. $\mathcal{L} \leftarrow E$
   Sort $\mathcal{L}$ in ascending order.
2. (Grow a subgraph $T = (V, E')$ into a tree)
   Initially $E' = \emptyset$
3. **while** $(|E'| < n - 1)$
4. Pick the next edge in $\mathcal{L}$.
5. **if** $T \cup \{e\}$ *creates a cycle*
6. Reject e.
7. **else**
8. $E' \leftarrow E' \cup \{e\}$.

$$w(G) = \sum_{e \in E'} w(e) = 8.$$

Question: How to check whether $T \cup \{e\}$ creates a cycle or not?

Correctness

Theorem

The spanning tree returned by Kruskal's algorithm is a MST.

Proof of the Theorem

Let T be returned by the algorithm and let S be a MST of G.

If T is equal to S then there is nothing to prove.

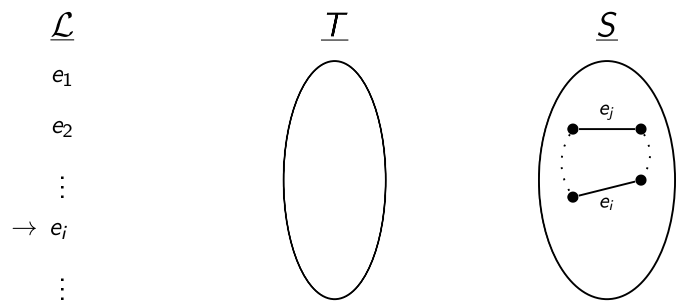
So, let $T \neq S$.

Strategy: From S form a spanning tree S' s.t., $w(S') \leq w(S)$ and S' has one more edge in common to T then S does.

Repeat the strategy at most $n-1$ times to obtain a tree S_0 which is equal to T and

$$w(T) = w(S_0) \leq \cdots \leq w(S') \leq w(S).$$

Proof of the Theorem



Since $S \neq T$, let $e_i \in \mathcal{L}$ be the first i, s.t., $e_i \in T$ but $e_i \notin S$.
Consider $S \cup \{e_i\}$.

- This must have created a cycle in S.
- Then there must be an edge e_j on this cycle which is not in T.
- Modify S:

$$S' \leftarrow (S \setminus \{e_j\}) \cup \{e_i\}.$$

So S' has one more edge in common to T than S and

$$w(S') = w(S) - w(e_j) + w(e_i).$$

Proof of the Theorem

Can e_j's position in $\mathcal{L}$ be earlier to e_i?

Proof of the Theorem

Can e_j's position in $\mathcal{L}$ be earlier to e_i? No.

- Notice that $e_j \notin T$ because it must have formed a cycle with edges in T which are earlier to e_j in $\mathcal{L}$.
- By definition of e_i, if $j < i$, then all these edges must be in S.
- Hence with e_j they form a cycle in S! $\Rightarrow\!\Leftarrow$

$\therefore$ $j > i$, which implies that $w(S') < w(S)$. Hence the theorem.

An Implementation Using Disjoint-set Data Structure

MST-KRUSKAL(G, w)
I/P: A connected, weighted undirected graph $G = (V, E)$ and the corresponding weight function w.
O/P: A list of edges of the MST.

```
A ← ∅

for each vertex v ∈ V
    MAKE-SET(v)

sort the edges of E into non-decreasing order of weight w

for each edge (u, v) ∈ E, taken in non-decreasing order of weight
    if (FIND-SET(u) ≠ FIND-SET(v))
        A ← A ∪ {(u, v)}
        UNION(u, v)

return A
```

Time Complexity

- The running time depends on the implementation of the disjoint-set data structure.
- Assume the disjoint-set-forest implementation with the union-by-rank and path-compression heuristics.
- Recall that it is the asymptotically fastest implementation known.

Time Complexity

- **Line 1:** Initializing the set A takes $\mathcal{O}(1)$ time.
- **Line 4:** Sort the list of edges takes $\mathcal{O}(|E| \log |E|)$ times.
- **Lines 2-3:** Performs $|V|$ $\textsc{Make-Set}$ operations.
- **Lines 5-8:** Performs $\mathcal{O}(|E|)$ $\textsc{Find-Set}$ and $\textsc{Union}$ operations.
- **Lines 2-3 + Lines 5-8:** Takes a total of

$$\begin{aligned} & \mathcal{O}((|V| + |E|)\alpha(|V|)) \\ = \;& \mathcal{O}(|E| \cdot \alpha(|V|)) \quad [\because \; |E| \geq |V| - 1 \text{ (G is connected)}] \end{aligned}$$

  time, where α is the very slowly growing function.
- Moreover, since $\alpha(|V|) = \mathcal{O}(\log |V|)$ (slowly growing w.r.t. V), the running time of Kruskal's algorithms is

$$\mathcal{O}(|E| \log |V|).$$

Prim's Algorithm

History

- 1^{st} developed in 1930 by Czech mathematician Vojtěch Jarník.
- Rediscovered and republished later by computer scientists
  - Robert C. Prim in 1957 and
  - Edsger W. Dijkstra in 1959.
- **Other Names:**
  - Jarník's algorithm,
  - Prim-Jarník algorithm,
  - Prim-Dijkstra algorithm or
  - the DJP algorithm.

Introduction

- Operates much like Dijkstra's algorithm for finding shortest paths in a graph.

- **Property:** The edges in the set A always form a single tree.

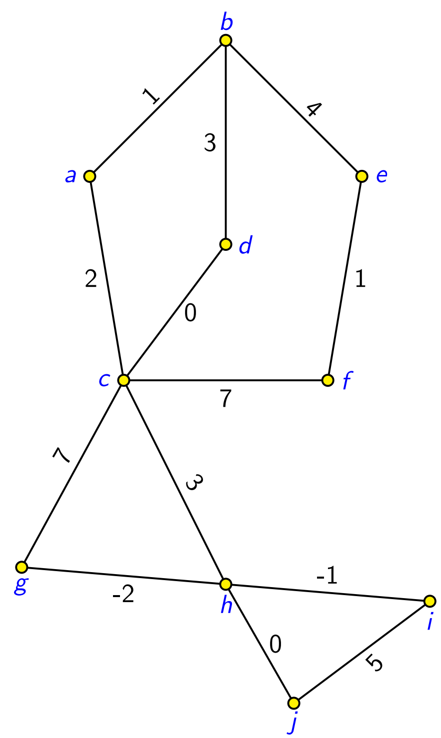
- **Key Idea:**
  - Starts from an arbitrary root vertex r.
  - Grows until the tree spans all the vertices in V.

- **Strategy:**
  - At each step, a light edge is added to the tree A that connects A to an isolated vertex of $G_A = (V, A)$.
  - Greedy: The tree is augmented at each step with an edge that contributes the minimum amount possible to the tree's weight.

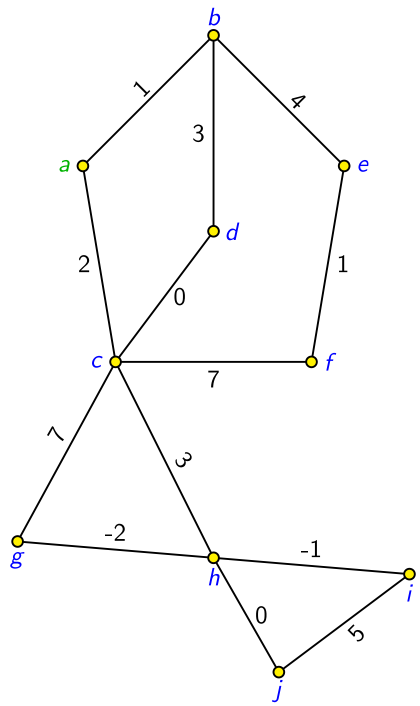
The Algorithm

- **Efficiency:** Depends on how easy it is to select a new edge to be added to the tree formed by the edges in A.
- **I/P:** A connected graph $G = (V, E)$ and the root r.
- All vertices that are not in the tree reside in a MIN-PRIORITY queue Q based on a key field.
- For each vertex v, $key[v]$ is the minimum weight of any edge connecting v to a vertex in the tree.
- By convention, $key[v] = \infty$ if there is no such edge.
- The field $\pi[v]$ names the parent of v in the tree.
- **During execution:** $A = \{(\pi[v], v) : v \in \{V \setminus \{r\}\} \setminus Q\}$.
- **Termination:** When the MIN-PRIORITY queue Q is empty.
  - $A = \{(\pi[v], v) : v \in V \setminus \{r\}\}$.

MST-PRIM(G, w, r)



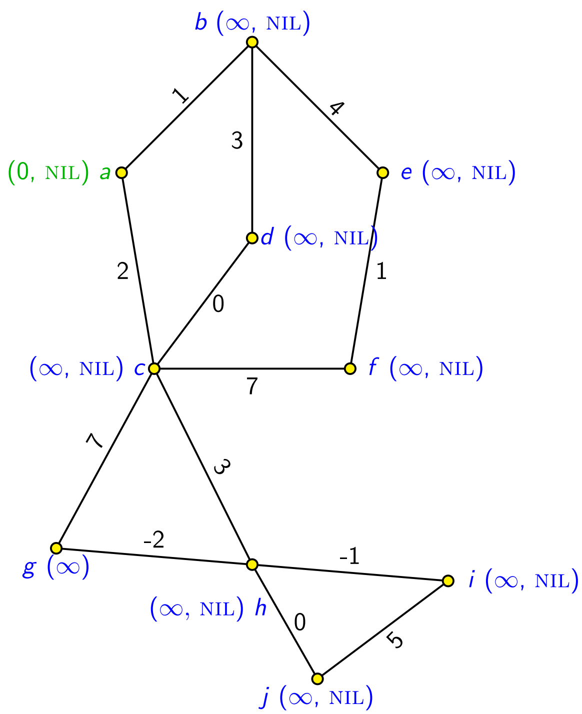
MST-PRIM(G, w, r)



I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

MST-PRIM(G, w, r)

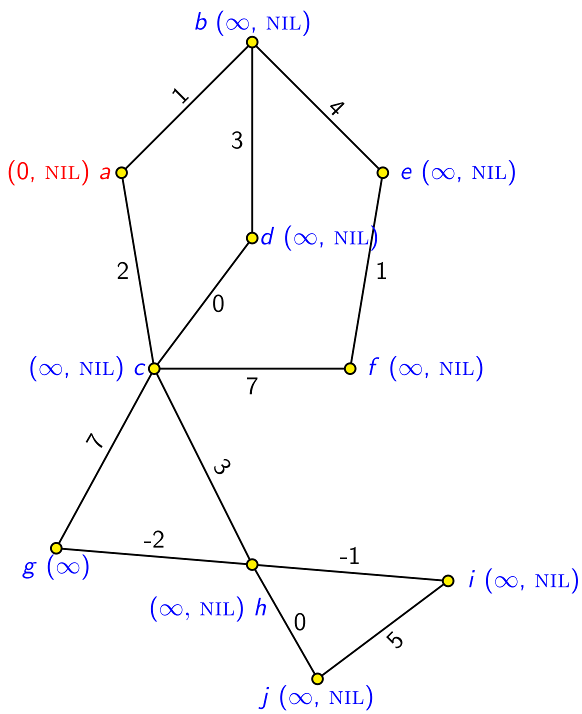


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow \text{NIL}$
4. $key[r] \leftarrow 0$

MST-PRIM(G, w, r)

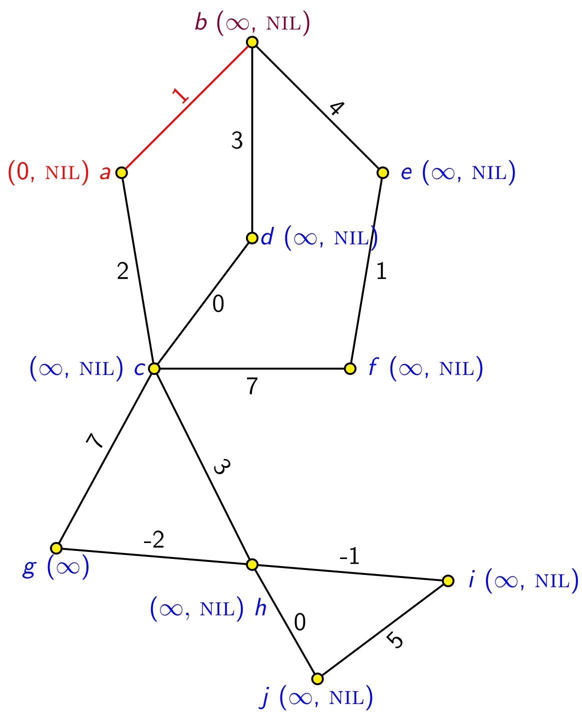


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow$ EXTRACT-MIN(Q)

MST-PRIM(G, w, r)

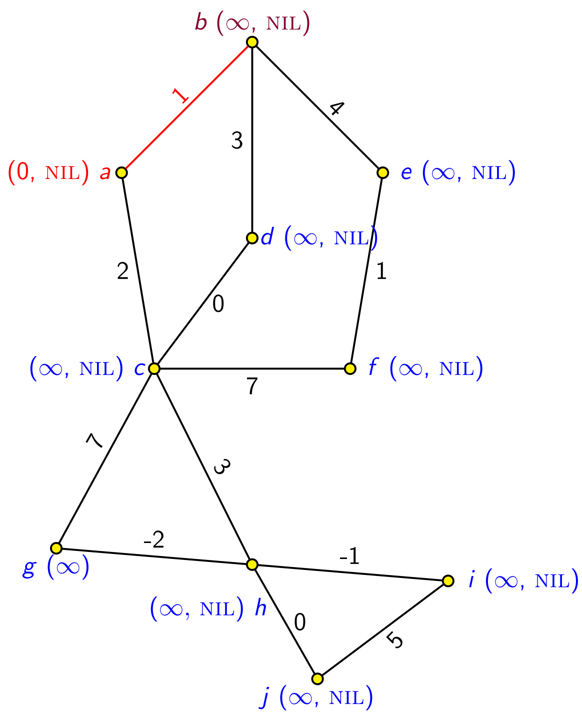


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow$ EXTRACT-MIN(Q)
8. **for** each $v \in Adj[u]$

MST-PRIM(G, w, r)

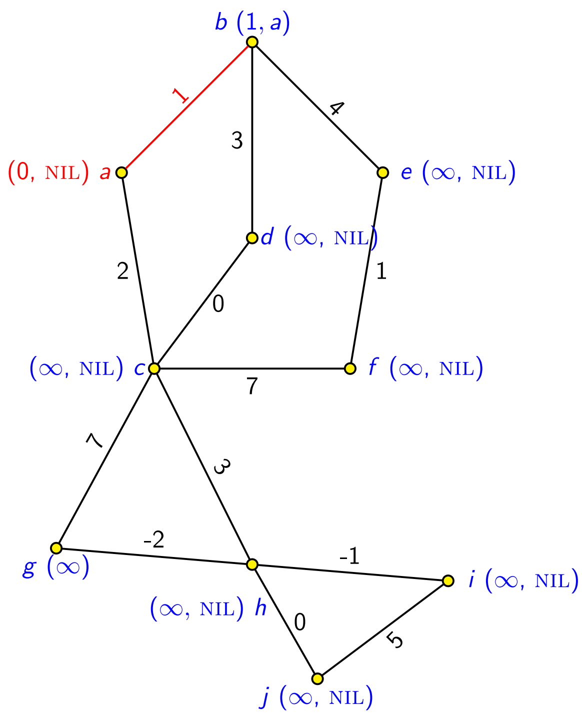


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$

MST-PRIM(G, w, r)

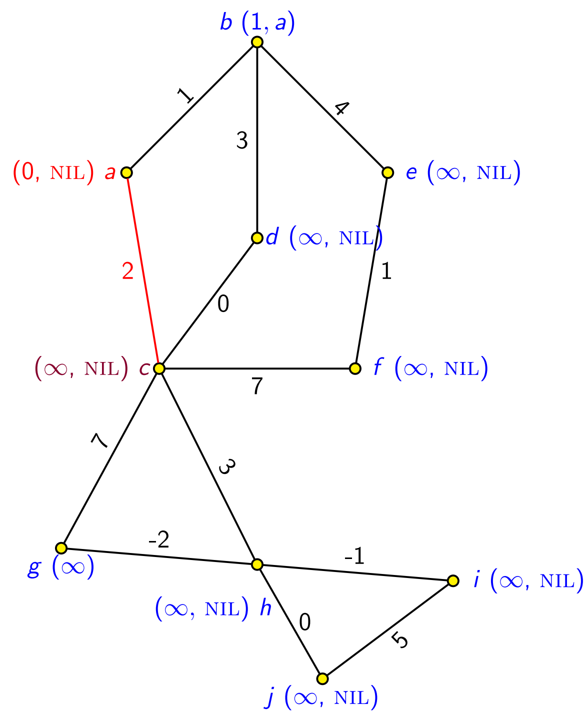


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow \text{EXTRACT-MIN}(Q)$
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

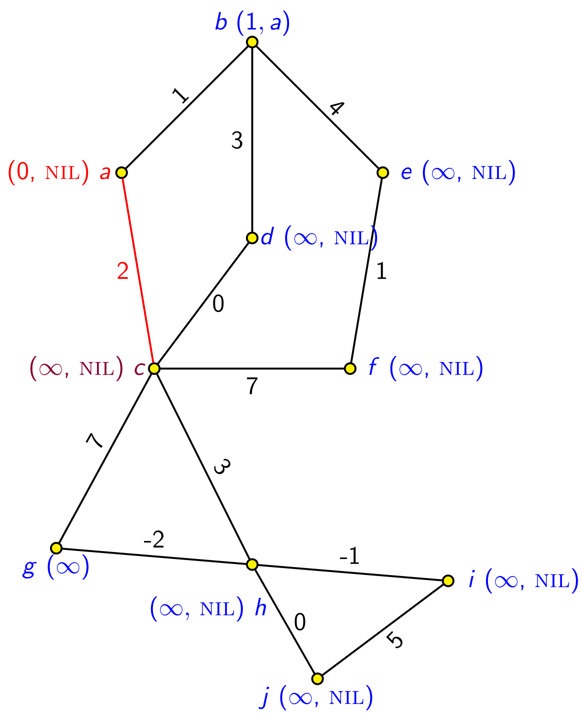


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

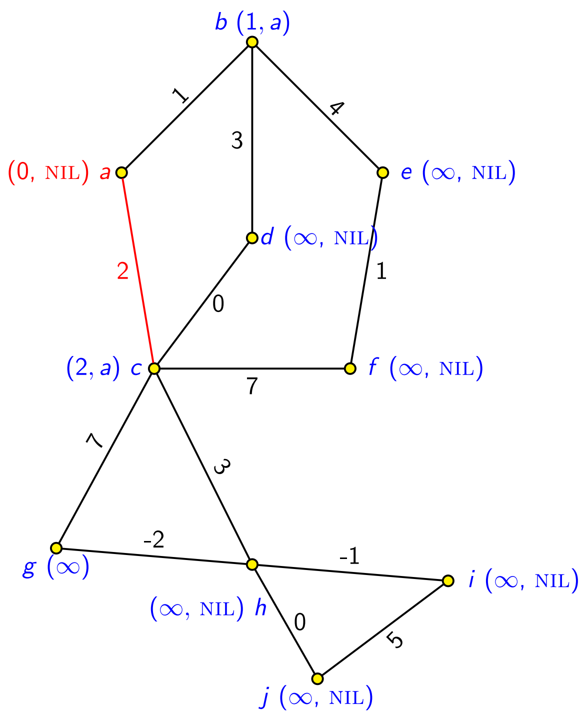


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

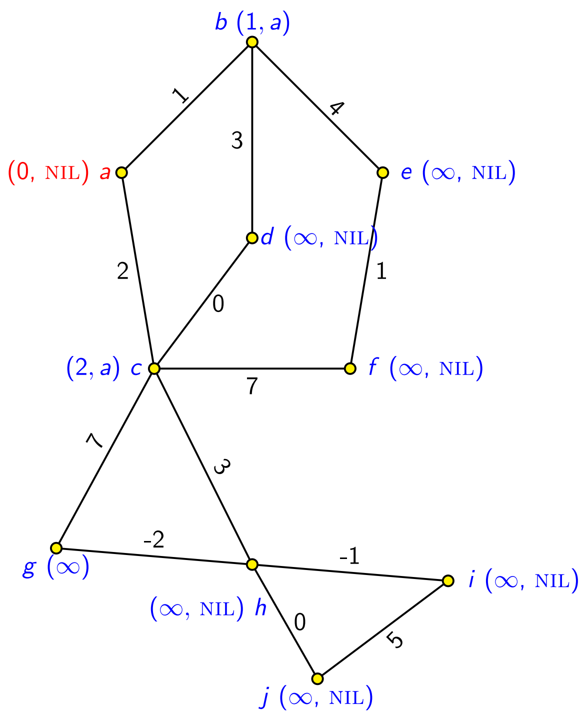


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

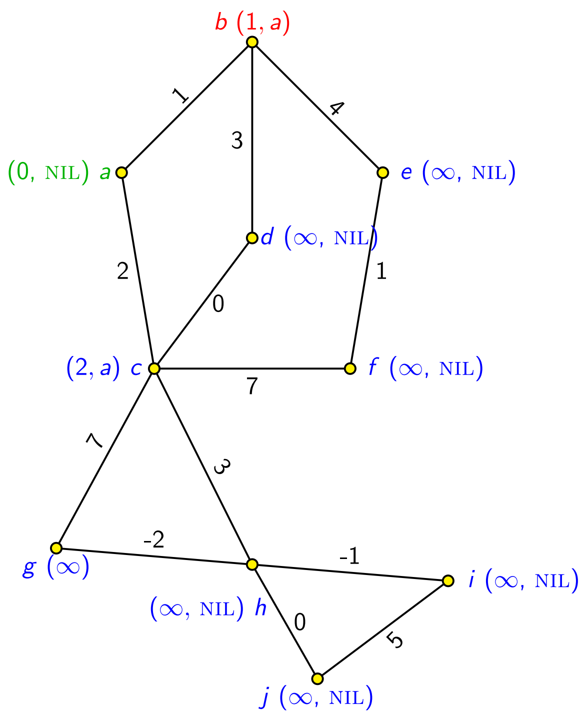


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow \text{NIL}$
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow \text{EXTRACT-MIN}(Q)$
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-Prim(G, w, r)

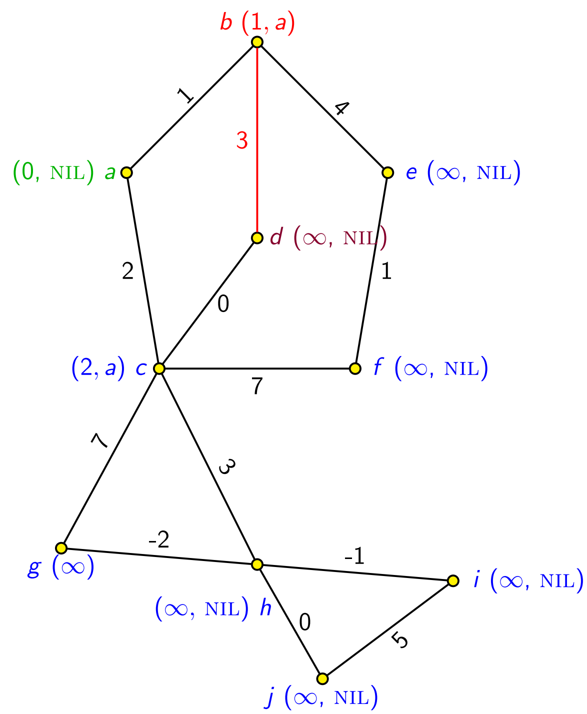


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow \text{Extract-Min}(Q)$
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-$\text{Prim}(G, w, r)$

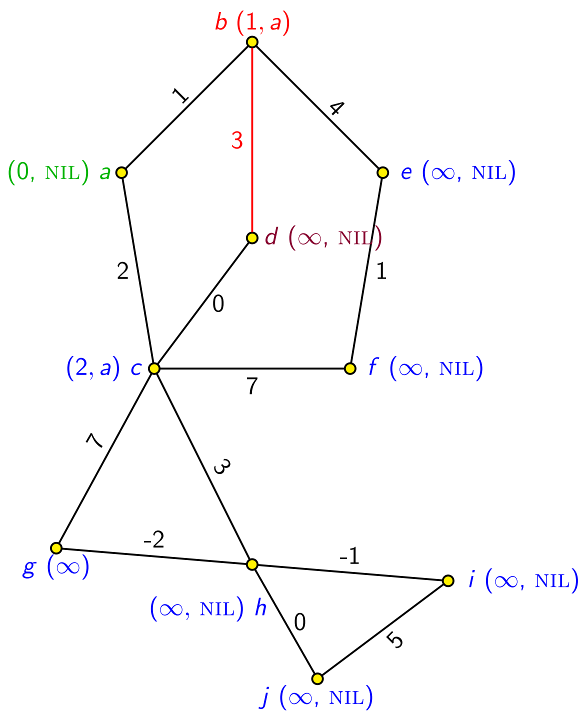


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

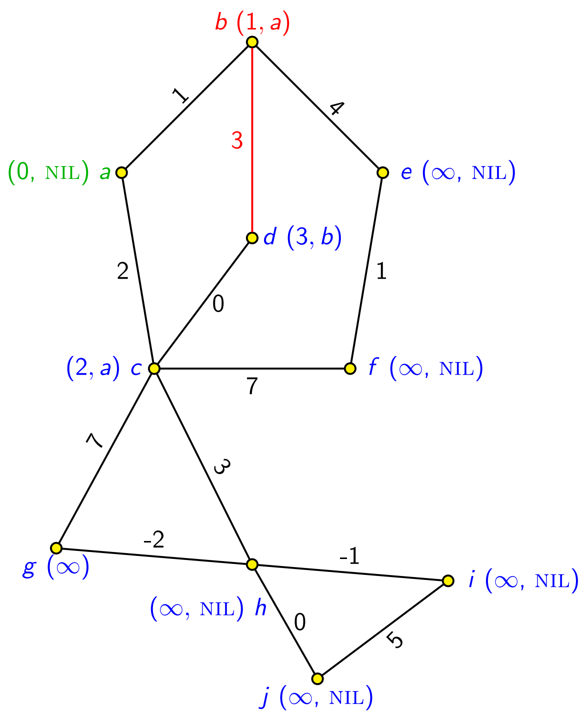


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

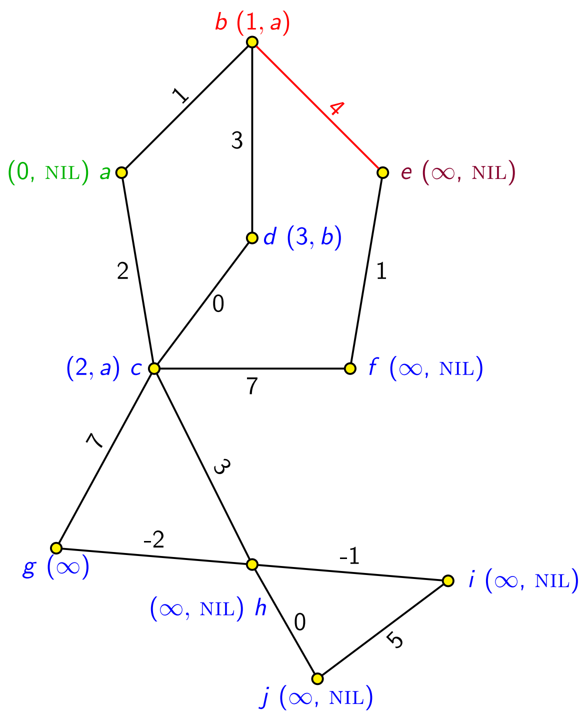


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

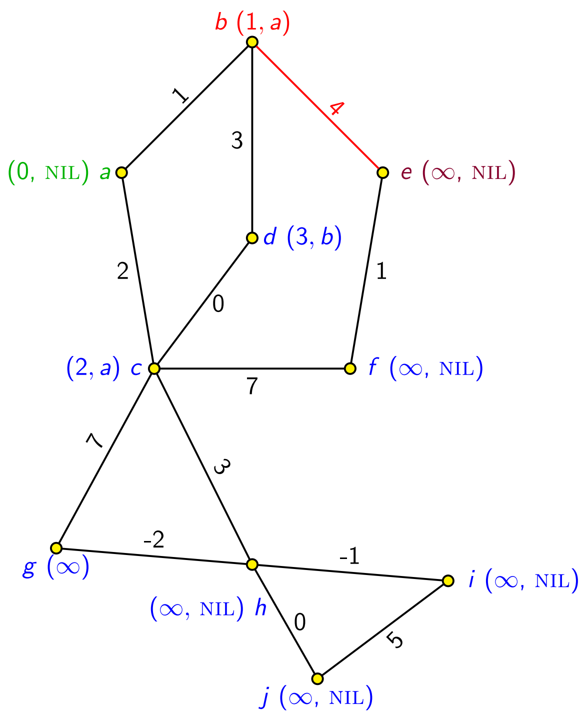


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

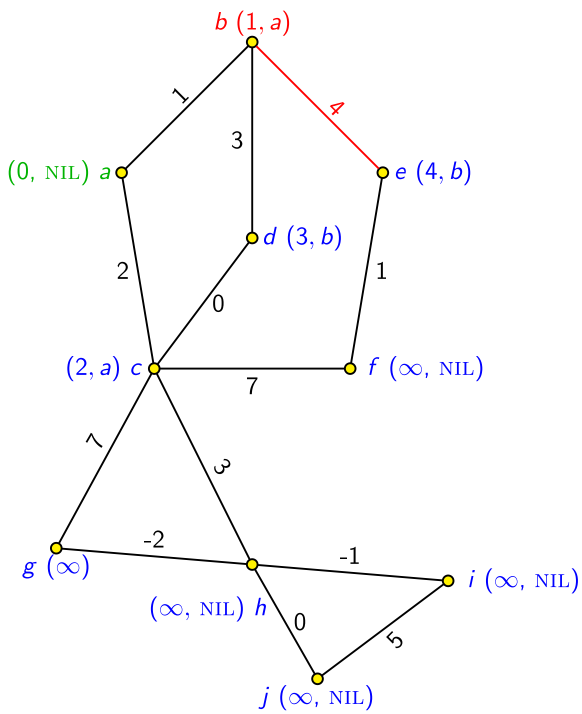


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

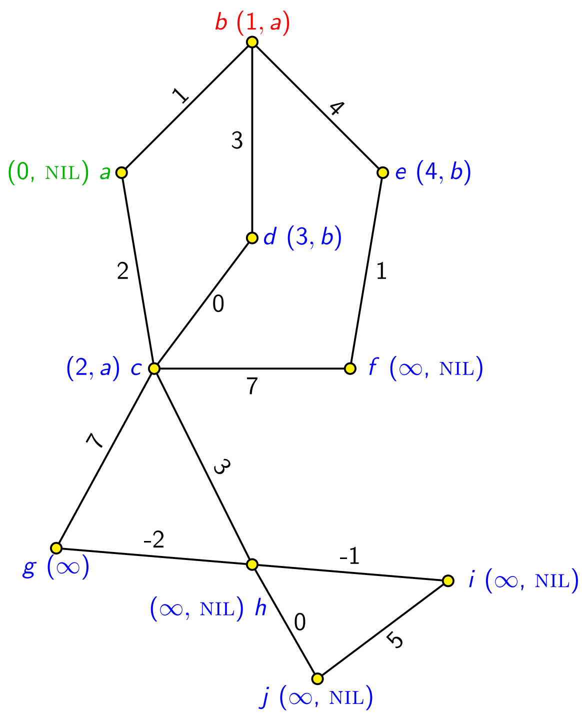


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow \text{NIL}$
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow \text{EXTRACT-MIN}(Q)$
8. **for** each $v \in Adj[u]$
9. **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\pi[v] \leftarrow u$
11. $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

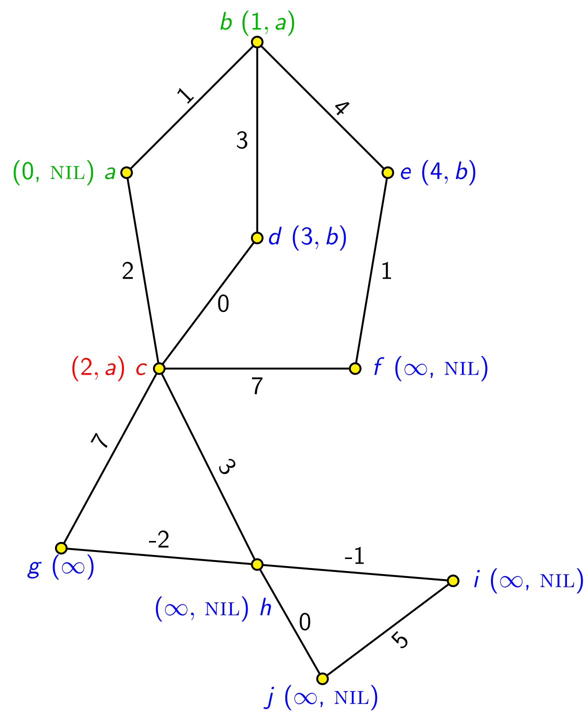


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

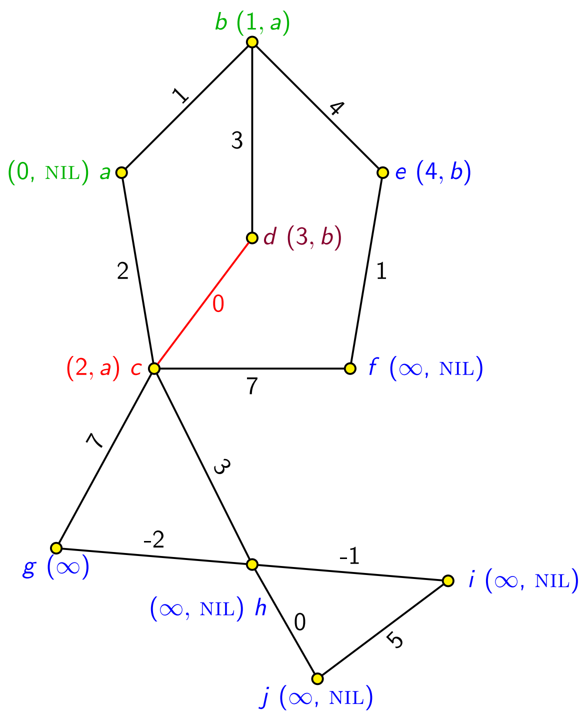


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

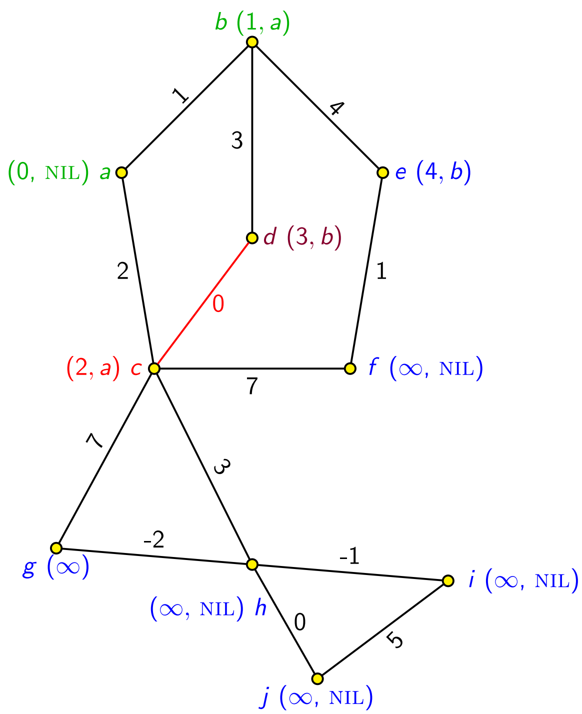


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow$ EXTRACT-MIN(Q)
8. **for** each $v \in Adj[u]$
9. **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\pi[v] \leftarrow u$
11. $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

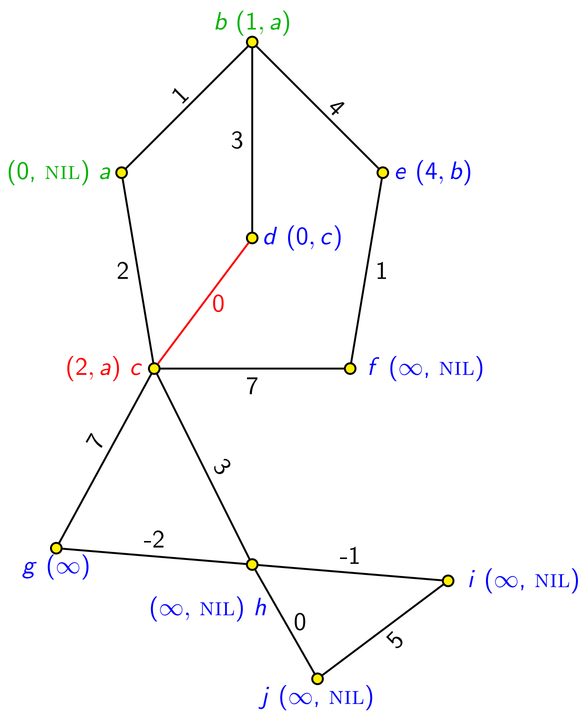


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow \text{NIL}$
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow \text{EXTRACT-MIN}(Q)$
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

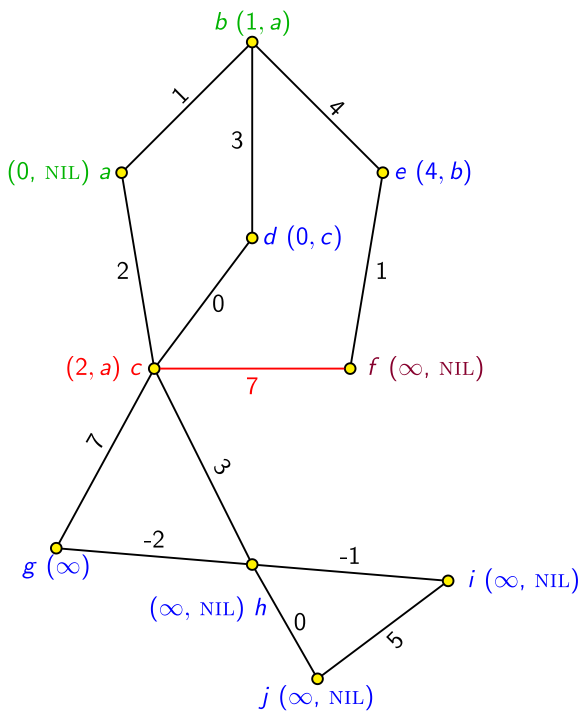


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow \text{NIL}$
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow \text{EXTRACT-MIN}(Q)$
8. **for** each $v \in Adj[u]$
9. **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\pi[v] \leftarrow u$
11. $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

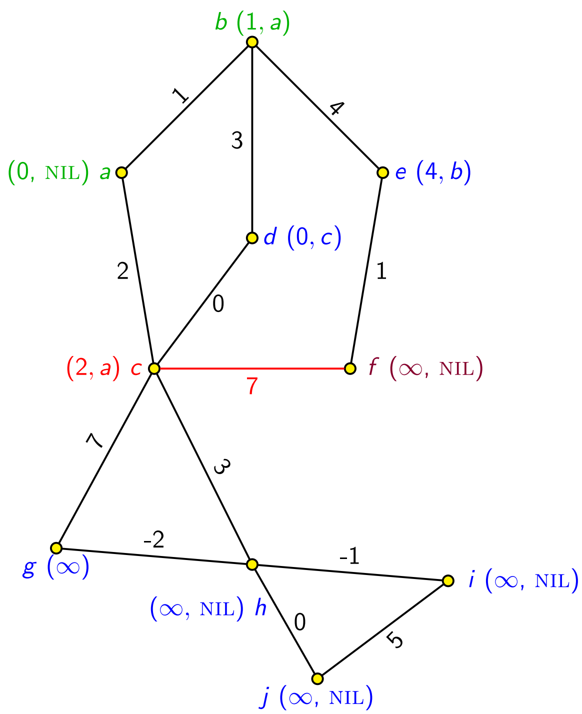


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

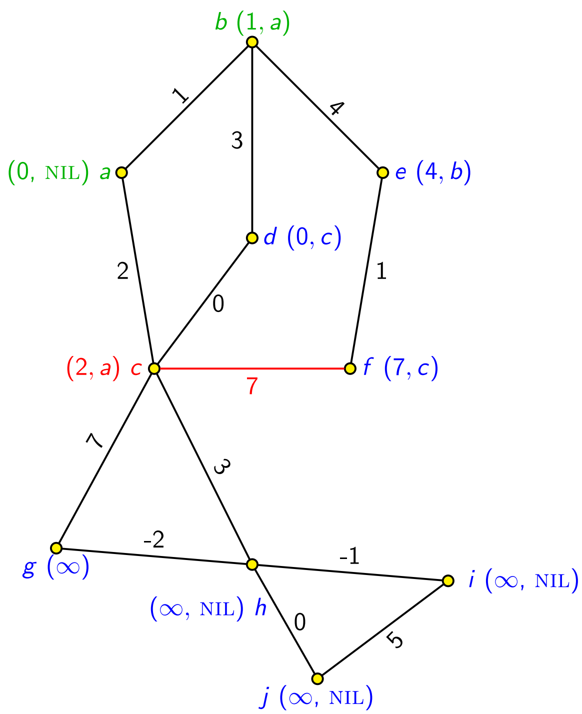


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

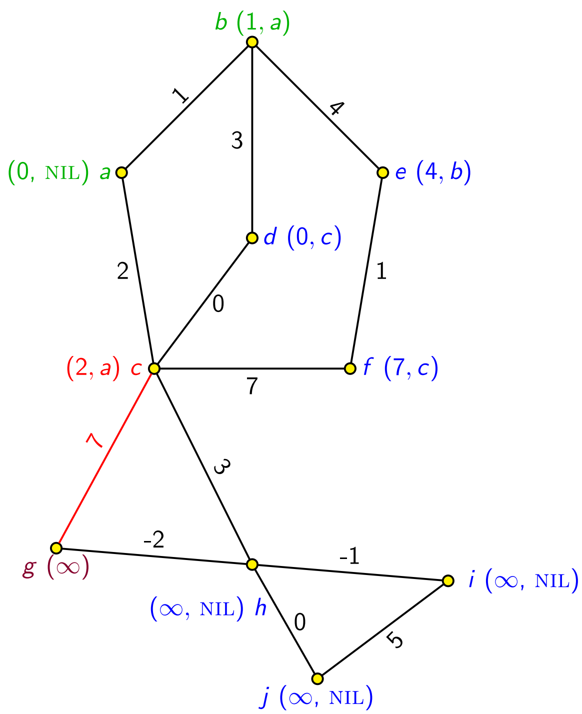


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

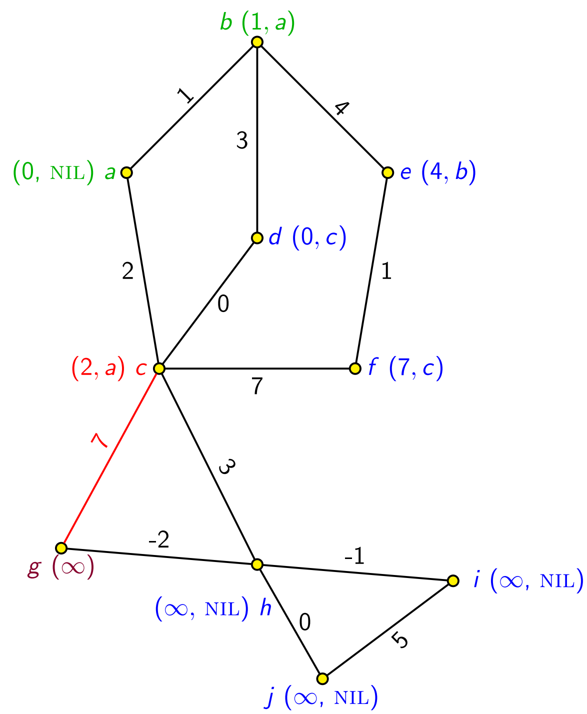


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow \text{EXTRACT-MIN}(Q)$
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

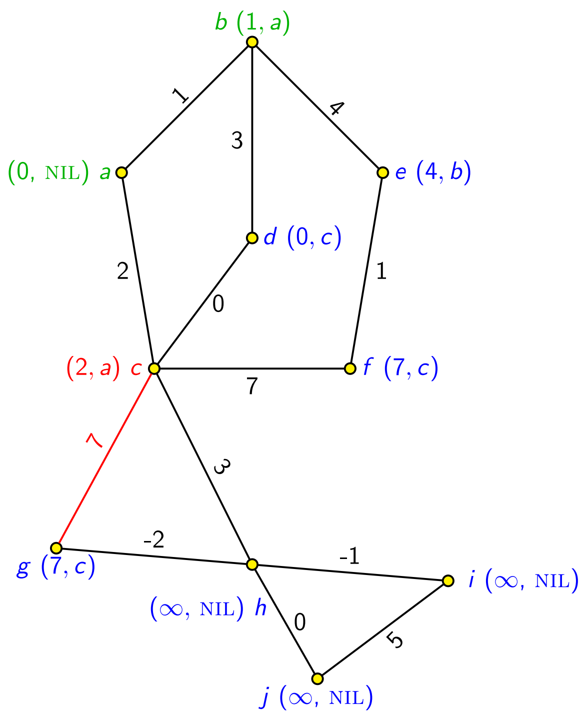


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

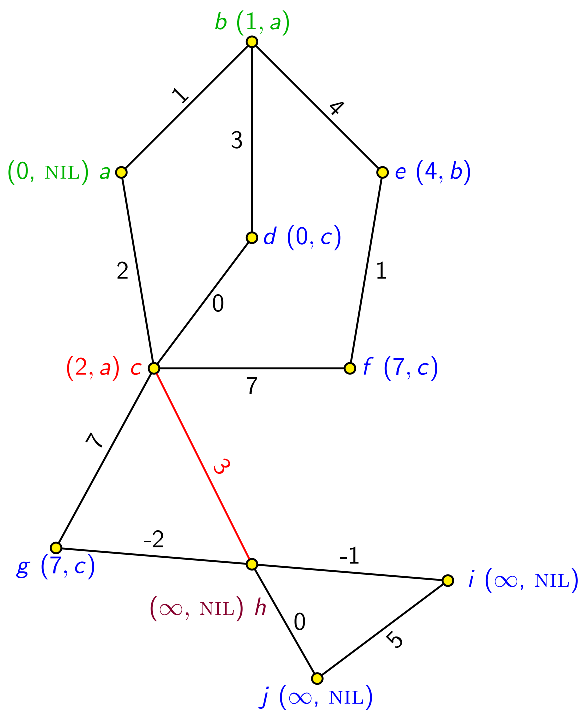


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-Prim(G, w, r)

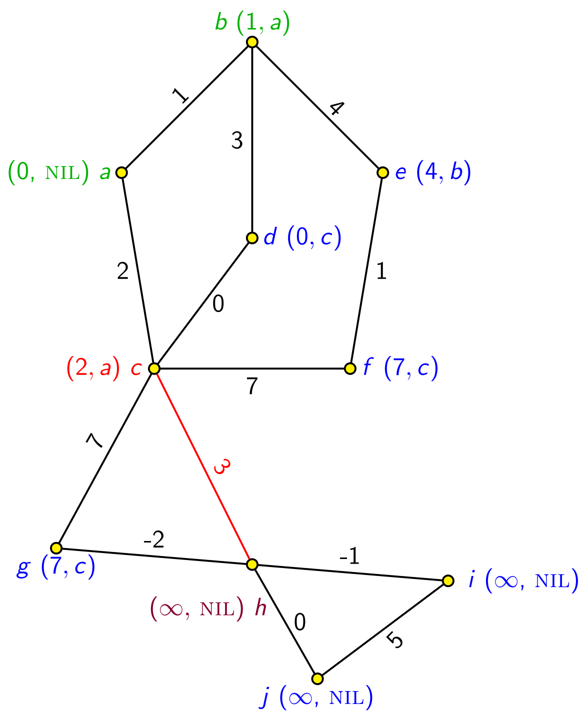


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow \text{Extract-Min}(Q)$
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

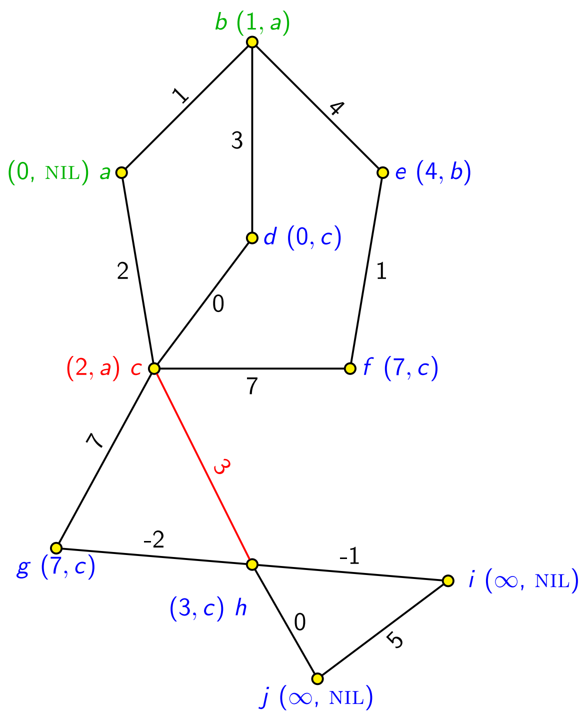


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-Prim(G, w, r)

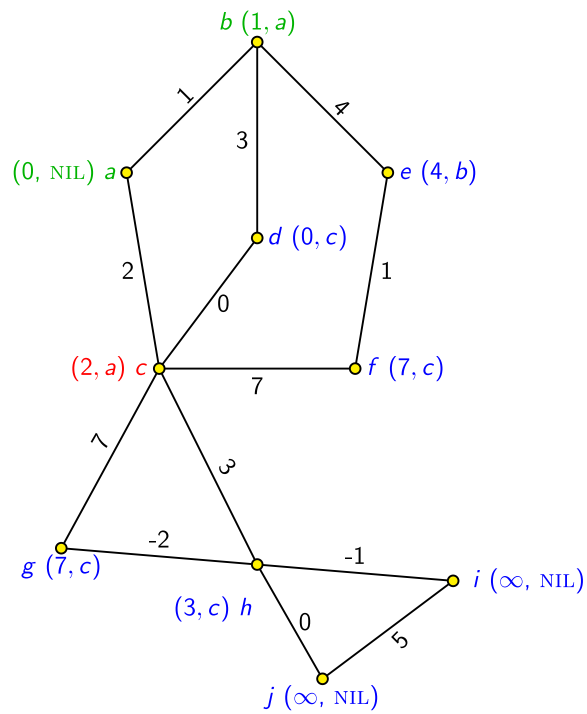


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ Extract-Min(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

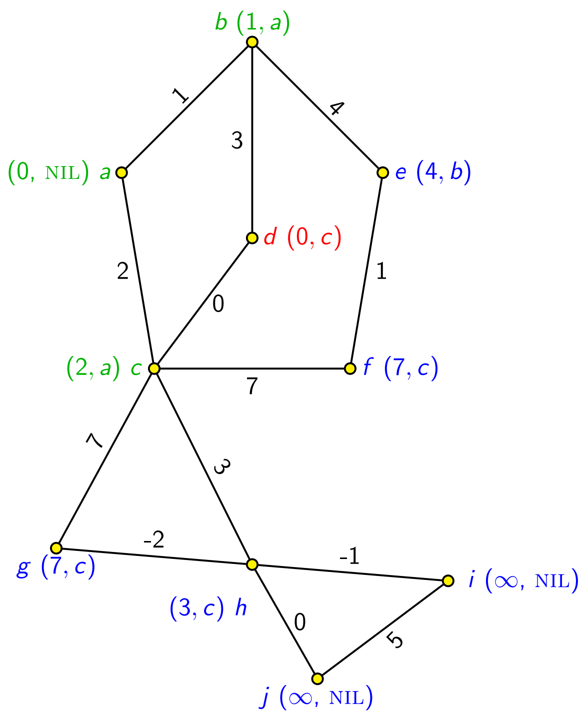


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

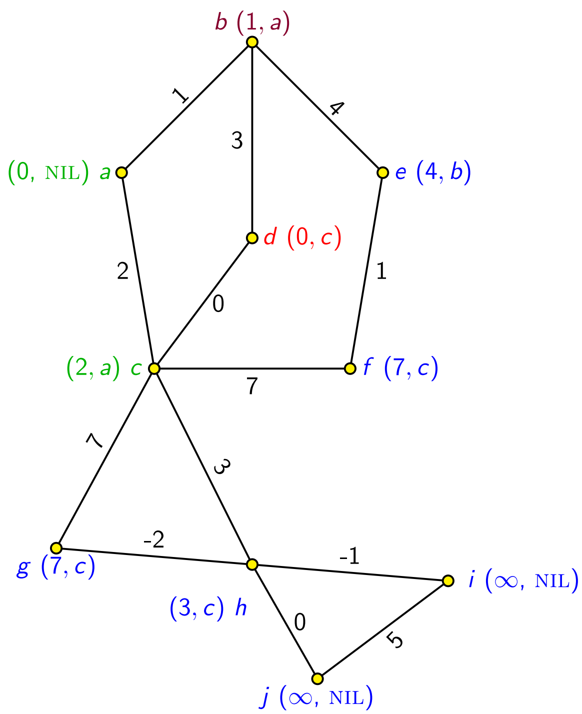


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

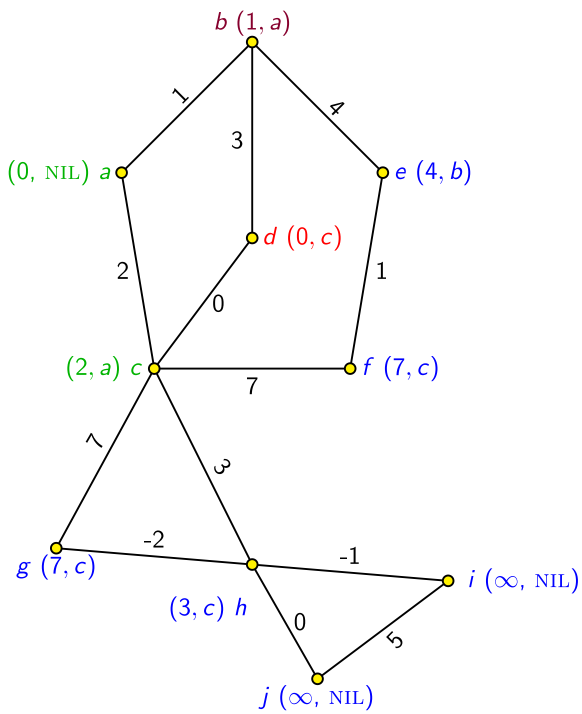


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

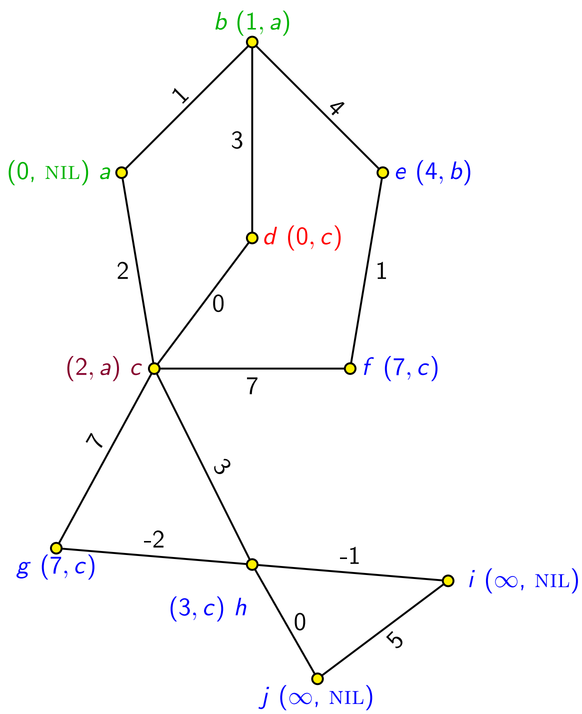


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

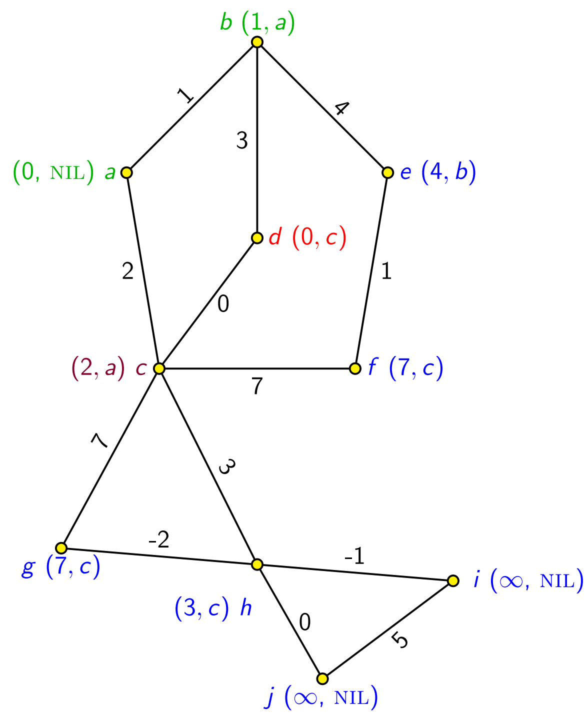


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

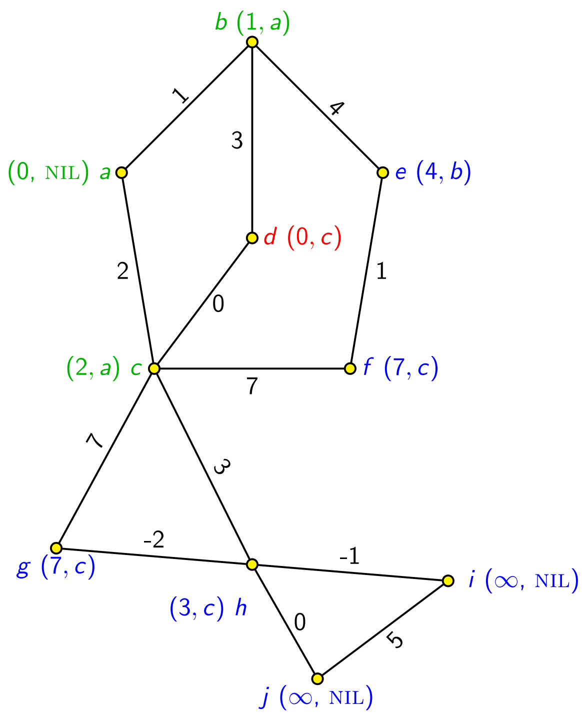


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

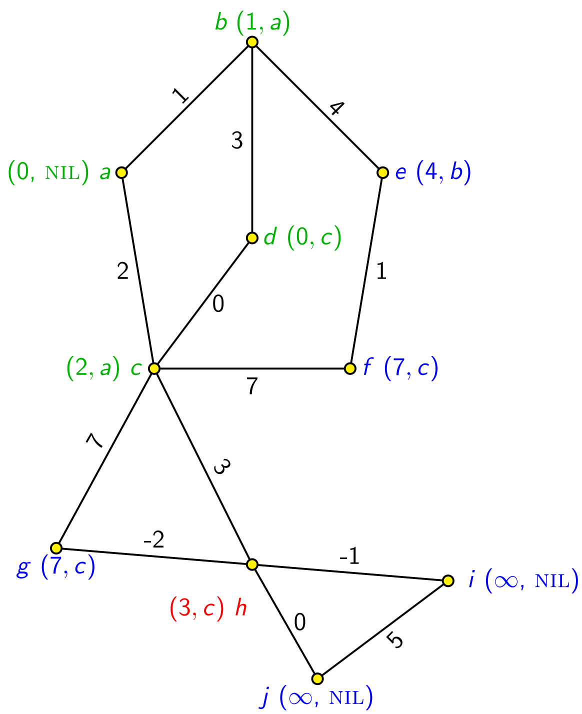


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

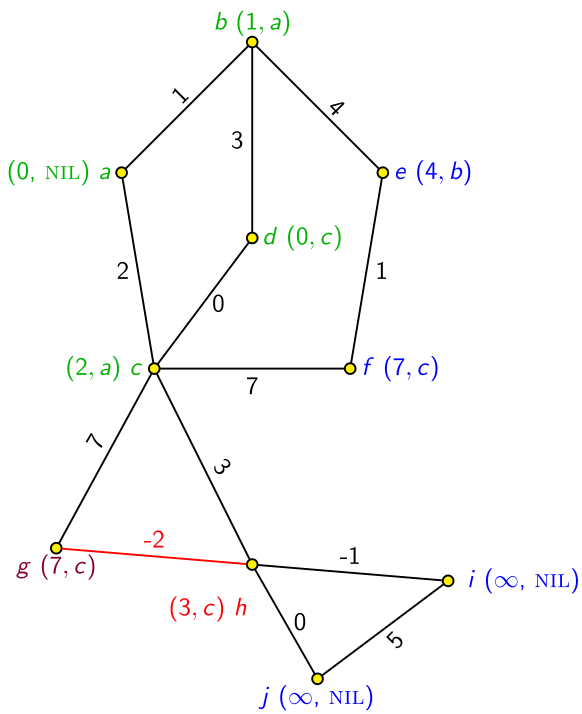


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow$ EXTRACT-MIN(Q)
8. **for** each $v \in Adj[u]$
9. **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\pi[v] \leftarrow u$
11. $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

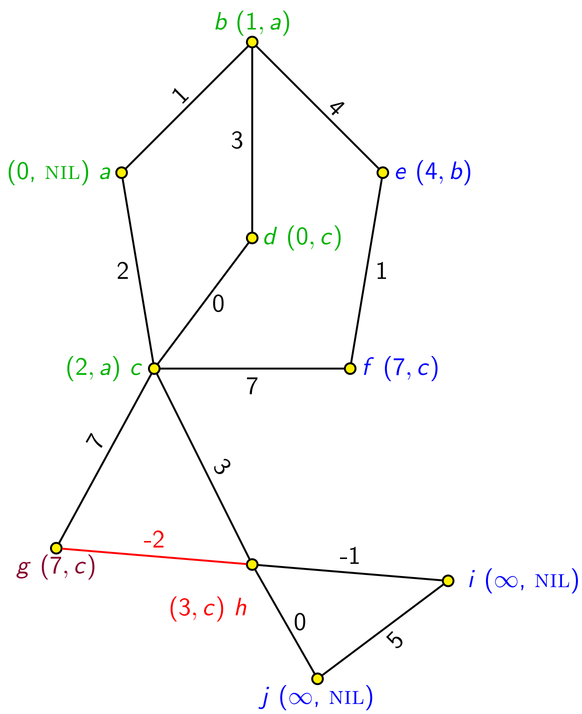


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

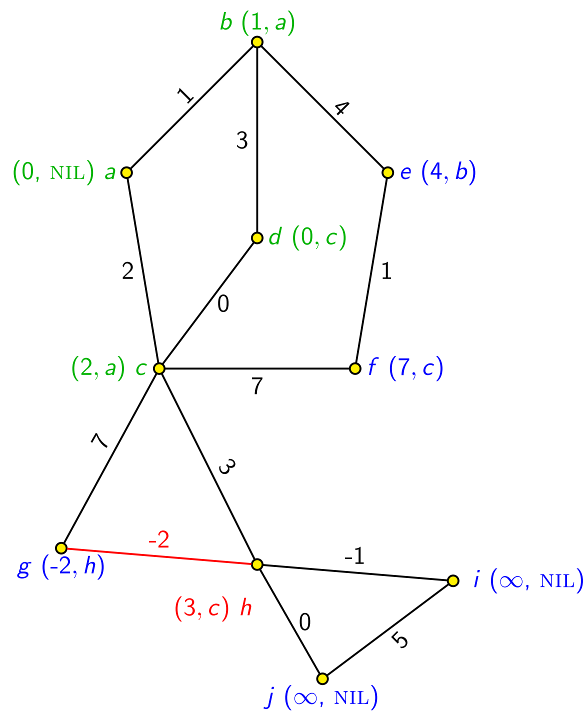


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow \text{EXTRACT-MIN}(Q)$
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

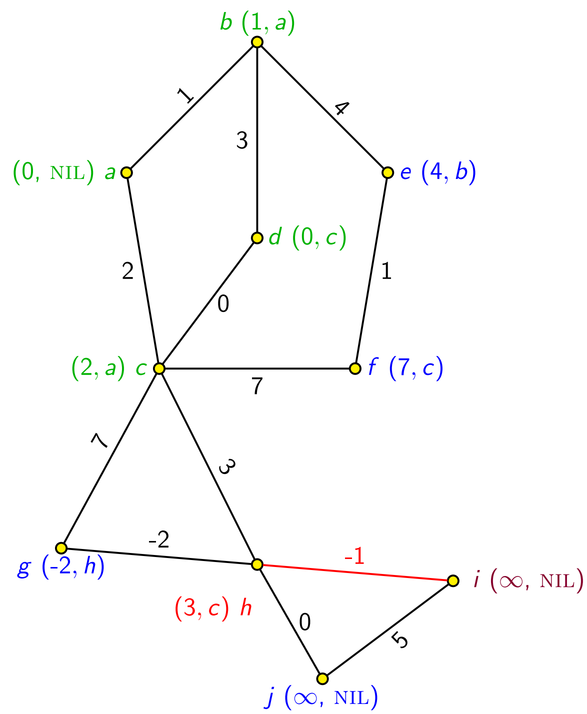


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

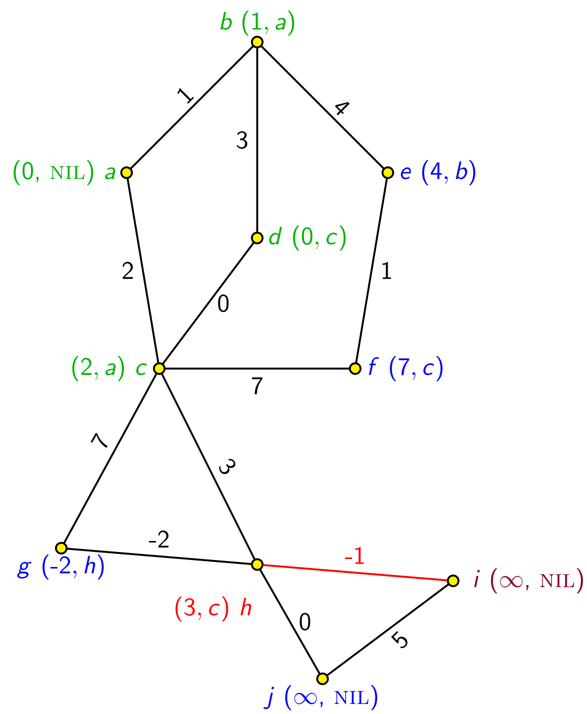


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow$ EXTRACT-MIN(Q)
8. **for** each $v \in Adj[u]$
9. **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\pi[v] \leftarrow u$
11. $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

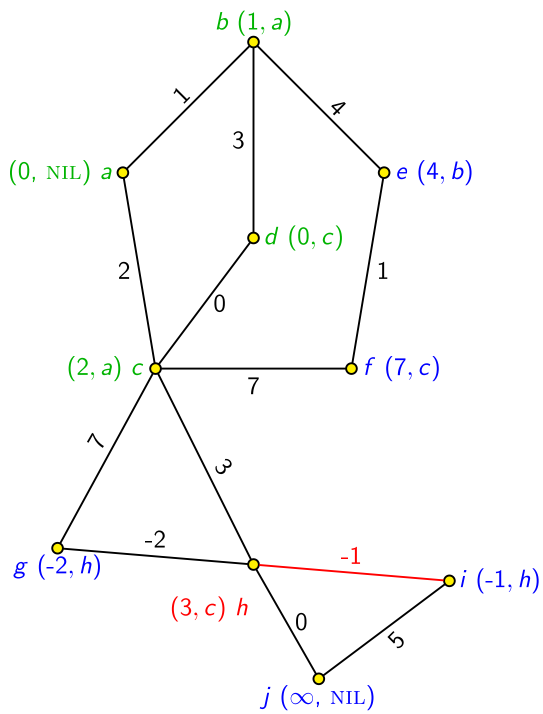


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

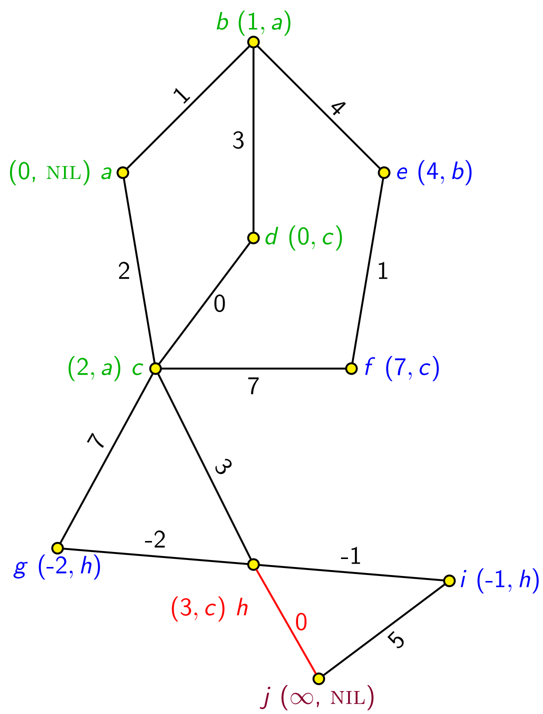


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

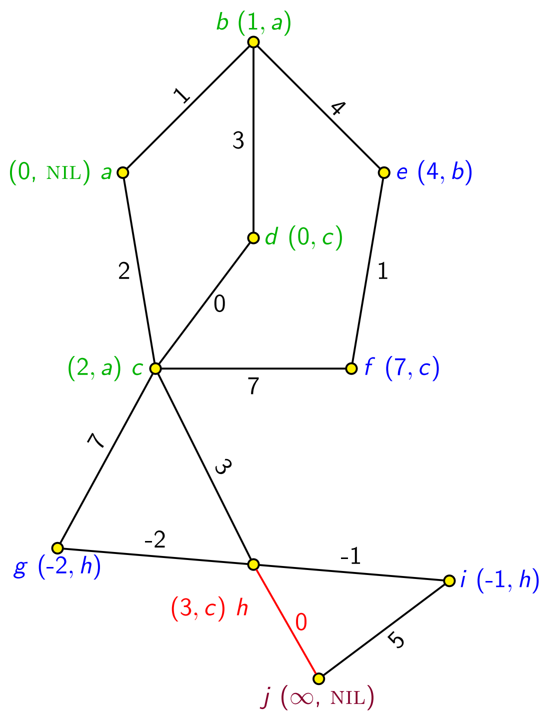


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

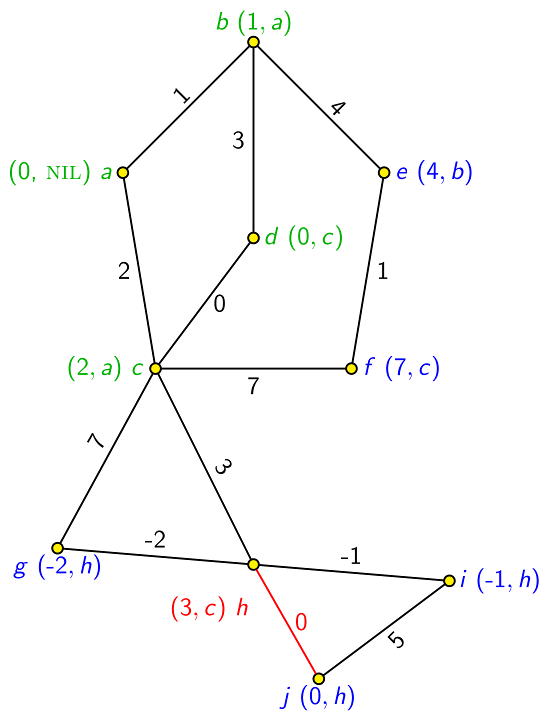


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

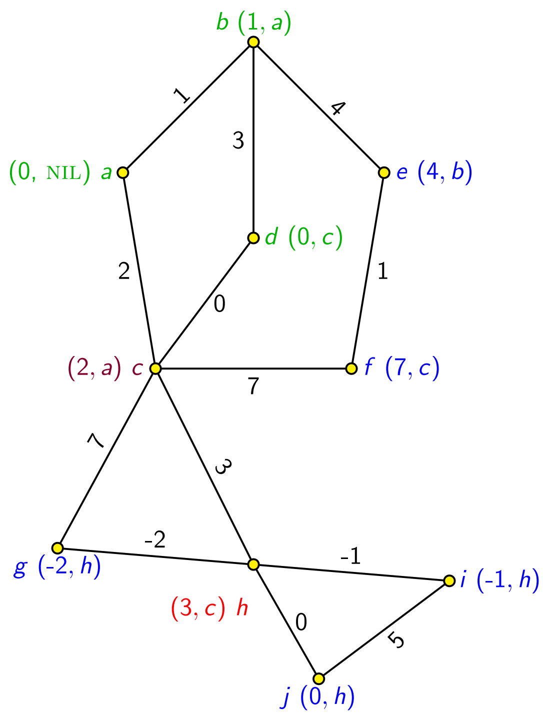


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

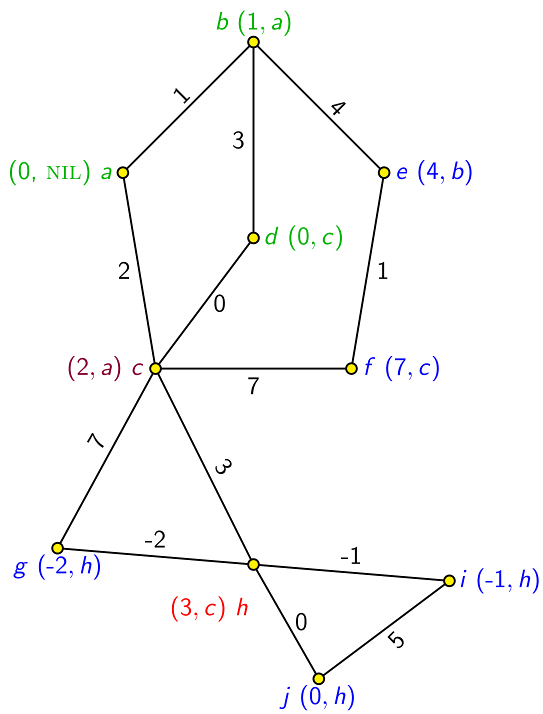


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

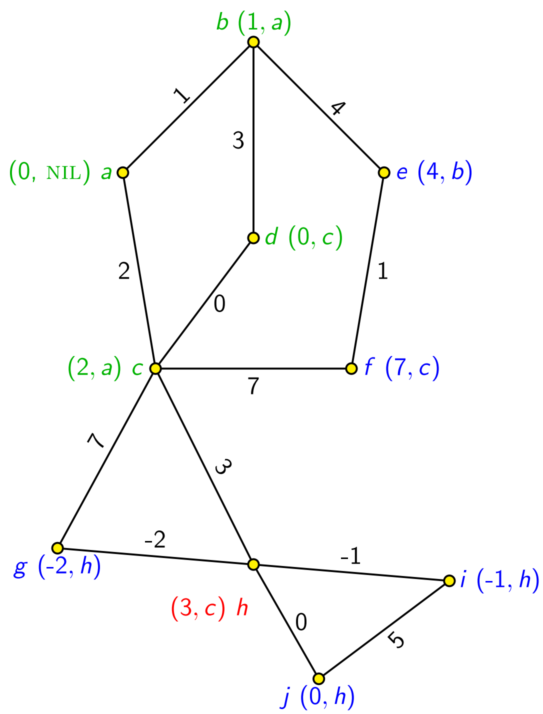


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow \text{NIL}$
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow \text{EXTRACT-MIN}(Q)$
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

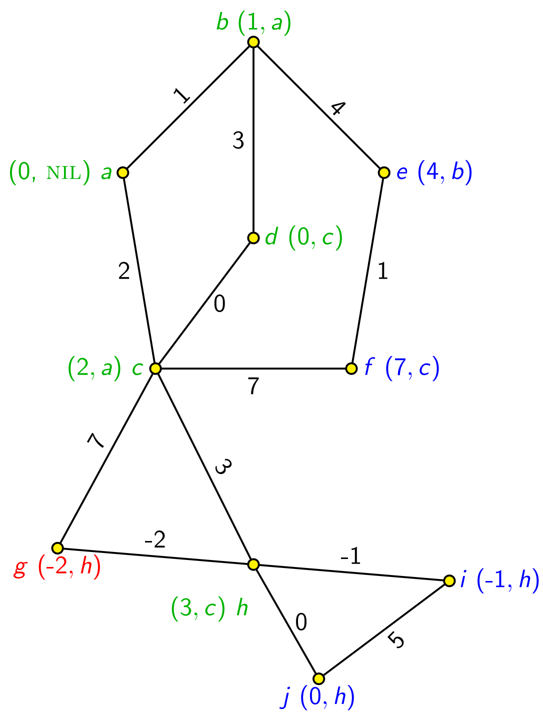


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

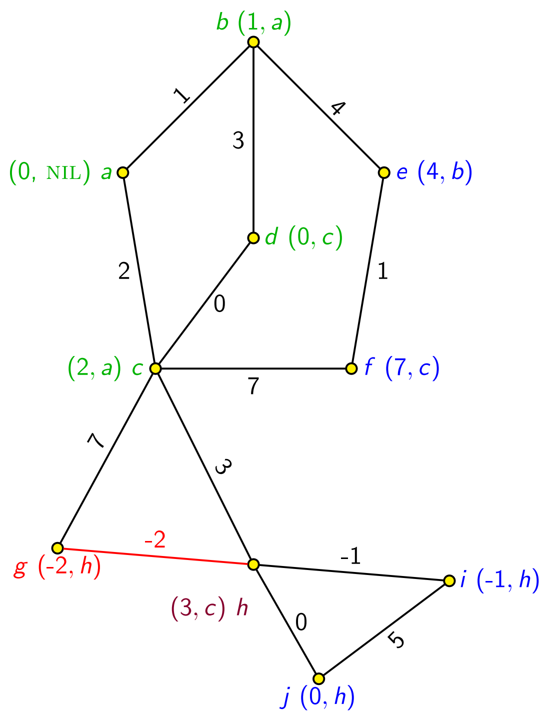


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

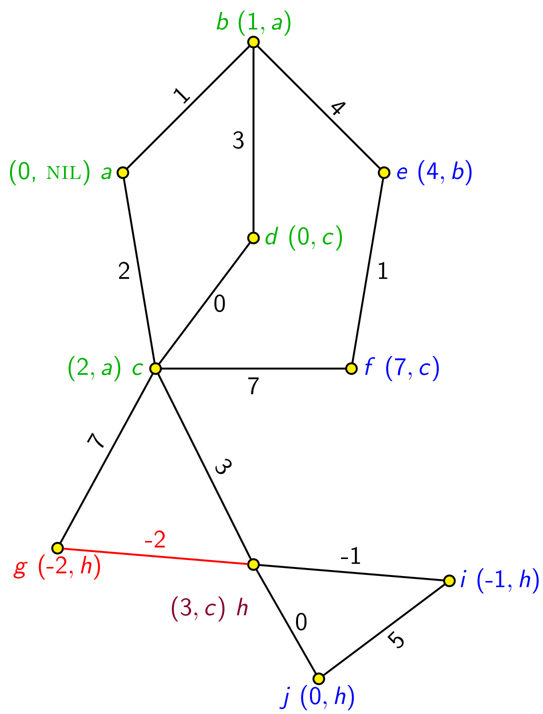


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

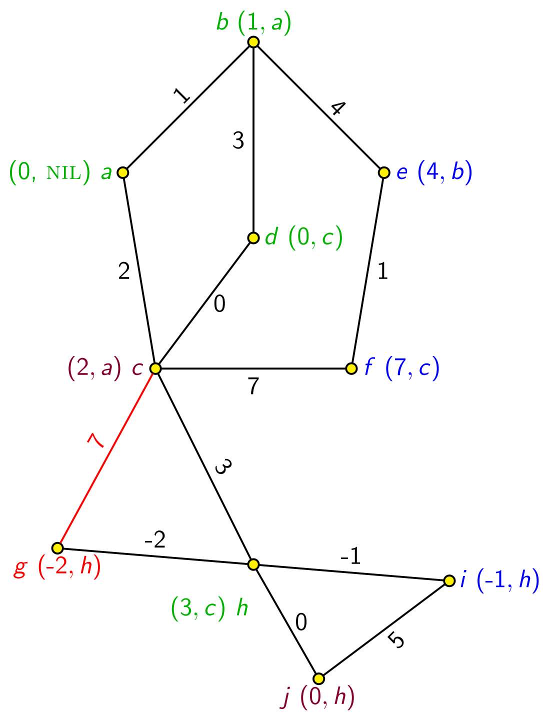


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow \text{EXTRACT-MIN}(Q)$
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

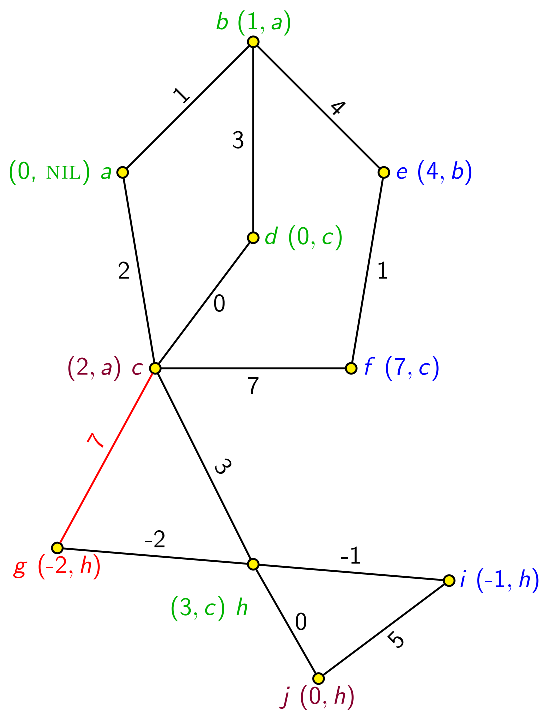


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

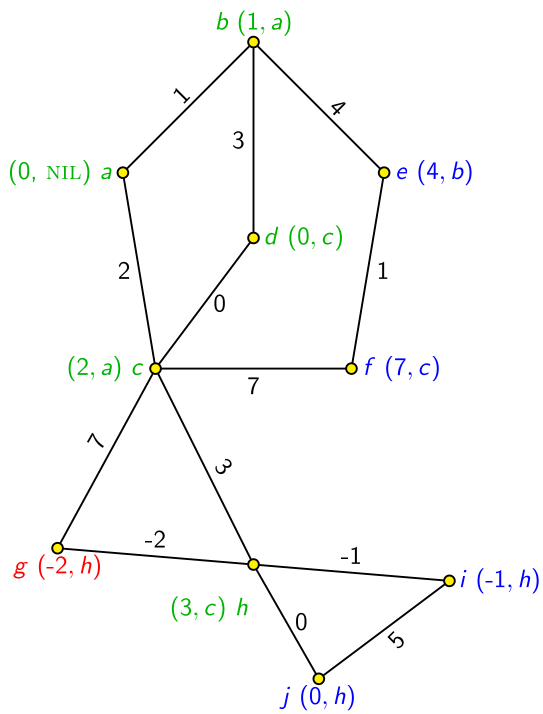


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-Prim(G, w, r)

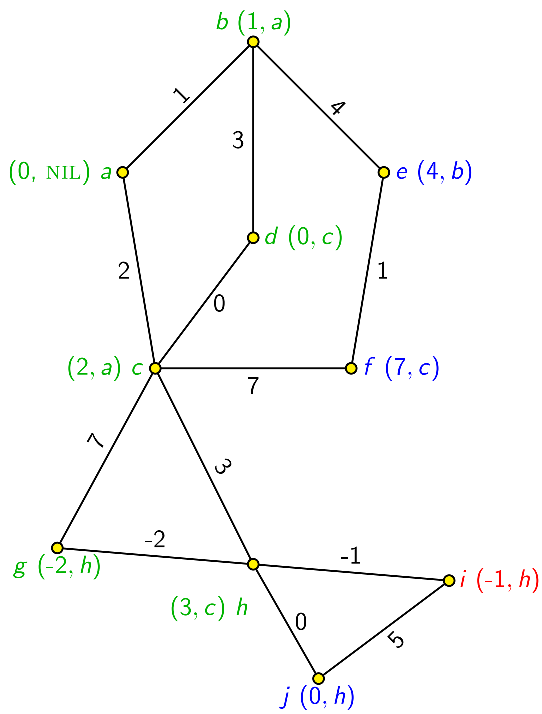


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow \text{Extract-Min}(Q)$
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-Prim(G, w, r)

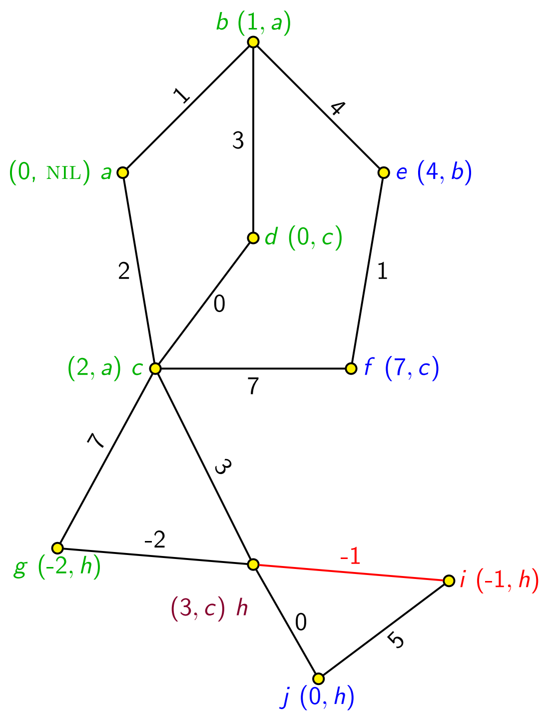


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ Extract-Min(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

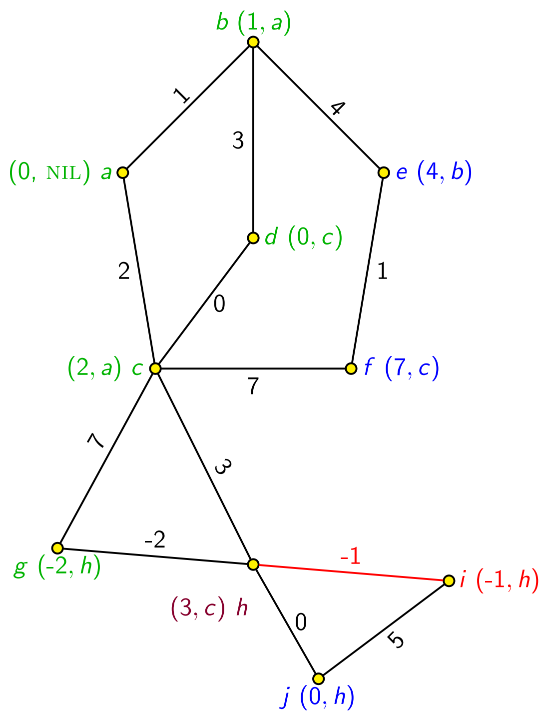


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

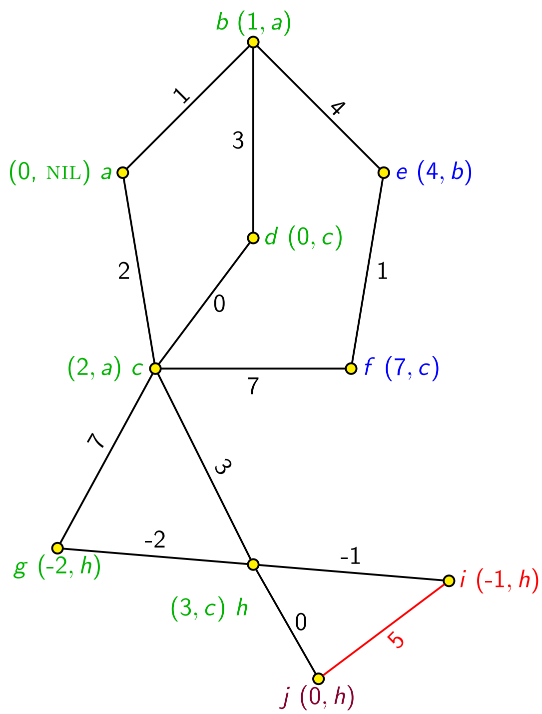


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

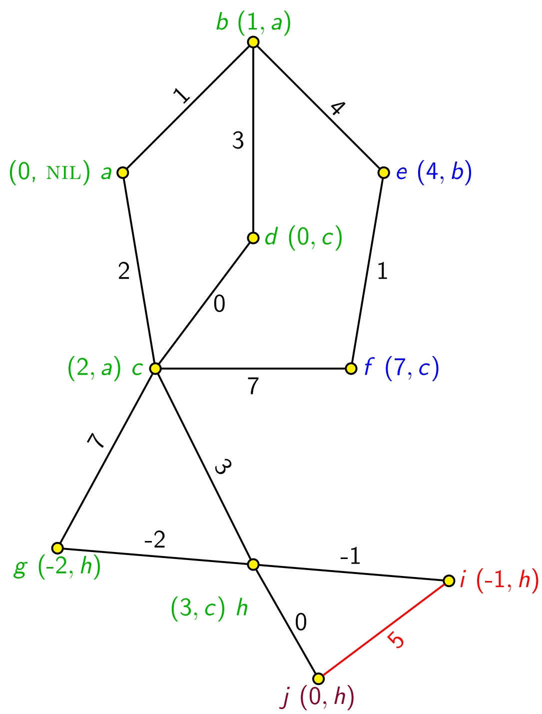


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

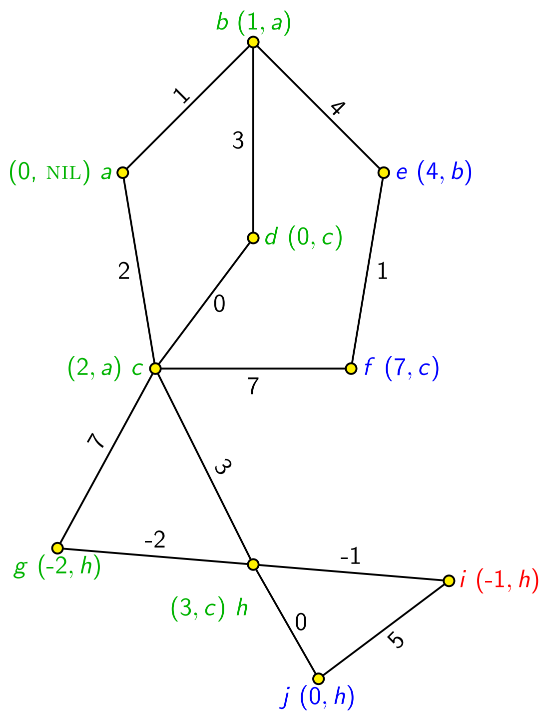


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

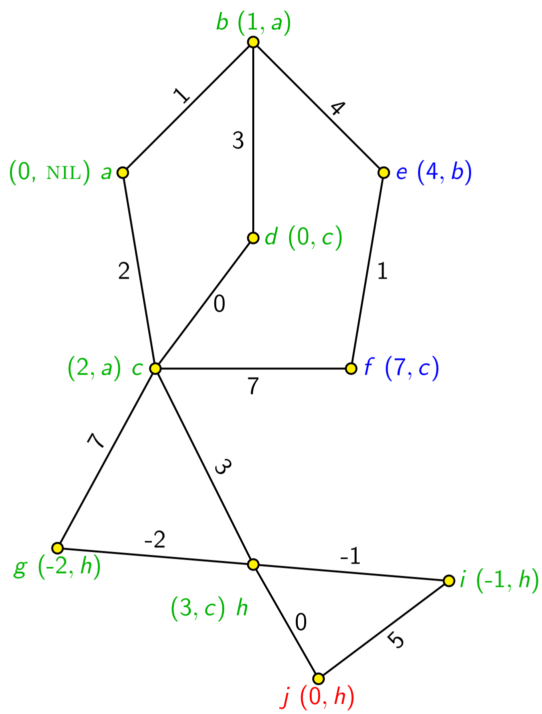


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

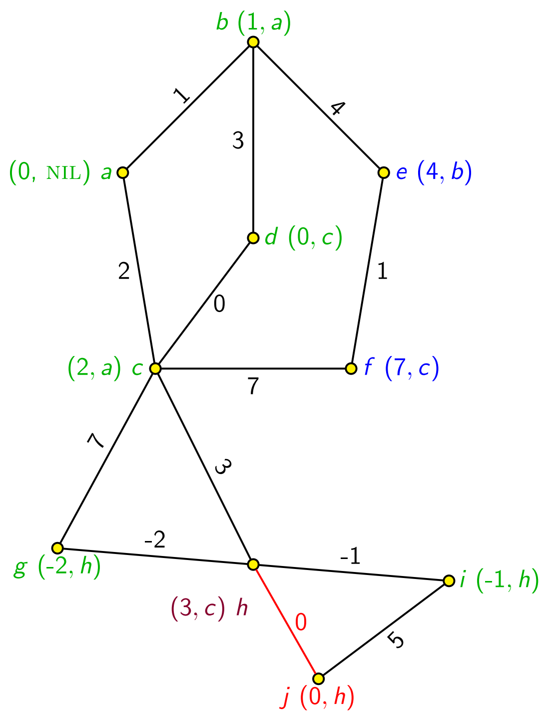


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-Prim(G, w, r)

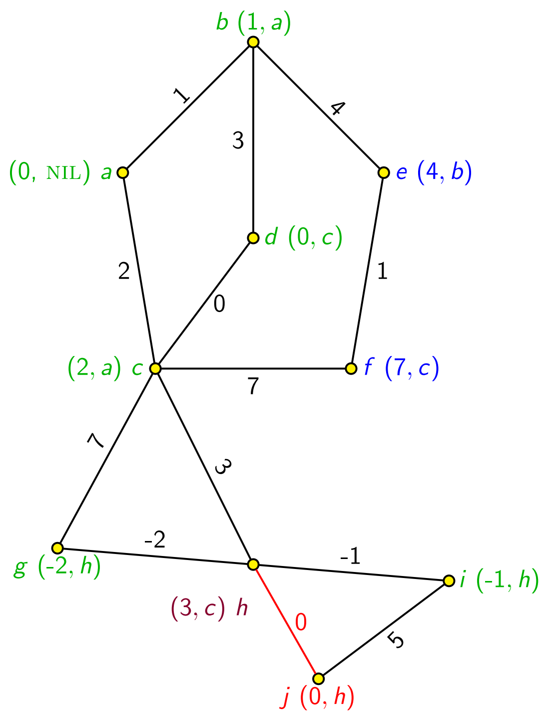


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ Extract-Min(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

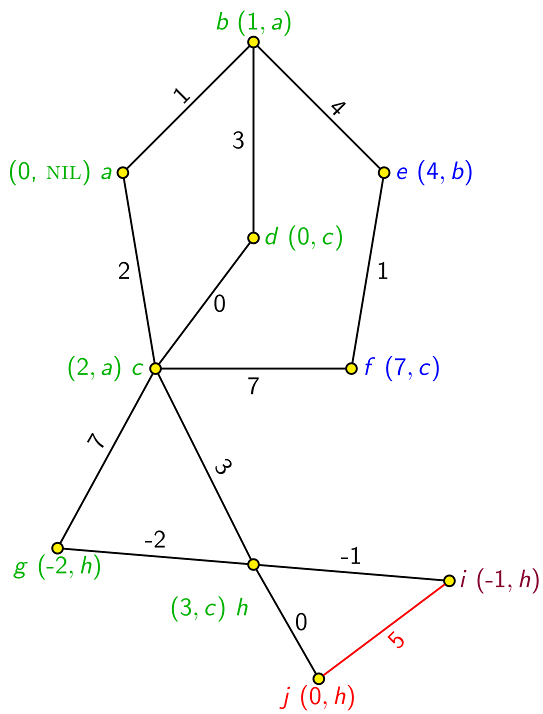


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\qquad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\qquad\quad \pi[v] \leftarrow u$
11. $\qquad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

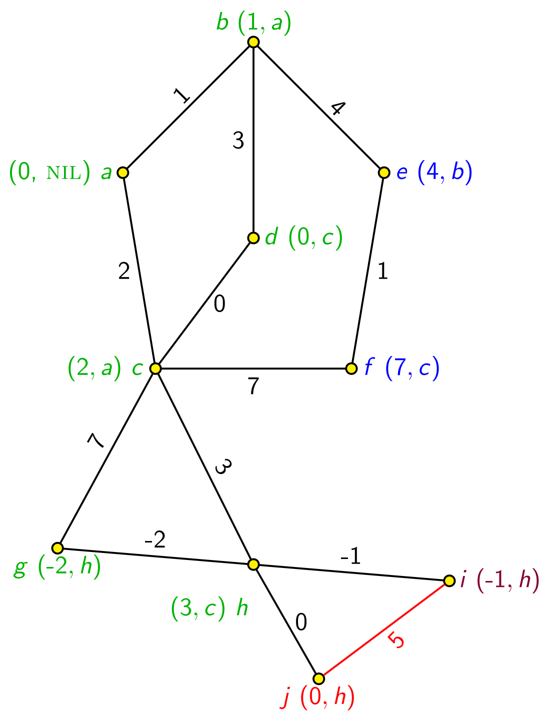


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

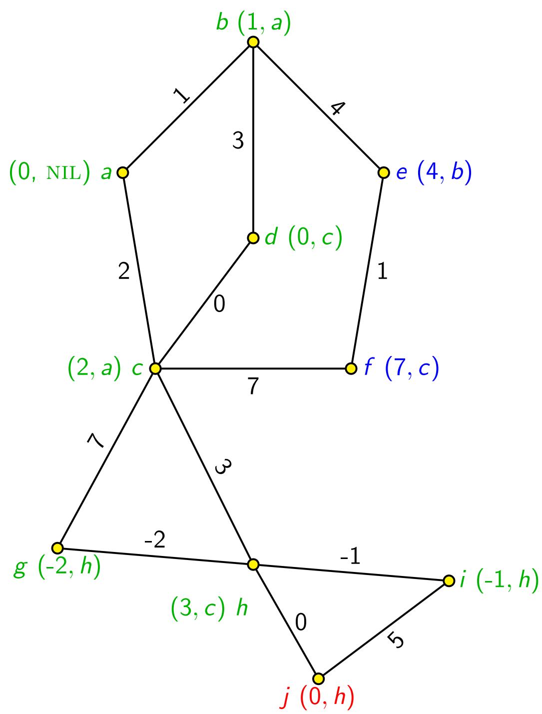


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

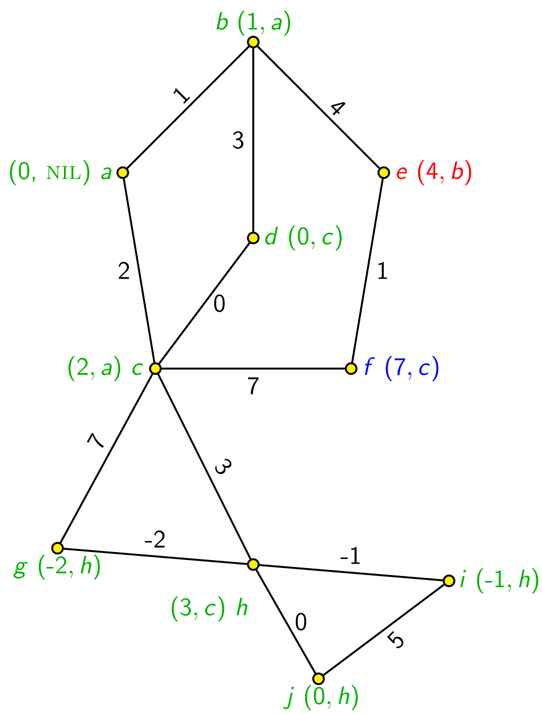


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

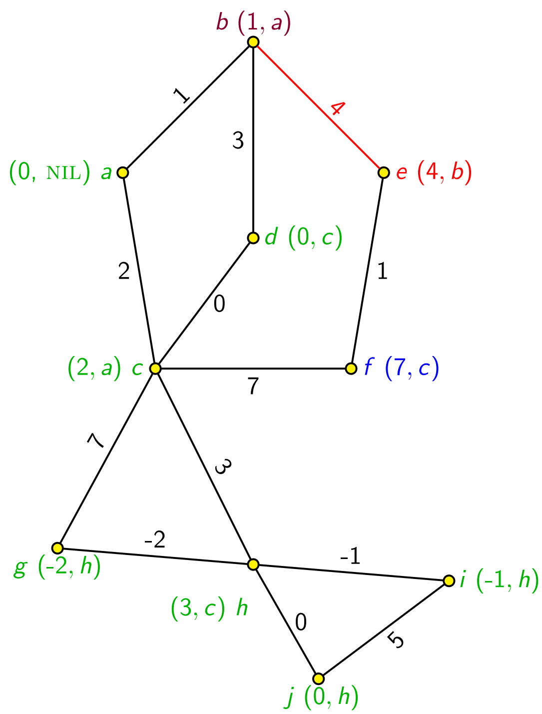


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow \text{EXTRACT-MIN}(Q)$
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-Prim(G, w, r)

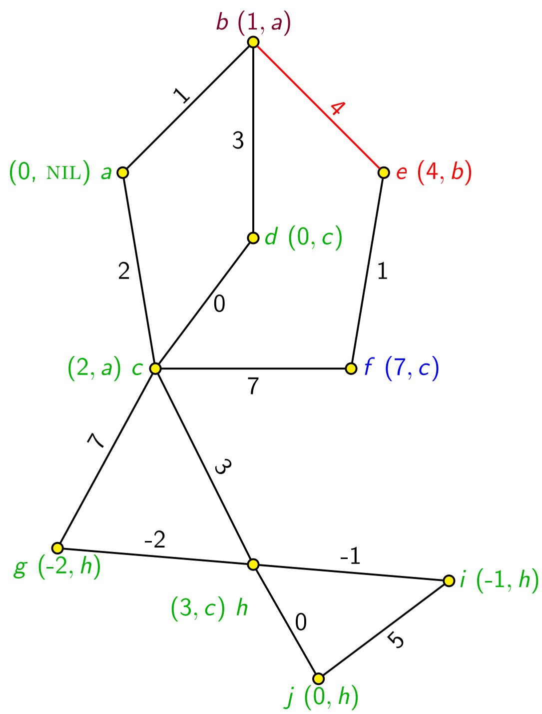


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ Extract-Min(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

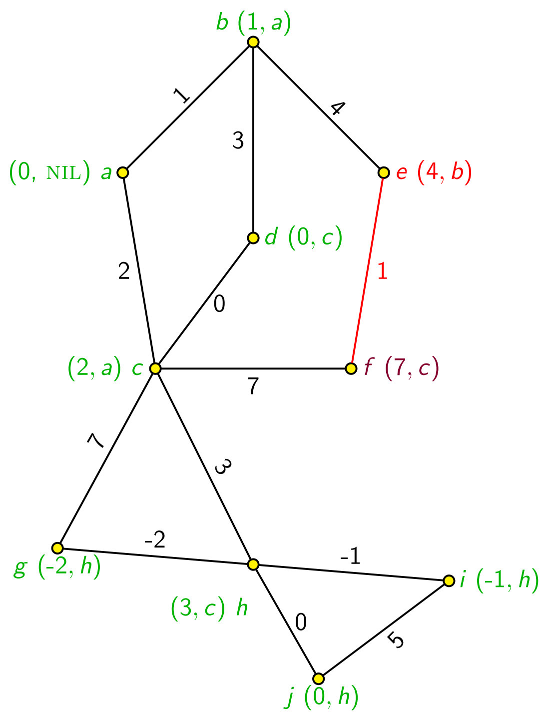


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

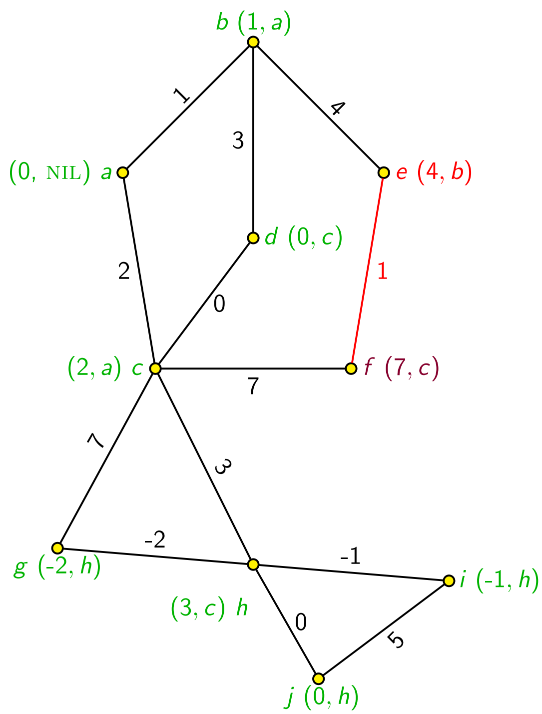


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
    $key[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
    $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
        if $(v \in Q)$ and $(w(u, v) < key[v])$
            $\pi[v] \leftarrow u$
            $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

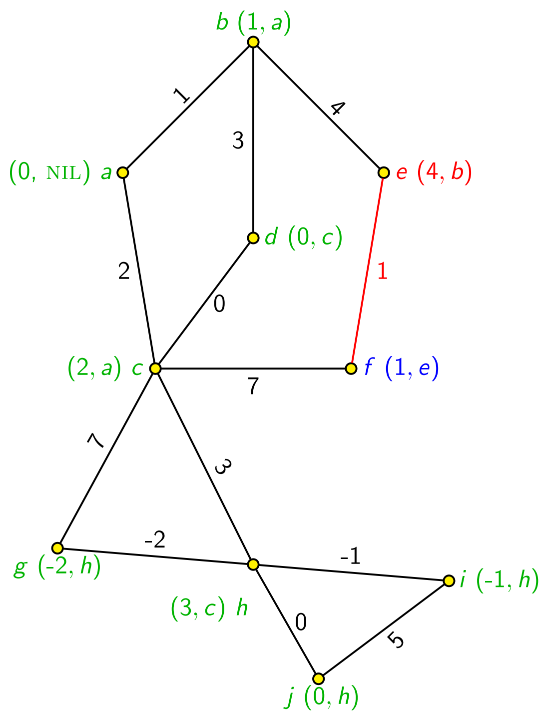


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $key[u] \leftarrow \infty$
3. $\pi[u] \leftarrow \text{NIL}$
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $u \leftarrow \text{EXTRACT-MIN}(Q)$
8. **for** each $v \in Adj[u]$
9. **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\pi[v] \leftarrow u$
11. $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

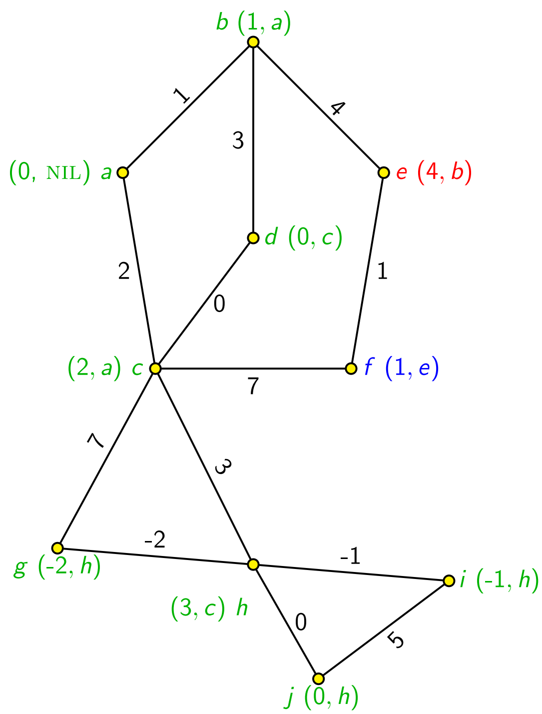


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-Prim(G, w, r)

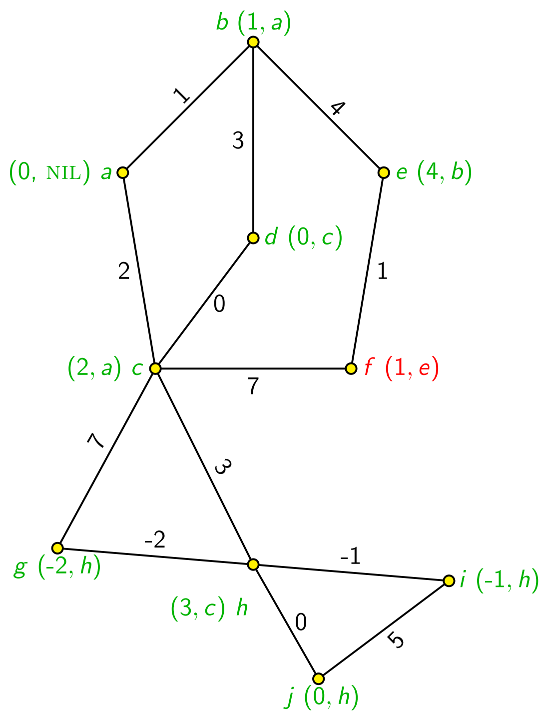


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ Extract-Min(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

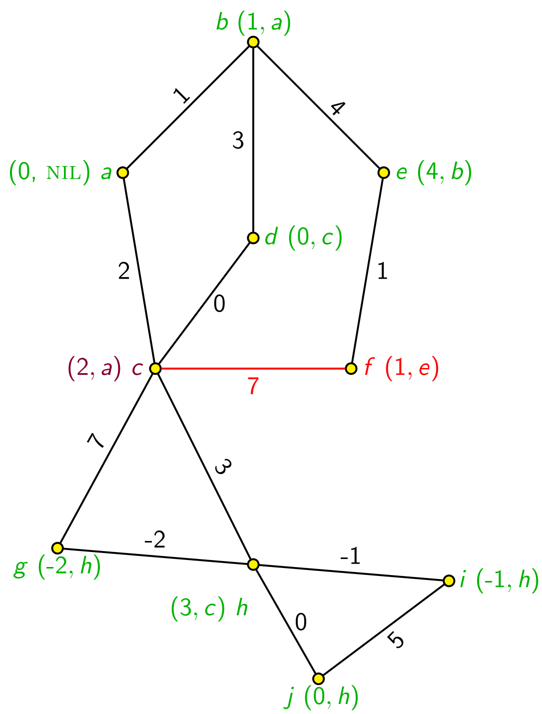


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

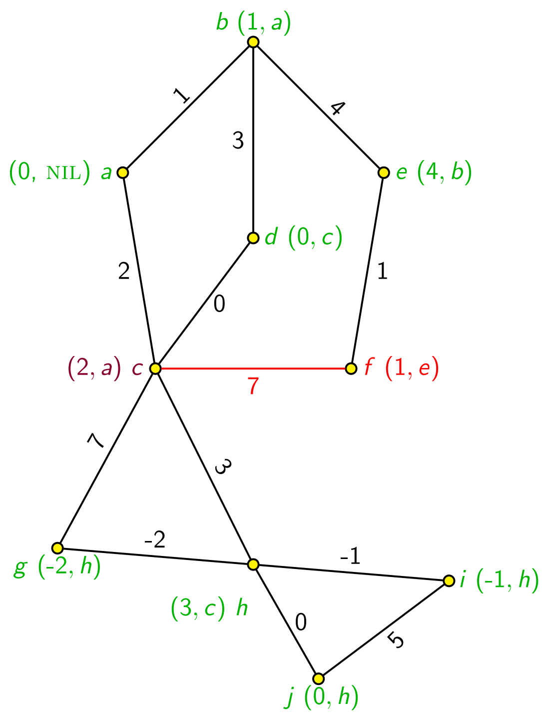


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\qquad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\qquad\quad \pi[v] \leftarrow u$
11. $\qquad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

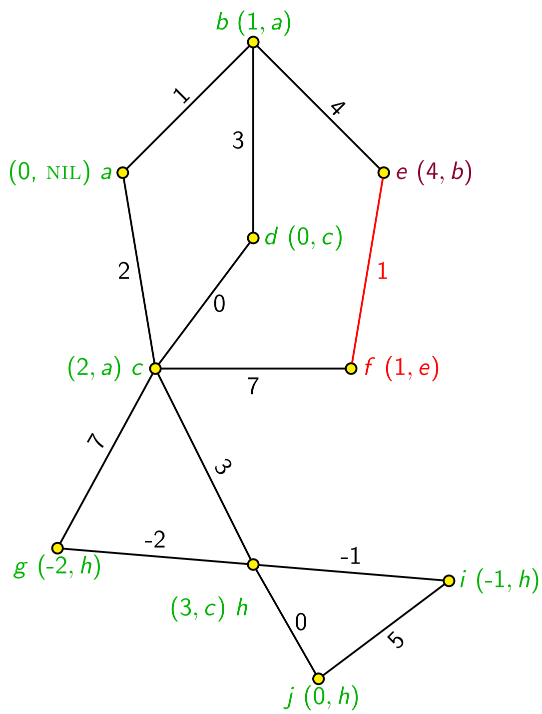


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

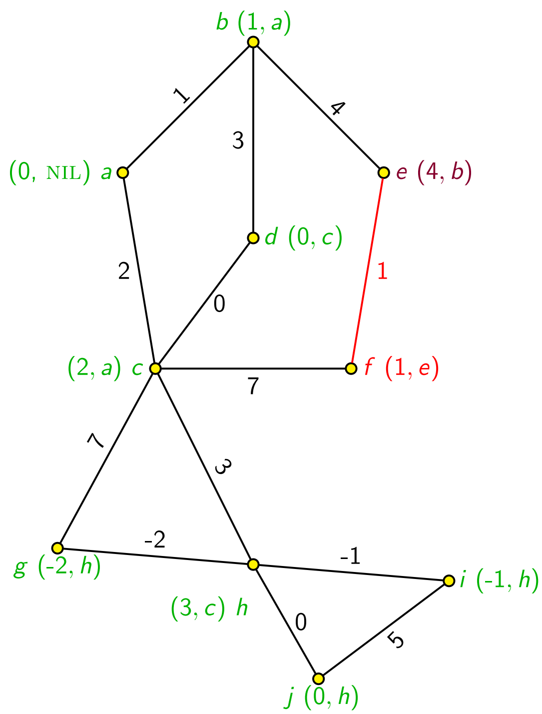


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

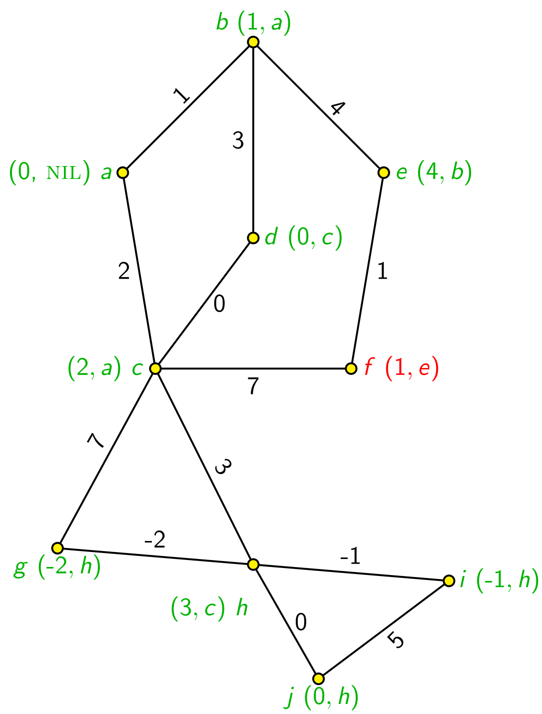


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

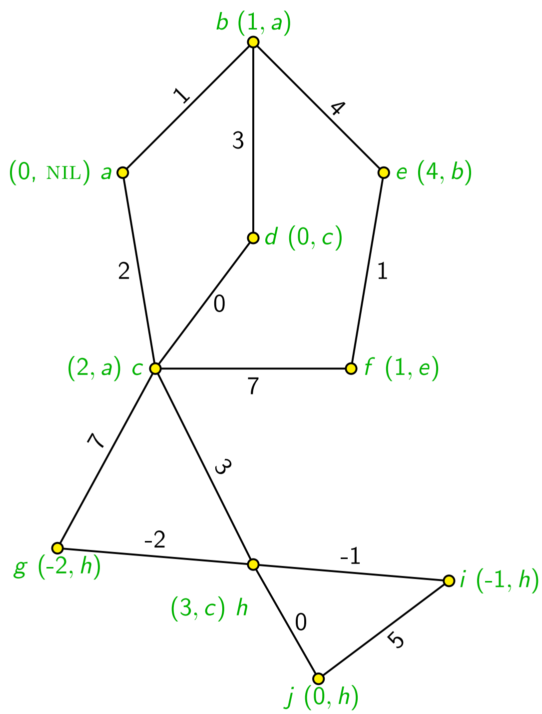


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
for each u ∈ V
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V
while (Q ≠ ∅)
    u ← EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if (v ∈ Q) and (w(u, v) < key[v])
            π[v] ← u
            key[v] ← w(u, v)
```

MST-PRIM(G, w, r)

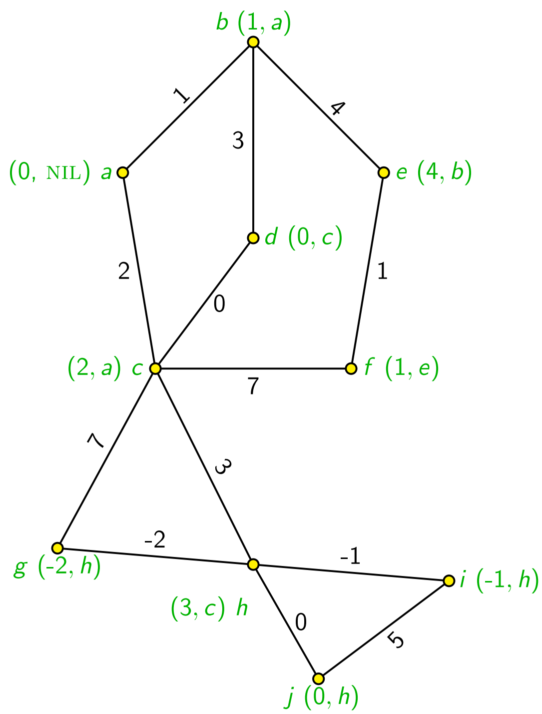


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

for each $u \in V$
  $key[u] \leftarrow \infty$
  $\pi[u] \leftarrow$ NIL
$key[r] \leftarrow 0$
$Q \leftarrow V$
while $(Q \neq \emptyset)$
  $u \leftarrow$ EXTRACT-MIN(Q)
  for each $v \in Adj[u]$
    if $(v \in Q)$ and $(w(u, v) < key[v])$
      $\pi[v] \leftarrow u$
      $key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)

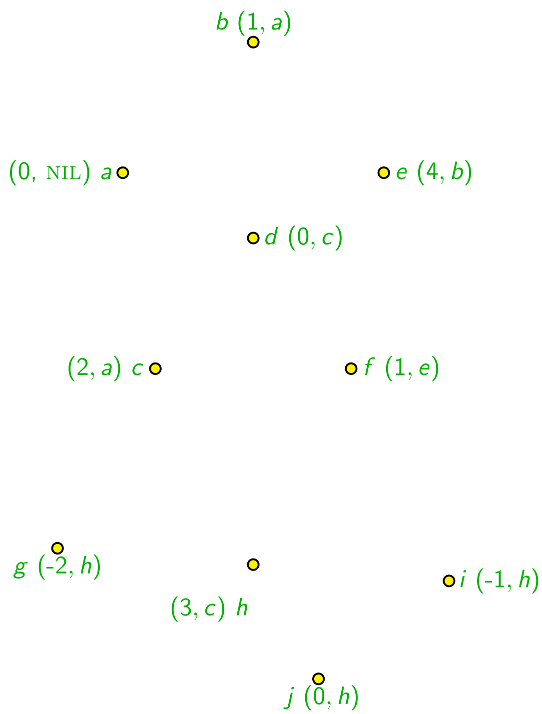


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

1. **for** each $u \in V$
2. $\quad key[u] \leftarrow \infty$
3. $\quad \pi[u] \leftarrow$ NIL
4. $key[r] \leftarrow 0$
5. $Q \leftarrow V$
6. **while** $(Q \neq \emptyset)$
7. $\quad u \leftarrow$ EXTRACT-MIN(Q)
8. $\quad$ **for** each $v \in Adj[u]$
9. $\quad\quad$ **if** $(v \in Q)$ and $(w(u, v) < key[v])$
10. $\quad\quad\quad \pi[v] \leftarrow u$
11. $\quad\quad\quad key[v] \leftarrow w(u, v)$

MST-PRIM(G, w, r)



I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

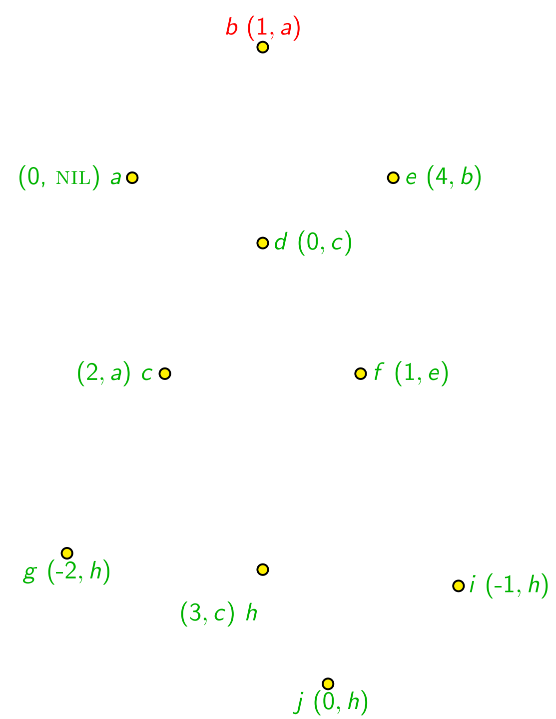
O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T

12. $T \leftarrow \emptyset$

16. **return** T

MST-PRIM(G, w, r)



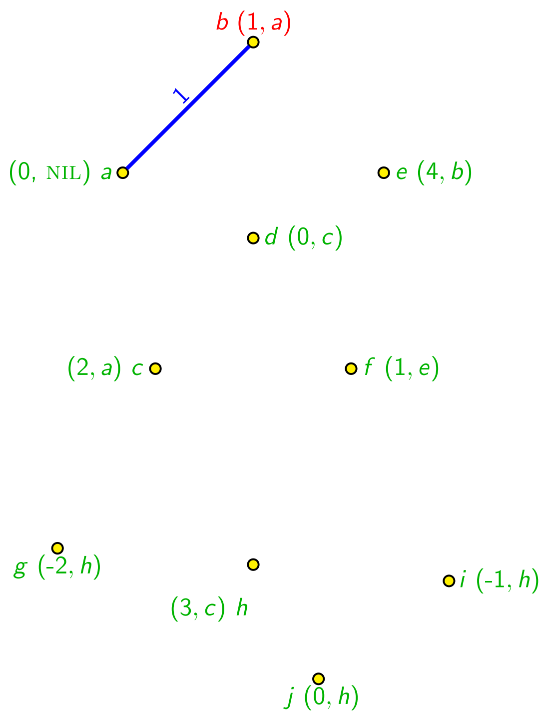
I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
   // Constructing the MST T
   T ← ∅
   for each v ∈ V
      if (v! = r)

   return T
```

MST-PRIM(G, w, r)

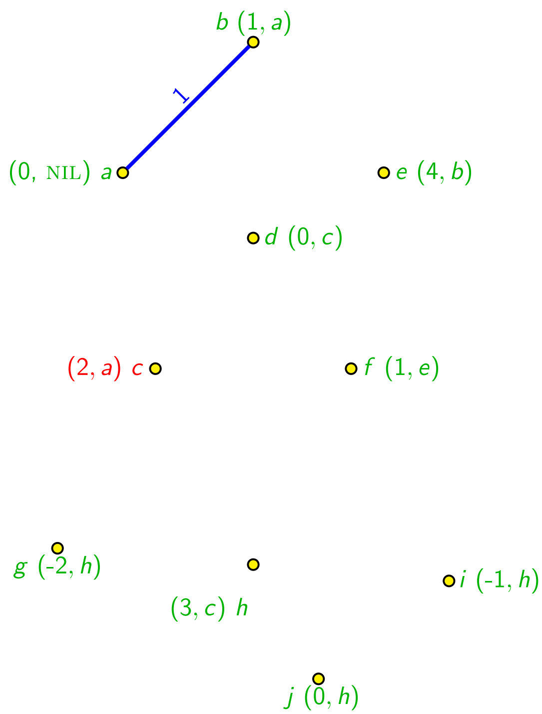


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
// Constructing the MST T
T ← ∅
for each v ∈ V
   if (v !=r)
      T ← T ∪ {(π[v], v)}
return T
```

MST-PRIM(G, w, r)

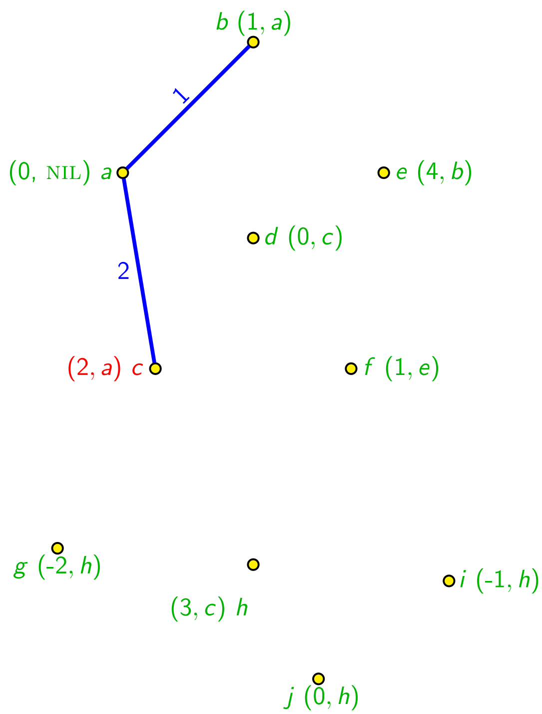


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T
- $T \leftarrow \emptyset$
- **for** each $v \in V$
  - **if** $(v! = r)$
    - $T \leftarrow T \cup \{(\pi[v], v)\}$
- **return** T

MST-Prim(G, w, r)

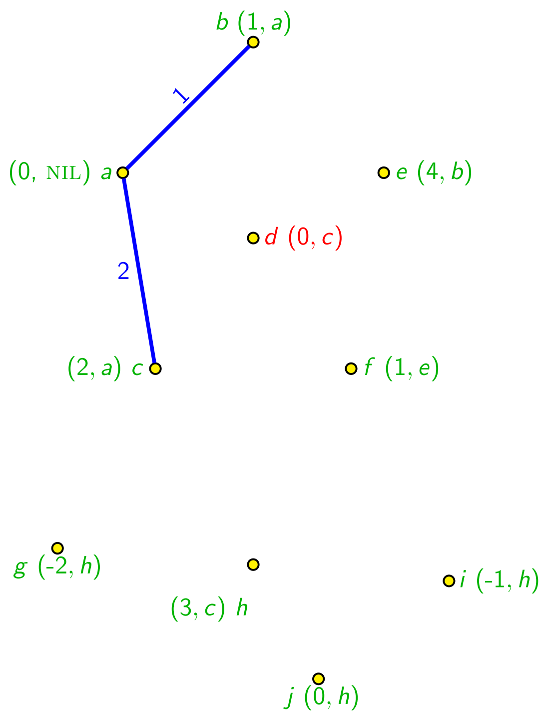


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
   // Constructing the MST T
   T ← ∅
   for each v ∈ V
      if (v != r)
         T ← T ∪ {(π[v], v)}
   return T
```

MST-PRIM(G, w, r)

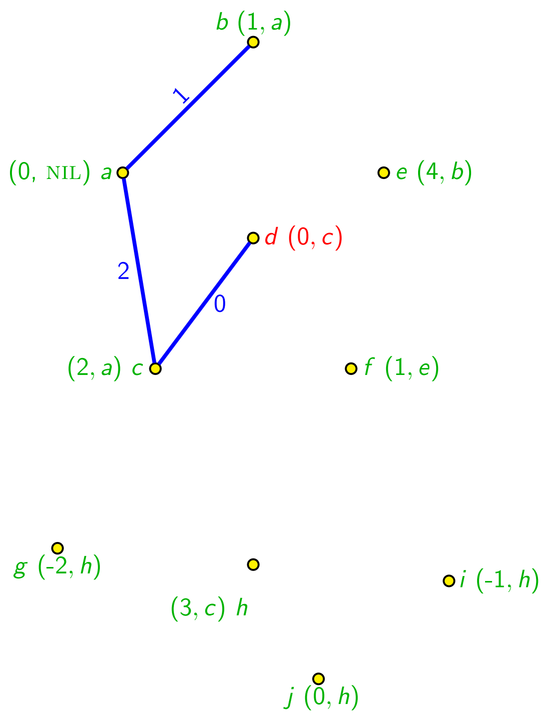


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T
12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** $(v! = r)$
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

MST-PRIM(G, w, r)

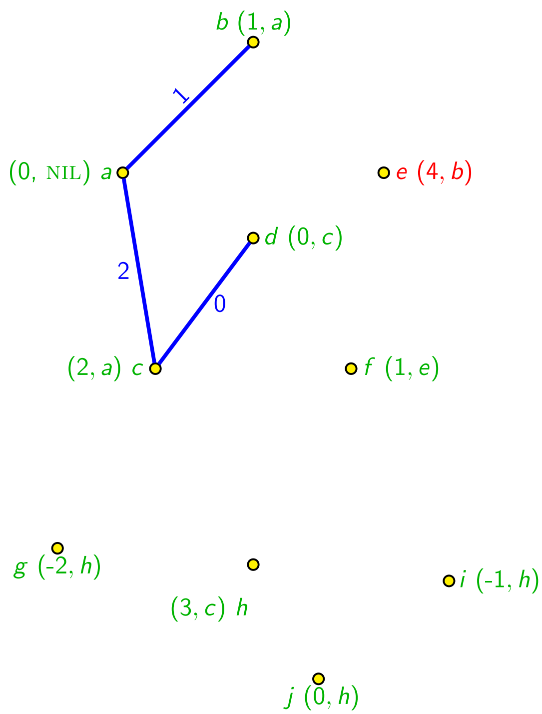


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T
12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** $(v \;!\!= r)$
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

MST-PRIM(G, w, r)

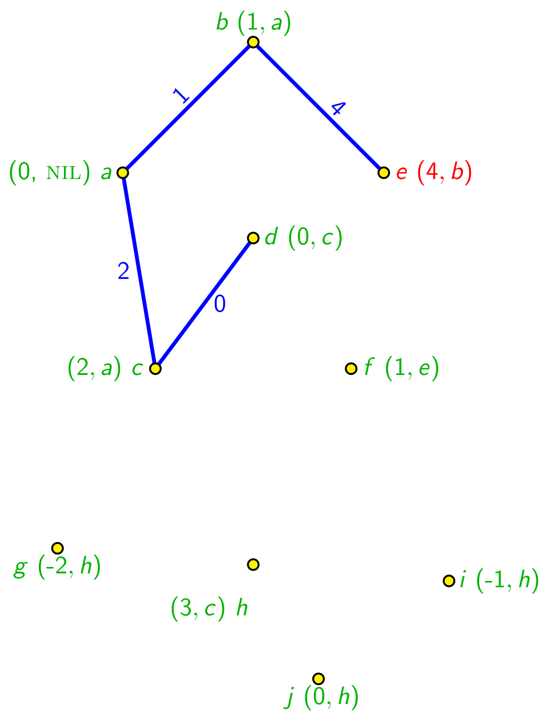


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
    // Constructing the MST T
12.   T ← ∅
13.   for each v ∈ V
14.      if (v! = r)
15.         T ← T ∪ {(π[v], v)}
16.   return T
```

MST-PRIM(G, w, r)



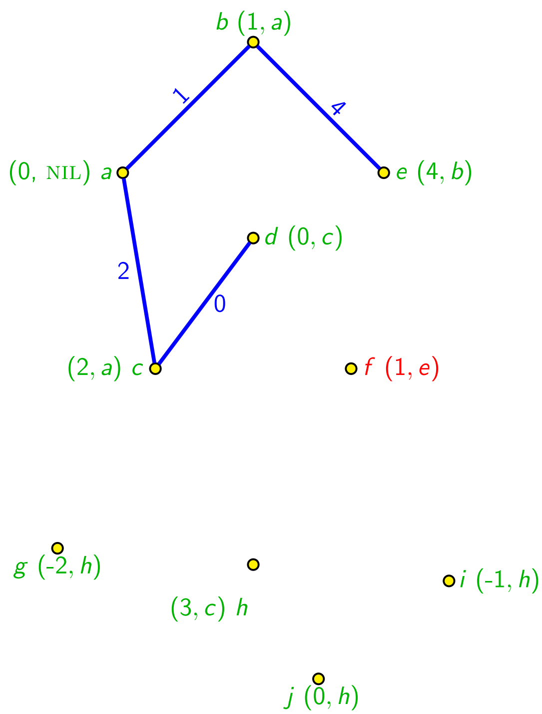
I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T

12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** (v !=r)
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

MST-PRIM(G, w, r)



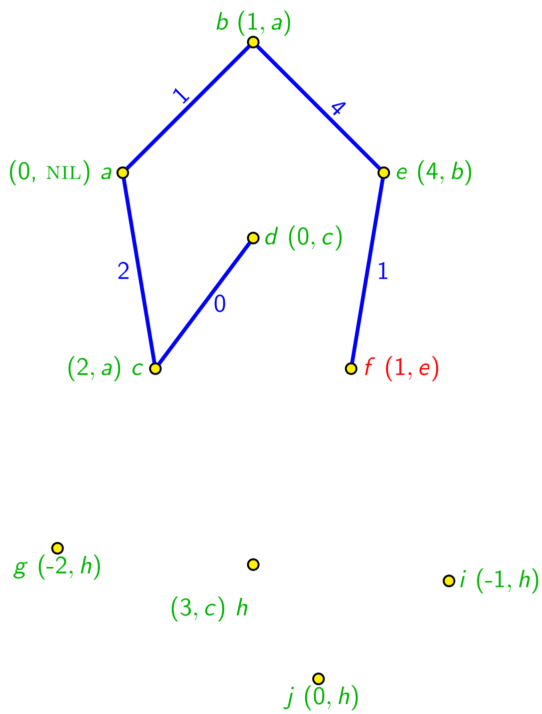
I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
   // Constructing the MST T
   T ← ∅
   for each v ∈ V
      if (v! = r)
         T ← T ∪ {(π[v], v)}

   return T
```

MST-PRIM(G, w, r)

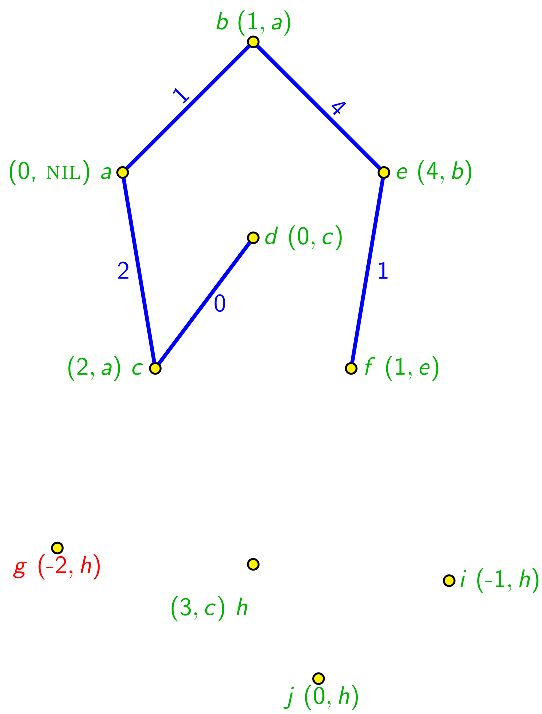


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T
12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** $(v \mathrel{!}= r)$
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

MST-PRIM(G, w, r)

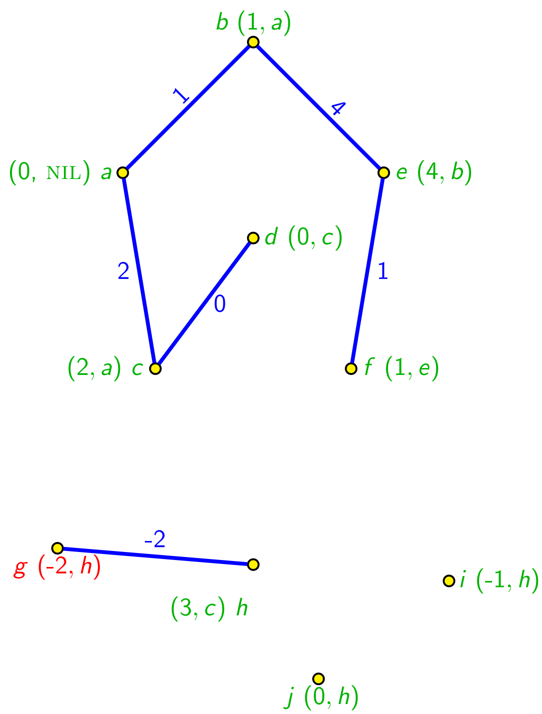


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
// Constructing the MST T
T ← ∅
for each v ∈ V
   if (v! = r)
      T ← T ∪ {(π[v], v)}
return T
```

MST-PRIM(G, w, r)



I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T

$T \leftarrow \emptyset$

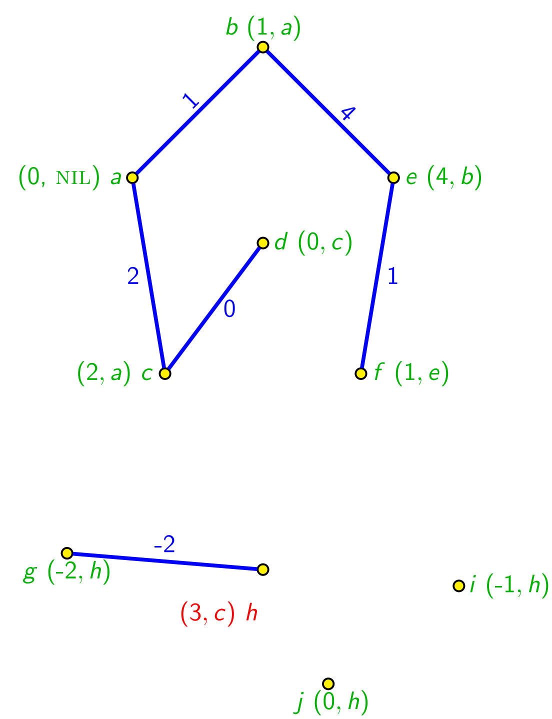
for each $v \in V$

  if (v !=r)

    $T \leftarrow T \cup \{(\pi[v], v)\}$

return T

MST-PRIM(G, w, r)

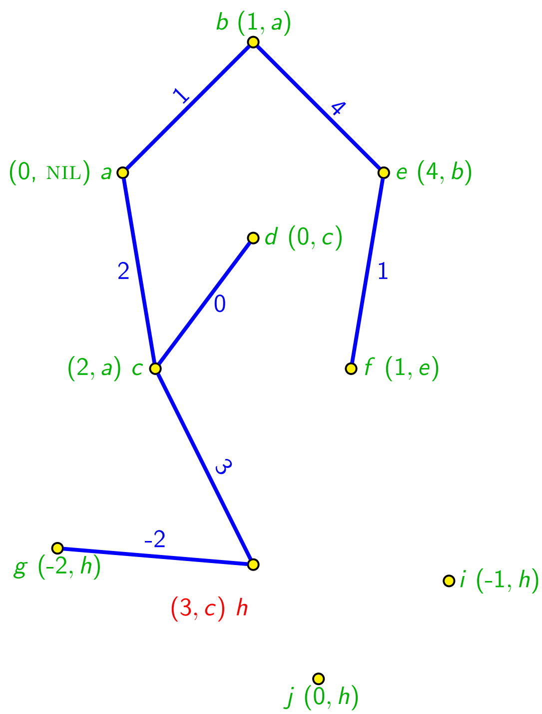


I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
    // Constructing the MST T
    T ← ∅
    for each v ∈ V
        if (v! = r)
            T ← T ∪ {(π[v], v)}
    return T
```

MST-PRIM(G, w, r)



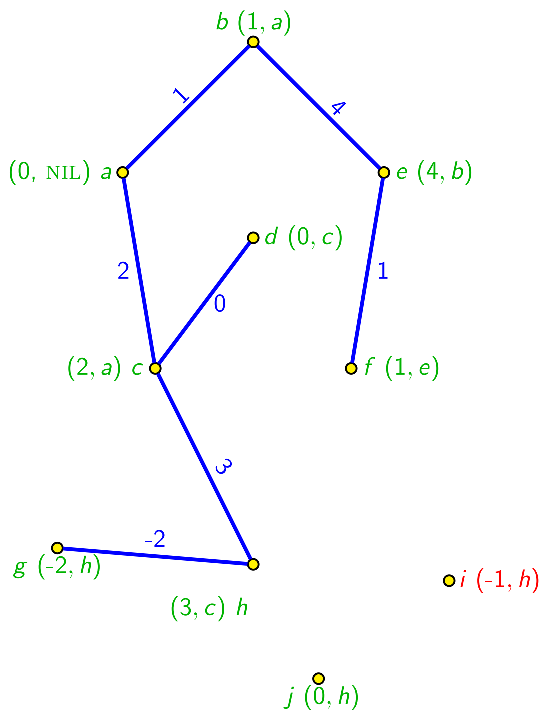
I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T

12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** (v !=r)
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

MST-PRIM(G, w, r)



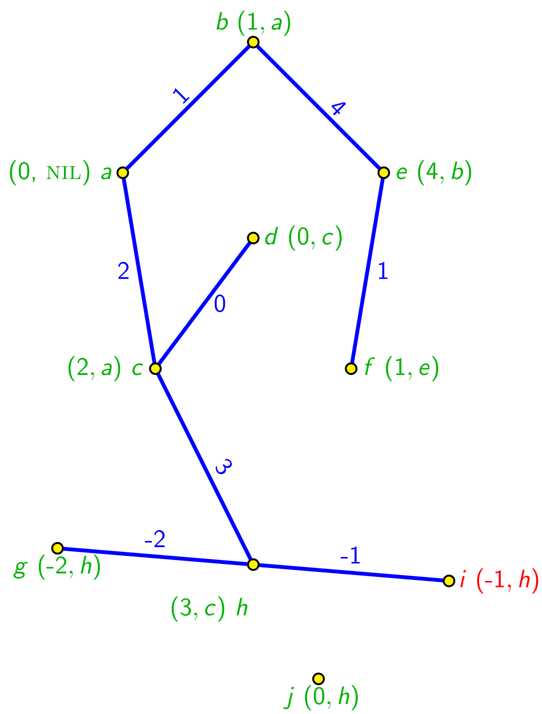
I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T

12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** $(v! = r)$
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

MST-PRIM(G, w, r)



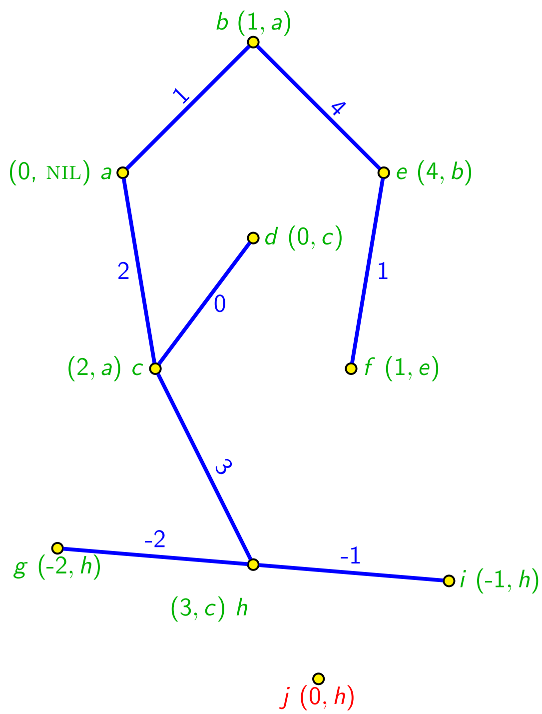
I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

```
   // Constructing the MST T
   T ← ∅
   for each v ∈ V
      if (v != r)
          T ← T ∪ {(π[v], v)}

   return T
```

MST-PRIM(G, w, r)



I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

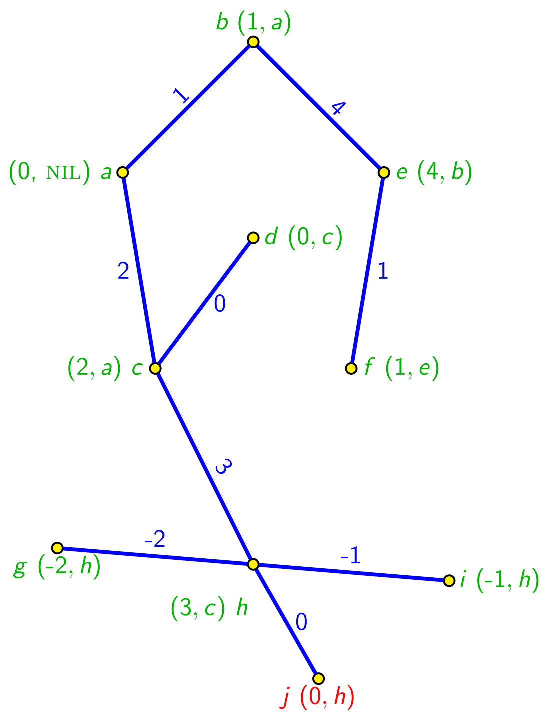
O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T

12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** $(v! = r)$
15. $T \leftarrow T \cup \{(\pi[v], v)\}$

16. **return** T

MST-PRIM(G, w, r)



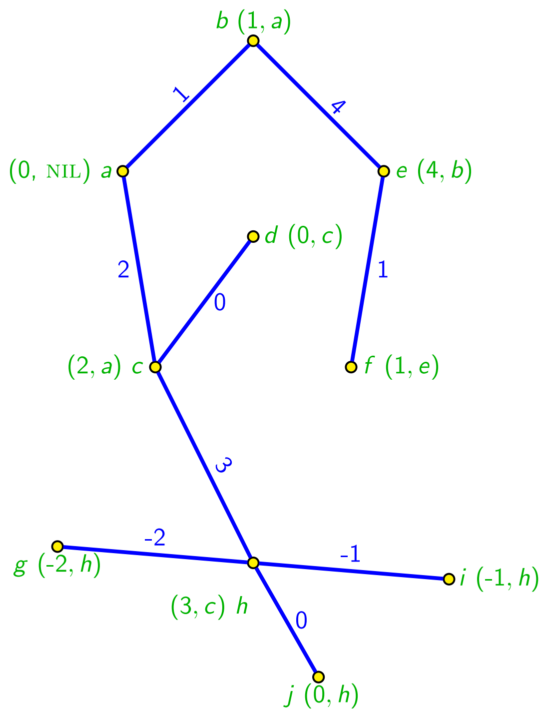
I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T

12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** $(v \; != r)$
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

MST-PRIM(G, w, r)



I/P: A weighted connected undirected graph $G = (V, E, w)$ and a root vertex r.

O/P: A MST $T = (V, A)$ of G.

// Constructing the MST T
12. $T \leftarrow \emptyset$
13. **for** each $v \in V$
14. **if** $(v \mathrel{!=} r)$
15. $T \leftarrow T \cup \{(\pi[v], v)\}$
16. **return** T

$$w(G) = \sum_{e \in A} w(e) = 8.$$

Correctness

The correctness follows from the following 3-part loop invariant:

- $A = \{(v, \pi[v]) : v \in \{V \setminus \{r\}\} \setminus Q\}$.
- The vertices already placed into the MST are those in $V \setminus Q$.
- For all vertices $v \in Q$, if $\pi[v] \neq \text{NIL}$, then $key[v] < \infty$ and $key[v]$ is the weight of a light edge $(v, \pi[v])$ connecting v to some vertex already placed into the minimum spanning tree.

  Cut: A *cut* $(S, V \setminus S)$ of an undirected graph $G = (V, E)$ is a partition of V.

  Light edge: An edge is a *light edge* crossing a cut if its weight is the minimum of any edge crossing the cut.

  Note: There can be more than one light edge crossing a cut in the case of ties.

Correctness

The correctness follows from the following 3-part loop invariant:

- $A = \{(v, \pi[v]) : v \in \{V \setminus \{r\}\} \setminus Q\}$.
- The vertices already placed into the MST are those in $V \setminus Q$.
- For all vertices $v \in Q$, if $\pi[v] \neq \text{NIL}$, then $key[v] < \infty$ and $key[v]$ is the weight of a light edge $(v, \pi[v])$ connecting v to some vertex already placed into the minimum spanning tree.

  Cut: A *cut* $(S, V \setminus S)$ of an undirected graph $G = (V, E)$ is a partition of V.

  Light edge: An edge is a *light edge* crossing a cut if its weight is the minimum of any edge crossing the cut.

  Note: There can be more than one light edge crossing a cut in the case of ties.

Homework: Read the proof the correctness of Prim's algorithm!

Complexity

- Depends on how the MIN-PRIORITY queue Q is implemented.
- Assume that Q is implemented as a binary MIN-HEAP.
- **Lines 1-5:** BUILD-MIN-HEAP procedure can be used to perform the initialization in $\mathcal{O}(|V|)$ time.
- **Note:** The body of the **while** loop is executed $|V|$ times.
  - Each EXTRACT-MIN operation takes $\mathcal{O}(\log |V|)$ time.

$\therefore$ $|V|$ calls to EXTRACT-MIN takes $\mathcal{O}(|V| \log |V|)$ time.

Complexity

- **Lines 8-11:** The **for** loop is executed $\mathcal{O}(|E|)$ times.
  - Recall that the sum of the lengths of all adjacency lists is $2|E|$.
  - **Line 9:** Test for membership in Q takes constant time.
    - Keep a bit for each vertex v to denote whether $v \in Q$ or not.
    - Update the bit when the vertex is removed from Q.
  - **Line 11:** Involves an implicit DECREASE-KEY operation.
    - Takes $\mathcal{O}(\log|V|)$ time in a binary MIN-HEAP.

- **Total time:** $\mathcal{O}(|V|\log|V| + |E|\log|V|) = \mathcal{O}(|E|\log|V|)$.

Books and Other Materials Consulted

1. Definitions and Krushkal's Algorithm taken from Discrete Mathematics Lecture Notes (M. Tech (CS), Monsoon Semester, 2007) taught by Prof. Palash Sarkar (ASU, ISI Kolkata).

2. *Introduction to Algorithms* by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.

Thank You for your kind attention!

Questions!!