

1. Minimum Spanning Tree

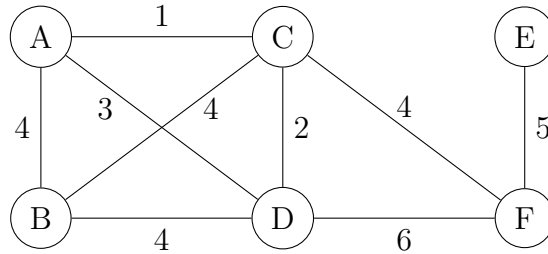


Figure 1: A weighted, connected undirected graph G_1

(a) Prim's algorithm

```
MST-PRIM( $G, w, r$ )
1   $Q = \text{EMPTY}$ 
2  for each vertex  $v \in V$ 
3       $v.\text{key} = +\text{INFTY}$ 
4       $v.\text{pi} = \text{NIL}$ 
5       $\text{INSERT}(Q, v)$ 
6   $\text{DECREASE-KEY}(Q, r, 0)$ 
7  while  $Q$  not  $\text{EMPTY}$ 
8       $u = \text{EXTRACT-MIN}(Q)$ 
9      for each  $v \in \text{Adj}[u]$ 
10         if  $v \in Q$  and  $w(u, v) < v.\text{key}$ 
11              $v.\text{pi} = u$ 
12              $v.\text{key} = w(u, v)$ 
```

(b) Kruskal's Algorithm

```
MST-KRUSKAL( $G, w$ )
1   $A = \text{EMPTY}$ 
2  for each vertex  $u \in V$ 
3       $\text{MAKE-SET}(u)$ 
4  Sort the edges  $E$  in nondecreasing order by  $w$ 
5  for each edge  $(u, v) \in E$ , taken in nondecreasing order by  $w$ 
6      if  $\text{FIND-SET}(u) \neq \text{FIND-SET}(v)$ 
7           $A = A \cup \{(u, v)\}$ 
8       $\text{UNION}(u, v)$ 
```

2. Single-Source Shortest Paths

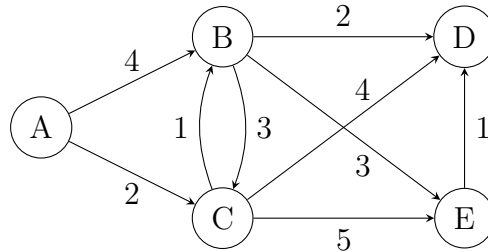


Figure 2: A weighted directed graph G_2

(a) The Bellman-Ford algorithm

```

BELLMAN-FORD( $G, w, s$ )
1  // initialization
2  for each vertex  $v \in V$ 
3       $v.d = +\infty$ 
4       $v.pi = \text{NIL}$ 
5   $s.d = 0$ 
6  for  $i = 1$  to  $|V| - 1$ 
7      for each edge  $(u, v) \in E$ 
8          // relax if needed
9          if  $v.d > u.d + w(u, v)$ 
10              $v.d = u.d + w(u, v)$ 
11              $v.pi = u$ 
12  // check whether there is negative-weight cycle
13  for each edge  $(u, v) \in E$ 
14      if  $v.d > u.d + w(u, v)$ 
15          return FALSE
16  return TRUE
  
```

	d					pi				
	A	B	C	D	E	A	B	C	D	E
Init	0	∞	∞	∞	∞	NIL	NIL	NIL	NIL	NIL
1										
2										
3										
4										
Check										

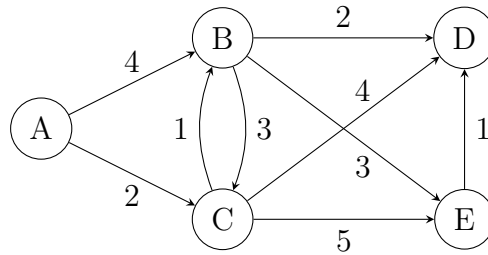


Figure 3: A weighted directed graph G_2

(b) Dijkstra's algorithm

```
DIJKSTRA( $G, w, s$ )
1  // initialization
2  for each vertex  $v \in V$ 
3       $v.d = +\text{INFTY}$ 
4       $v.pi = \text{NIL}$ 
5   $s.d = 0$ 
6  // priority queue keyed by  $d$ 
7   $Q = V$ 
8  while  $Q$  not EMPTY
9       $u = \text{EXTRACT-MIN}(Q)$ 
10     for each edge  $(u, v) \in E$ 
11         // relax if needed
12         if  $v.d > u.d + w(u, v)$ 
13              $v.d = u.d + w(u, v)$ 
14              $v.pi = u$ 
```

3. Minimum Spanning Tree VS Single-Source Shortest Paths

Some might consider that Minimum Spanning Tree and Single-Source Shortest Paths algorithms share a lot in common. Actually, they vary significantly in essence.

In MST, the goal is to connect all of the vertices and to minimize total weights of edges among all possible connections. Therefore, the solution must be a tree. In SSSP, the objective is to reach destination vertex from source vertex with lowest possible weights. This leads the solution to be a path.

We can also compare the four algorithms discussed above in the following aspects.

	MST		SSSP	
	Prim's	Kruskal's	Bellman-Ford	Dijkstra's
Directed or Undirected	Undirected		Directed	
Cycles Allowed	No		Yes*	Yes
Negative Weights Allowed	Yes		Yes*	No
Data Structures Used	Priority Queue	Disjoint Set	N/A	Priority Queue
Time Complexity	$O(E \lg V)$	$O(E \lg V)$	$\Theta(V \times E)$	Varies

*: Negative cycles are not applicable.

4. Proofs of Shortest Paths (optional)