Prepared by:
Ji Wang

Discussion 9

ECS 122A
May 29, 2019

1. Minimum Spanning Tree

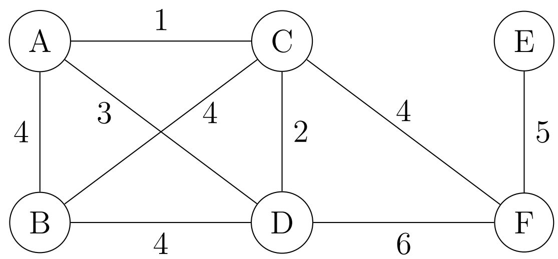

Figure 1: A weighted, connected undirected graph G_1

(a) Prim's algorithm

```
MST-PRIM(G, w, r)
Q = EMPTY
for each vertex v ∈ V
    v.key = +INFTY
    v.pi = NIL
    INSERT(Q, v)
DECREASE-KEY(Q, r, 0)
while Q not EMPTY
    u = EXTRACT-MIN(Q)
    for each v ∈ Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.pi = u
            v.key = w(u, v)
return {(v, v.pi) : v ∈ V - {r}}
```

(b) Kruskal's Algorithm

```
MST-KRUSKAL(G, w)
A = EMPTY
for each vertex u ∈ V
    MAKE-SET(u)
Sort the edges E in nondecreasing order by w
for each edge (u, v) ∈ E, taken in nondecreasing order by w
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

2. Single-Source Shortest Paths

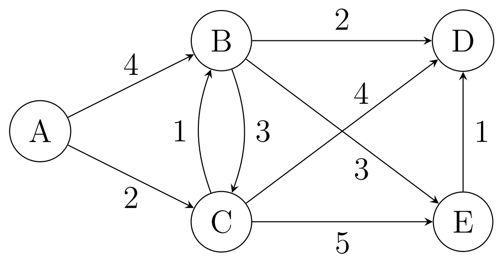

Figure 2: A weighted directed graph G_2

(a) The Bellman-Ford algorithm

```
BELLMAN-FORD(G, w, s)
// initialization
for each vertex v ∈ V
    v.d = +INFTY
    v.pi = NIL
s.d = 0
for i = 1 to |V| − 1
    for each edge (u, v) ∈ E
        // relax if needed
        if v.d > u.d + w(u, v)
            v.d = u.d + w(u, v)
            v.pi = u
// check whether there is negative-weight cycle
for each edge (u, v) ∈ E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE, {v.d : v ∈ V}, {v.pi : v ∈ V}
```

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

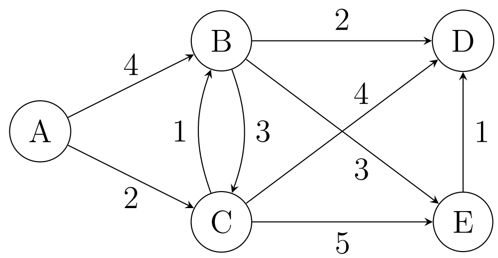

Figure 3: A weighted directed graph G_2

(b) Dijkstra's algorithm

```
DIJKSTRA(G, w, s)
// initialization
for each vertex v ∈ V
    v.d = +INFTY
    v.pi = NIL
s.d = 0
// priority queue keyed by d
Q = V
while Q not EMPTY
    u = EXTRACT-MIN(Q)
    for each edge (u, v) ∈ E
        // relax if needed
        if v.d > u.d + w(u, v)
            v.d = u.d + w(u, v)
            v.pi = u
return {v.d : v ∈ V}, {v.pi : v ∈ V}
```

Discussion 9

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 vertices from source vertex with lightest total weights. This leads the solution to be a set of paths.

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

	MST		**SSSP**	
	Prim's	**Kruskal's**	**Bellman-Ford**	**Dijkstra's**
Directed or Undirected	Undirected		Directed	
Source Vertex as Input	Yes	No	No	
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.