6. GRAPH ALGORITHMS HANDBOOK

Data Structures and Algorithms [19ECSC201]

PRAKASH HEGADE

III C   2019

SOCSE, KLE TU, Hubballi-31

CONTENTS

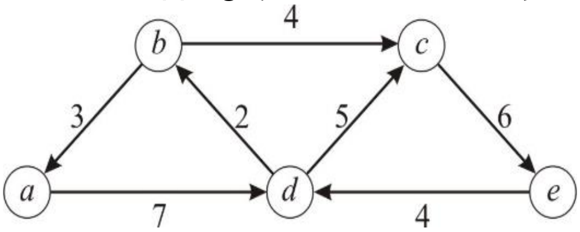
Note: Bellman-Ford is referenced from CLRS and others from Levitin text book.

1. Dijkstra's Algorithm

ALGORITHM Dijkstra(G, s)
// Dijkstra's algorithm for single source shortest path
// Input: A weighted connected graph G(V, E) with non-negative weights and its vertex s
// Output: the length d_v of a shortest path from s to v and its penultimate vertex p_v for
every vertex v in V
Initialize(Q)   // Initialize vertex priority queue to empty
for every vertex v in V do
   $d_v \leftarrow \infty$
   $p_v \leftarrow$ null
   Insert (Q, v, d_v)   // Initialize vertex priority in priority queue
$d_s \leftarrow 0$
Decrease (Q, s, d_s)   // Update priority of s with d_s
$V_T \leftarrow \emptyset$
for i $\leftarrow$ 0 to |V| - 1 do
   u* $\leftarrow$ DeleteMin(Q)
   $V_T = V_T \cup \{u^*\}$
   for every vertex u in V – V_T that is adjacent to u* do
      if $d_{u^*} + w(u^*, u) < d_u$
         $d_u \leftarrow d_{u^*} + w(u^*, u)$
         $p_u \leftarrow u^*$
         Decrease (Q, u, d_u)

Intuition:

Example: Find the shortest path form vertex c to all other vertices.
Vertices Mapping: {a:0, b:1, c:2, d:3, e:4}



cost [5][5] =

	0	1	2	3	4
0					
1					
2					
3					
4					

Initialization:

S = { 2 }

V – S = { 0, 1, 3, 4}

dist

0	
1	
2	
3	
4	

path

0	
1	
2	
3	
4	

Iteration 01:

u =

dist[u] =

S =

V – S =

dist

0	
1	
2	
3	
4	

path

0	
1	
2	
3	
4	

Iteration 02:

u =

dist[u] =

S =

V – S =

dist

0	
1	
2	
3	
4	

path

0	
1	
2	
3	
4	

Working Space:

Iteration 03:

u = 　　　　　　　　S =

dist[u] = 　　　　　　V – S =

dist

0	
1	
2	
3	
4	

path

0	
1	
2	
3	
4	

Iteration 04:

u = 　　　　　　　　S =

dist[u] = 　　　　　　V – S =

dist

0	
1	
2	
3	
4	

path

0	
1	
2	
3	
4	

Trace the path from vertex 2 to vertex 4:

Design Technique:

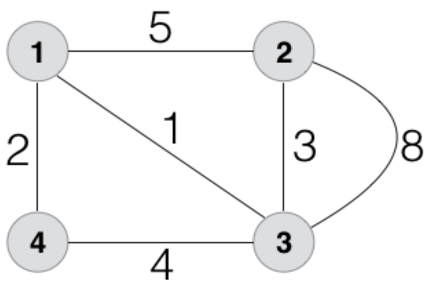
Efficiency Analysis:

2. Floyd's Algorithm

Intuition:

ALGORITHM Floyd (W[1..n,1..n])
// Implements Floyd's algorithm for all pair shortest path problem
// Input: The weight matrix W of the graph with no negative length cycle
// Output: The distance matrix of the shortest path's lengths
D ← W
for k ← 1 to n do
   for i ← 1 to n do
      for j ← 1 to n do
         D [i, j] ← min {D[i, j], D[i, k] +D[k, j]}
return D

Example: Apply Floyd's algorithm on the given graph:



$D^{(0)} =$

$D^{(1)} =$

$D^{(2)} =$

$D^{(3)} =$

$D^{(4)} =$

Design Technique:

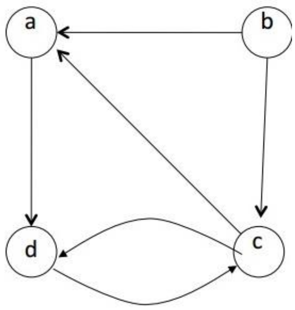
Efficiency Analysis:

3. Warshall's Algorithm

Intuition:

```
ALGORITHM Warshall (A[1..n,1..n])
// Implements Warshall's algorithm for computing transitive closure
// Input: The adjacency matrix A of a diagraph with n vertices
// Output: The transitive closure of the diagraph
R(0) ← A
for k ← 1 to n do
   for i ← 1 to n do
      for j ← 1 to n do
         R(k) [i, j] ← R(k-1) [i, j] or (R(k-1) [i, k] and R(k-1) [k, j])
return R(n)
```

Example:



$R^{(0)} =$

$R^{(1)} =$

$R^{(2)} =$

$R^{(3)} =$

$R^{(4)} =$

Design Technique:

Efficiency Analysis:

4. Kruskal's Algorithm

ALGORITHM Kruskal(G)
// Kruskal's algorithm to construct a minimum spanning tree
// Input: A weighted connected graph G(V, E)
// Output: E_T, the set of edges composing of MST of G
sort E in nondecreasing order of the edge weights $w(e_{i1}) <= \dots <= w(e_{i|E|})$
$E_T \leftarrow \emptyset$
ecounter $\leftarrow$ 0
k $\leftarrow$ 0
while ecounter < |V| - 1 do
   k $\leftarrow$ k + 1
   if E_T U $\{e_{ik}\}$ is acyclic
      $E_T \leftarrow E_T$ U $\{e_{ik}\}$
      ecounter $\leftarrow$ecounter + 1
return E_T

Disjoint-Set:
For the given array, perform the said union operations:

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

Union (3, 4)

0	1	2	3	4	5	6	7	8

Union (1, 4)

0	1	2	3	4	5	6	7	8

Union (3, 7)

0	1	2	3	4	5	6	7	8

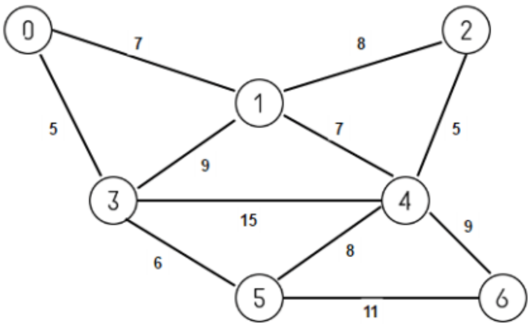
Union (6, 8)

0	1	2	3	4	5	6	7	8

Union (6, 5)

0	1	2	3	4	5	6	7	8

Example:



Sorted Edges:

1.	2.	3.	4.
5.	6.	7.	8.
9.	10.	11.	

steps	(u, v)	i=find(u) j=find(v)	output	Union(i, j)						
				0	1	2	3	4	5	6
init										
1										
2										
3										
4										
5										
6										
7										
8										
9										

Design Technique:

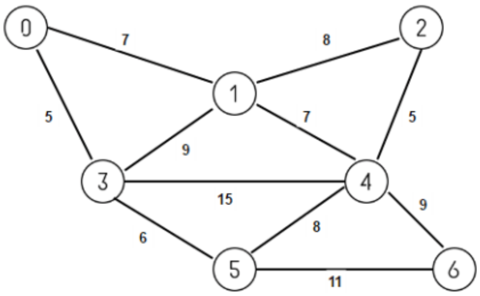
Efficiency Analysis:

5. Prim's Algorithm

ALGORITHM Prim(G)
// Prim's algorithm to construct a minimum spanning tree
// Input: A weighted connected graph G(V, E)
// Output: E_T, the set of edges composing of MST of G
$V_T \leftarrow \{v_0\}$
$E_T \leftarrow \varnothing$
for i ← 1 to |V| - 1 do
  find a minimum weight edge e* = (v*,u*) along all the edges (v, u) such that
  v is in V_T and u is in $V - V_T$
  $V_T \leftarrow V_T \cup \{u^*\}$
  $E_T \leftarrow E_T \cup \{e^*\}$
return E_T

Intuition:

Example:



cost[ ][ ] =

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

Initialization: Step 0

	dist	path
0		
1		
2		
3		
4		
5		
6		

S =
V – S =
u =
dist[u] =
output =

Initialization: Step 1

	dist	path
0		
1		
2		
3		
4		
5		
6		

S =
V – S =
u =
dist[u] =
output =

Initialization: Step 2

	dist	path
0		
1		
2		
3		
4		
5		
6		

S =
V – S =
u =
dist[u] =
output =

Initialization: Step 3

	dist	path
0		
1		
2		
3		
4		
5		
6		

S =
V – S =
u =
dist[u] =
output =

Initialization: Step 4

	dist	path
0		
1		
2		
3		
4		
5		
6		

S =
V – S =
u =
dist[u] =
output =

Initialization: Step 5

	dist	path
0		
1		
2		
3		
4		
5		
6		

S =
V – S =
u =
dist[u] =
output =

Initialization: Step 6

	dist	path
0		
1		
2		
3		
4		
5		
6		

S =
V – S =

Cost of MST is ____________

Design Technique:

Efficiency Analysis:

6. Bellman Ford Algorithm

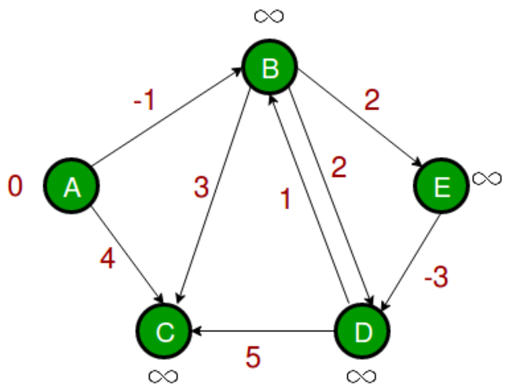
$d[s] \leftarrow 0$
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
$\Bigg\}$ initialization

for $i \leftarrow 1$ **to** $|V| - 1$
  do for each edge $(u, v) \in E$
    do if $d[v] > d[u] + w(u, v)$
      then $d[v] \leftarrow d[u] + w(u, v)$
$\Big\}$ ***relaxation step***

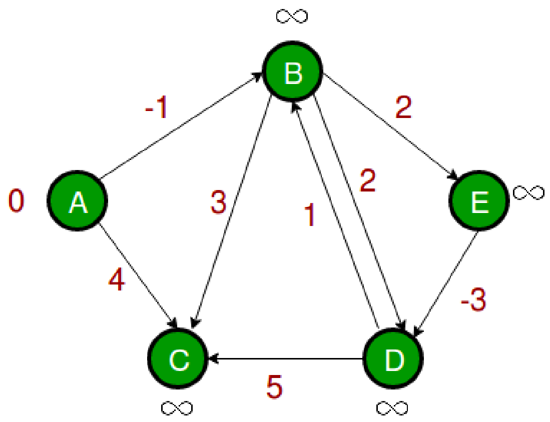
for each edge $(u, v) \in E$
  do if $d[v] > d[u] + w(u, v)$
    then report that a negative-weight cycle exists

At the end, $d[v] = \delta(s, v)$, if no negative-weight cycles.

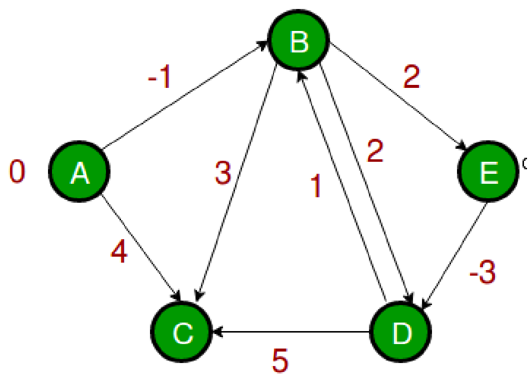
Example:
Edge List: (B, E), (D, B), (B, D), (A, B), (A, C), (D, C), (B, C), (E, D).



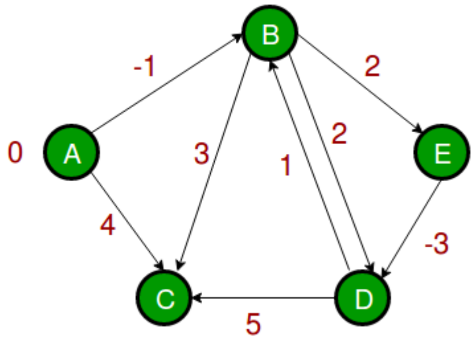
Iteration 01:



Iteration 02:



Iteration 03:



Design Technique:

Efficiency Analysis:

~*~*~*~*~