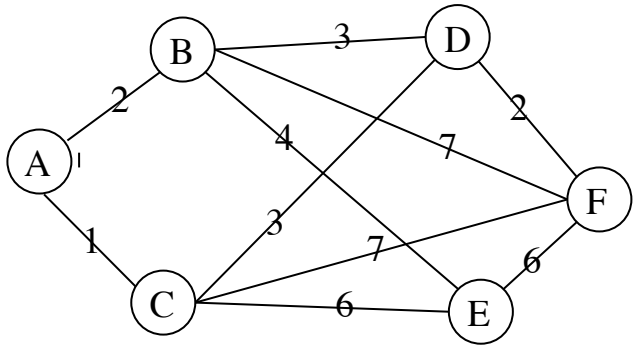
Demo: Shortest Path Routing

- Notations
  - *S*: the set of nodes for which the cheapest path from *A* is known

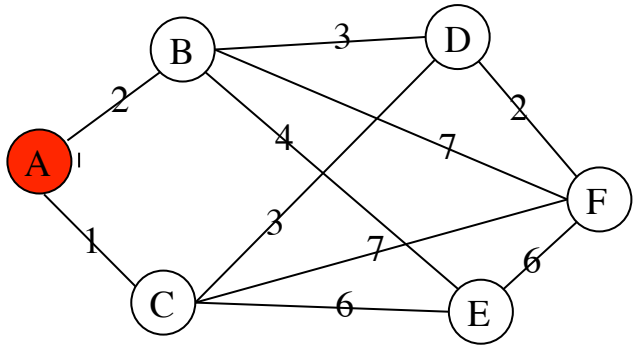
    [red colour]
  - *W*: the set of nodes not in S, but are neighbours of at least one node in *S* via a direct link

    [blue colour]
  - *Cost(X)*: the cost of the cheapest path from *A* to *X* for which intermediate nodes are in *S*
  - *Prior(X)*: is the node preceding X in the cheapest route



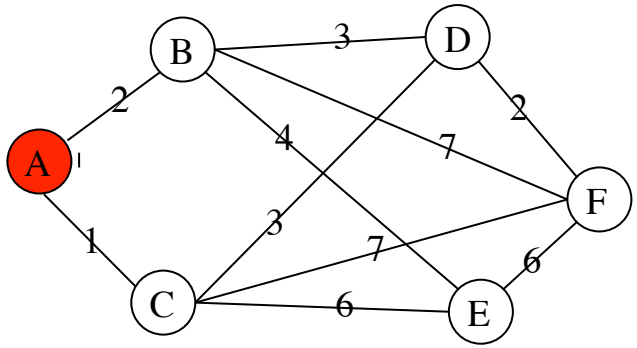
Step 1:
S={A}; Cost(X) is the cost of the link from *A* to *X*; if no such link exists, *Cost(X)* is infinite. For those nodes linked to *A,* define *Prior(X)=A.*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1													
2													
3													
4													
5													



Step 1:
S={A}; Cost(X) is the cost of the link from *A* to *X*; if no such link exists, *Cost(X)* is infinite. For those nodes linked to *A,* define *Prior(X)=A*.

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}			2	1	∞	∞	∞	A	A	–	–	–
2													
3													
4													
5													

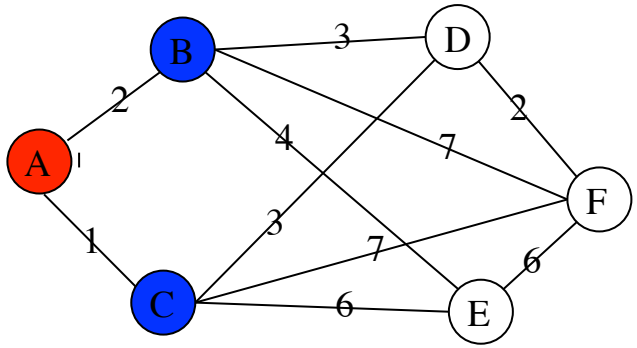


Step 2:
Find *W*, the nodes which are not in *S*, but are connected to a node in *S*.

Check the neighbours of each node in *S* one by one.

Since *S* ={*A*}, its neighbours are *B* and *C*.
Since both *B* and *C* are not in *S*, *W*={*B*, *C*}

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}			2	1	∞	∞	∞	A	A	–	–	–
2													
3													
4													
5													

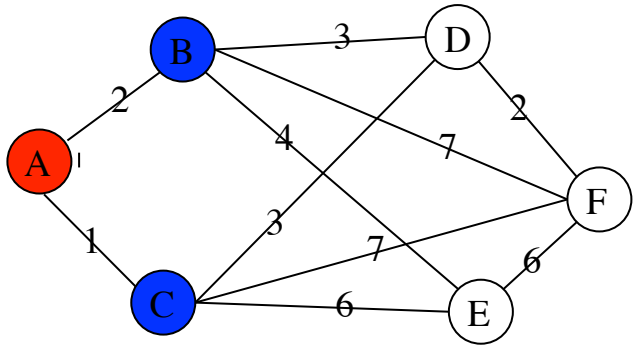


Step 2:
Find *W*, the nodes which are not in *S*, but are connected to a node in *S*.

Check the neighbours of each node in *S* one by one.

Since *S* ={*A*}, its neighbours are *B* and *C*.
Since both *B* and *C* are not in *S*, *W*={*B*, *C*}

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}		2	1	∞	∞	∞	A	A	–	–	–
2													
3													
4													
5													



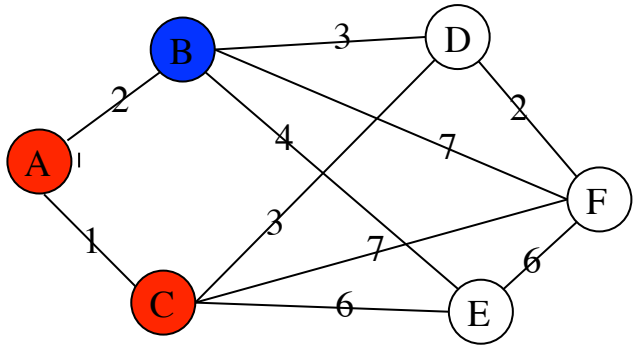
Step 3:
Choose a node X in W for which Cost(X) is minimum and add X to the set S

W={B, C}

Cost(B)= Cost(A,B) = 2
Cost(C) = Cost(A,C) = 1

Since Cost(C) < Cost(B), X = C

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}		2	1	∞	∞	∞	A	A	–	–	–
2													
3													
4													
5													



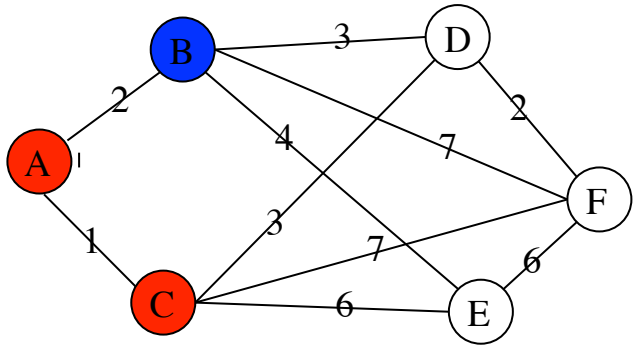
Step 3:
Choose a node X in W for which Cost(X) is minimum and add X to the set S

W={B, C}

Cost(B)= Cost(A,B) = 2
Cost(C) = Cost(A,C) = 1

Since Cost(C) < Cost(B), X = C

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}												
3													
4													
5													

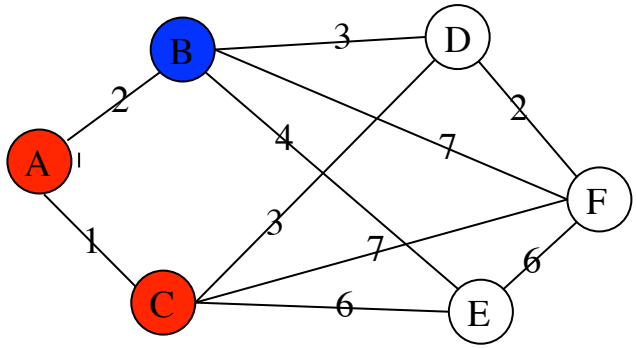


Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {B,D,E,F}

Cost(B) = minimum{ Cost(*B*), Cost(*C*)+Cost(*C*,*B*)
= minimum{2, 1+ ∞}
= 2

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}			2	1	∞	∞	∞	A	A	–	–	–
3													
4													
5													



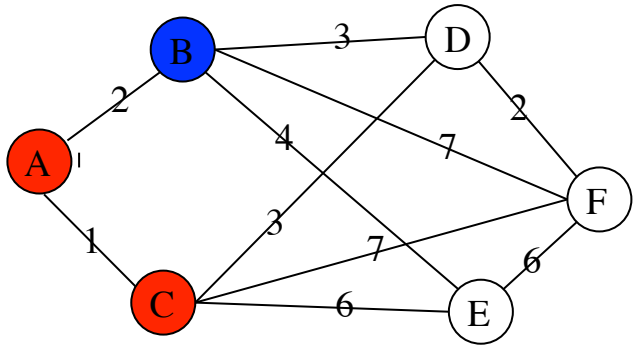
Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {B,D,E,F}

Cost(B)= minimum{ Cost(*B*), Cost(*C*)+Cost(*C,B*)
= minimum{2, 1+ ∞}
= 2

Cost(D)= minimum{ Cost(*D*), Cost(*C*)+Cost(*C,D*)
= minimum{∞, 1+ 3}
= 4

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}			2	1	4	∞	∞	A	A	C	–	–
3													
4													
5													



Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

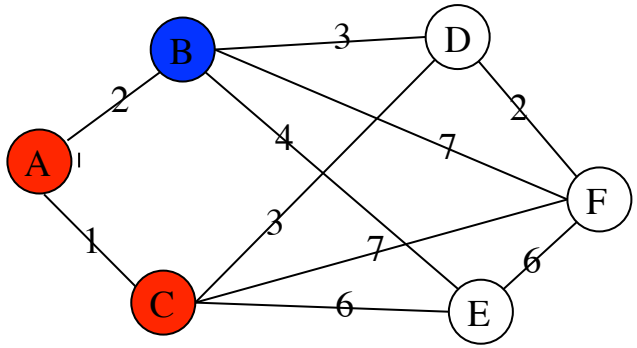
The set of nodes that are not in S is {B,D,E,F}

Cost(B) = minimum{ Cost(*B*), Cost(*C*)+Cost(*C,B*)
= minimum{2, 1+ ∞}
= 2

Cost(D) = minimum{ Cost(*D*), Cost(*C*)+Cost(*C,D*)
= minimum{∞, 1+ 3}
= 4

Cost(E) = minimum{ Cost(*E*), Cost(*C*)+Cost(*C,E*)
= minimum{∞, 1+ 6}
= 7

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}			2	1	4	7	∞	A	A	C	C	–
3													
4													
5													



Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {B,D,E,F}

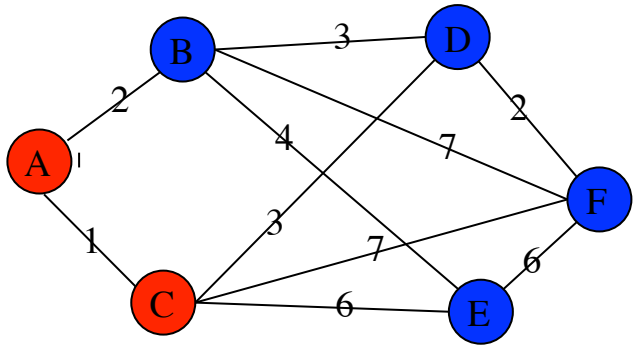
Cost(B)= minimum{ Cost(*B*), Cost(*C*)+Cost(*C,B*)
= minimum{2, 1+ ∞}
= 2

Cost(D)= minimum{ Cost(*D*), Cost(*C*)+Cost(*C,D*)
= minimum{∞, 1+ 3}
= 4

Cost(E)= minimum{ Cost(*E*), Cost(*C*)+Cost(*C,E*)
= minimum{∞, 1+ 6}
= 7

Cost(F)= minimum{ Cost(*F*), Cost(*C*)+Cost(*C,F*)
= minimum{∞, 1+ 7}
= 8

				Cost(X)					Prior(X)				
	S	W	X	B	C	D	E	F	B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}			2	1	4	7	8	A	A	C	C	C
3													
4													
5													

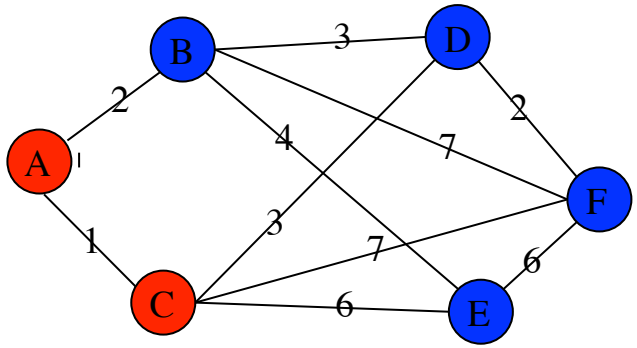


Step 2:
Find *W*, the nodes which are not in *S*, but are connected to a node in *S*.

Check the neighbours of each node in *S* one by one.

A's neighbours are B, C, D, E, F.
C's neighbours are A, B, D, E, F.
Since *A and C* are already in *S, W={B, D, E, F}*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}		2	1	4	7	8	A	A	C	C	C
3													
4													
5													



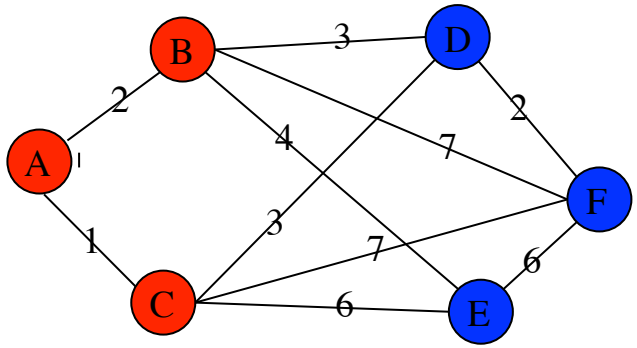
Step 3:
Choose a node X in W for which Cost(X) is minimum and add X to the set S

W={B, D, E, F}

Cost(B)= Cost(A,B) = 2
Cost(D) = 4
Cost(E) = 7
Cost(F) =8

Since Cost(B) = minimum {Cost(*B*), Cost(*D*), Cost(*E*), Cost(*F*)}, *X = B*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}		2	1	4	7	8	A	A	C	C	C
3													
4													
5													

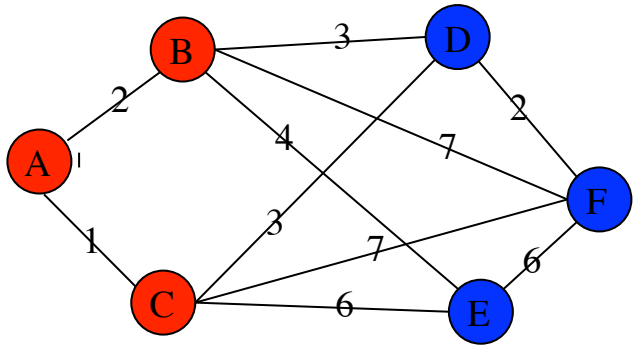


Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {D,E,F}

Cost(D)= minimum{ Cost(*D*), Cost(*B*)+Cost(*B*,*D*)}
= minimum{4, 2+ 3}
= 4

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}			2	1	4	7	8	A	A	C	C	C
4													
5													



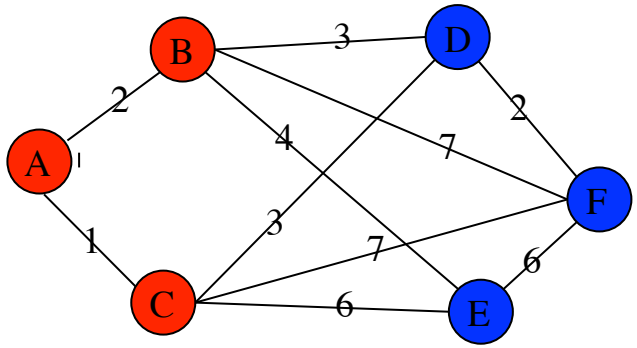
Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {D,E,F}

Cost(D)= minimum{ Cost(*D*), Cost(*B*)+Cost(*B*,*D*)}
= minimum{4, 2+ 3}
= 4

Cost(E)= minimum{ Cost(*E*), Cost(*B*)+Cost(*B*,*E*)}
= minimum{7, 2+ 4}
= 6

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}			2	1	4	**6**	8	A	A	C	**B**	C
4													
5													



Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

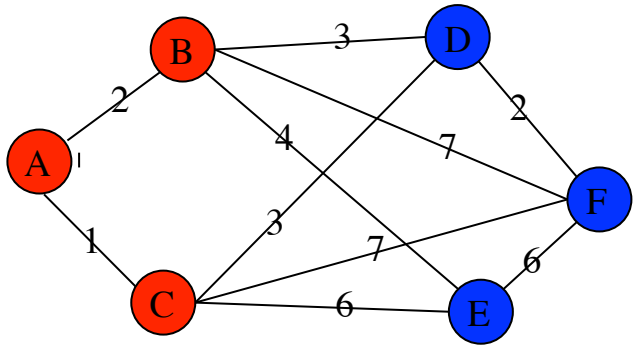
The set of nodes that are not in S is {D,E,F}

Cost(D) = minimum{ Cost(*D*), Cost(*B*)+Cost(*B*,*D*)}
= minimum{4, 2+ 3}
= 4

Cost(E) = minimum{ Cost(*E*), Cost(*B*)+Cost(*B*,*E*)}
= minimum{7, 2+ 4}
= 6

Cost(F) = minimum{ Cost(*F*), Cost(*B*)+Cost(*B*,*F*)}
= minimum{8, 2+ 7}
= 8

	S	W	X	Cost(X)					Prior(X)				
				B	C	D	E	F	B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}			2	1	4	6	8	A	A	C	B	C
4													
5													



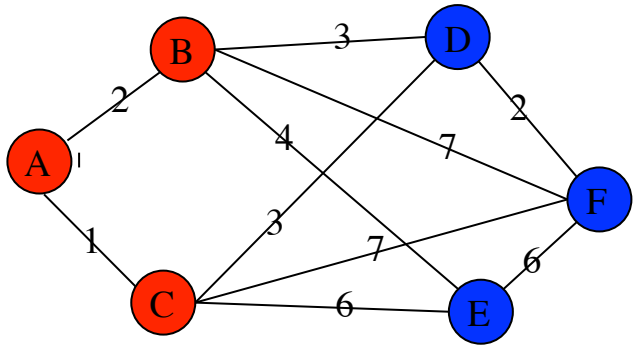
Step 2:
Find *W*, the nodes which are not in *S*, but are connected to a node in *S*.

Check the neighbours of each node in *S* one by one.

A's neighbours are B, C, D, E, F.
B's neighbours are A, D, E, F.
C's neighbours are A, B, D, E, F.

Since *A, B and C* are already in *S, W={ D, E, F}*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}		2	1	4	6	8	A	A	C	B	C
4													
5													



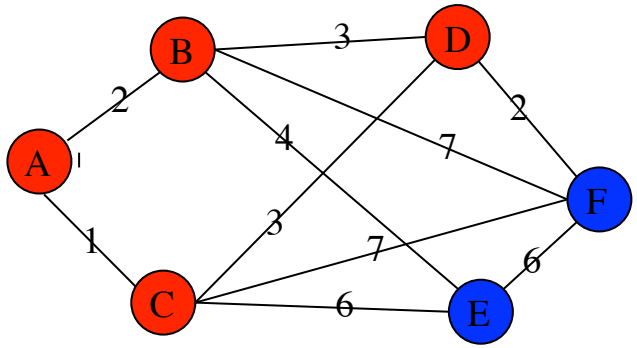
Step 3:
Choose a node X in W for which Cost(X) is minimum and add X to the set S

W={D, E, F}

Cost(D) = 4
Cost(E) = 6
Cost(F) =8

Since Cost(D) = minimum {Cost(*D*), Cost(*E*), Cost(*F*)}, *X = D*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}		2	1	4	6	8	A	A	C	B	C
4													
5													



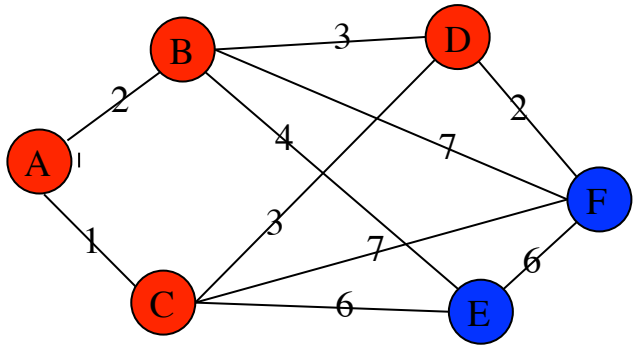
Step 3:
Choose a node X in W for which Cost(X) is minimum and add X to the set S

W={D, E, F}

Cost(D) = 4
Cost(E) = 6
Cost(F) =8

Since Cost(D) = minimum {Cost(*D*), Cost(*E*), Cost(*F*)}, *X = D*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	8	A	A	C	B	C
4	{A,B,C,D}												
5													

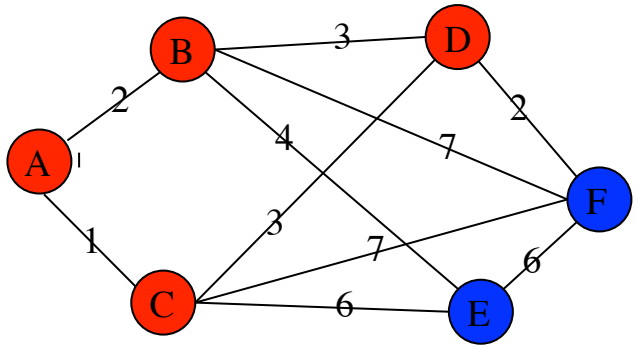


Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {E,F}

Cost(E) = minimum{ Cost(*E*), Cost(*D*)+Cost(*D,E*)}
= minimum{6, 4+ ∞}
= 6

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	8	A	A	C	B	C
4	{A,B,C,D}												
5													



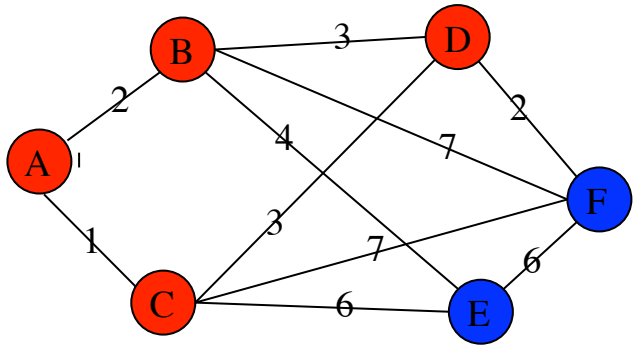
Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {E,F}

Cost(E) = minimum{ Cost(*E*), Cost(*D*)+Cost(*D*,*E*)}
= minimum{6, 4+ ∞}
= 6

Cost(F) = minimum{ Cost(*F*), Cost(*D*)+Cost(*D*,*F*)}
= minimum{8, 4+ 2}
= 6

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	**6**	A	A	C	B	**D**
4	{A,B,C,D}												
5													



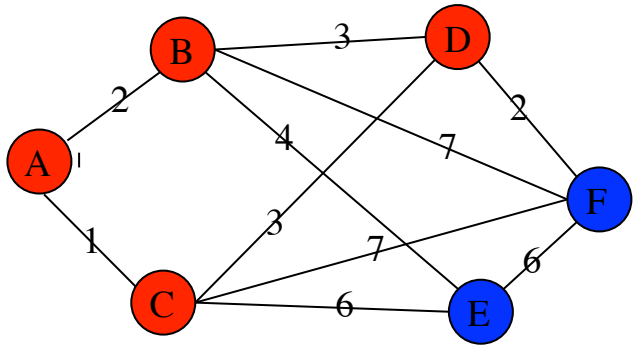
Step 2:
Find *W*, the nodes which are not in *S*, but are connected to a node in *S*.

Check the neighbours of each node in *S* one by one.

A's neighbours are B, C, D, E, F.
B's neighbours are A, D, E, F.
C's neighbours are A, B, D, E, F.
D's neighbours are B, C, F.

Since *A, B, C and D* are already in *S, W={ E, F}*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}		2	1	4	6	6	A	A	C	B	D
5													



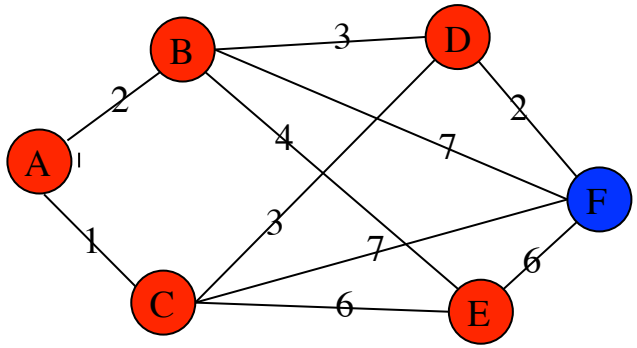
Step 3:
Choose a node X in W for which Cost(X) is minimum and add X to the set S

W={E, F}

Cost(E) = 6
Cost(F) =6

Since Cost(E) = Cost(F), we can choose any one. Suppose we choose *E.*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}		2	1	4	6	6	A	A	C	B	D
5													



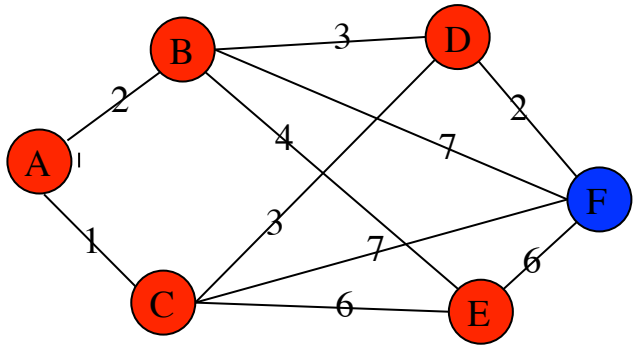
Step 3:
Choose a node X in W for which Cost(X) is minimum and add X to the set S

W={E, F}

Cost(E) = 6
Cost(F) =6

Since Cost(E) = Cost(F), we can choose any one. Suppose we choose *E.*

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A,B,C,D,E}												

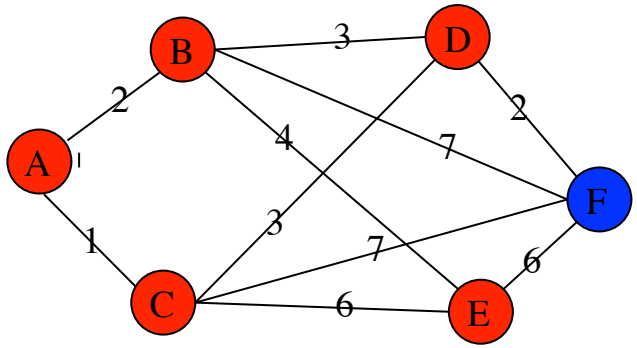


Step 4:
For each V not in S, Cost(V)=minimum{Cost(V), Cost(X)+ Cost(X,V)}.
If Cost(V) is changed, Prior(V)=X.

The set of nodes that are not in S is {F}

Cost(F) = minimum{ Cost(*F*), Cost(*E*)+Cost(*E,F*)}
= minimum{6, 6+ 6}
= 6

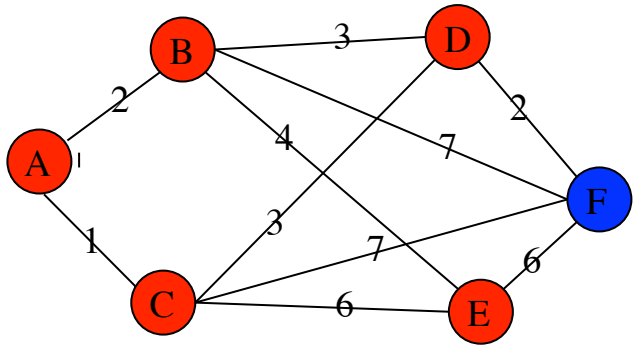
	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A,B,C,D,E}			2	1	4	6	6	A	A	C	B	D



Step 2:
Find *W*, the nodes which are not in *S*, but are connected to a node in *S*.

We only have F left. Hence, W={F}

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A,B,C,D,E}	{F}		2	1	4	6	6	A	A	C	B	D

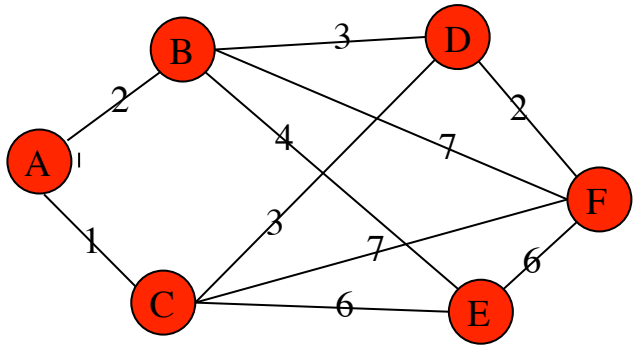


Step 3:

Choose a node X in W for which Cost(X) is minimum and add X to the set S

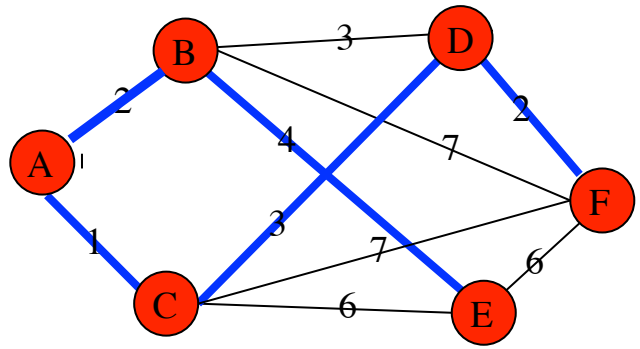
Since we only have F left, we have to choose F.

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A,B,C,D,E}	{F}		2	1	4	6	6	A	A	C	B	D



Since all nodes have been added to *S*, the algorithm terminates.

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A,B,C,D,E}	{F}	F	2	1	4	6	6	A	A	C	B	D
6	{A,B,C,D,E,F}			2	1	4	6	6	A	A	C	B	D



A spanning tree rooted at source *A*.

	S	W	X	Cost(X) B	C	D	E	F	Prior(X) B	C	D	E	F
1	{A}	{B,C}	C	2	1	∞	∞	∞	A	A	–	–	–
2	{A,C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, C}	{D, E, F}	D	2	1	4	6	6	A	A	C	B	D
4	{A,B,C,D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A,B,C,D,E}	{F}	F	2	1	4	6	6	A	A	C	B	D
6	{A,B,C,D,E,F}			2	1	4	6	6	A	A	C	B	D