Kruskal’s Algorithm for Computing MSTs

Presented by: Raj Kumar Ranabhat

M.E in Computer Engineering(I/I)

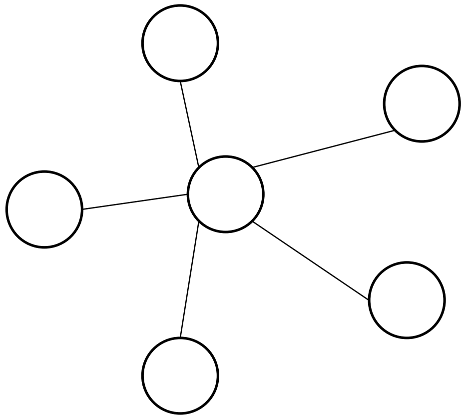
Kathmandu University

Tree

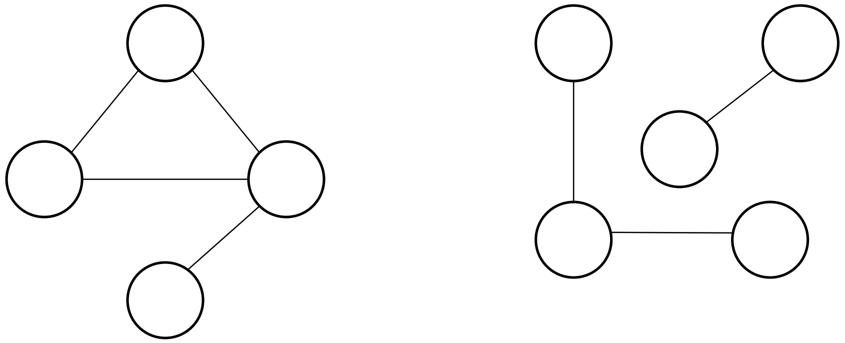
A tree is a graph with the following properties:

- The graph is connected (can go from anywhere to anywhere)

- There are no cycles(acyclic)



Tree



Graphs that are not trees

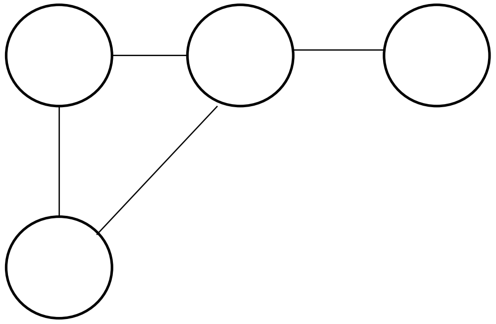
Minimum Spanning Tree (MST)

Let G=(V,E) be an undirected connected graph.

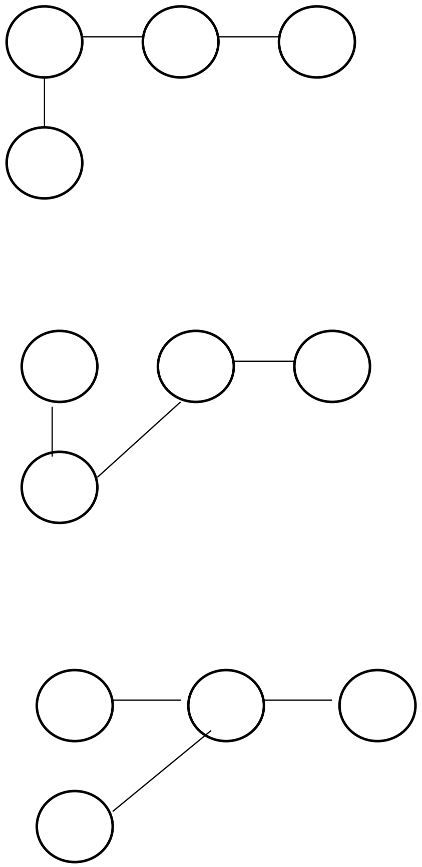
A sub graph T=(V,E’) of G is a spanning tree of G if :-

- T is a tree (i.e., it is acyclic)
- T covers all the vertices ***V***
  - contains **|*V*| - *1*** edges
- T has minimum total weight
- A single graph can have many different spanning trees.

Connected undirected graph

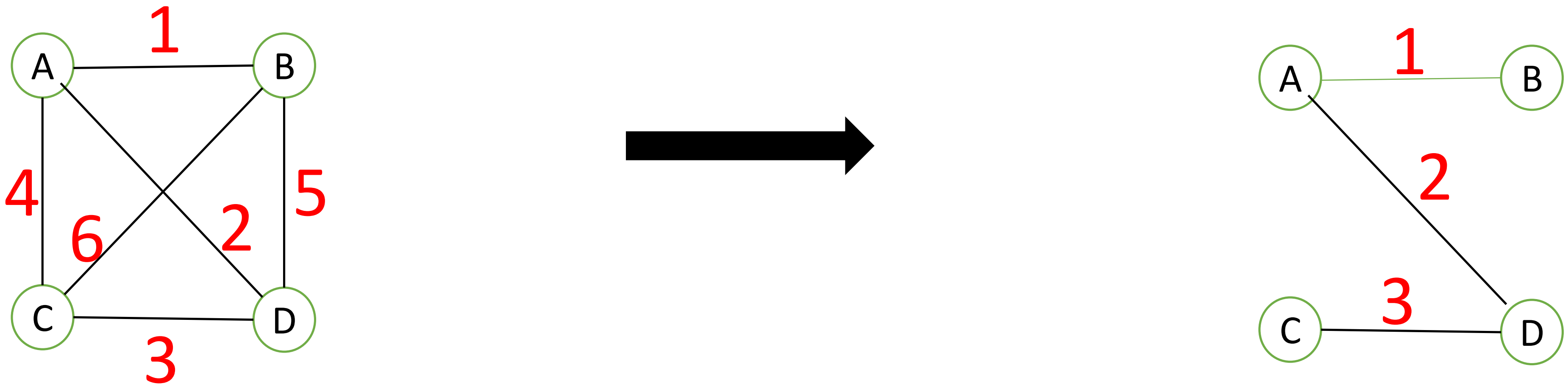


Spanning Tree



Kruskal’s Algorithm

- It is a algorithm used to find a minimum cost spanning tree for connected weighted undirected graph
- This algorithm first appeared in Proceedings of the American Mathematical Society in 1956, and was written by Joseph Kruskal



Connected Weighted

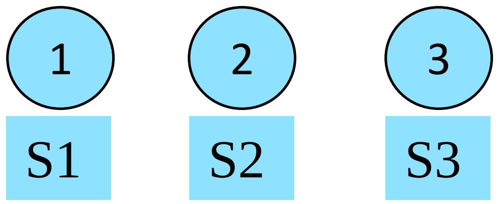
- It's a spanning tree because it connects all vertices without loops.
- Tree weight is minimum of all possibilities hence minimum cost spanning tree

Disjoint Sets

- A disjoint set is a data structure which keeps track of all elements that are separated by a number of disjoint (not connected) subsets.
- It supports three useful operations
  - MAKE-SET(x): Make a new set with a single element x
  - UNION (S1,S2): Merge the set S1 and set S2
  - FIND-SET(x): Find the set containing the element x

Disjoint Sets

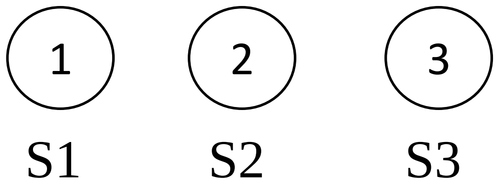
- A disjoint set is a data structure which keeps track of all elements that are separated by a number of disjoint (not connected) subsets.
- It supports three useful operations
  - MAKE-SET(x): Make a new set with a single element x
  - UNION (S1,S2): Merge the set S1 and set S2
  - FIND-SET(x): Find the set containing the element x



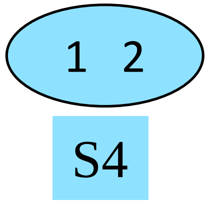
MAKE-SET(1), MAKE-SET(2), MAKE-SET(3) creates new set S1,S2,S3

Disjoint Sets

- A disjoint set is a data structure which keeps track of all elements that are separated by a number of disjoint (not connected) subsets.
- It supports three useful operations
  - MAKE-SET(x): Make a new set with a single element x
  - UNION (S1,S2): Merge the set S1 and set S2
  - FIND-SET(x): Find the set containing the element x



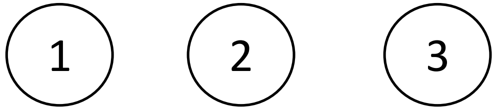
MAKE-SET(1), MAKE-SET(2), MAKE-SET(3) creates new set S1,S2,S3



UNION (1,2) merge set S1 and set S2 to create set S4

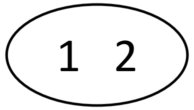
Disjoint Sets

- A disjoint set is a data structure which keeps track of all elements that are separated by a number of disjoint (not connected) subsets
- It supports three useful operations
    - MAKE-SET(x): Make a new set with a single element x
    - UNION (S1,S2): Merge the set S1 and set S2
    - FIND-SET(x): Find the set containing the element x



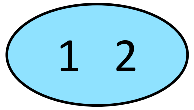
S1 S2 S3

MAKE-SET(1), MAKE-SET(2), MAKE-SET(3) creates new set S1,S2,S3



S4

UNION (1,2) merge set S1 and set S2 to create set S4



S4

FIND-SET(2) returns set S4

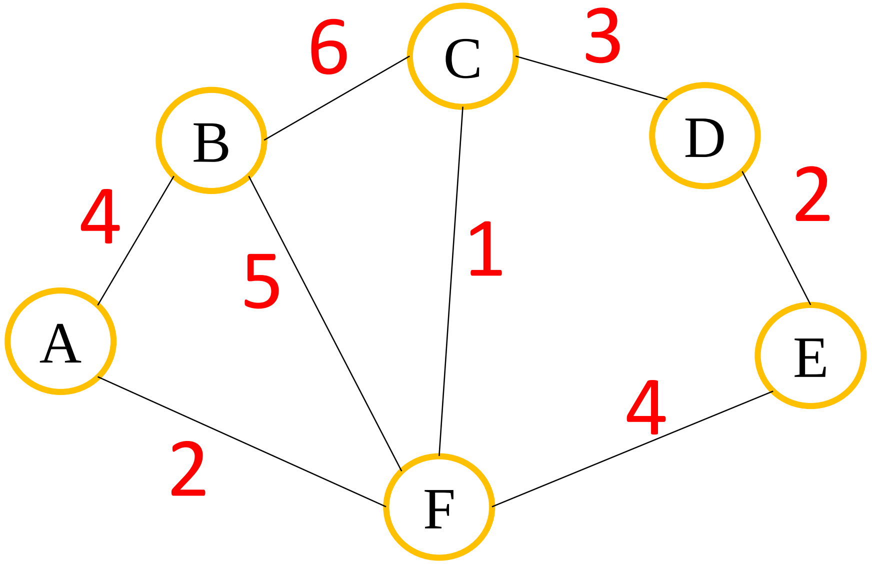
Kruskal's Algorithm

KRUSKAL(V,E):

A = ∅
foreach v∈ V:
  MAKE-SET(v)
Sort E by weight increasingly
foreach (v_1,v_2) ∈ E:
  if FIND-SET(v_1) ≠ FIND-SET(v_2):
    A = A ∪ {(v_1,v_2)}
    UNION(v_1,v_2)
  else
    Remove edge (v_1,v_2)
return A

KRUSKAL(V,E):

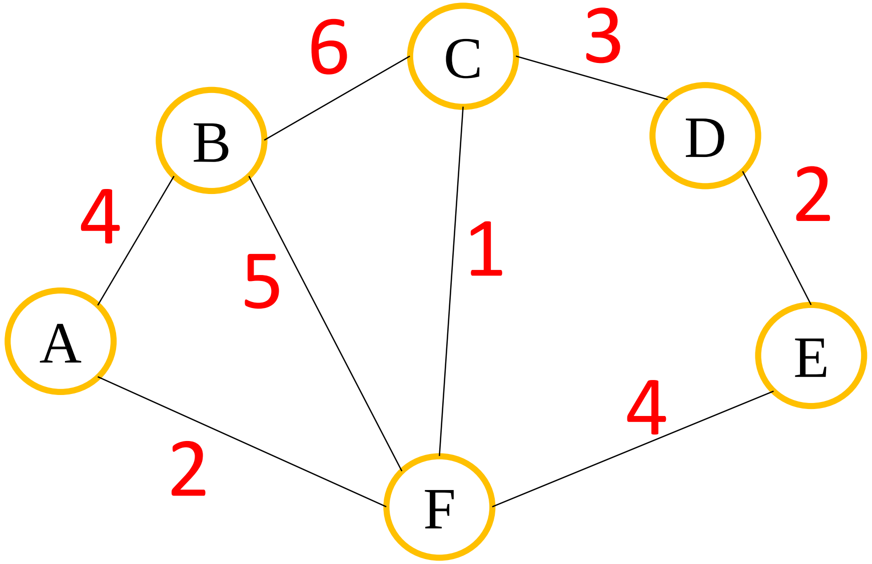
$A = \emptyset$
foreach $v \in V$:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in E$:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        $A = A \cup \{(v_1, v_2)\}$
        UNION(v_1, v_2)
    else
        Remove edge (v_1, v_2)
return A



Edges	Weight
AB	4
BC	6
CD	3
DE	2
EF	4
AF	2
BF	5
CF	1

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A $\cup$ $\{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A

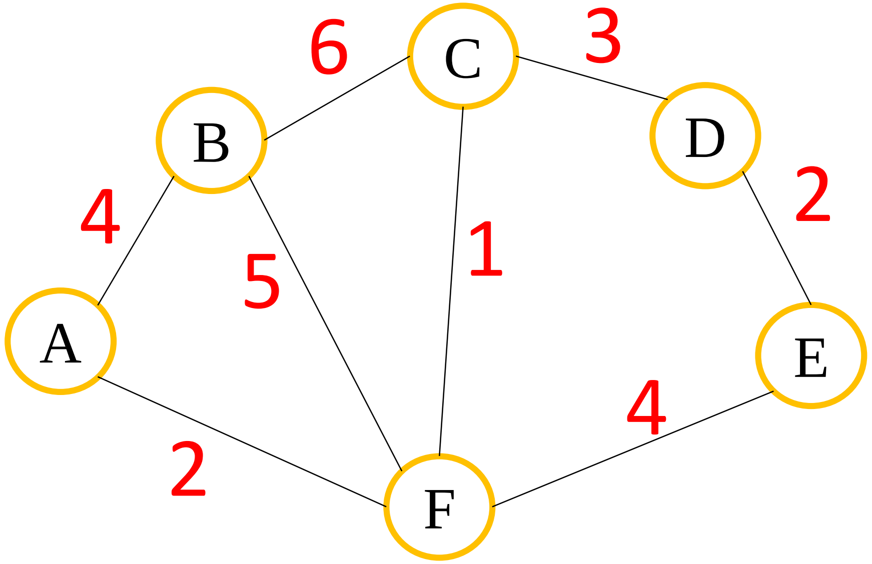


A ={ }

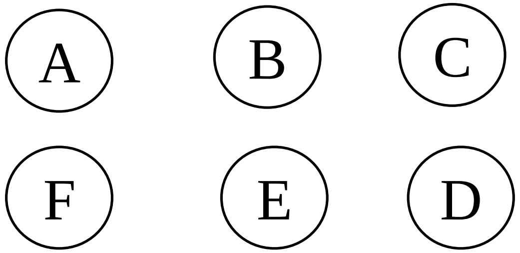
Edges	Weight
AB	4
BC	6
CD	3
DE	2
EF	4
AF	2
BF	5
CF	1

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ $\{(v_1, v_2)\}$
        UNION(v_1, v_2)
    else
        Remove edge (v_1, v_2)
return A



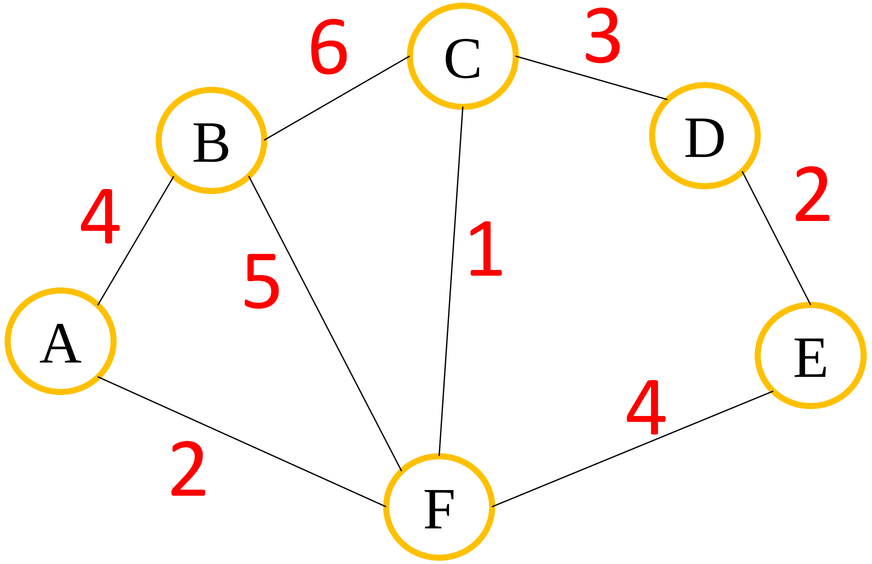
A ={ }



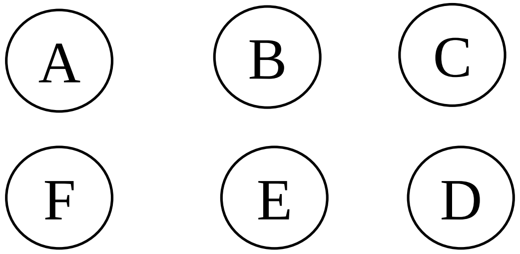
Edges	Weight
AB	4
BC	6
CD	3
DE	2
EF	4
AF	2
BF	5
CF	1

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ $\{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



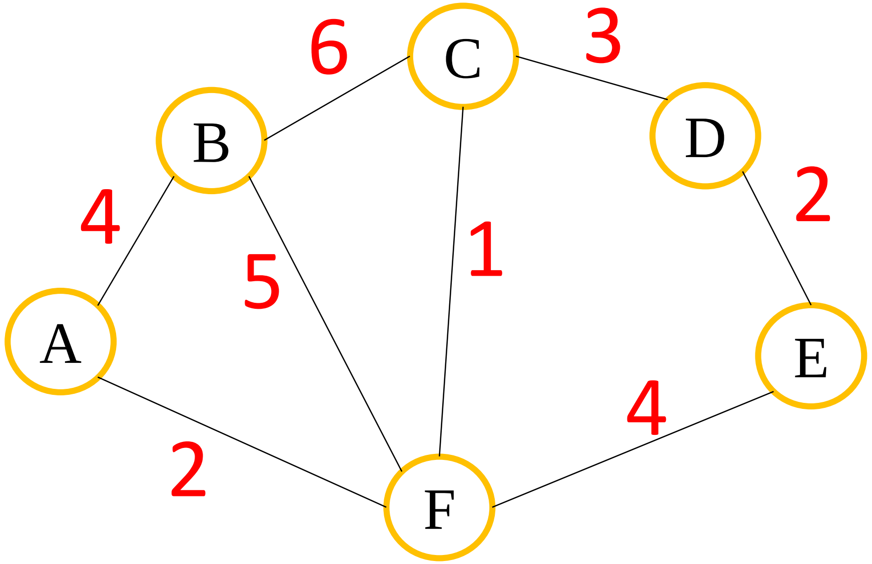
A ={ }



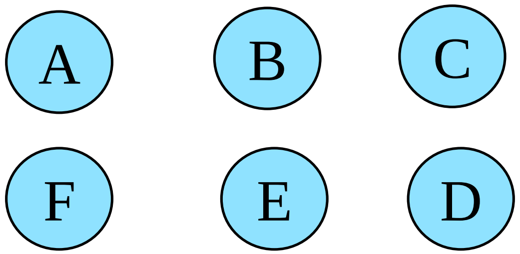
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ $\{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



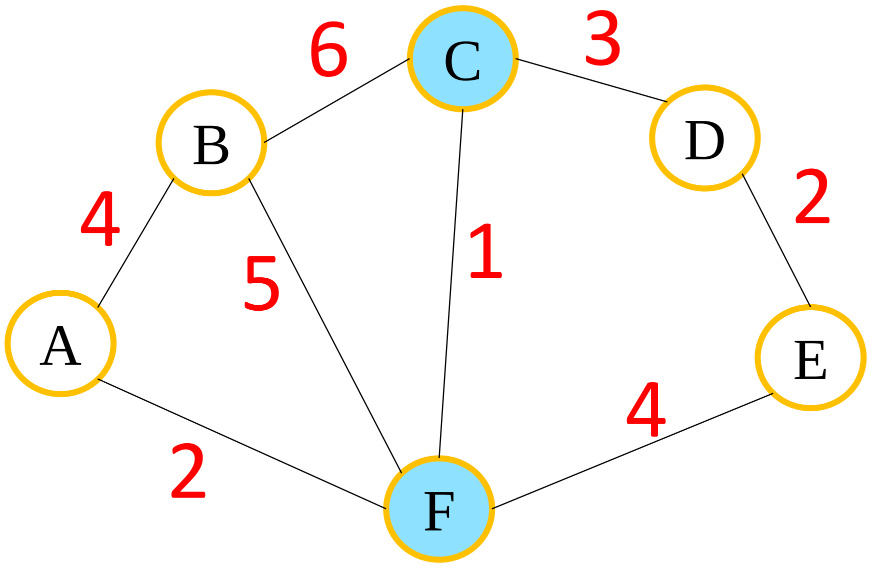
A ={ }



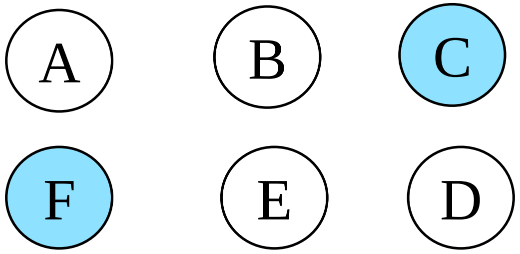
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in$ V:
  MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    $A = A \cup \{(v_1,v_2)\}$
    UNION(v_1,v_2)
  else
    Remove edge (v_1,v_2)
return A



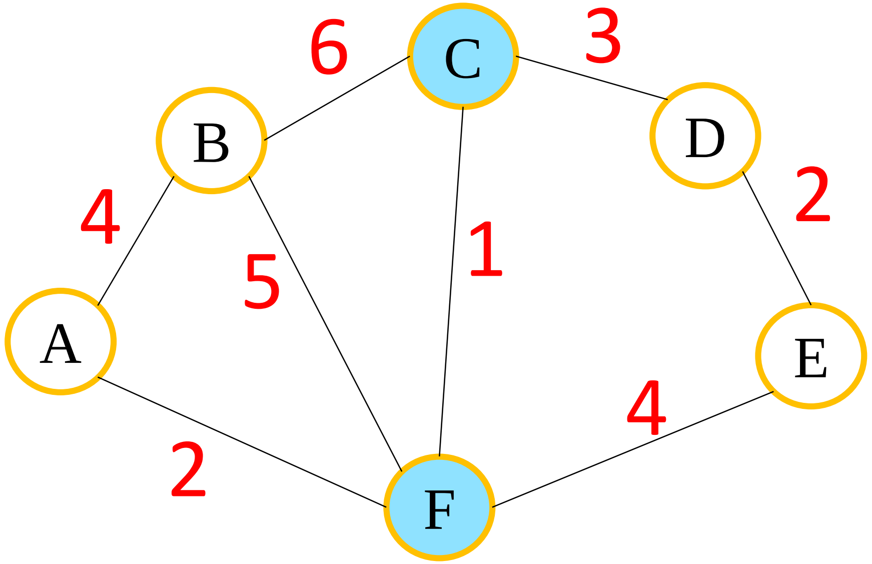
A ={ }



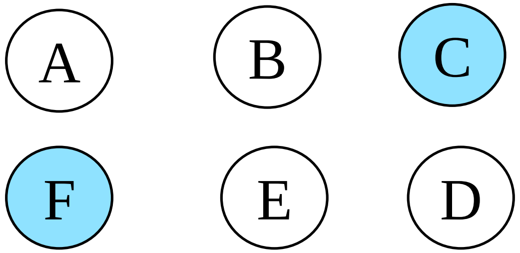
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in E$:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        $A = A \cup \{(v_1, v_2)\}$
        UNION(v_1, v_2)
    else
        Remove edge (v_1, v_2)
return A



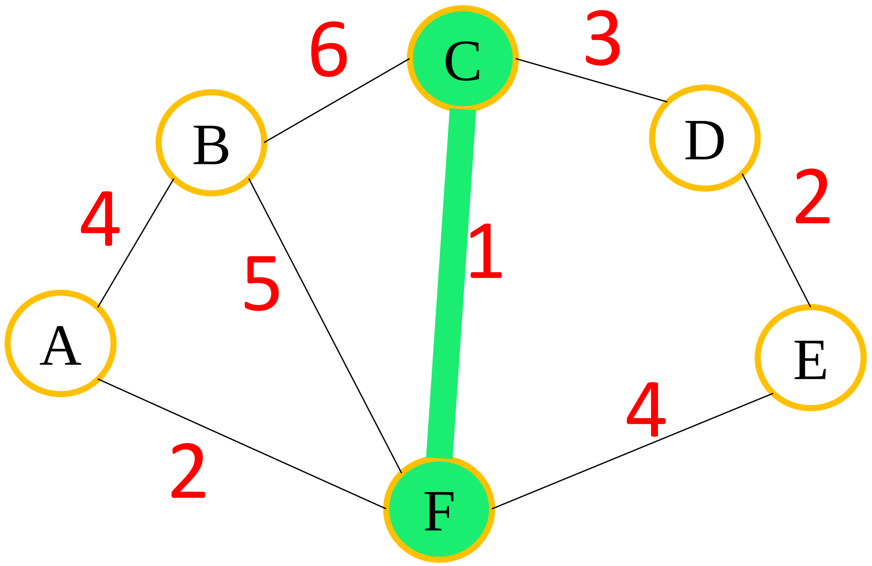
A ={(C,F)}



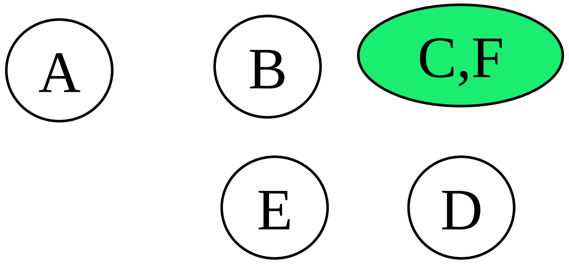
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
  MAKE-SET(v)
Sort E by weight increasingly
foreach (v_1,v_2) ∈ E:
  if FIND-SET(v_1) ≠ FIND-SET(v_2):
    A = A ∪ {(v_1,v_2)}
    UNION(v_1,v_2)
  else
    Remove edge (v_1,v_2)
return A



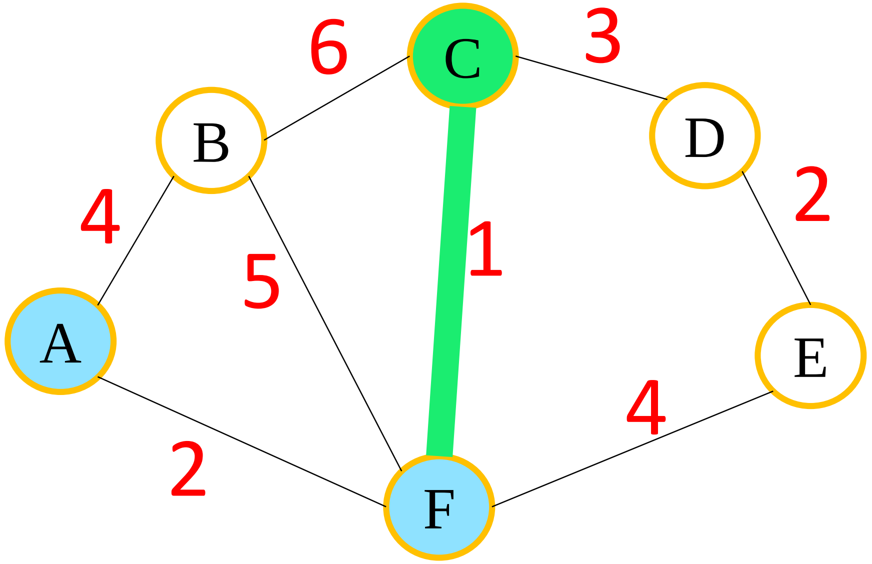
A ={(C,F)}



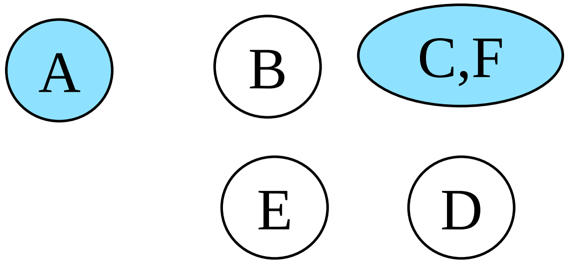
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in V$:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in E$:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ $\{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



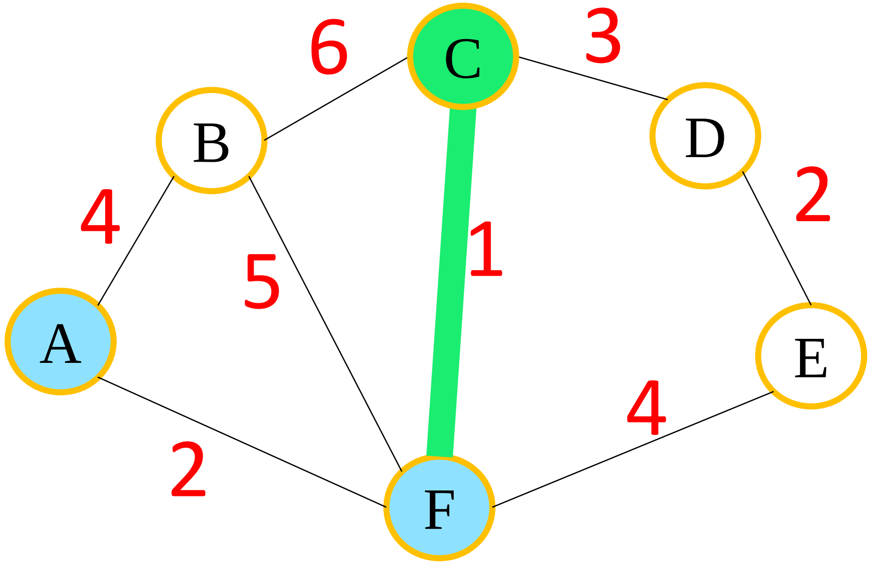
A ={(C,F)}



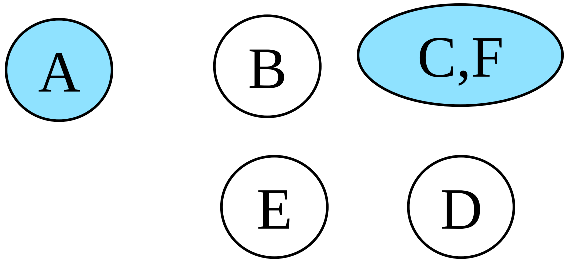
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ $\{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



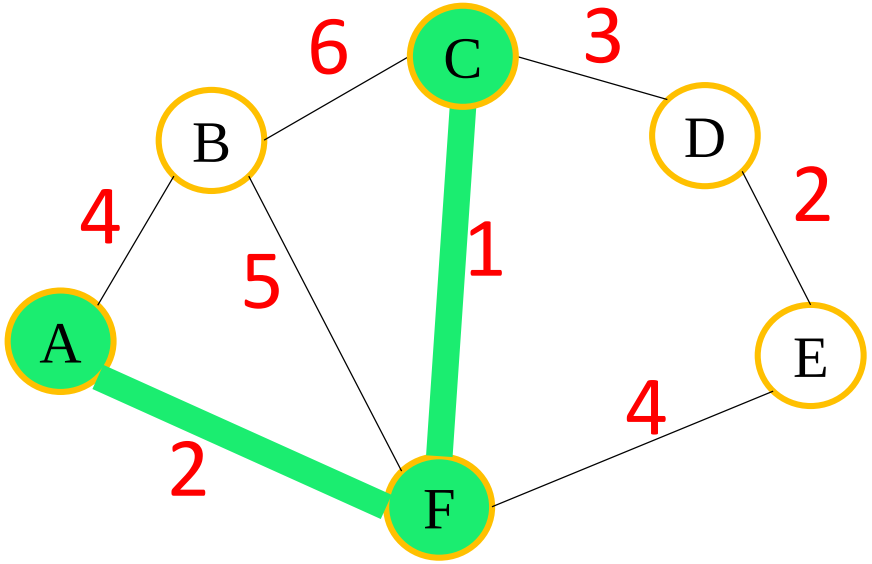
A ={(C,F),(A,F)}



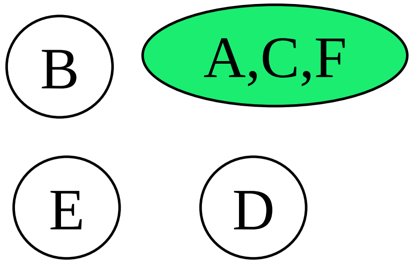
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
if FIND-SET(v_1) ≠ FIND-SET(v_2):
A = A ∪ $\{(v_1,v_2)\}$
UNION(v_1,v_2)
else
Remove edge (v_1,v_2)
return A



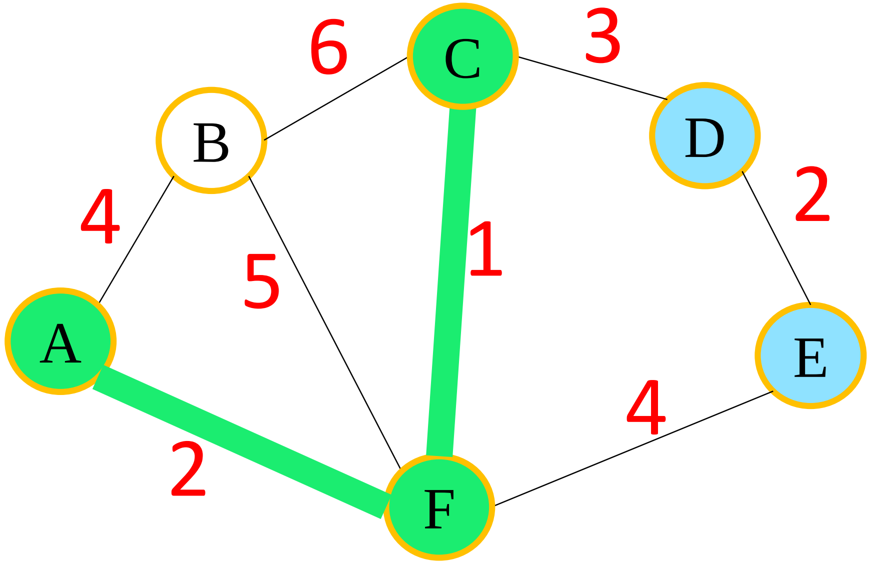
A ={(C,F),(A,F)}



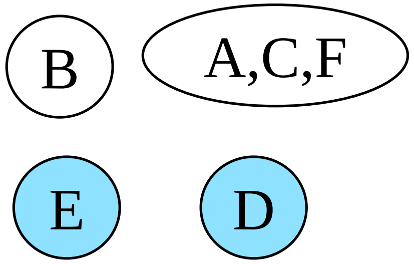
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
  MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in E$:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    $A = A \cup \{(v_1, v_2)\}$
    UNION(v_1, v_2)
  else
    Remove edge (v_1, v_2)
return A



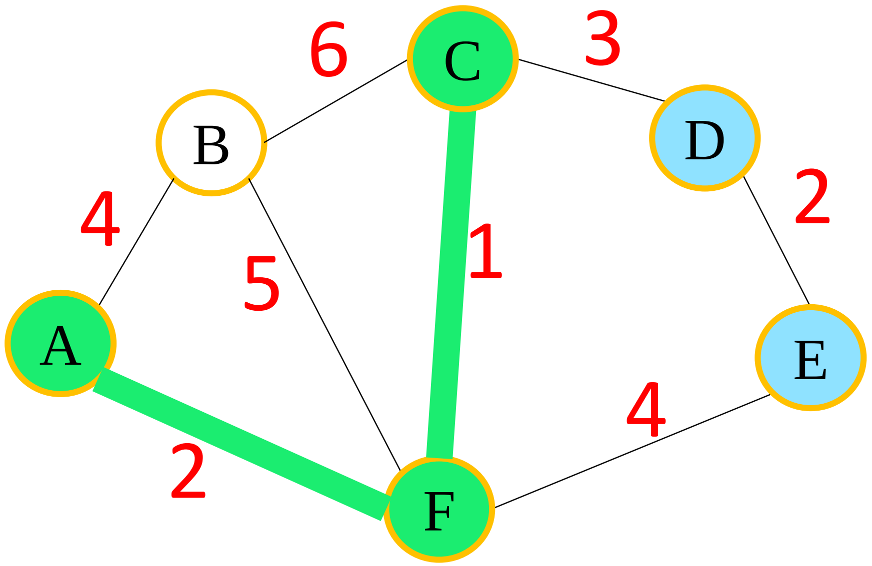
A ={(C,F),(A,F)}



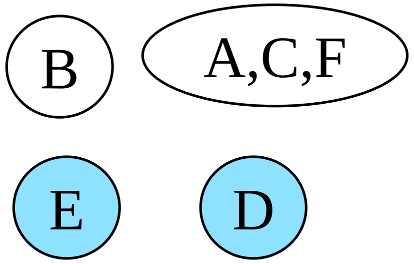
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
  MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in$ E:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    A = A ∪ {(v_1, v_2)}
    UNION(v_1, v_2)
  else
    Remove edge (v_1, v_2)
return A



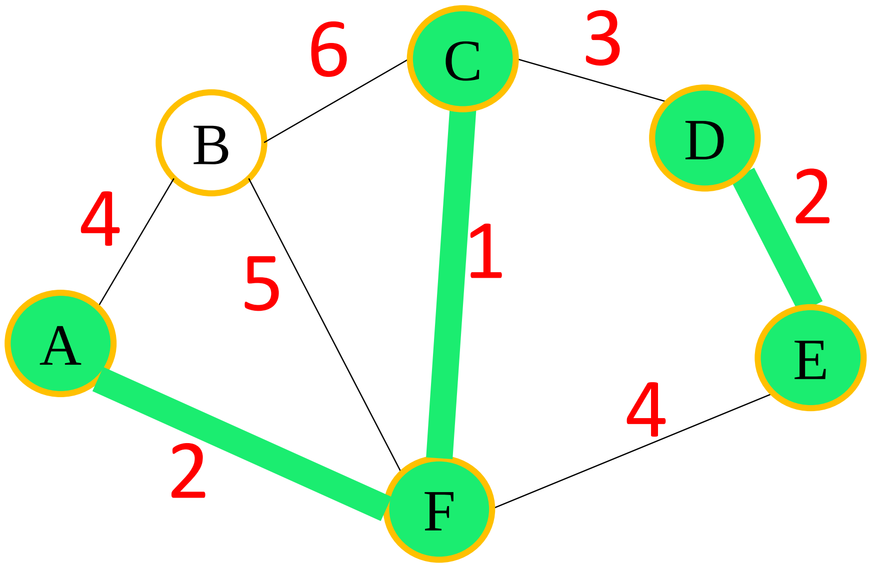
A ={(C,F),(A,F),(D,E)}



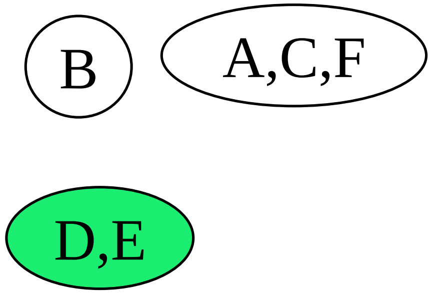
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
  MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in E$:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    $A = A \cup \{(v_1,v_2)\}$
    UNION(v_1,v_2)
  else
    Remove edge (v_1,v_2)
return A



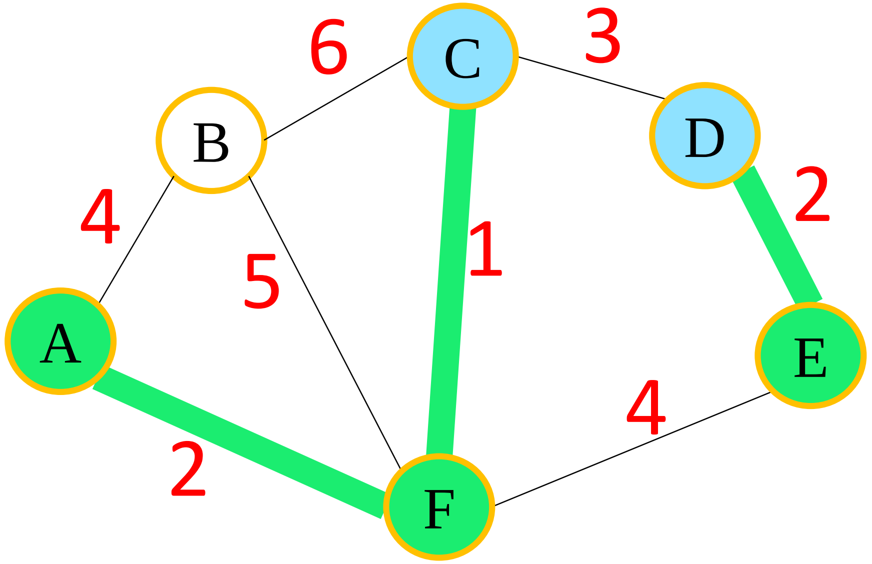
$A = \{(C,F),(A,F),(D,E)\}$



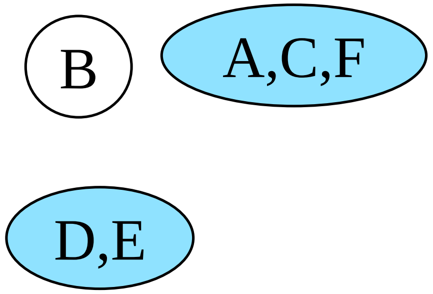
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in$ E:
    if FIND-SET(v_1) ≠ FIND-SET(v_2):
        A = A ∪ {(v_1, v_2)}
        UNION(v_1, v_2)
    else
        Remove edge (v_1, v_2)
return A



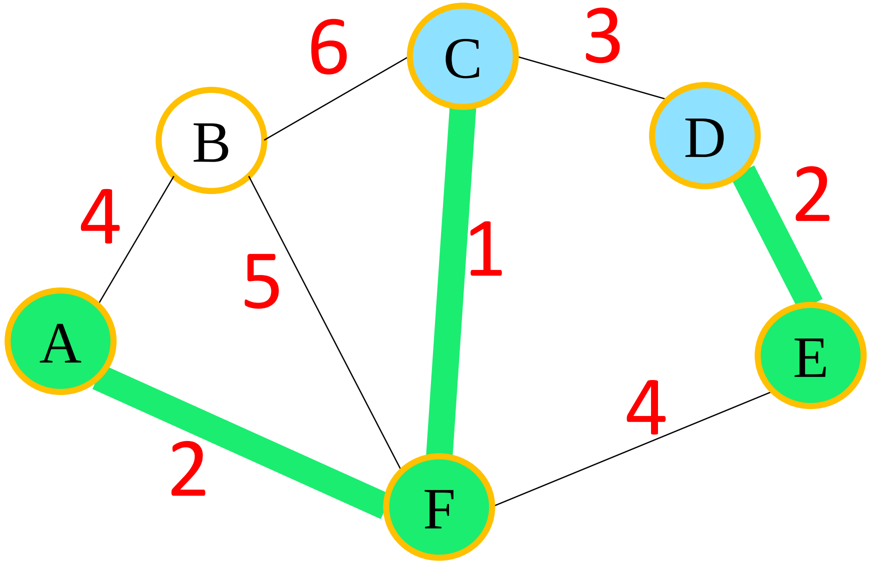
A ={(C,F),(A,F),(D,E)}



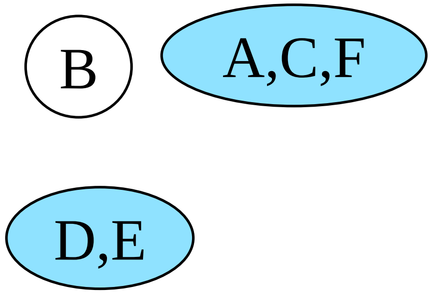
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in E$:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        $A = A \cup \{(v_1, v_2)\}$
        UNION(v_1, v_2)
    else
        Remove edge (v_1, v_2)
return A



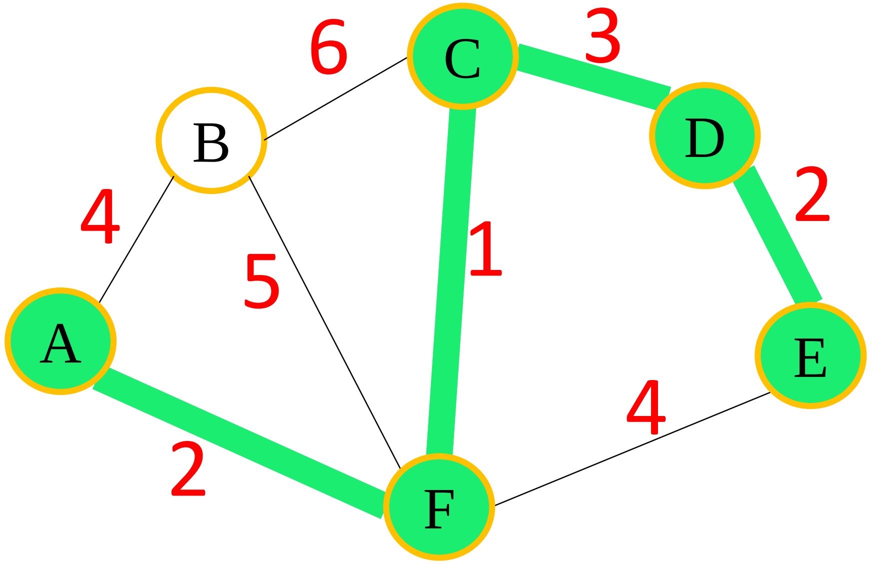
A ={(C,F),(A,F),(D,E),(C,D)}



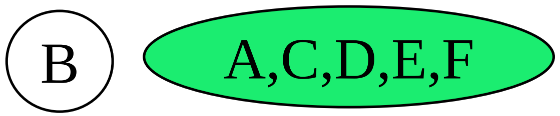
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ {(v_1,v_2)}
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



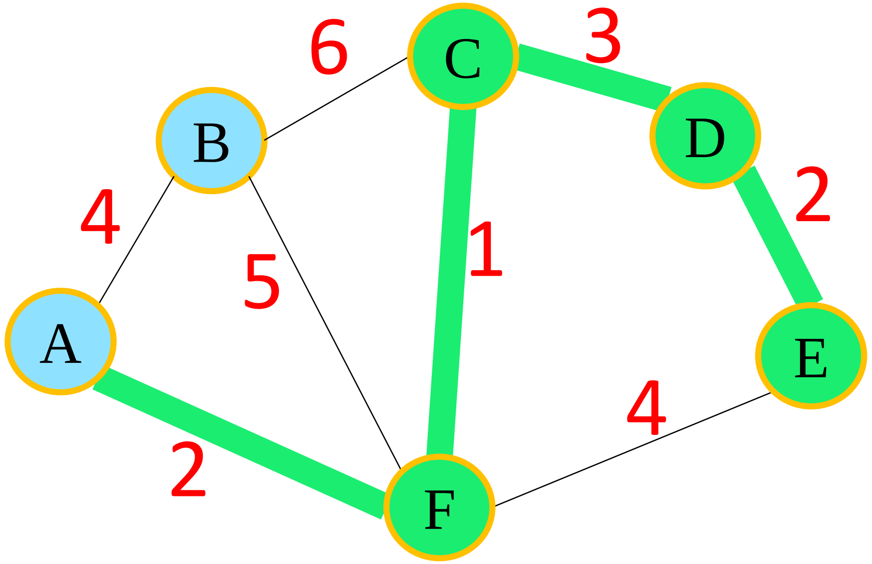
A ={(C,F),(A,F),(D,E),
(C,D)}



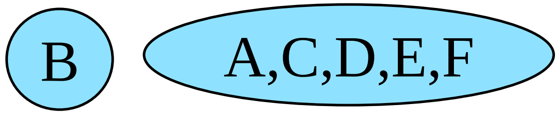
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in E$:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        $A = A \cup \{(v_1, v_2)\}$
        UNION(v_1, v_2)
    else
        Remove edge (v_1, v_2)
return A



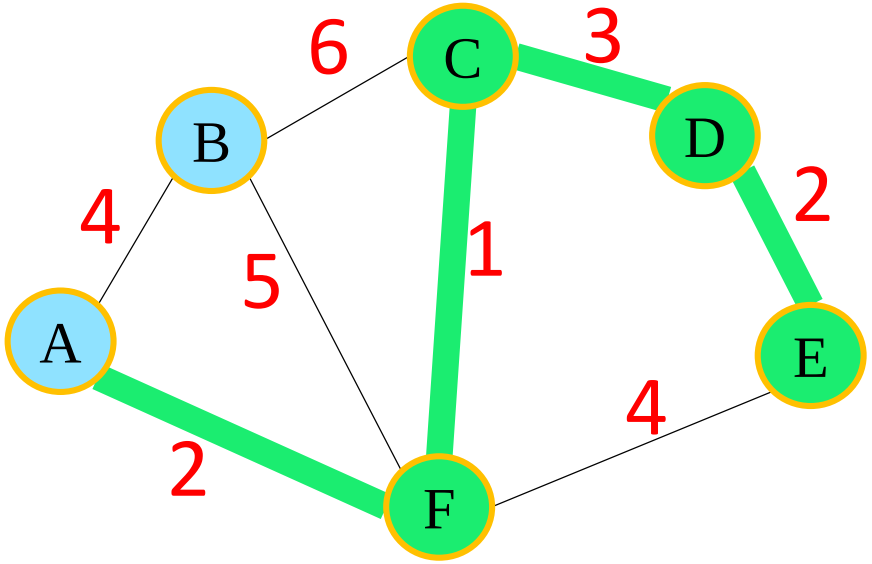
A ={(C,F),(A,F),(D,E),(C,D)}



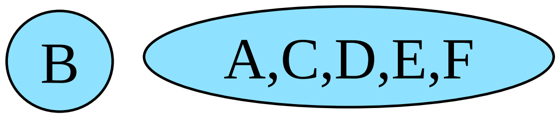
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

  A = ∅
  foreach $v \in$ V:
    MAKE-SET(v)
  Sort E by weight increasingly
  foreach $(v_1, v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
      A = A ∪ $\{(v_1, v_2)\}$
      UNION(v_1, v_2)
    else
      Remove edge (v_1, v_2)
  return A



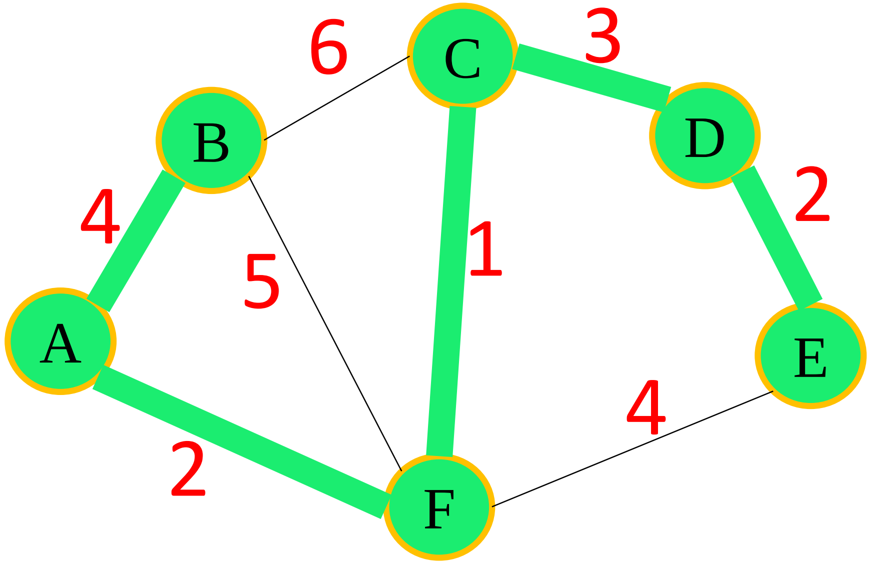
A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}



Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
  MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in E$:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    $A = A \cup \{(v_1,v_2)\}$
    UNION(v_1,v_2)
  else
    Remove edge (v_1,v_2)
return A



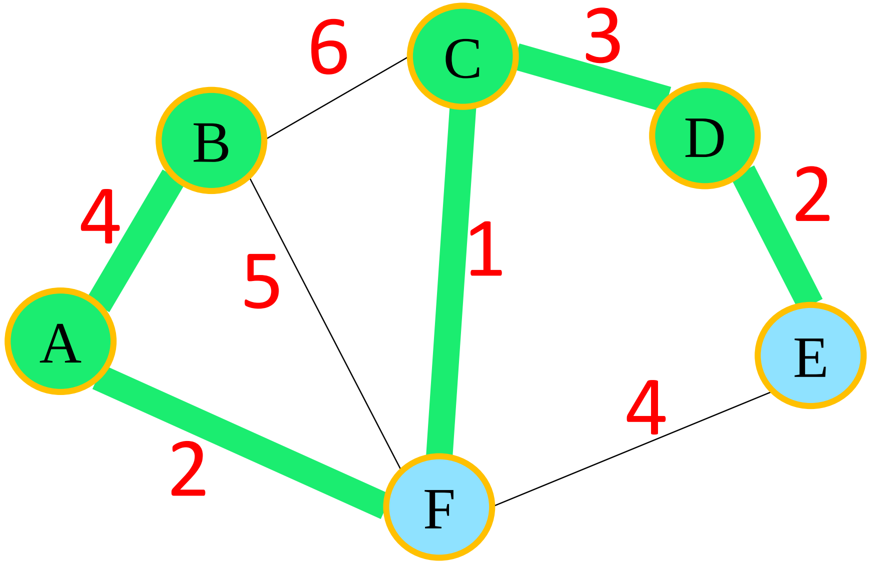
A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}

A,B,C,D,E,F

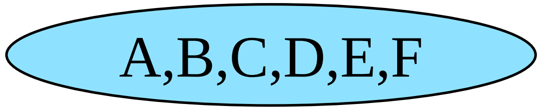
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in E$:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        $A = A \cup \{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



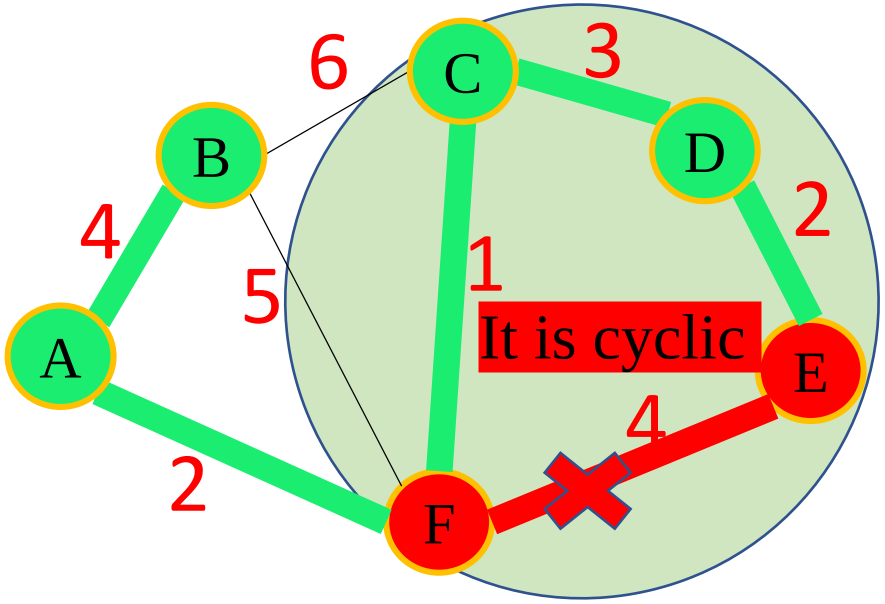
$A = \{(C,F),(A,F),(D,E),(C,D),(A,B)\}$



Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in V$:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in E$:
    if FIND-SET(v_1) ≠ FIND-SET(v_2):
        A = A ∪ {(v_1,v_2)}
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}

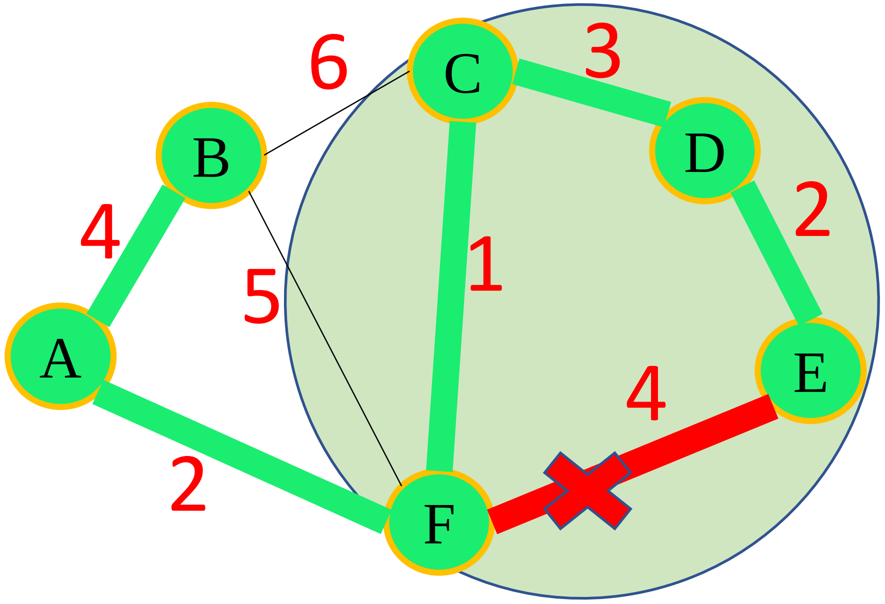
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

A,B,C,D,E,F

Is in a same set.

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ {(v_1,v_2)}
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



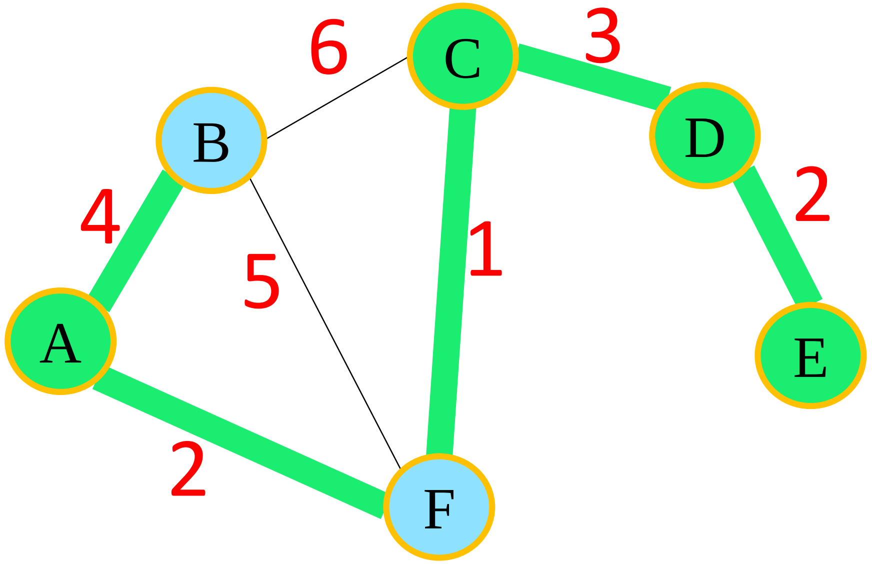
A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}

A,B,C,D,E,F

Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

> A = ∅
> **foreach** $v \in$ V:
>   MAKE-SET(v)
> Sort E by weight increasingly
> **foreach** $(v_1,v_2) \in$ E:
>   **if** FIND-SET(v_1) $\neq$ FIND-SET(v_2):
>   A = A ∪ $\{(v_1,v_2)\}$
>   UNION(v_1,v_2)
>   **else**
>   Remove edge (v_1,v_2)
> **return** A



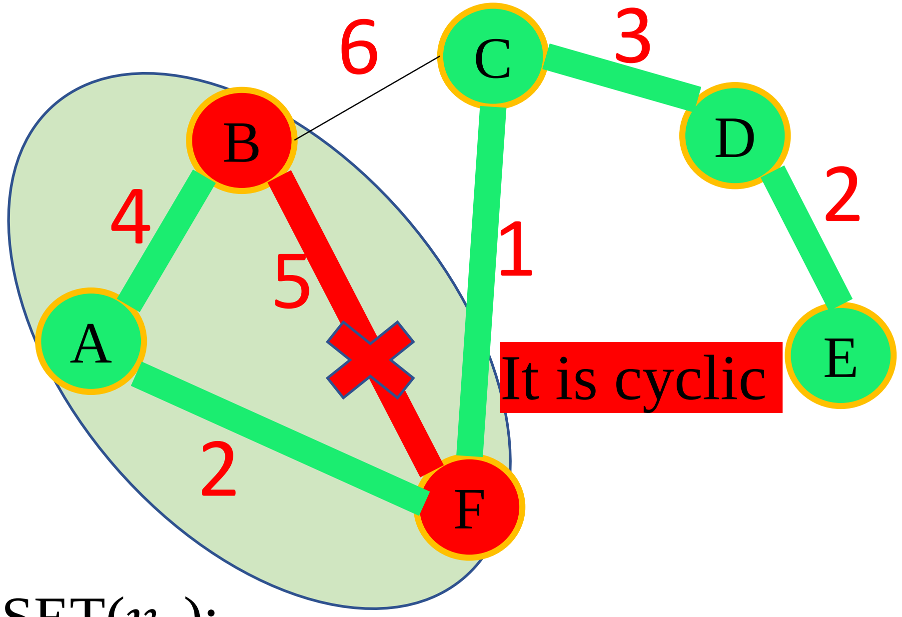
A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}

A,B,C,D,E,F

Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach v ∈ V:
  MAKE-SET(v)
Sort E by weight increasingly
foreach (v_1,v_2) ∈ E:
  if FIND-SET(v_1) ≠ FIND-SET(v_2):
    A = A ∪ {(v_1,v_2)}
    UNION(v_1,v_2)
  else
    Remove edge (v_1,v_2)
return A



A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}

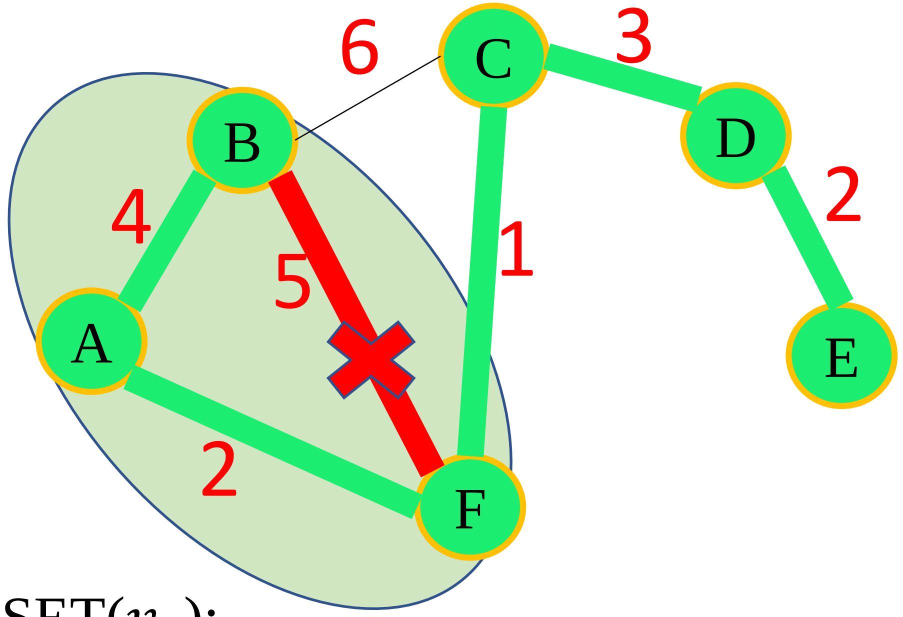
A,B,C,D,E,F

Is in a same set.

Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
MAKE-SET(v)
Sort E by weight increasingly
foreach (v_1,v_2) ∈ E:
if FIND-SET(v_1) ≠ FIND-SET(v_2):
A = A ∪ {(v_1,v_2)}
UNION(v_1,v_2)
else
Remove edge (v_1,v_2)
return A



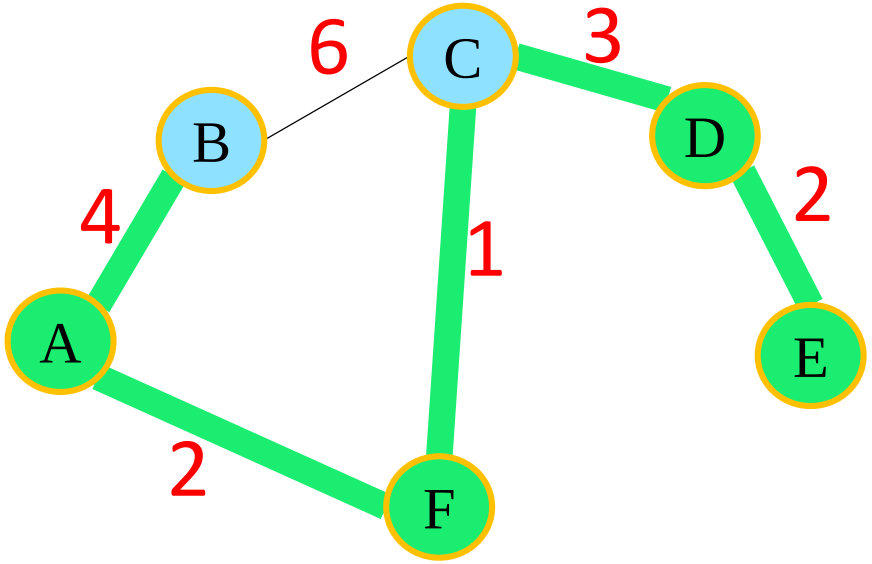
A ={(C,F),(A,F),(D,E),(C,D),(A,B)}

A,B,C,D,E,F

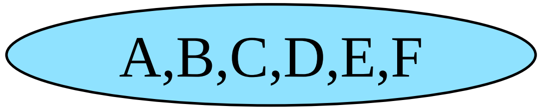
Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in E$:
if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
$A = A \cup \{(v_1,v_2)\}$
UNION(v_1,v_2)
else
Remove edge (v_1,v_2)
return A



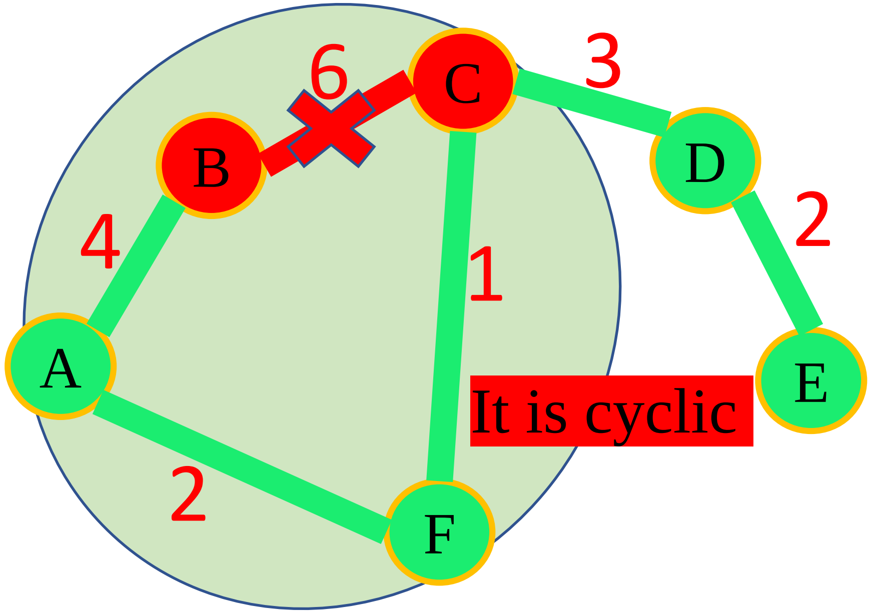
A ={(C,F),(A,F),(D,E),(C,D),(A,B)}



Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A $\cup$ $\{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)
return A



A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}

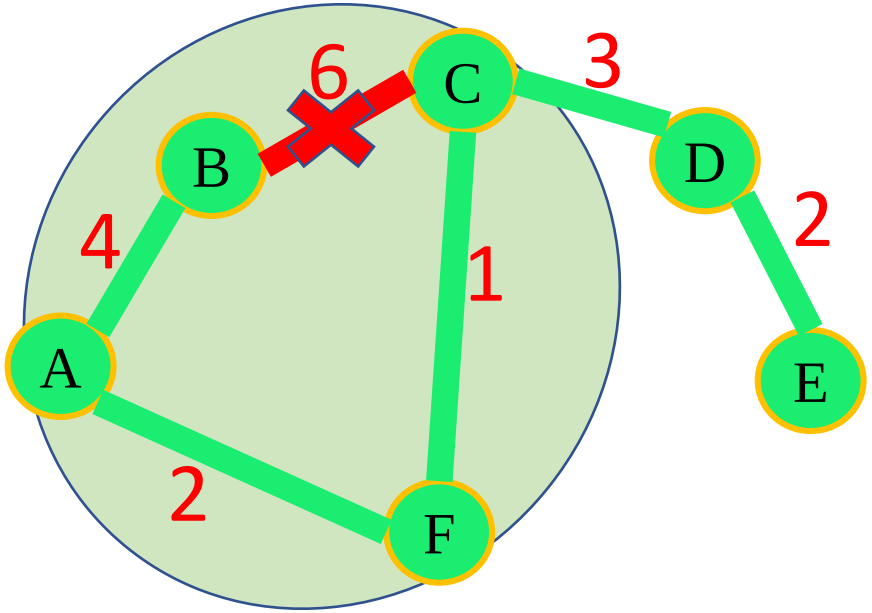
A,B,C,D,E,F

Is in a same set.

Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
  MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in E$:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    $A = A \cup \{(v_1, v_2)\}$
    UNION(v_1, v_2)
  else
    Remove edge (v_1, v_2)
return A



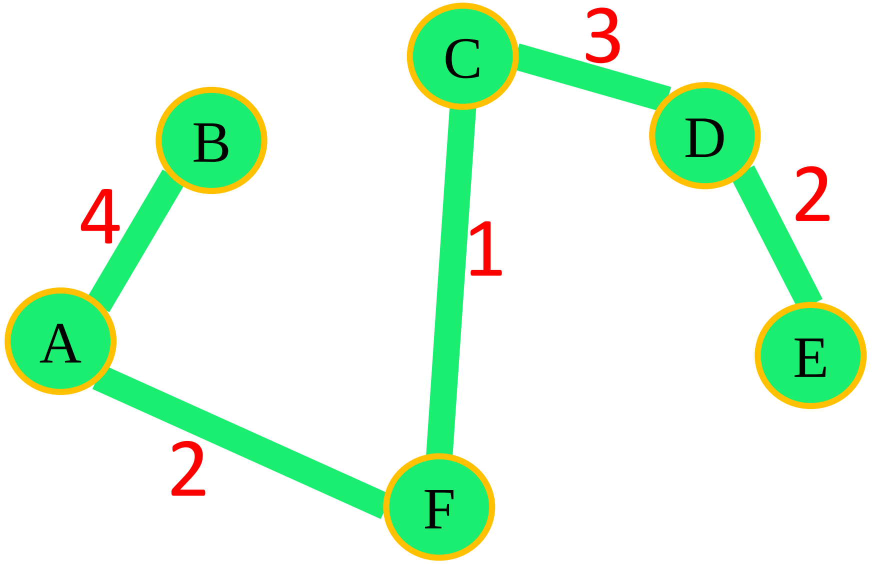
$A = \{(C,F),(A,F),(D,E),(C,D),(A,B)\}$

A,B,C,D,E,F

Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

$A = \emptyset$
foreach $v \in V$:
  MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1,v_2) \in E$:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    $A = A \cup \{(v_1,v_2)\}$
    UNION(v_1,v_2)
  else
    Remove edge (v_1,v_2)
return A

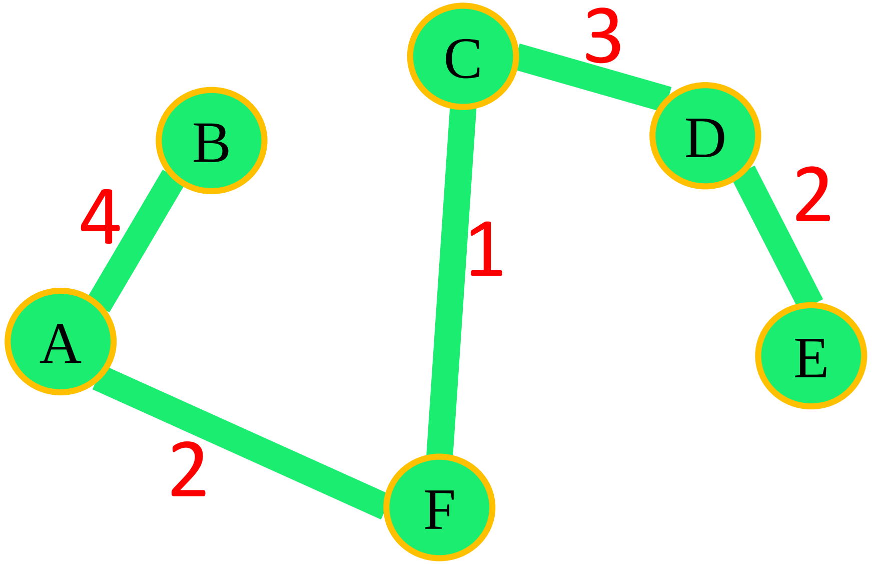


A ={(C,F),(A,F),(D,E),
(C,D),(A,B)}

Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

KRUSKAL(V,E):

A = ∅
foreach $v \in$ V:
    MAKE-SET(v)
Sort E by weight increasingly
foreach $(v_1, v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A ∪ $\{(v_1, v_2)\}$
        UNION(v_1, v_2)
    else
        Remove edge (v_1, v_2)
return A



Edges	Weight
CF	1
AF	2
DE	2
CD	3
AB	4
FE	4
BF	5
BC	6

A ={(C,F),(A,F),(D,E),(C,D),(A,B)}

Total Weight :- 1 + 2 + 2 + 3 + 4
= 12

Time Complexity

KRUSKAL(V,E):

O(1) — $A = \emptyset$

O(V) — **foreach** $v \in$ V:
    MAKE-SET(v)

O(E log E) — Sort E by weight increasingly

O(E log V) — **foreach** $(v_1,v_2) \in$ E:
    if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
        A = A $\cup$ $\{(v_1,v_2)\}$
        UNION(v_1,v_2)
    else
        Remove edge (v_1,v_2)

return A

Time Complexity

KRUSKAL(V,E):

O(1)
A = ∅

O(V)
foreach $v \in$ V:
  MAKE-SET(v)

O(E log E)
Sort E by weight increasingly

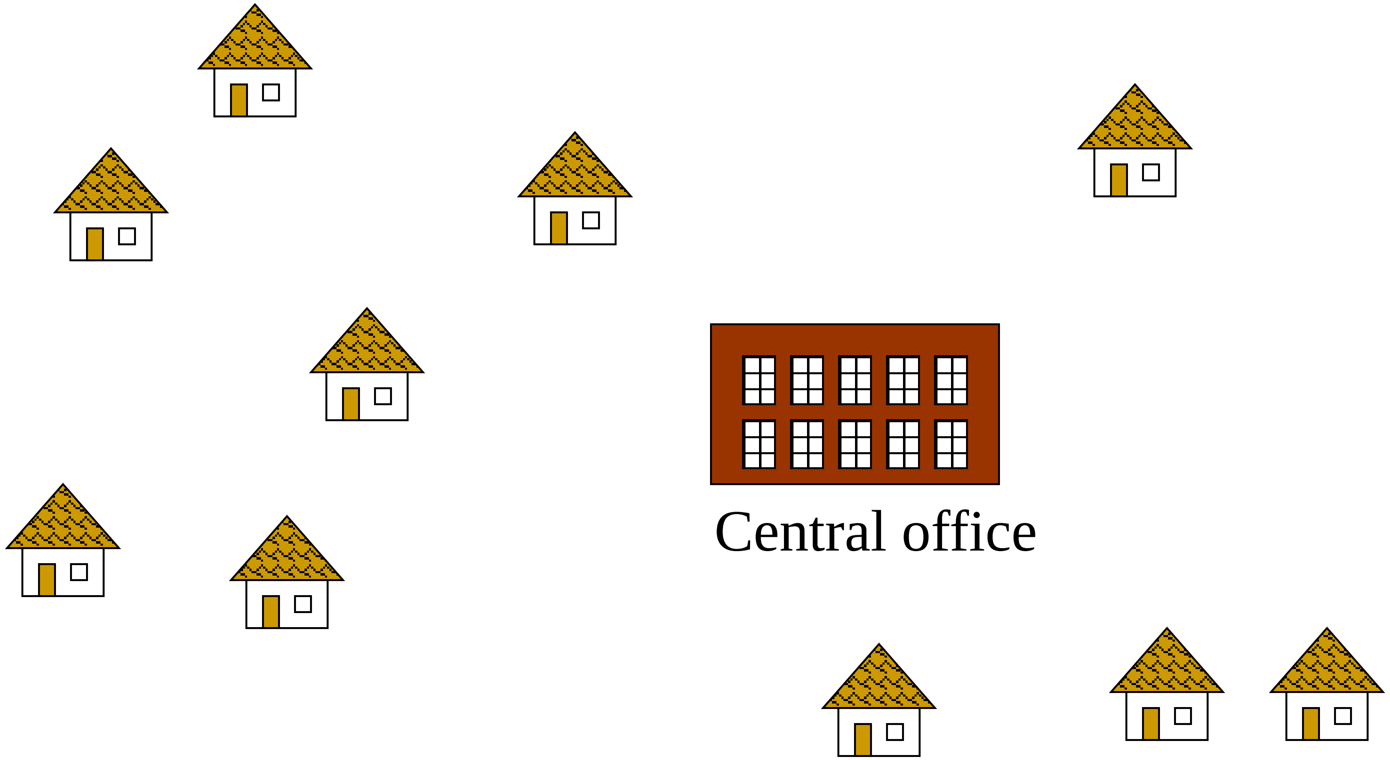
O(E log V)
foreach $(v_1, v_2) \in$ E:
  if FIND-SET(v_1) $\neq$ FIND-SET(v_2):
    A = A $\cup$ $\{(v_1, v_2)\}$
    UNION(v_1, v_2)
  else
    Remove edge (v_1, v_2)
return A

Time Complexity = O(1)+ O(V)+O(E log E)+O(E log V)
= O(E log E) + O(E log V)
Since, $|E| \leq |V|^2 \Rightarrow \log |E| = O(2 \log V) = O(\log V)$.
= O(E log V) + O(E log V)
= O(E log V)

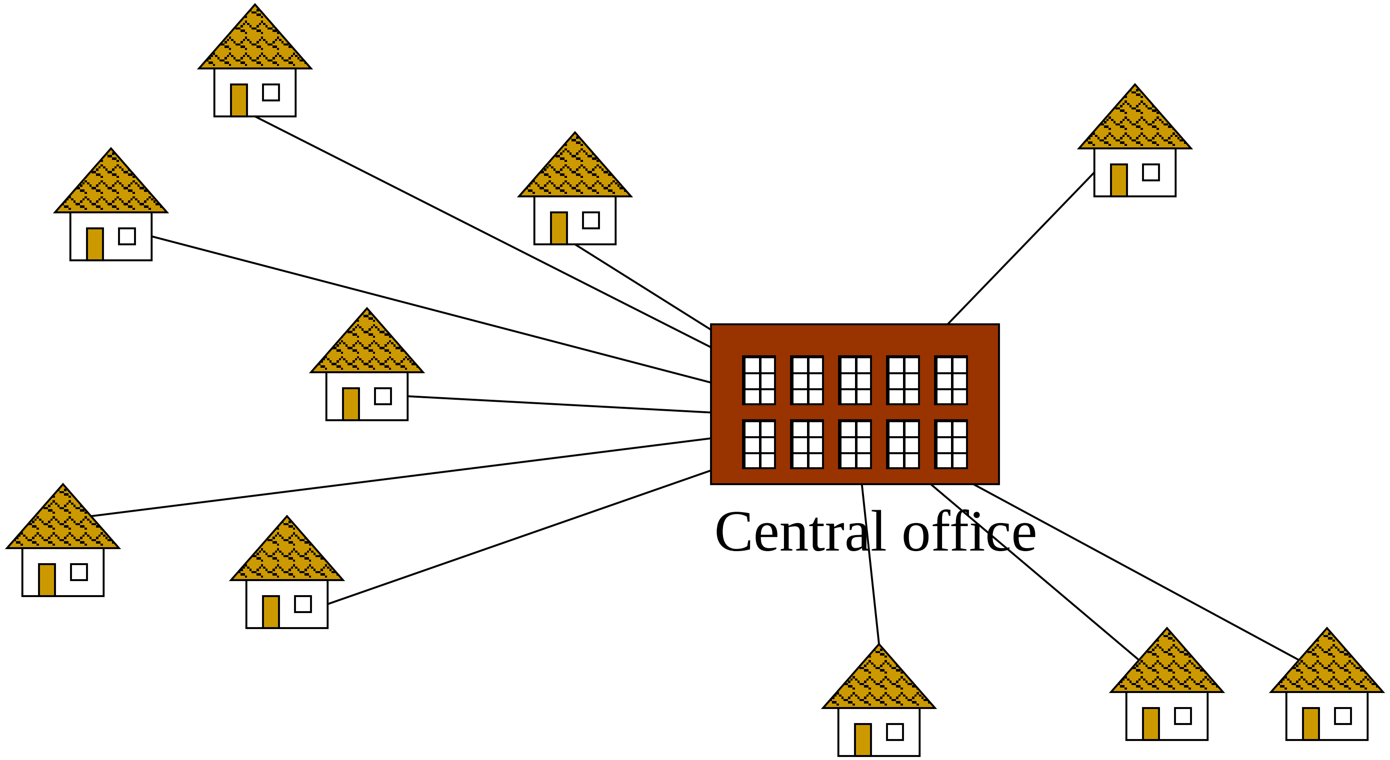
Real-life applications of Kruskal's algorithm

- Landing Cables
- TV Network
- Tour Operations
- Computer networking
- Study of Molecular bonds in Chemistry
- Routing Algorithms

Problem: Laying Telephone Wire

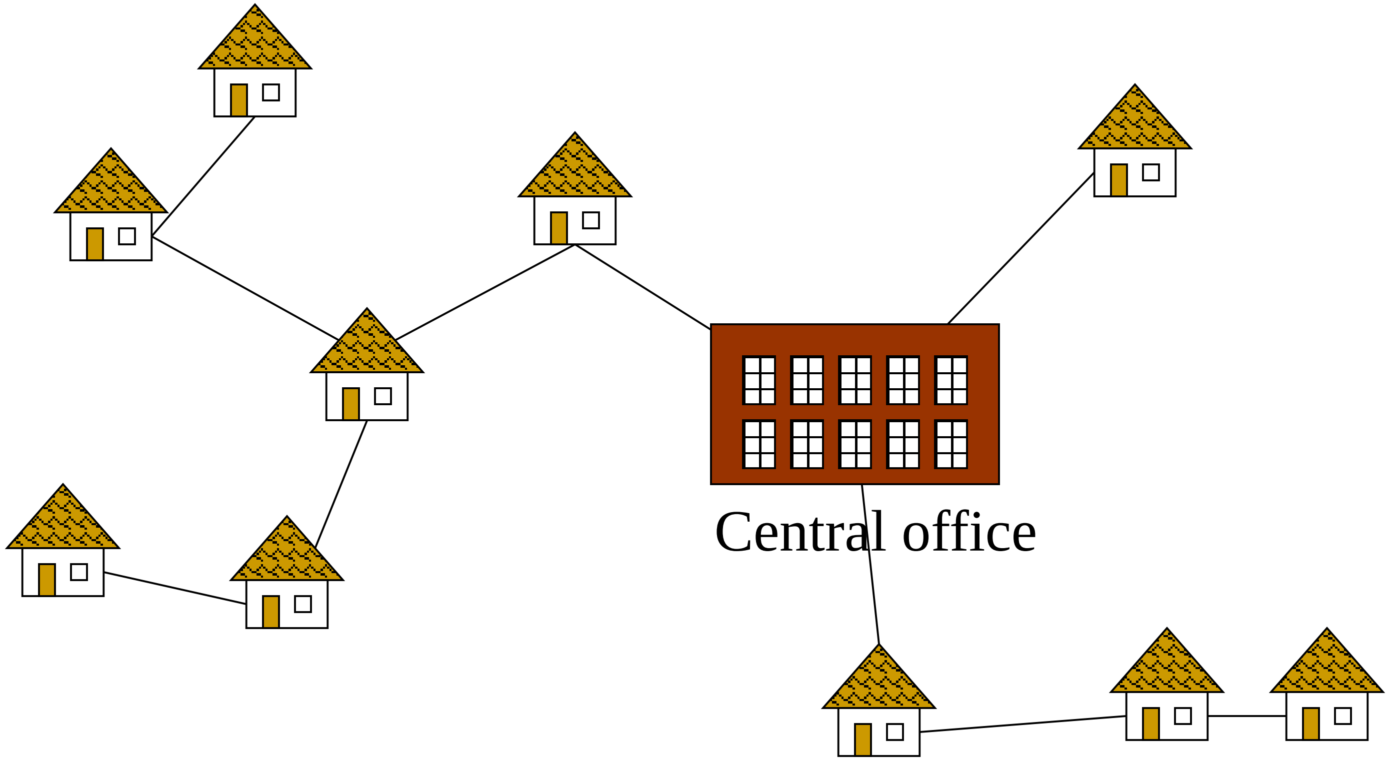


Wiring: Naïve Approach



Expensive!

Wiring: Better Approach



Minimize the total length of wire connecting the customers