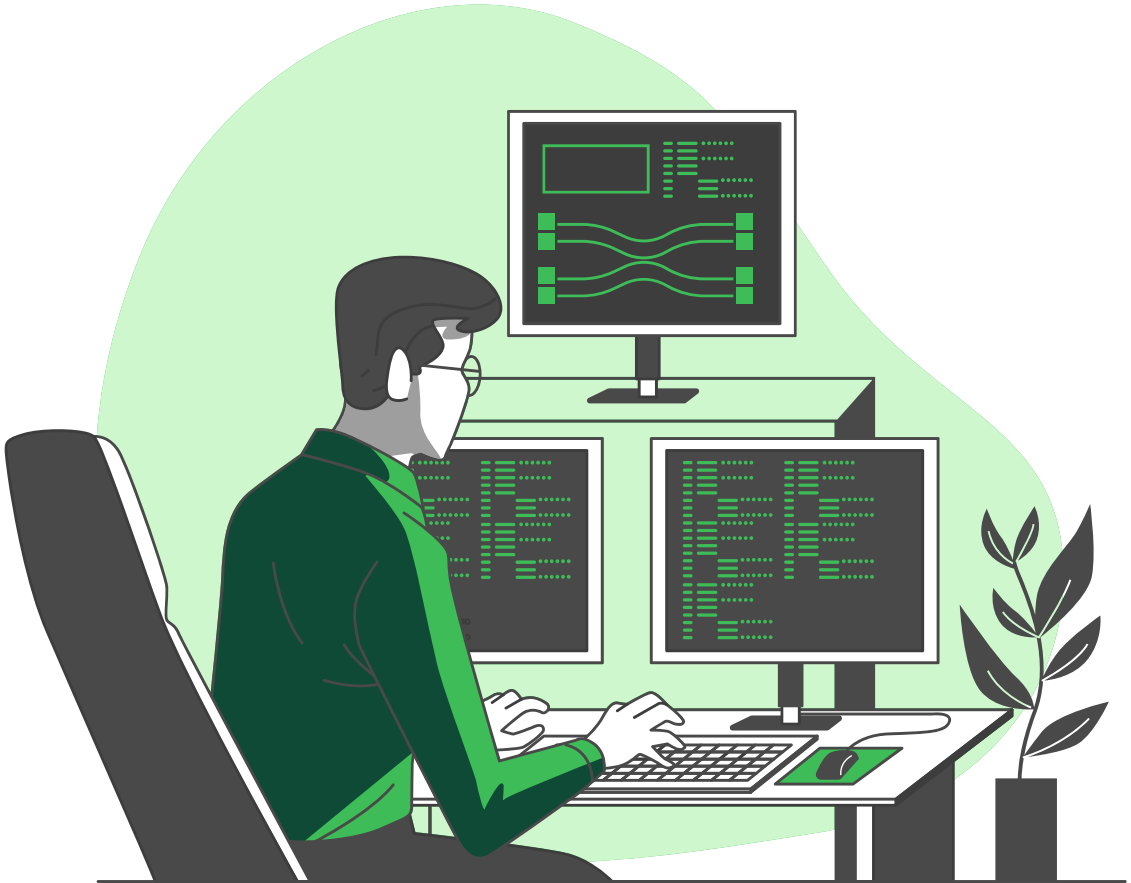
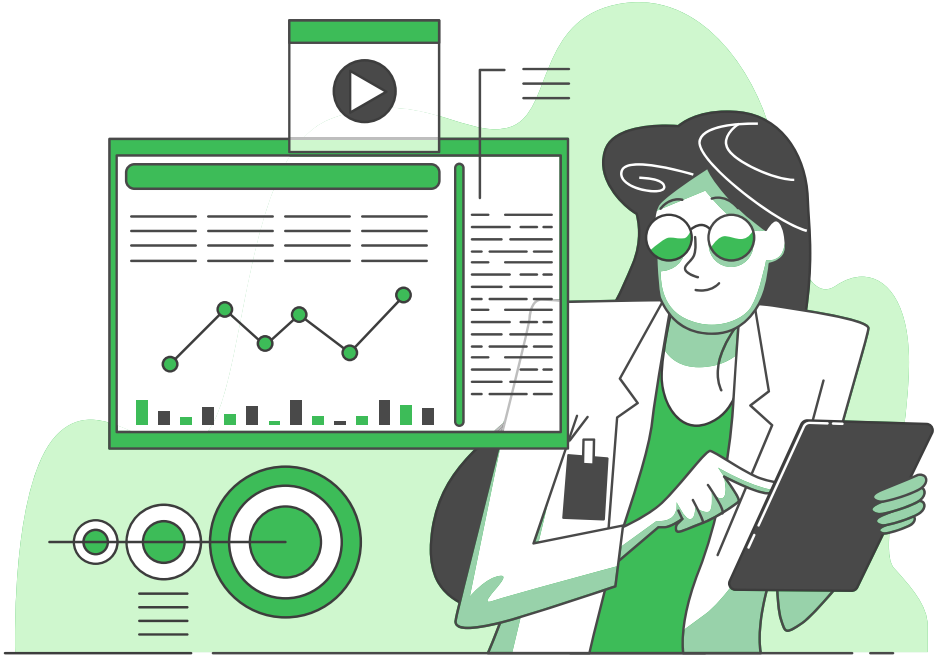


Kruskal's Algorithm

CS 251 - Data Structures and Algorithms

Note:
Slides complement the discussion in class

Table of Contents

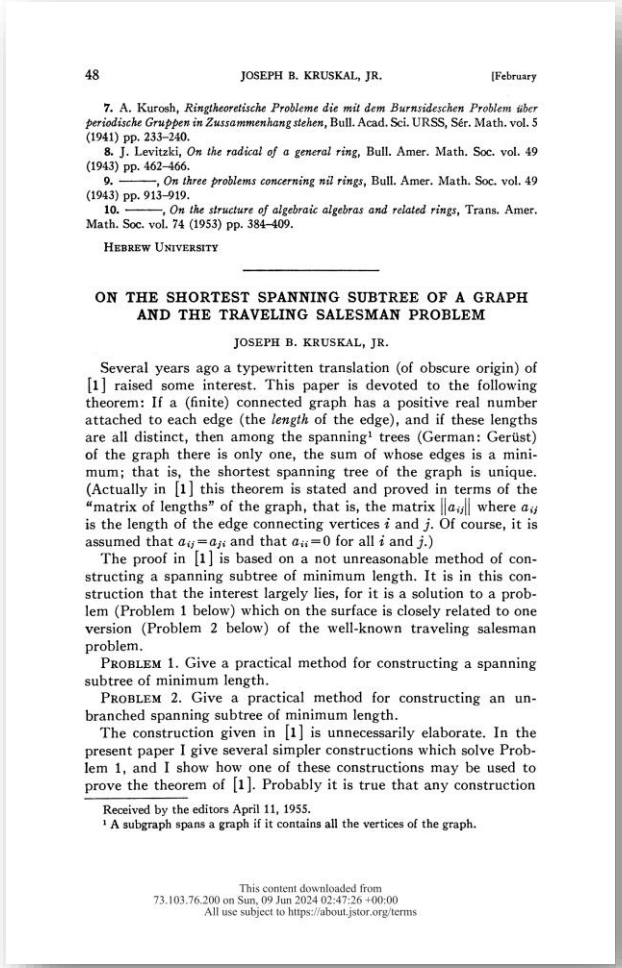


01

Kruskal's Algorithm

A way to find an MST

Joseph B. Kruskal. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem." Proceedings of the American Mathematical Society, vol. 7, no. 1, pp. 48–50, 1956.



7. A. Kurosh, *Ringtheoretische Probleme die mit dem Burnsideschen Problem über periodische Gruppen in Zussammenhang stehen*, Bull. Acad. Sci. URSS, Sér. Math. vol. 5 (1941) pp. 233–240.
8. J. Levitzki, *On the radical of a general ring*, Bull. Amer. Math. Soc. vol. 49 (1943) pp. 462–466.
9. ———, *On three problems concerning nil rings*, Bull. Amer. Math. Soc. vol. 49 (1943) pp. 913–919.
10. ———, *On the structure of algebraic algebras and related rings*, Trans. Amer. Math. Soc. vol. 74 (1953) pp. 384–409.

HEBREW UNIVERSITY

ON THE SHORTEST SPANNING SUBTREE OF A GRAPH AND THE TRAVELING SALESMAN PROBLEM

JOSEPH B. KRUSKAL, JR.

Several years ago a typewritten translation (of obscure origin) of [1] raised some interest. This paper is devoted to the following theorem: If a (finite) connected graph has a positive real number attached to each edge (the *length* of the edge), and if these lengths are all distinct, then among the spanning[1] trees (German: Gerüst) of the graph there is only one, the sum of whose edges is a minimum; that is, the shortest spanning tree of the graph is unique. (Actually in [1] this theorem is stated and proved in terms of the "matrix of lengths" of the graph, that is, the matrix $\|a_{ij}\|$ where a_{ij} is the length of the edge connecting vertices i and j. Of course, it is assumed that $a_{ij}=a_{ji}$ and that $a_{ii}=0$ for all i and j.)

The proof in [1] is based on a not unreasonable method of constructing a spanning subtree of minimum length. It is in this construction that the interest largely lies, for it is a solution to a problem (Problem 1 below) which on the surface is closely related to one version (Problem 2 below) of the well-known traveling salesman problem.

PROBLEM 1. Give a practical method for constructing a spanning subtree of minimum length.

PROBLEM 2. Give a practical method for constructing an unbranched spanning subtree of minimum length.

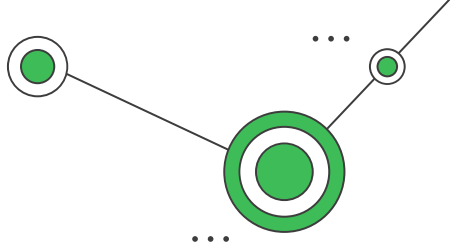
The construction given in [1] is unnecessarily elaborate. In the present paper I give several simpler constructions which solve Problem 1, and I show how one of these constructions may be used to prove the theorem of [1]. Probably it is true that any construction

Received by the editors April 11, 1955.

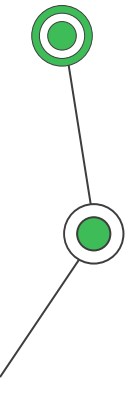
[1] A subgraph spans a graph if it contains all the vertices of the graph.

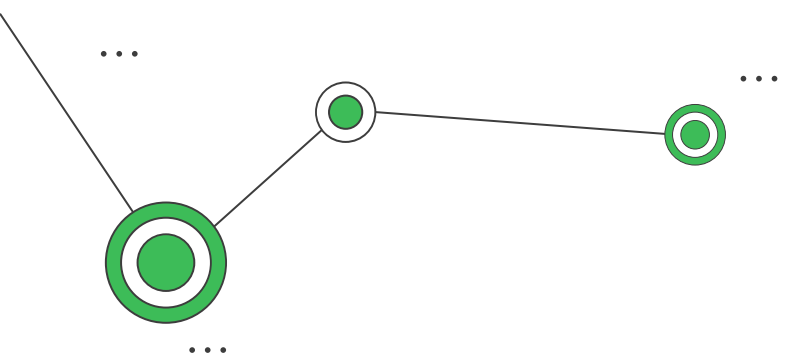
https://www.jstor.org/stable/2033241

Kruskal's Algorithm



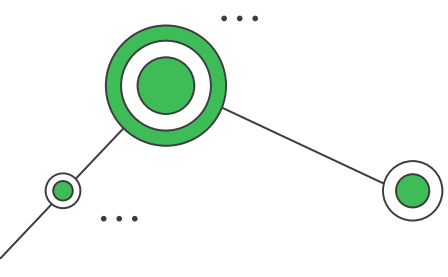
- Process edges, adding edges to the MST smallest-weight first (do not form a cycle on the process).
- The algorithm creates a forest that eventually merges into a single tree.
- Uses Union-Find for each separate component. Starts with $|V|$ independent components.
- Combine sets by adding edges, which reduces the number of components until there is only one.



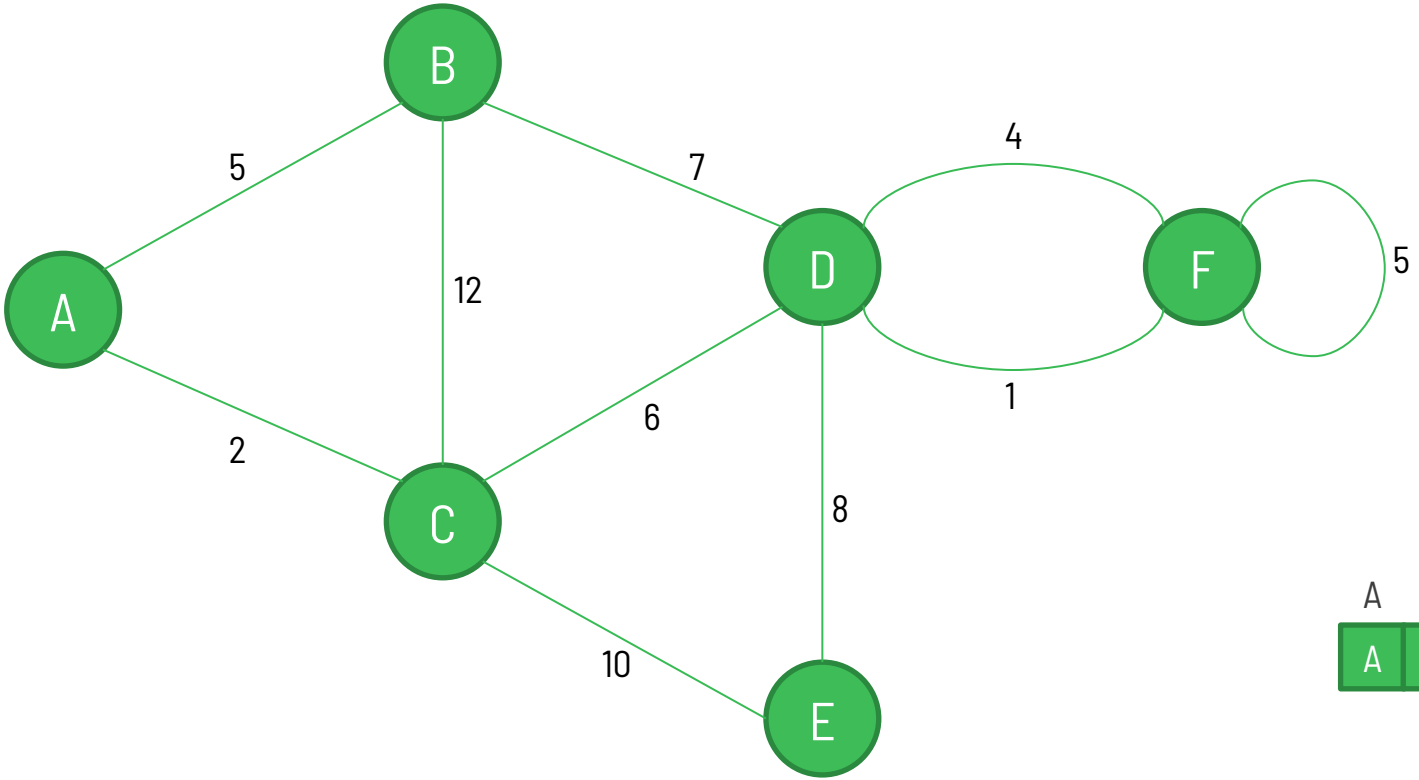


Kruskal's MST Algorithm

```
algorithm KruskalMST(G(V,E))

   let Q be an empty min-heap
   let UF be a Union-Find with |V| components

   for each e ∈ E do
      Q.insert(weight(e), e)
   end for

   T ← {}

   while |T| < |V| − 1 do
      (u,v) ← Q.getMin()

      if u and v are not connected in UF then
         T.insert((u,v))
         UF.union(u,v)
      end if
   end while

   return T
end algorithm
```



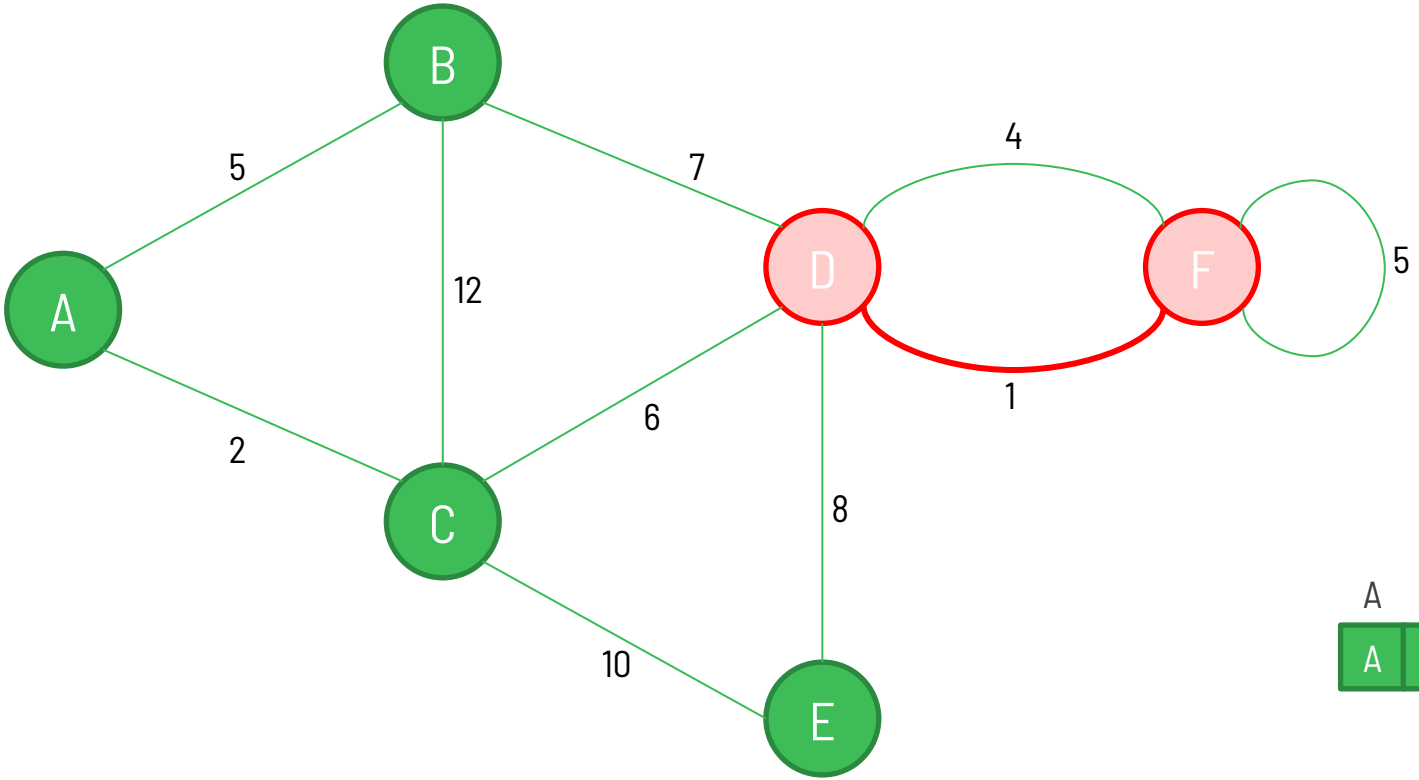
Kruskal's algorithm:



(D, F): 1
(A, C): 2
(D, F): 4
(F, F): 5
(A, B): 5
(C, D): 6
(B, D): 7
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
A	B	C	D	E	F

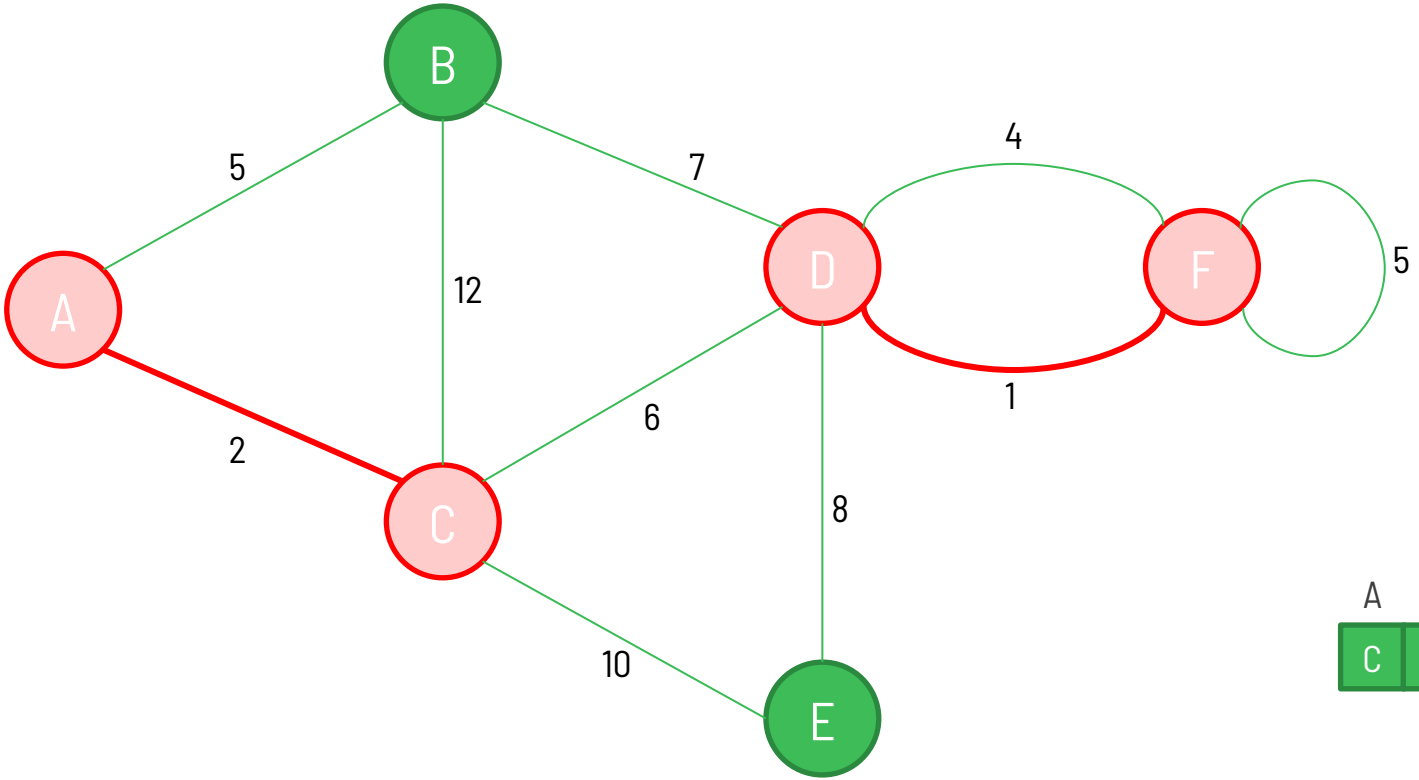
Kruskal's algorithm:



~~(D, F): 1~~
(A, C): 2
(D, F): 4
(F, F): 5
(A, B): 5
(C, D): 6
(B, D): 7
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
A	B	C	F	E	F

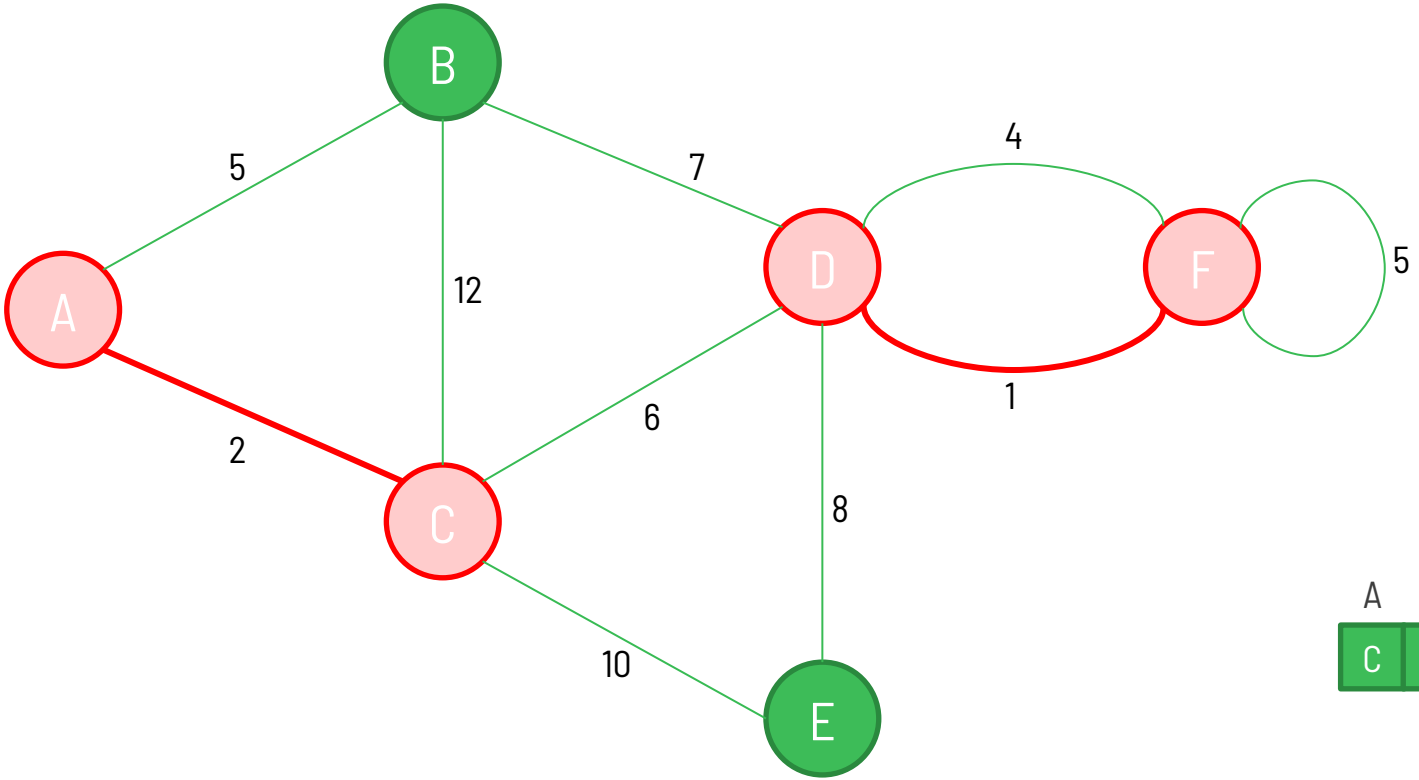
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
(D, F): 4
(F, F): 5
(A, B): 5
(C, D): 6
(B, D): 7
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
C	B	C	F	E	F

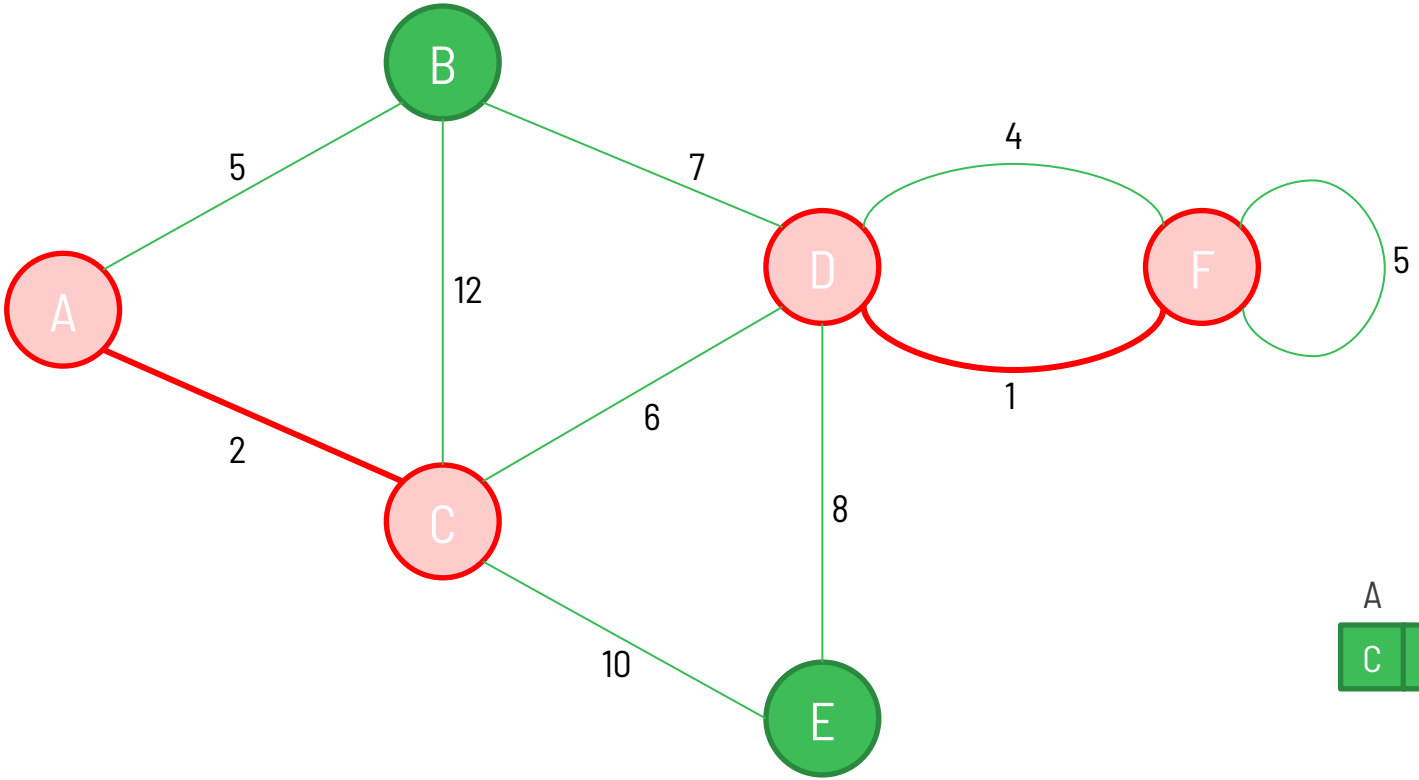
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
(F, F): 5
(A, B): 5
(C, D): 6
(B, D): 7
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
C	B	C	F	E	F

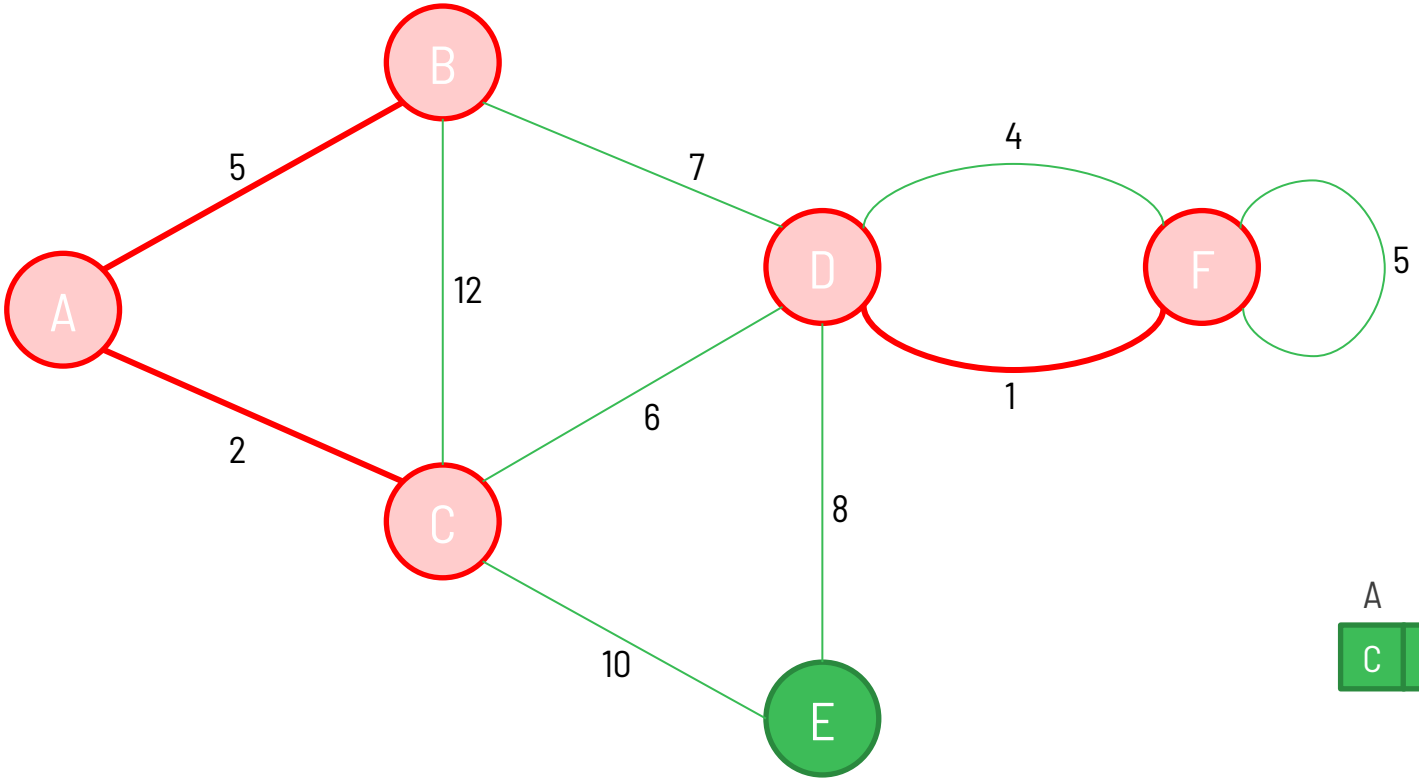
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
~~(F, F): 5~~
(A, B): 5
(C, D): 6
(B, D): 7
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
C	B	C	F	E	F

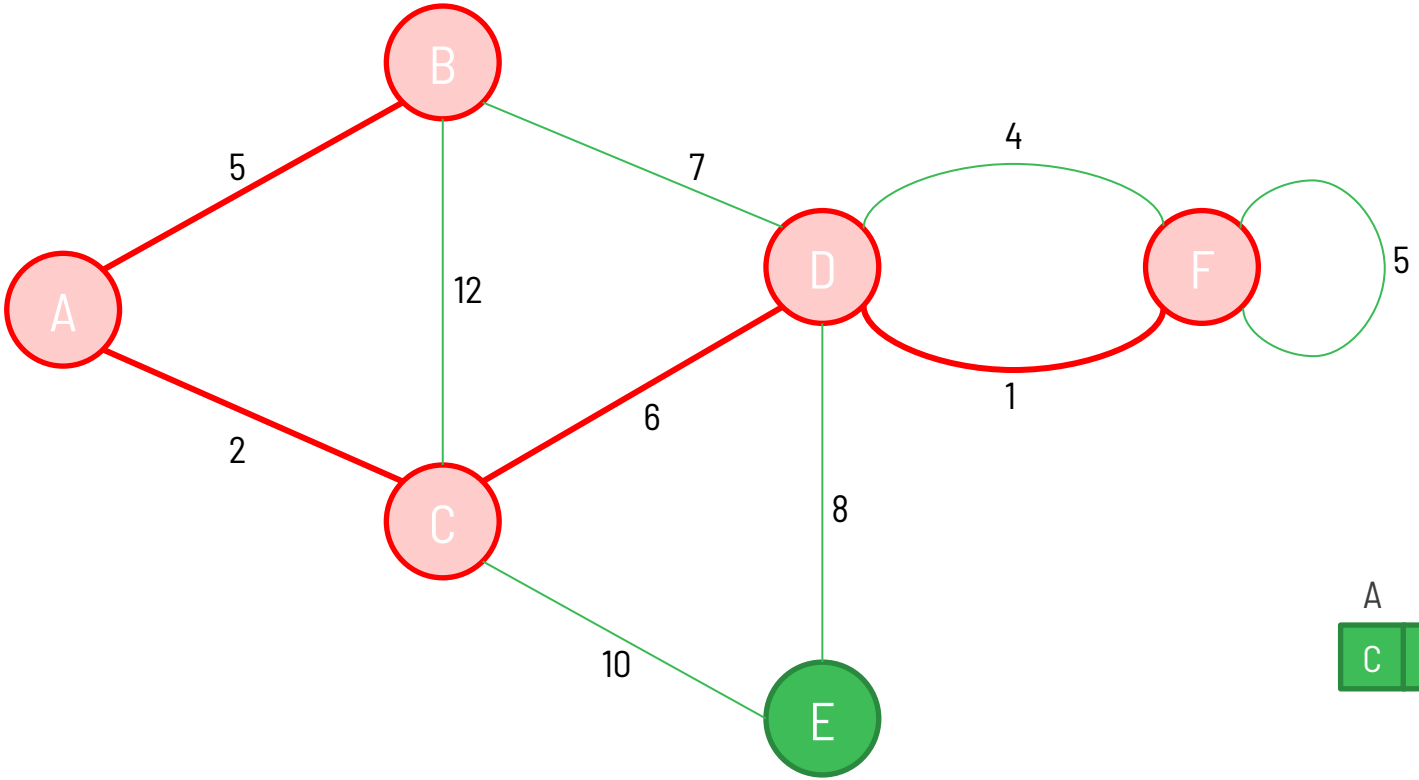
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
~~(F, F): 5~~
~~(A, B): 5~~
(C, D): 6
(B, D): 7
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
C	B	B	F	E	F

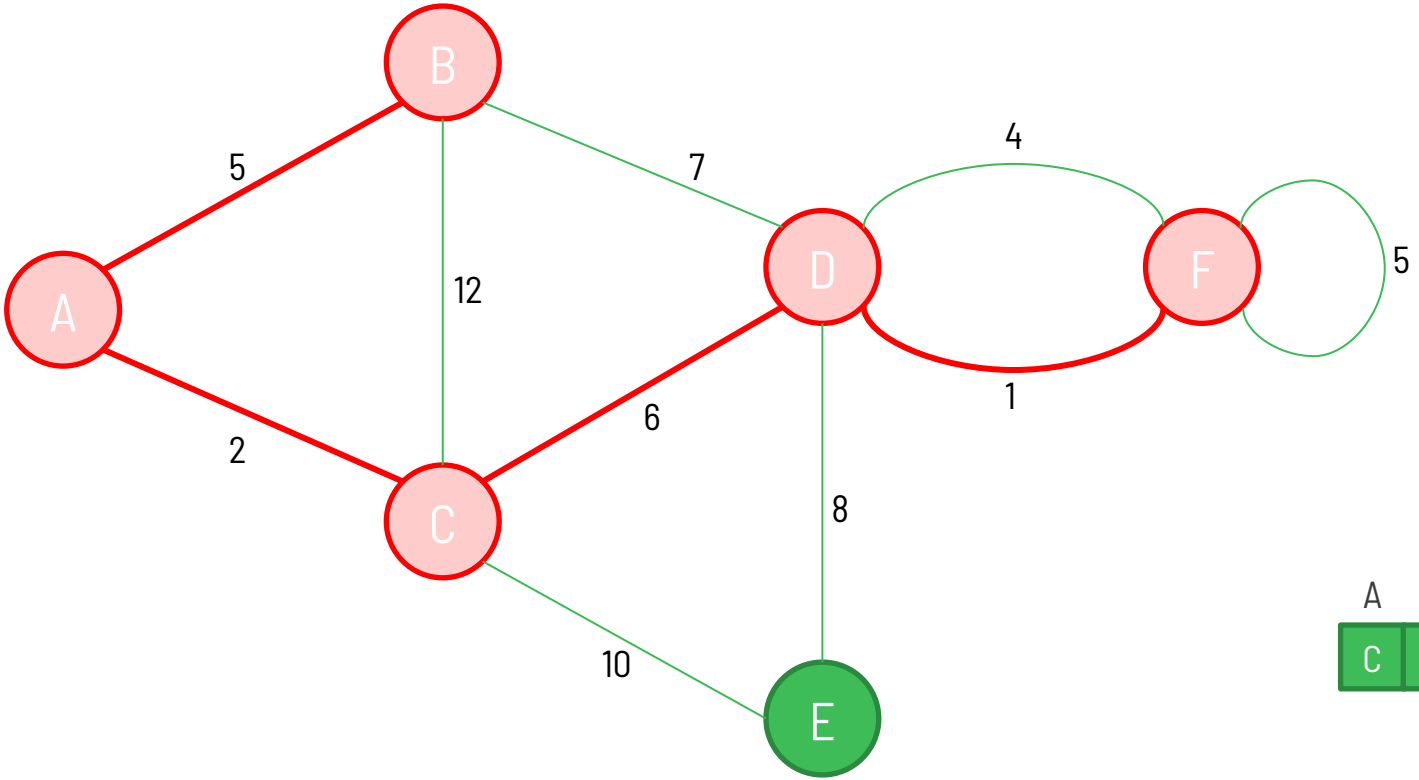
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
~~(F, F): 5~~
~~(A, B): 5~~
~~(C, D): 6~~
(B, D): 7
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
C	F	B	F	E	F

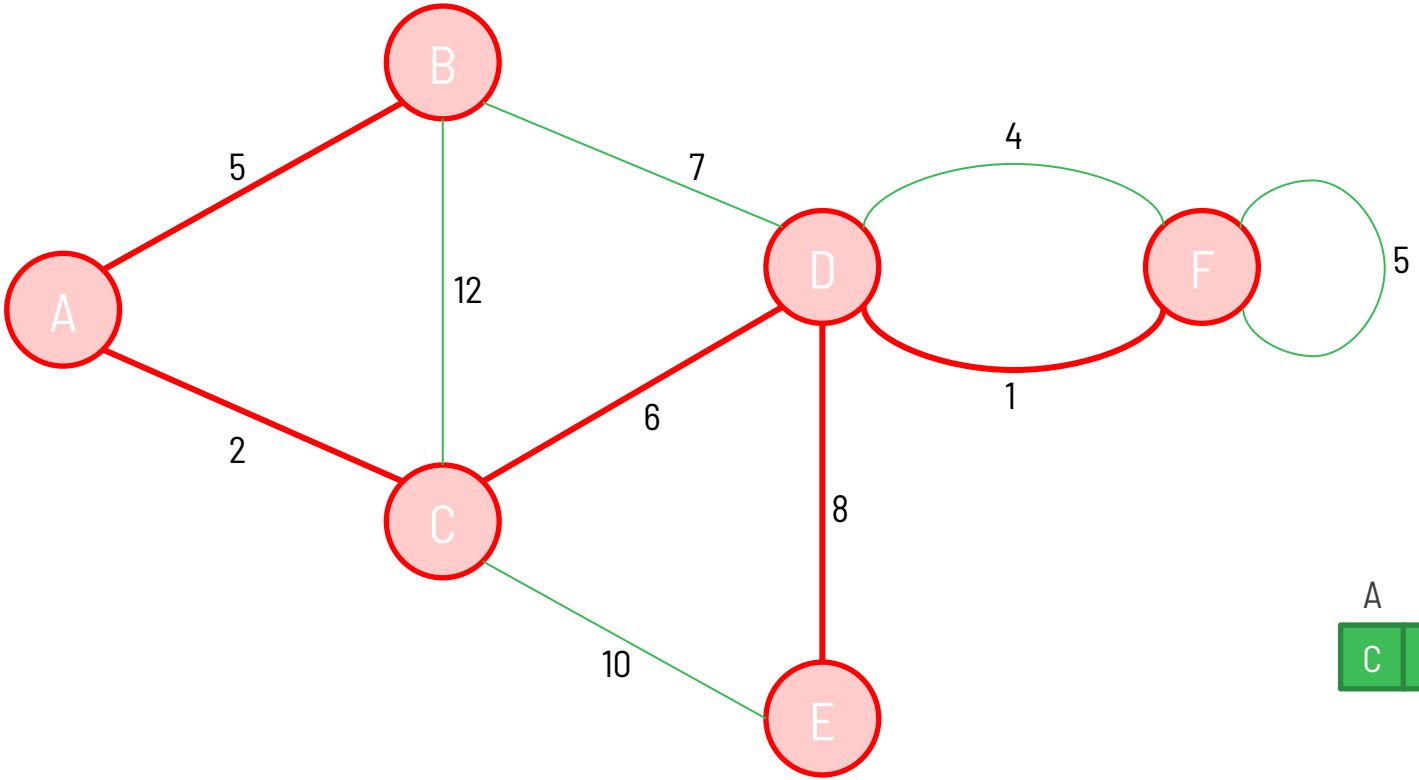
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
~~(F, F): 5~~
~~(A, B): 5~~
~~(C, D): 6~~
~~(B, D): 7~~
(D, E): 8
(C, E): 10
(B, C): 12

A	B	C	D	E	F
C	F	B	F	E	F

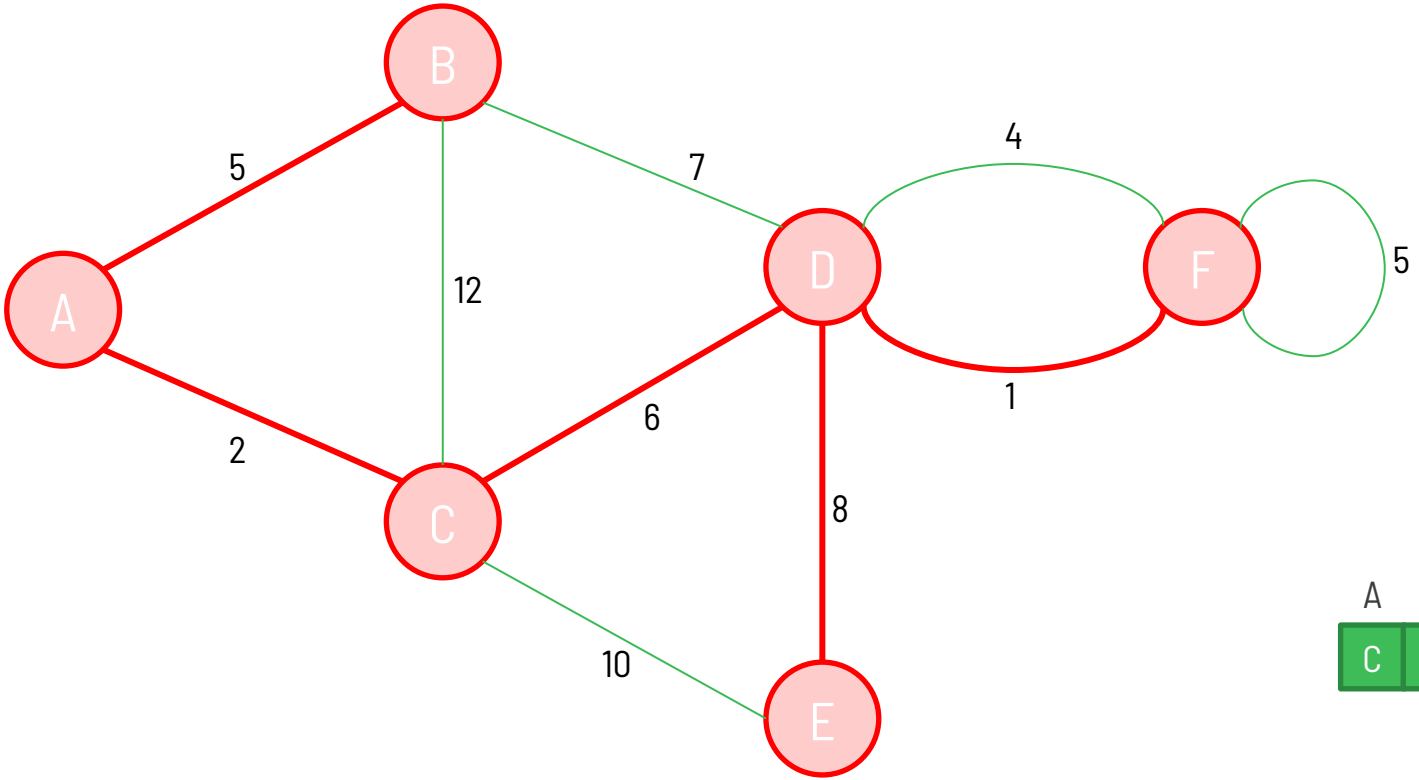
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
~~(F, F): 5~~
~~(A, B): 5~~
~~(C, D): 6~~
~~(B, D): 7~~
~~(D, E): 8~~
(C, E): 10
(B, C): 12

A	B	C	D	E	F
C	F	B	F	E	E

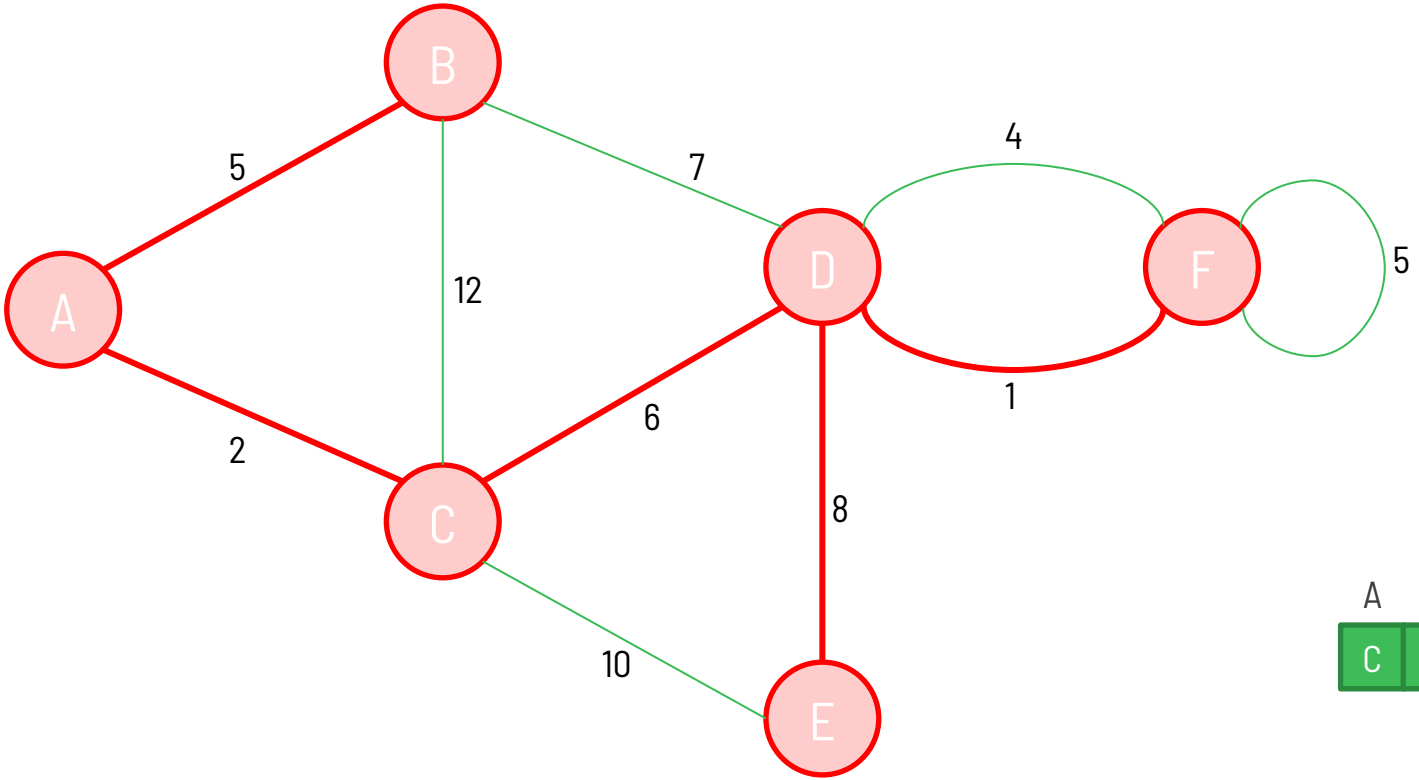
Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
~~(F, F): 5~~
~~(A, B): 5~~
~~(C, D): 6~~
~~(B, D): 7~~
~~(D, E): 8~~
~~(C, E): 10~~
(B, C): 12

A	B	C	D	E	F
C	F	B	F	E	E

Kruskal's algorithm:



~~(D, F): 1~~
~~(A, C): 2~~
~~(D, F): 4~~
~~(F, F): 5~~
~~(A, B): 5~~
~~(C, D): 6~~
~~(B, D): 7~~
~~(D, E): 8~~
~~(C, E): 10~~
~~(B, C): 12~~

A	B	C	D	E	F
C	F	B	F	E	E

```
algorithm KruskalMST(G(V,E))

   let Q be an empty min-heap
   let UF be a Union-Find with |V| components

   for each e ∈ E do
      Q.insert(weight(e), e)
   end for

   T ← {}

   while |T| < |V| − 1 do
      (u,v) ← Q.getMin()

      if u and v are not connected in UF then
         T.insert((u,v))
         UF.union(u,v)
      end if
   end while

   return T
end algorithm
```

$O(|V|)$

$O(|E|\log(|E|))$

$O(|E|\log(|V|))$

One More Left

Do you have any questions?