# GRAPH THEORY

## Problems and their Applications

MINIMUM SPANNING TREE PROBLEM

TRAVELLING SALESPERSON PROBLEM

GRAPH COLORING

SHORTEST PATH PROBLEM

GRAPH MATCHING

# MINIMUM SPANNING TREE PROBLEM

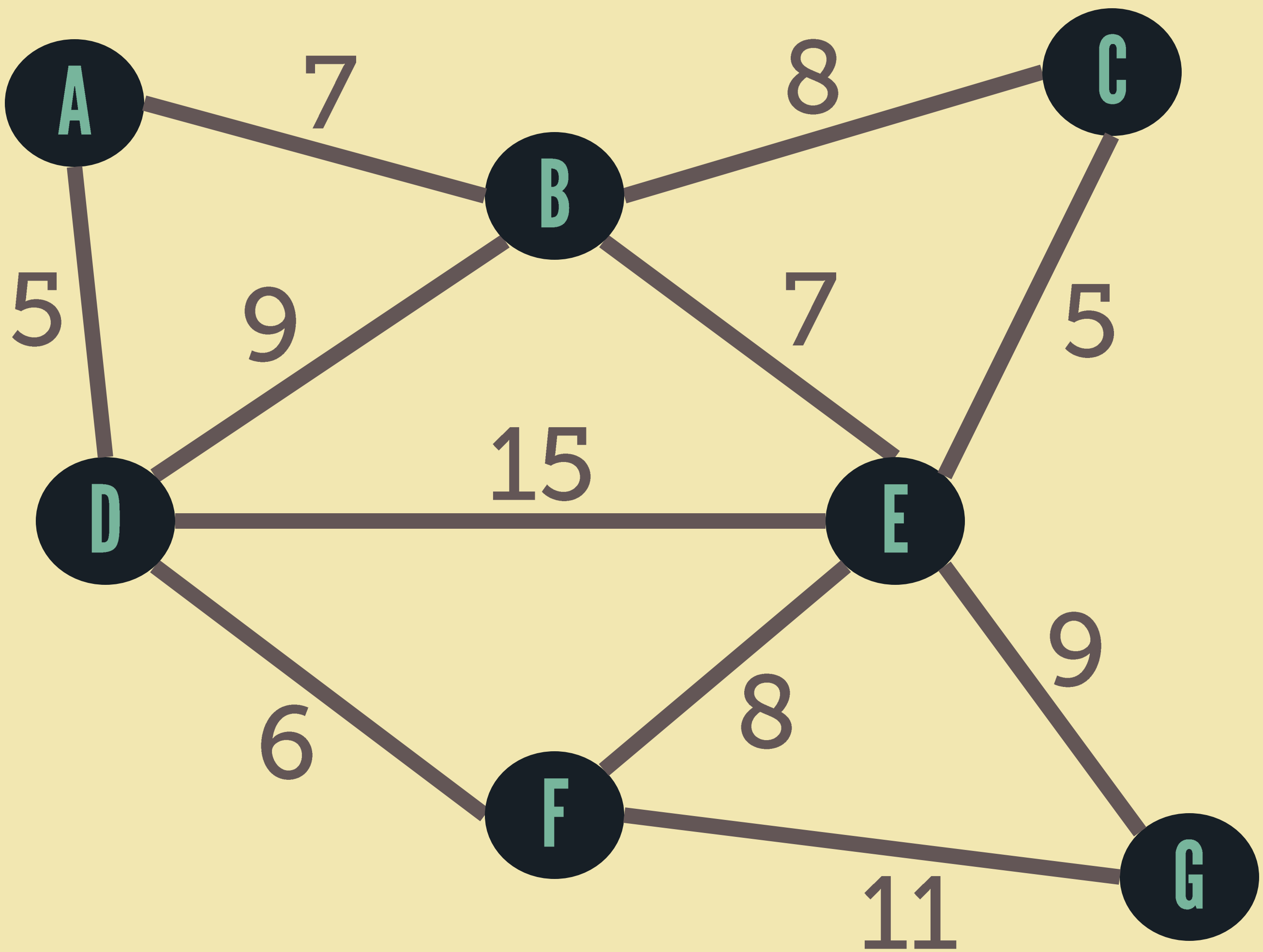
Given a weighted graph, find a **SPANNING TREE** such that the sum of the weights of the edges is the **SMALLEST** possible.

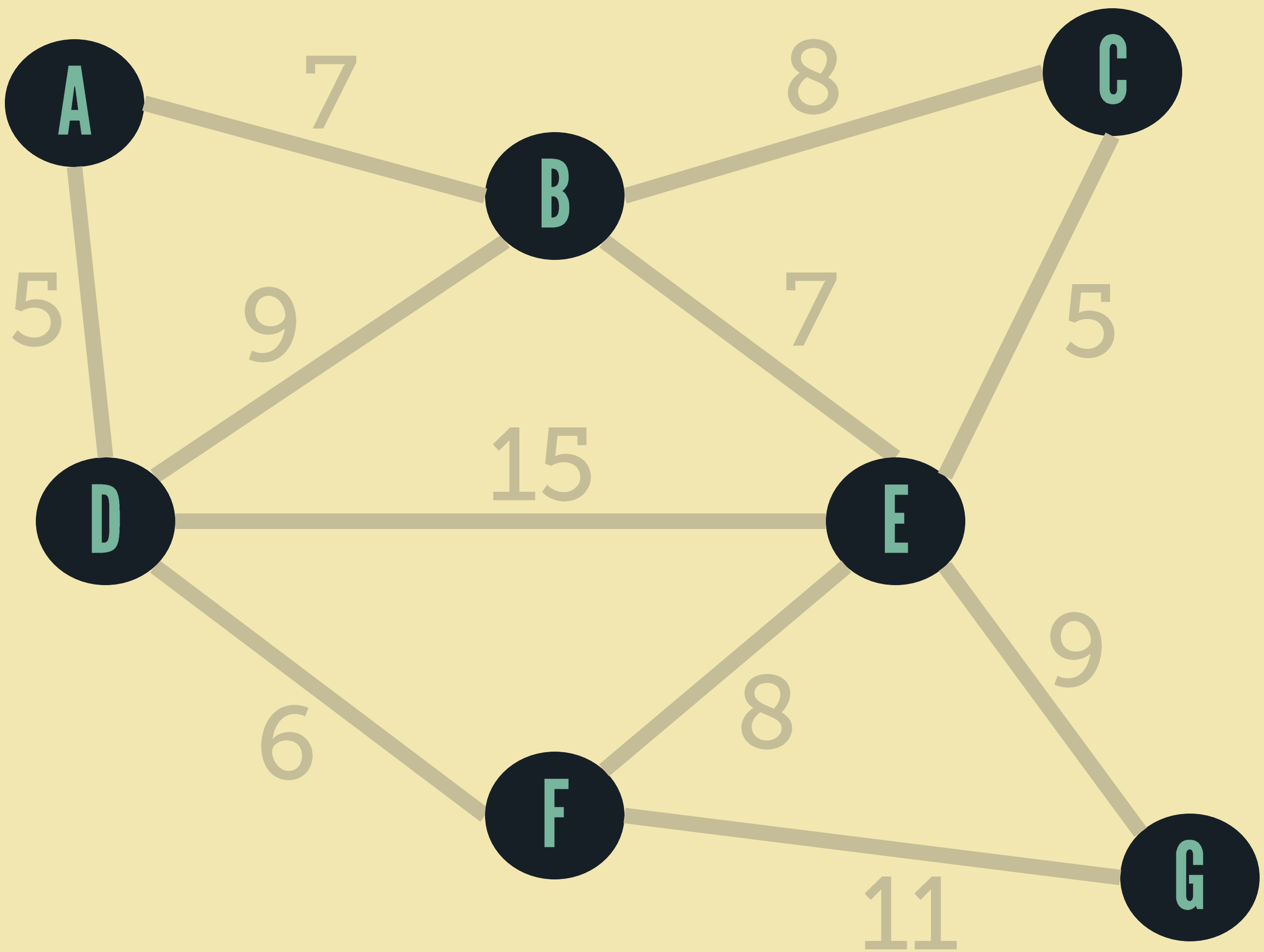
# KRUSKAL'S algorithm

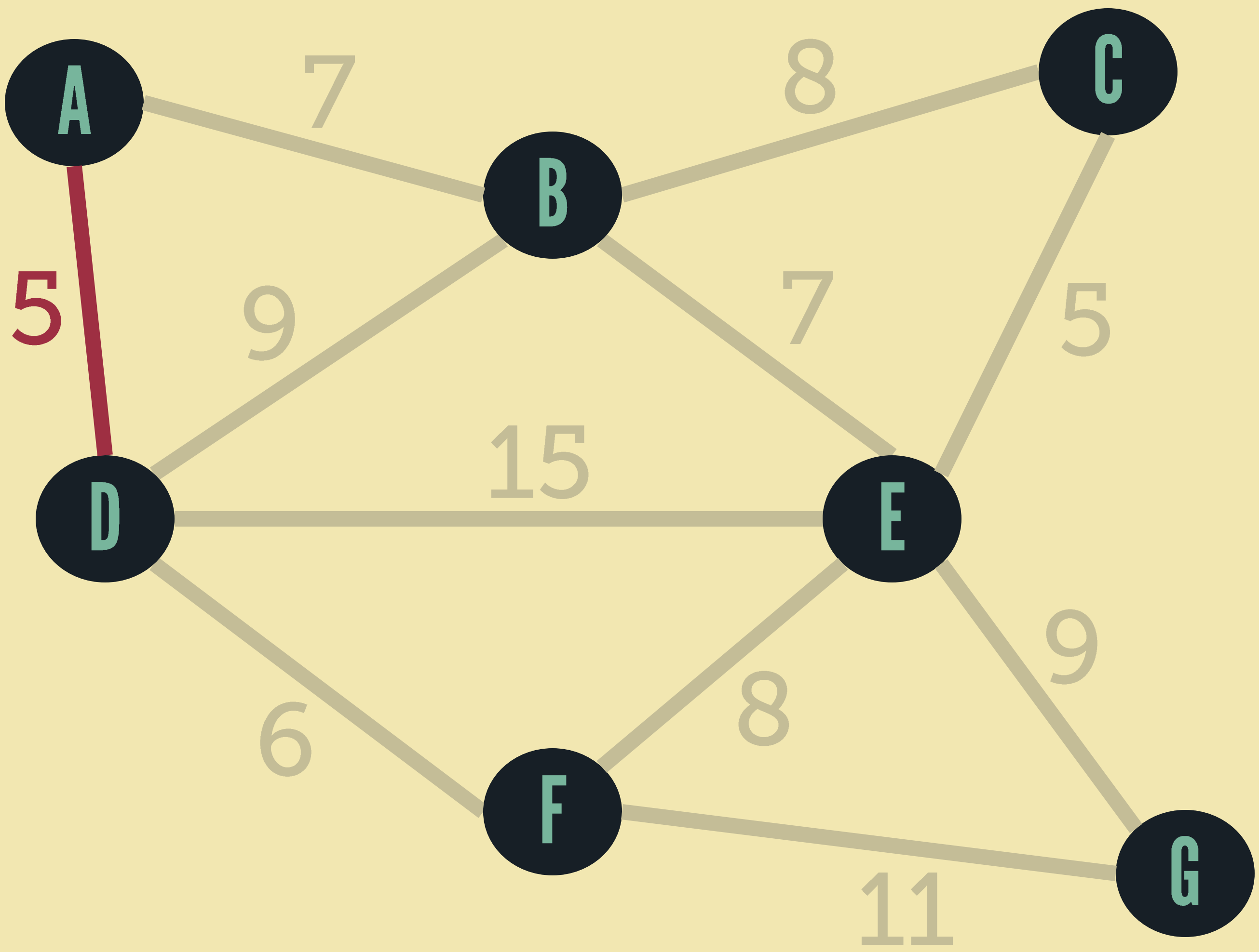
- Start with a null graph T with vertices of G.

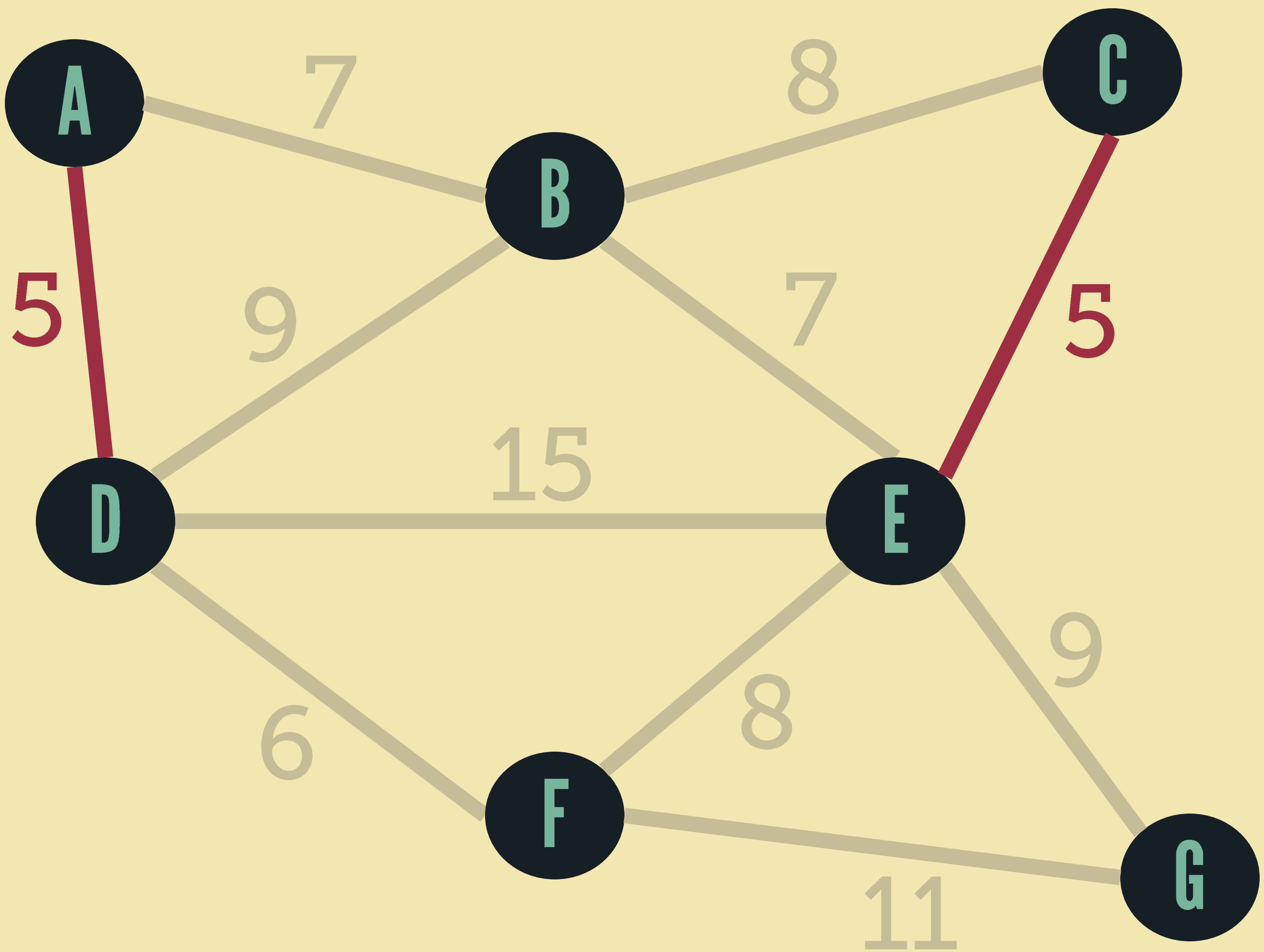
- Add currently cheapest edge from G to T that will not form a cycle in T.

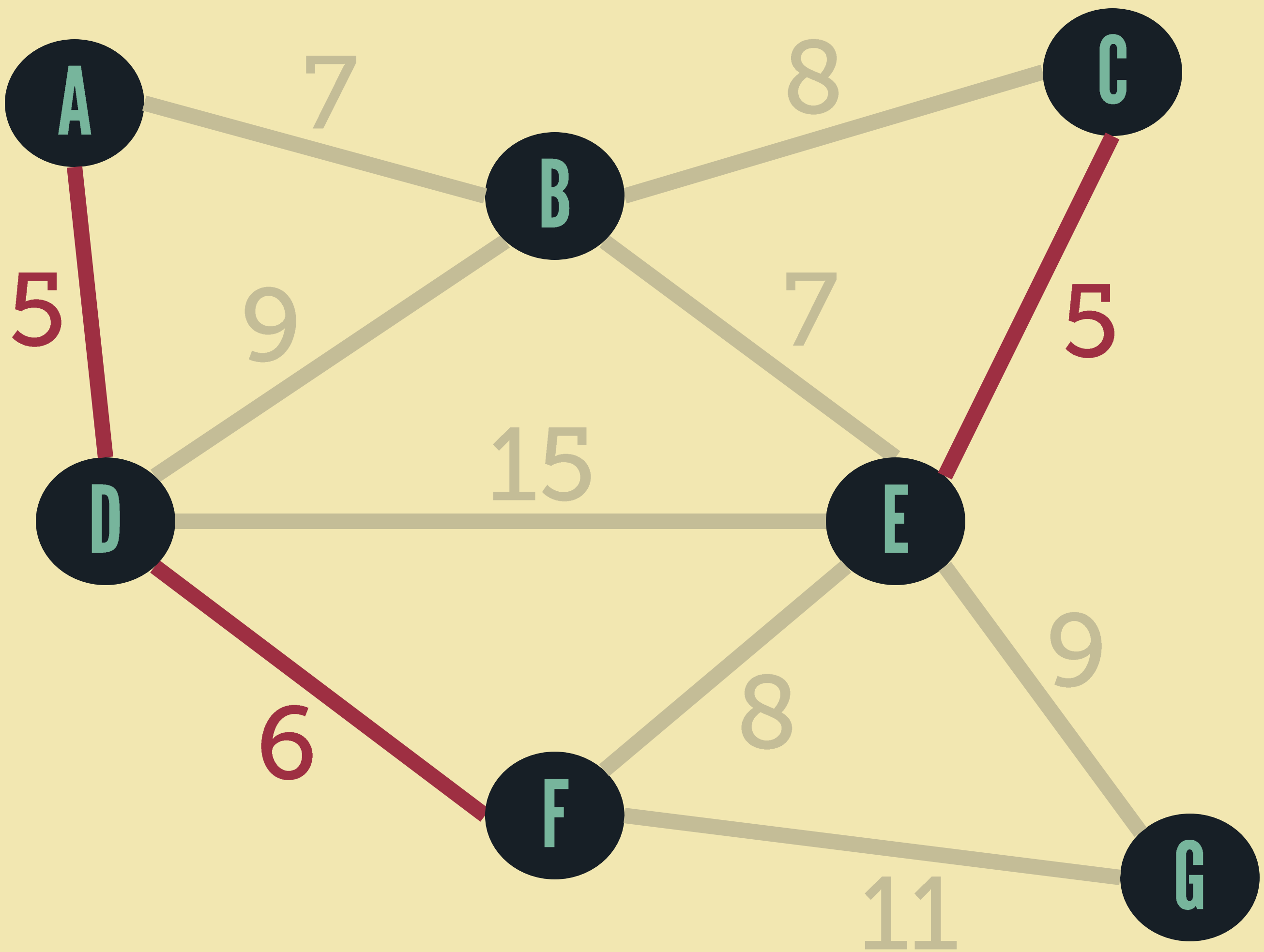
- Repeat previous step until a spanning tree is obtained.


A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11

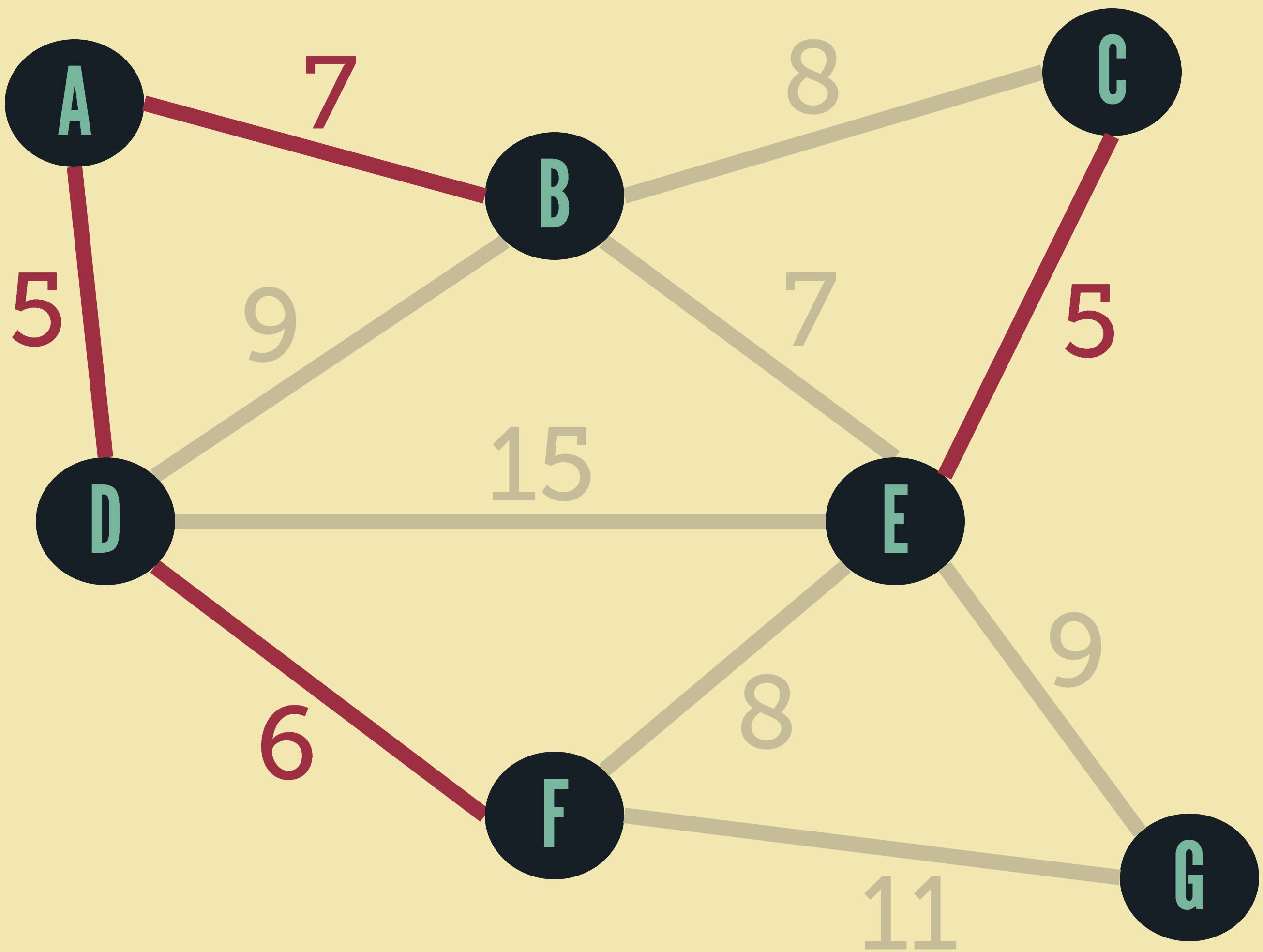

A
B
C
D
E
F
G
7
8
5
9
7
5
15
9
6
8
11

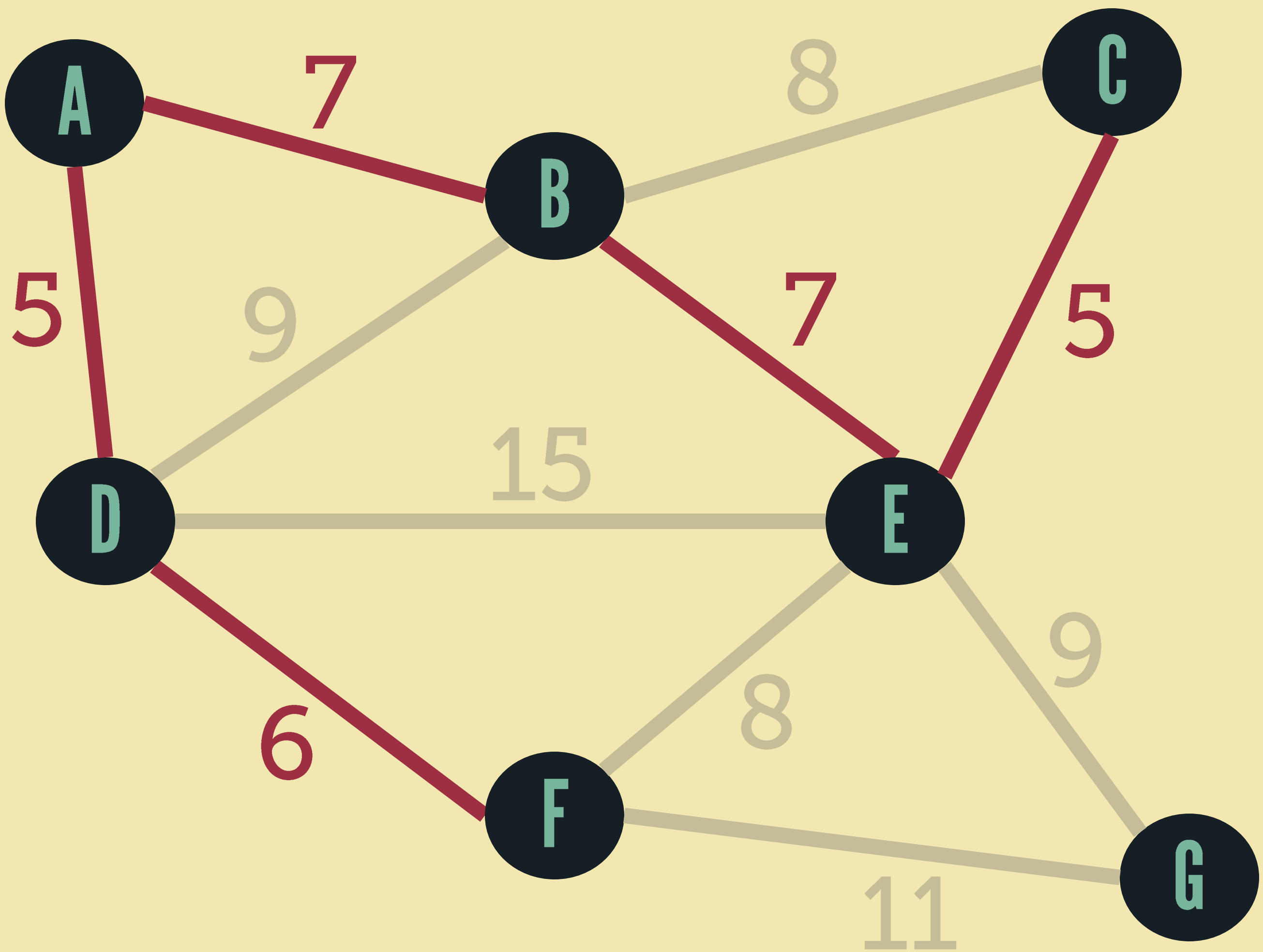

A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11

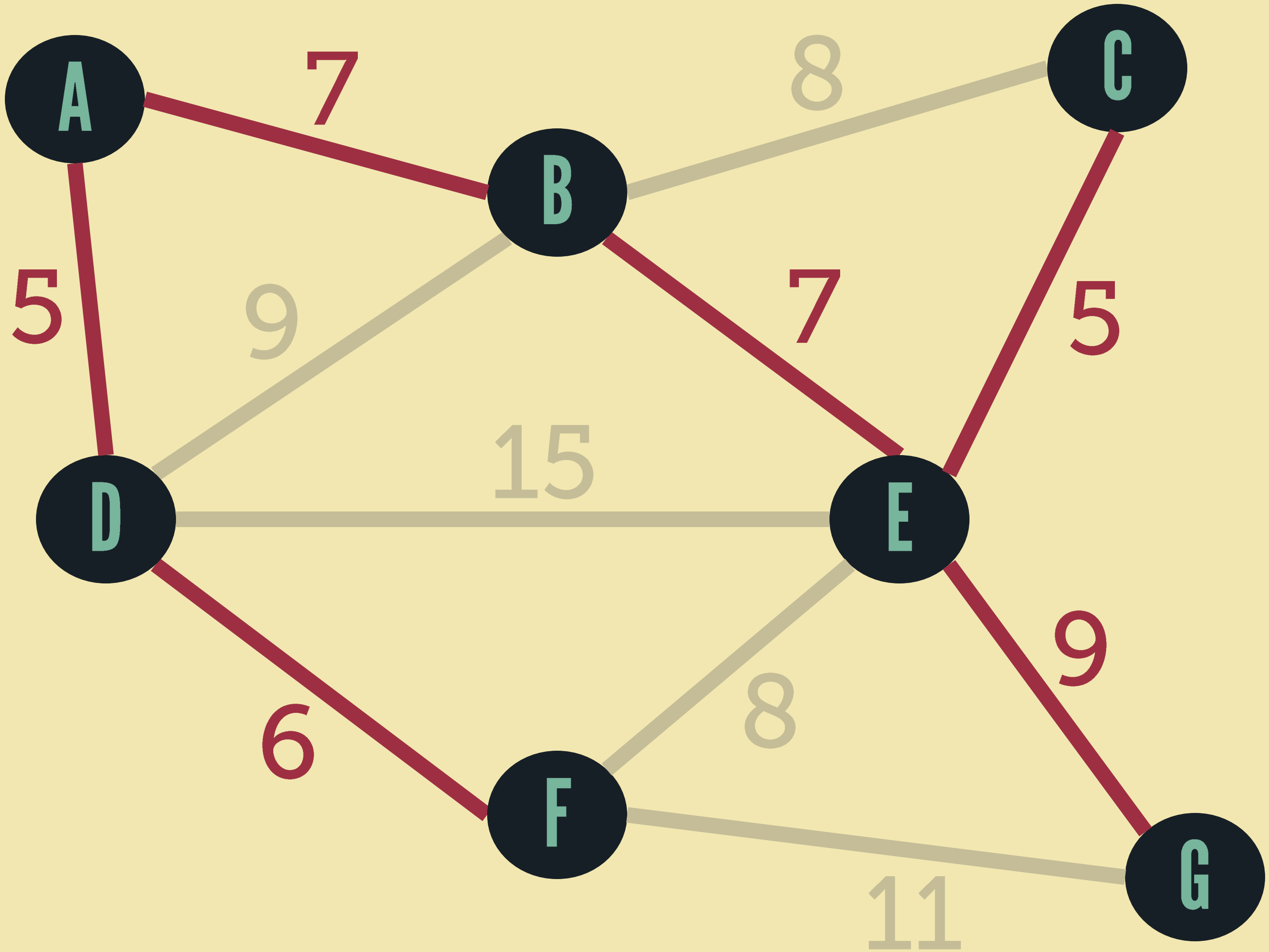

A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11



A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11


A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11


A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11

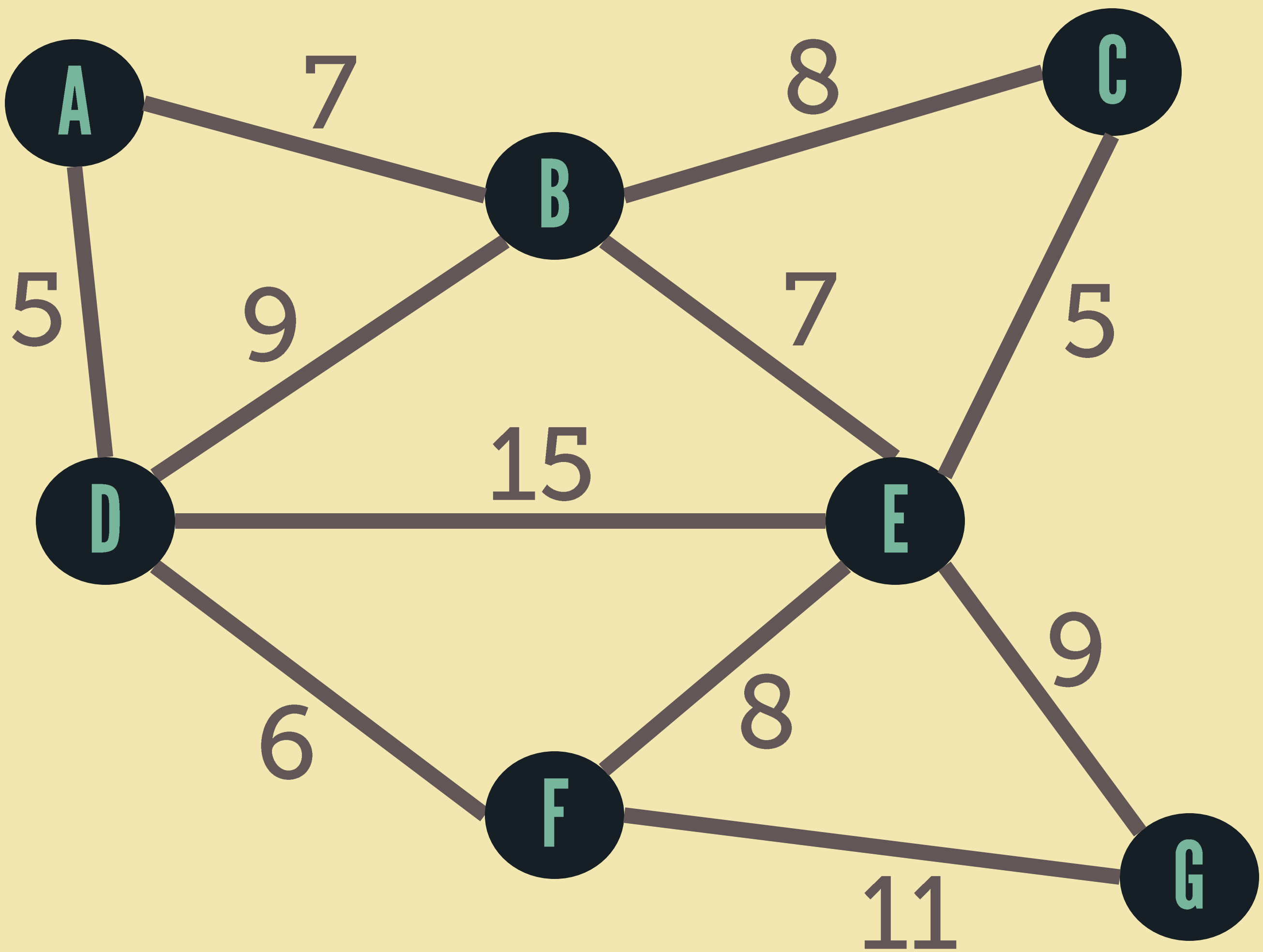

A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11

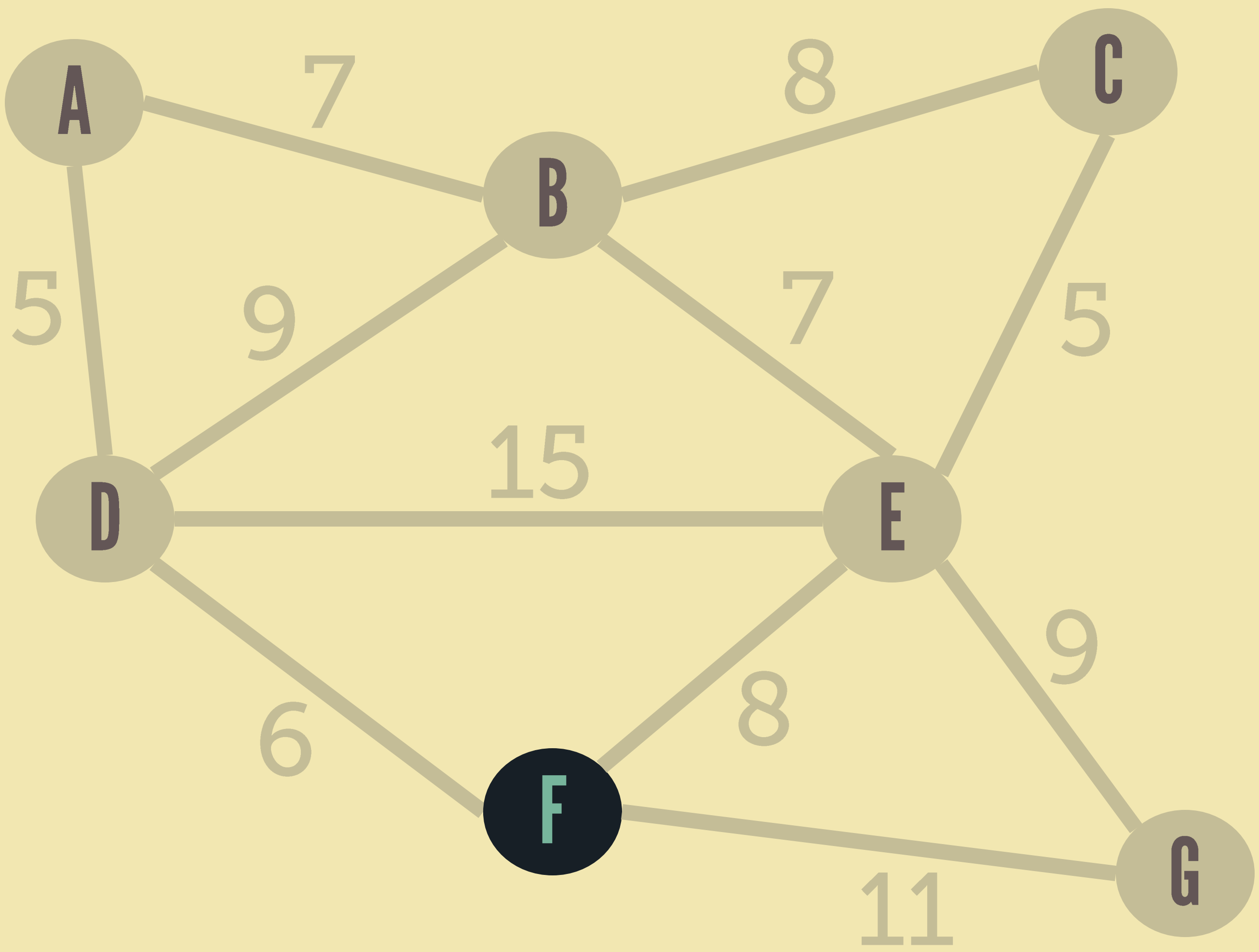
# PRIM'S algorithm

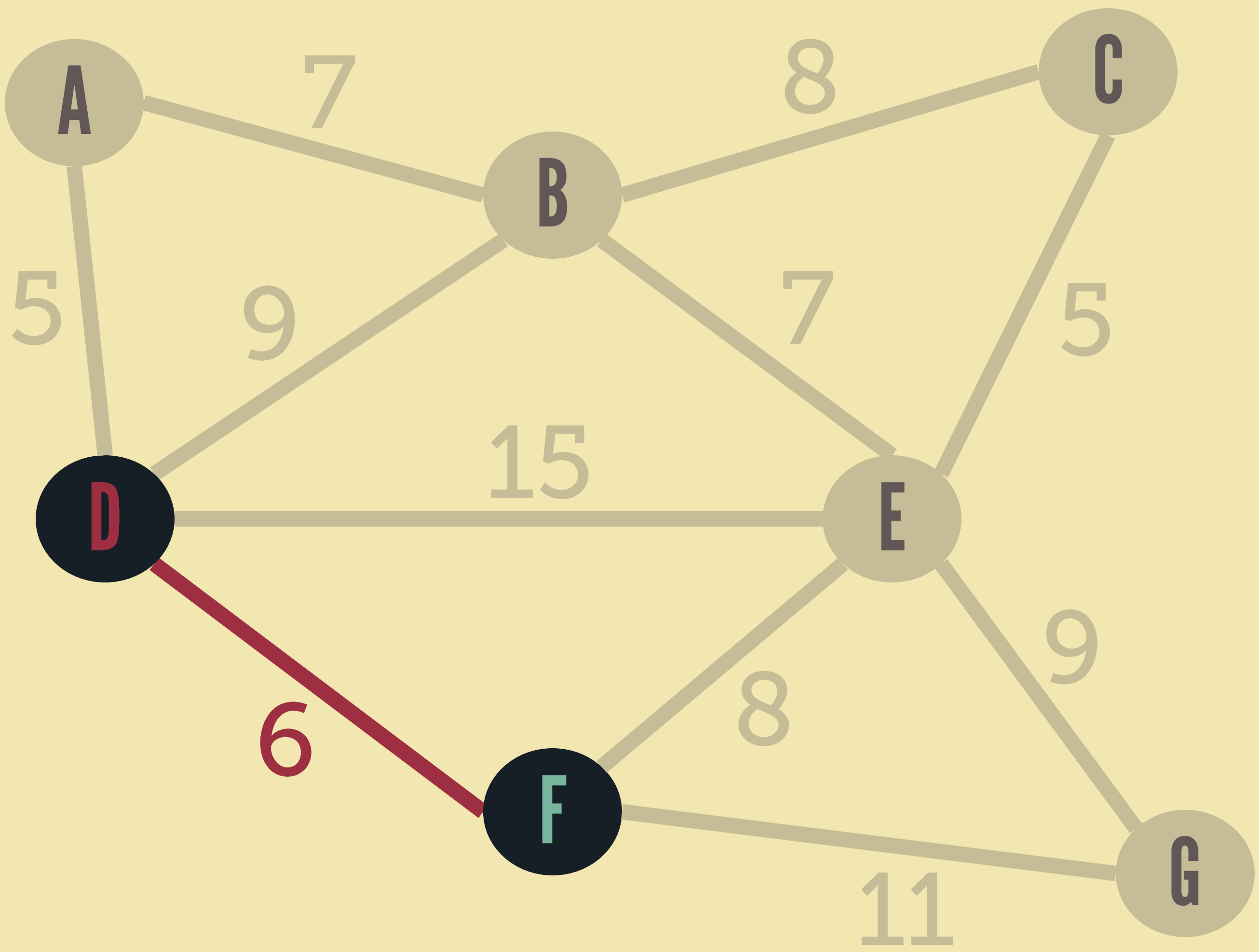
- Start with a trivial graph T with a single arbitrary vertex from G.

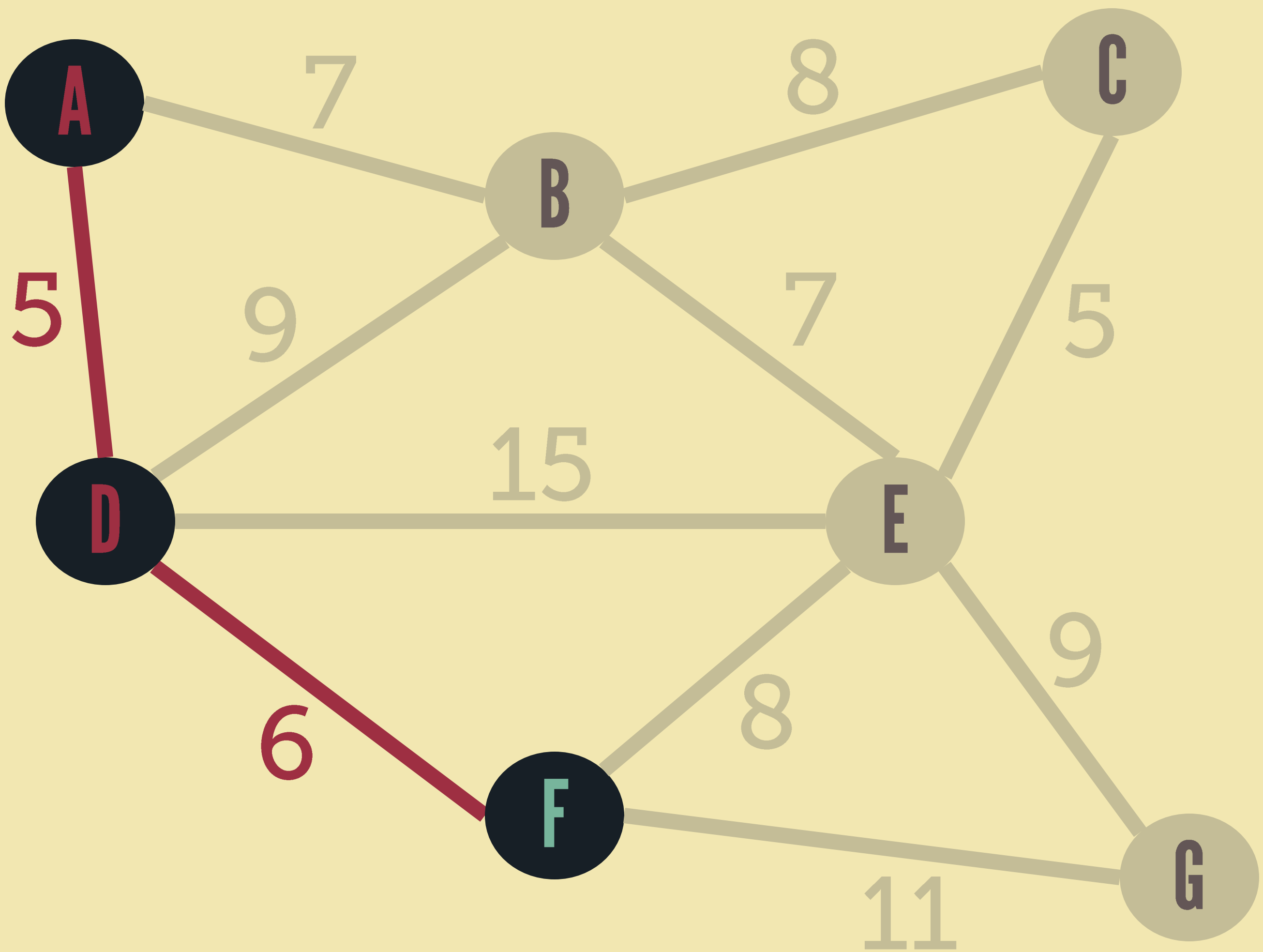
- Add the vertex y and edge e = (x,y) such that e is the cheapest edge connecting y to some vertex x already in T (and will not form a cycle)

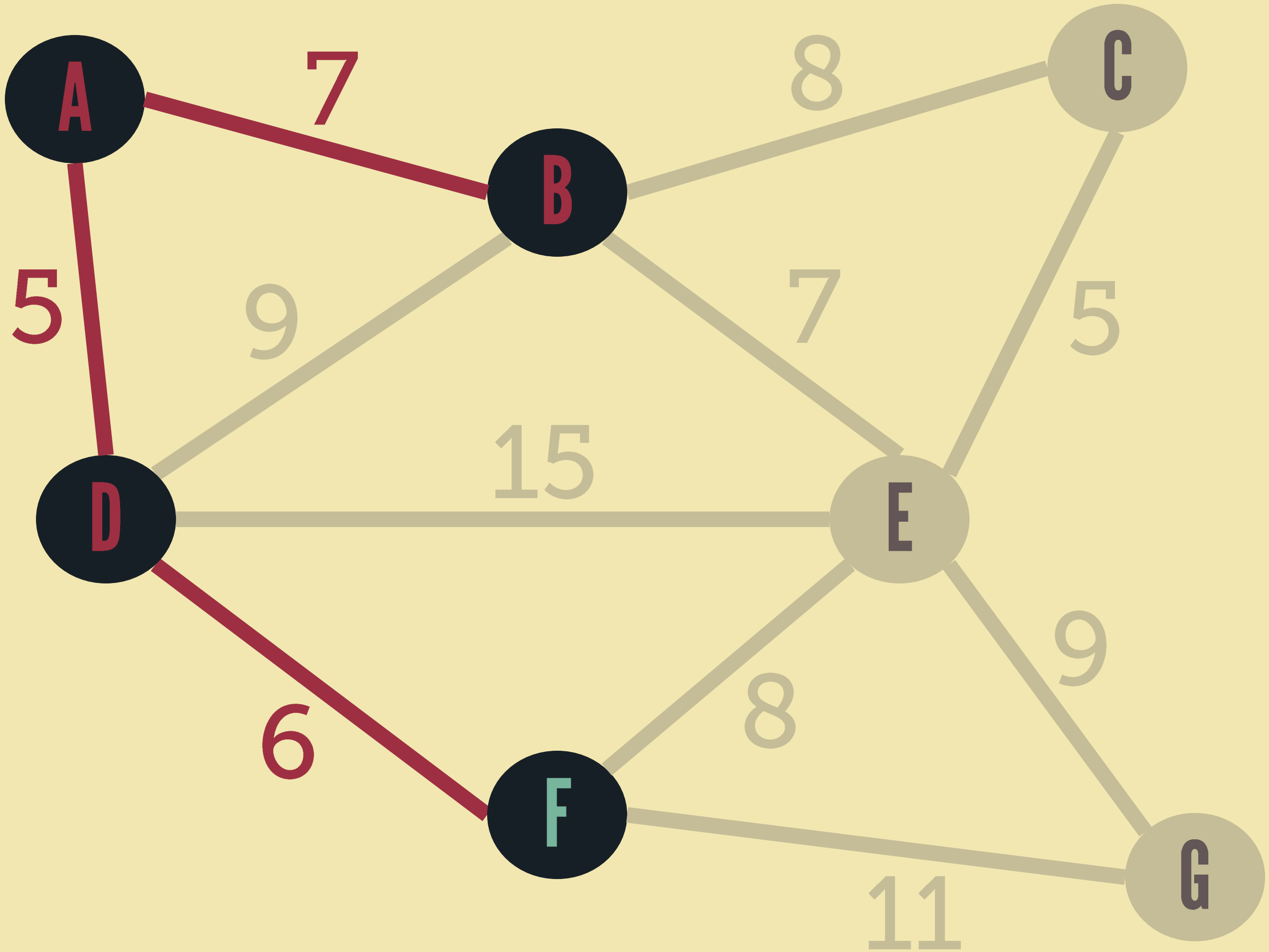
- Repeat previous step until a spanning tree is obtained.

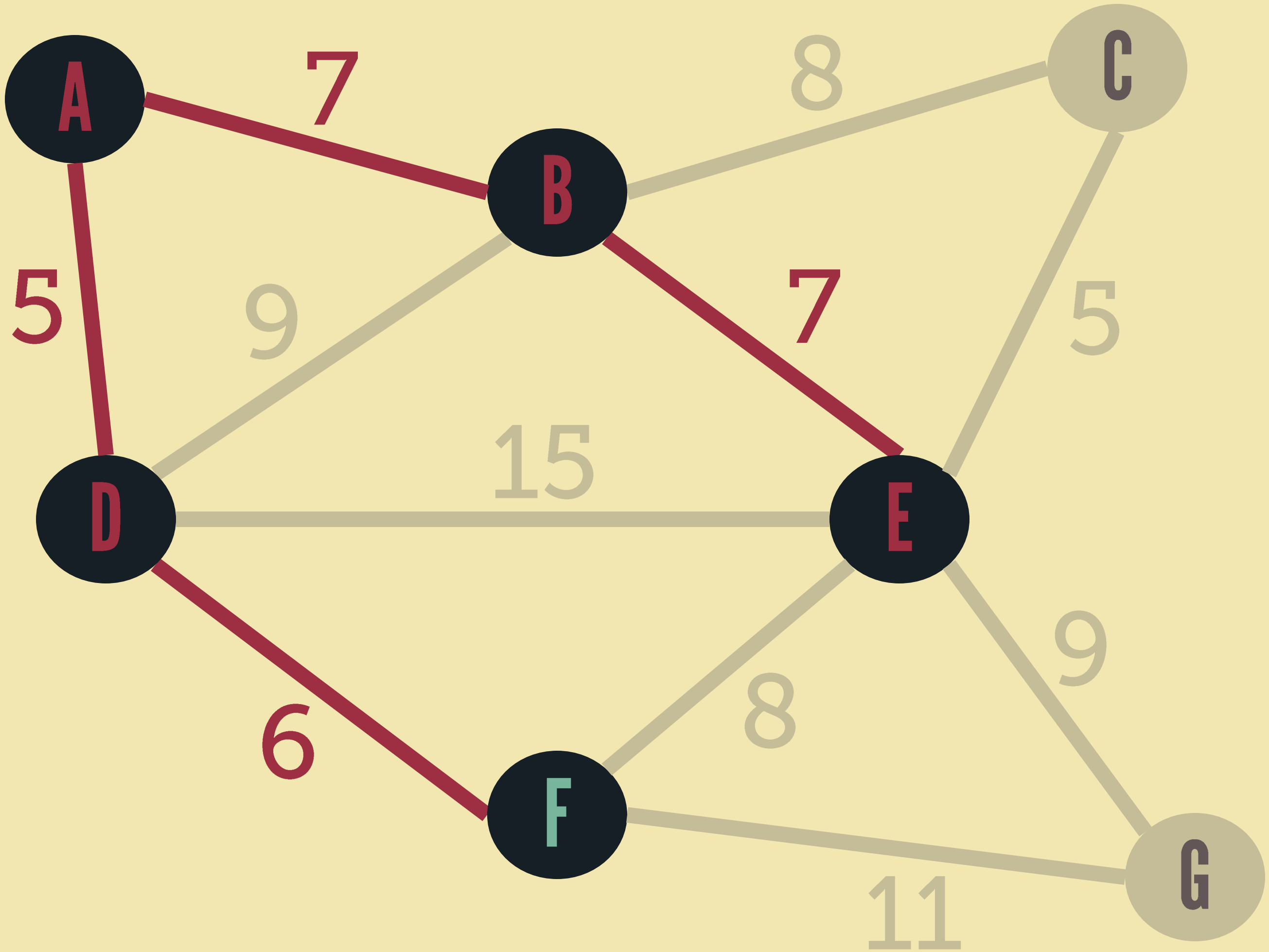

A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11

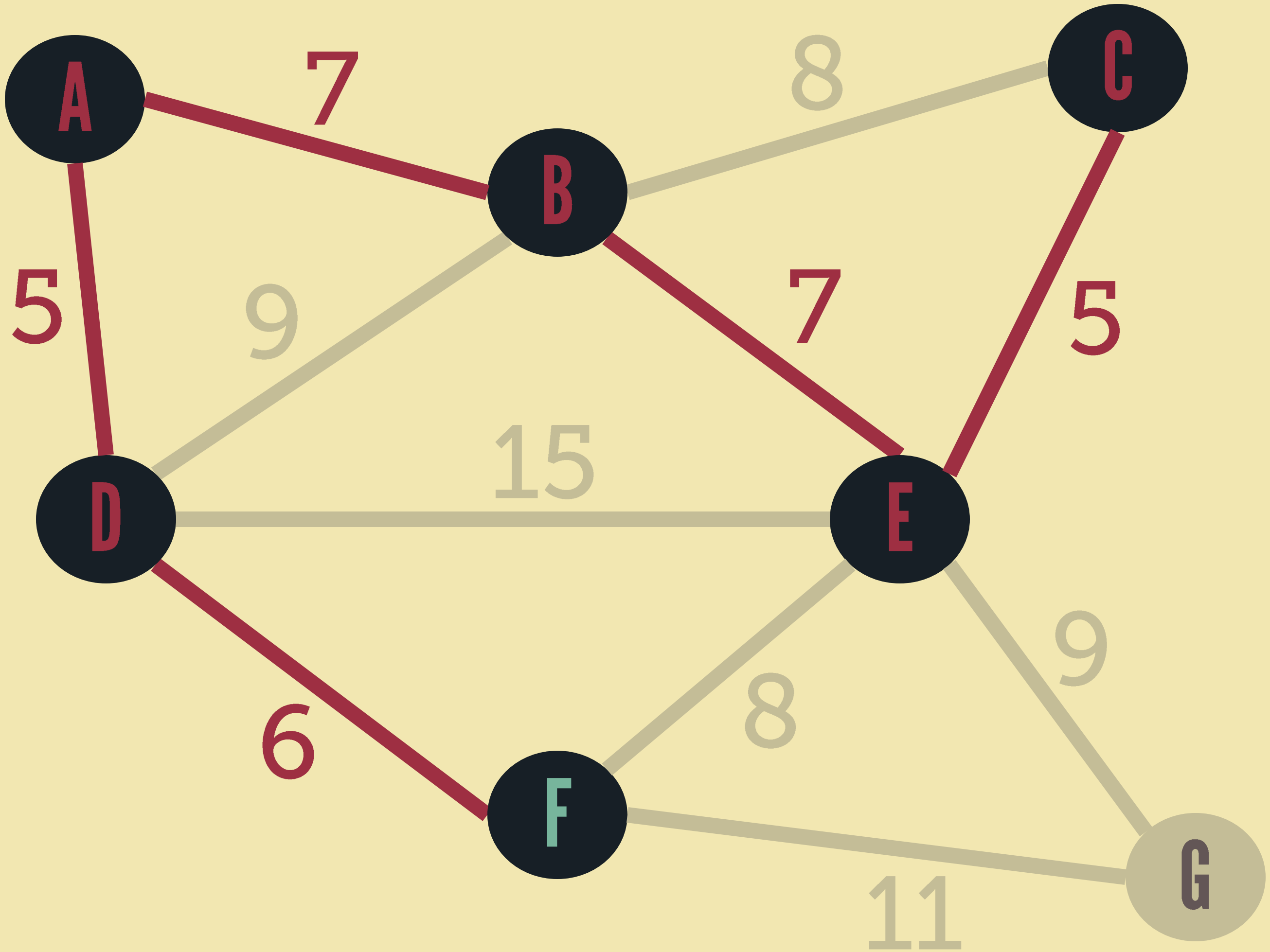

A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11

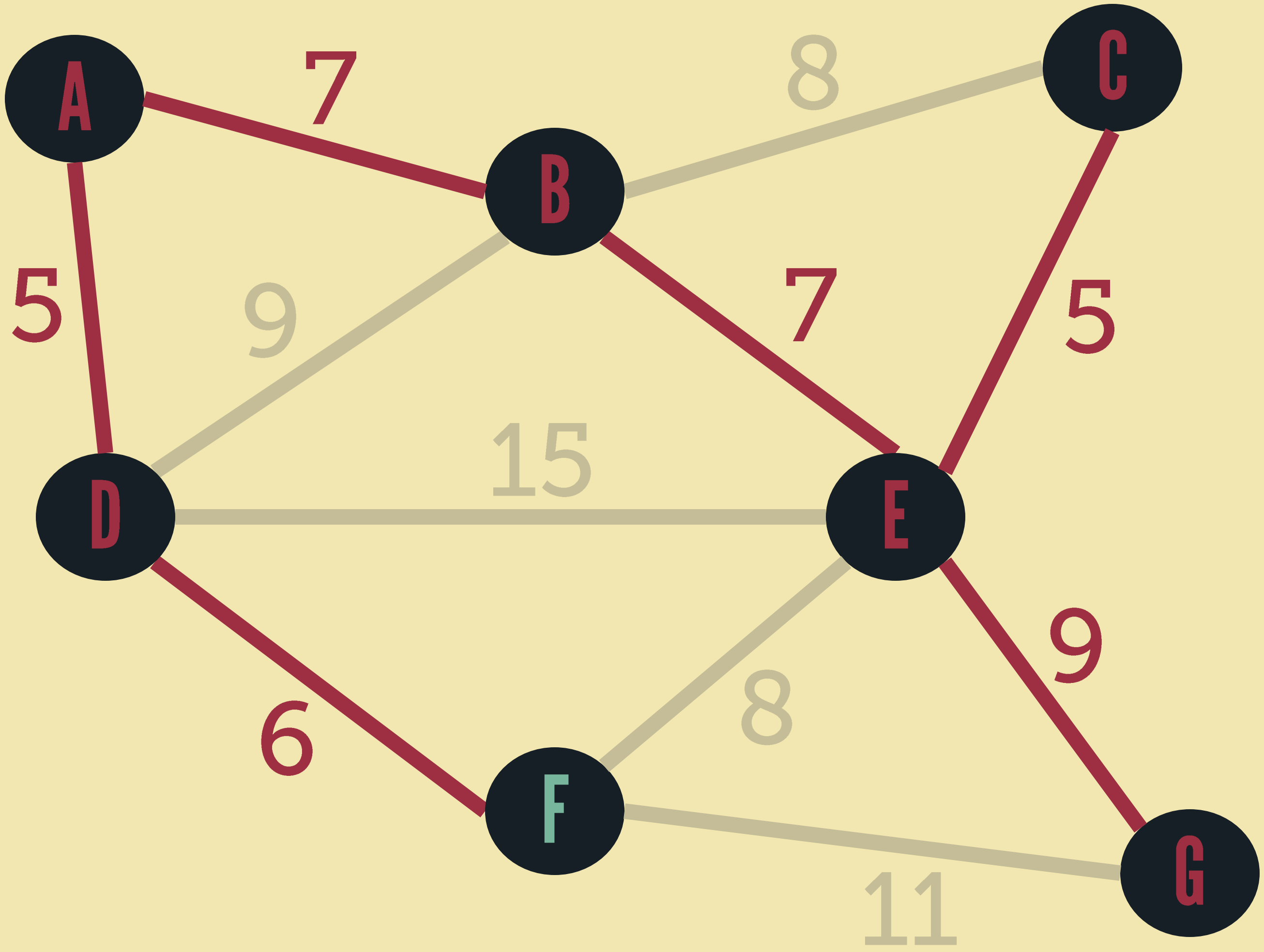

A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11


A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11


A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11


A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11


A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11



A
B
C
D
E
F
G
7
8
5
9
7
5
15
6
8
9
11

# TRAVELLING SALESPERSON PROBLEM

Given a weighed graph, find a **HAMILTONIAN CYCLE** such that the sum of the weights of the edges is the **SMALLEST** possible.

# BRUTE FORCE algorithm

- Generate all possible Hamiltonian cycles

- Select the Hamiltonian cycle with the least cost.

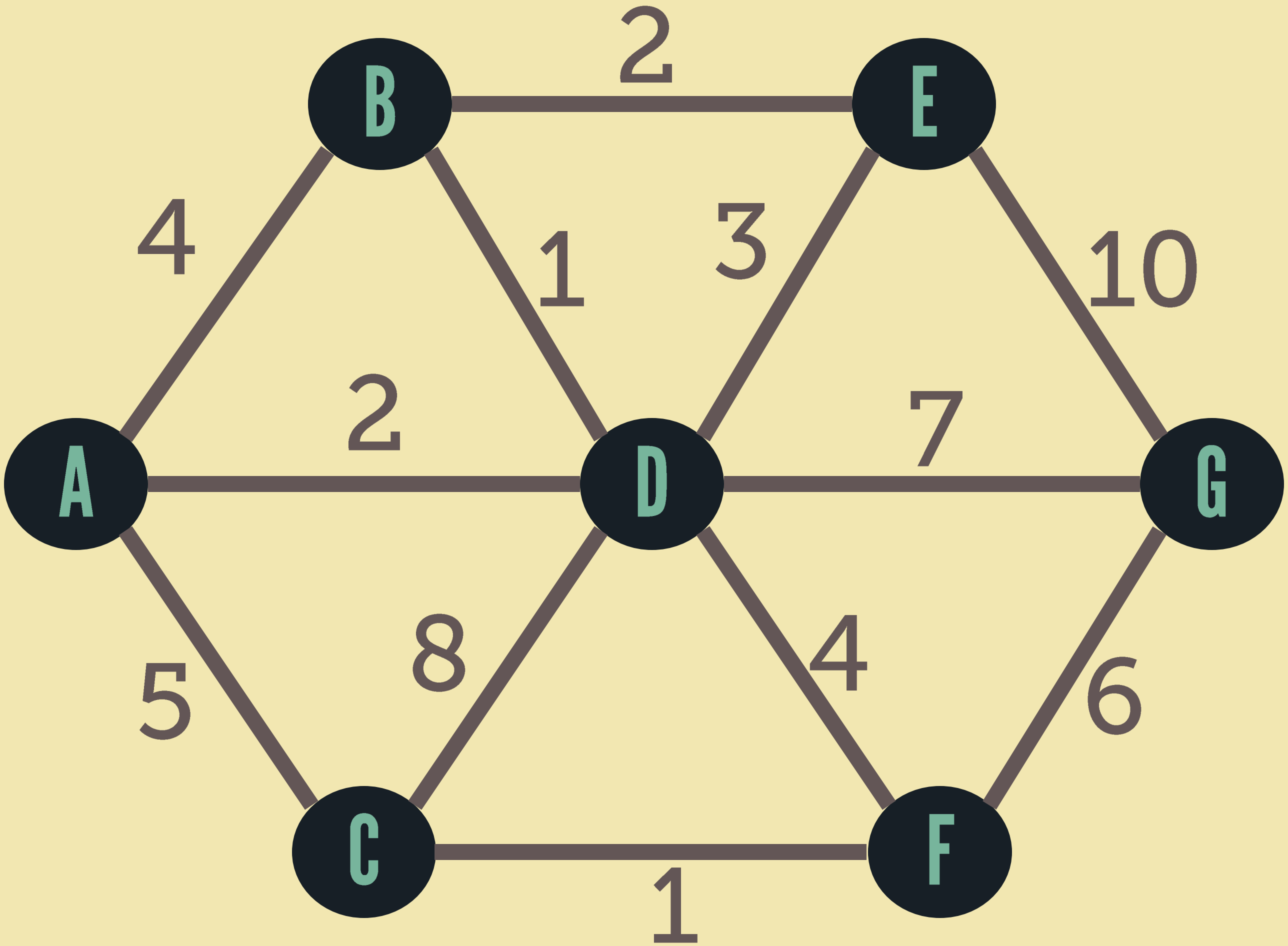
# GREEDY algorithm

Modified Kruskal's algorithm

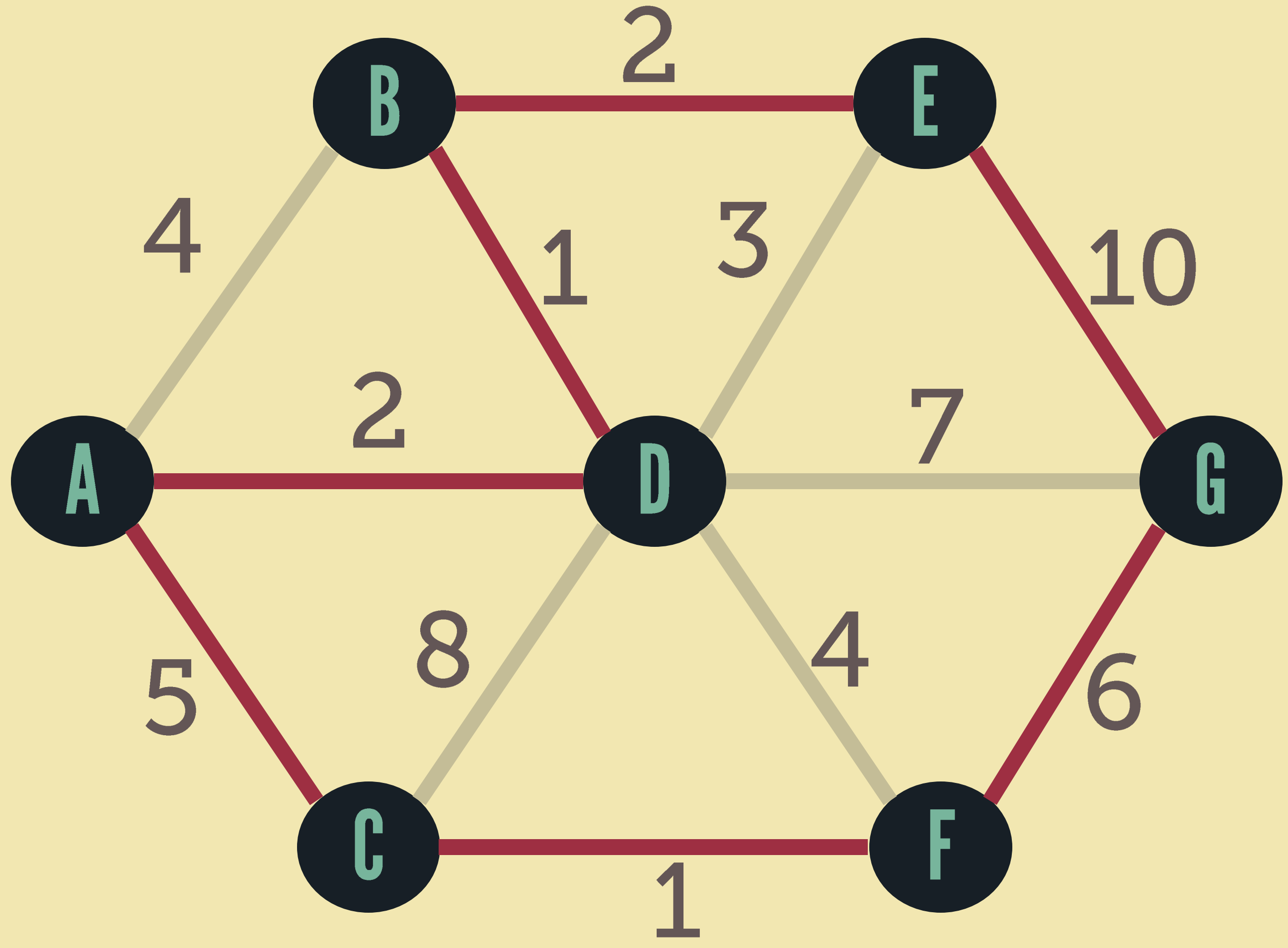
- Start with a null graph T with vertices of G.

- Add currently cheapest edge from G to T that
    - will not form a cycle in T
    - will not cause a vertex in T to have a degree of 3 or more.

- Repeat previous step until # edges = # vertices.



B
E
A
D
G
C
F
2
4
1
3
10
2
7
5
8
4
6
1



A
B
C
D
E
F
G
2
4
1
3
10
2
7
5
8
4
6
1

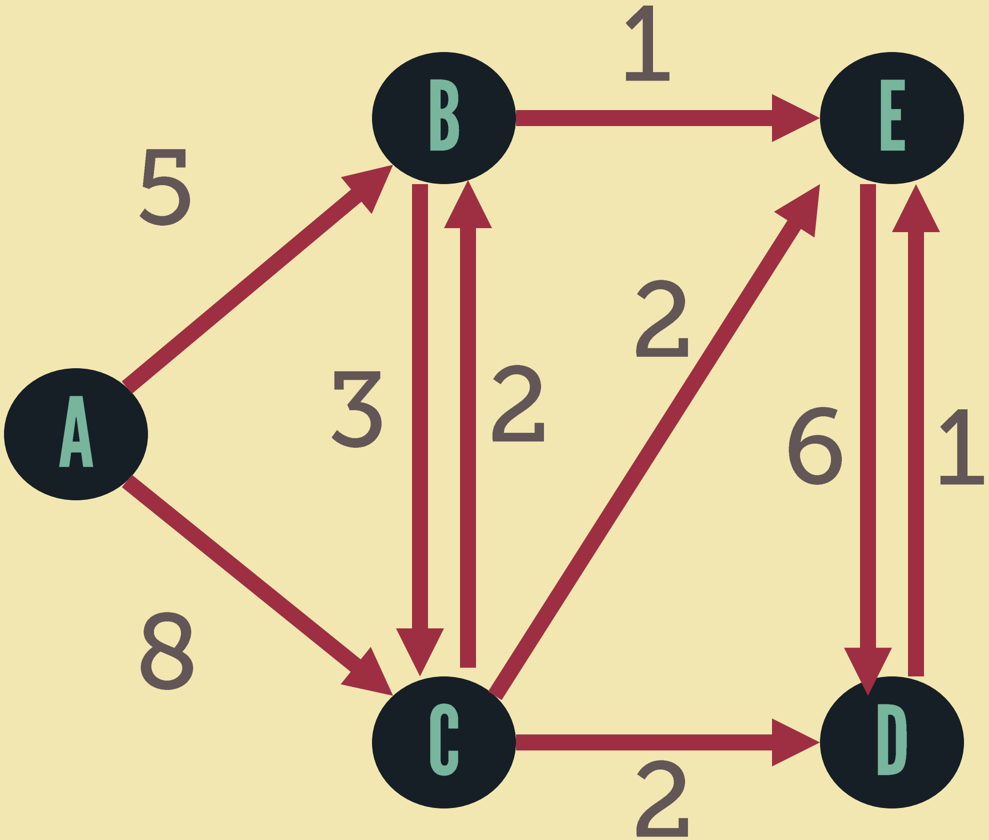
# SHORTEST PATH PROBLEM

Given a weighed (directed) graph, find the **SHORTEST PATH** from vertex u to v.

DIJKSTRA'S algorithm

FLOYD'S algorithm

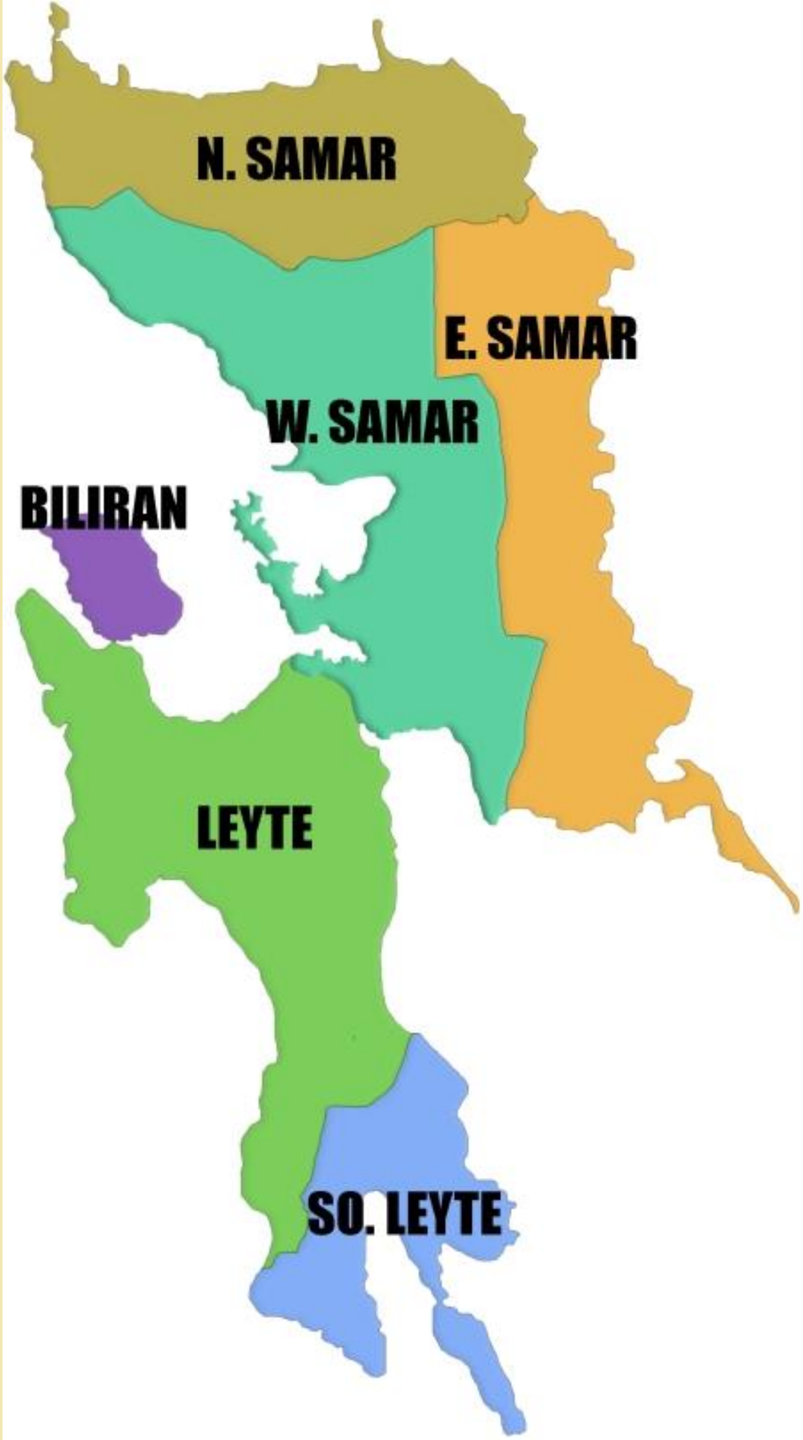
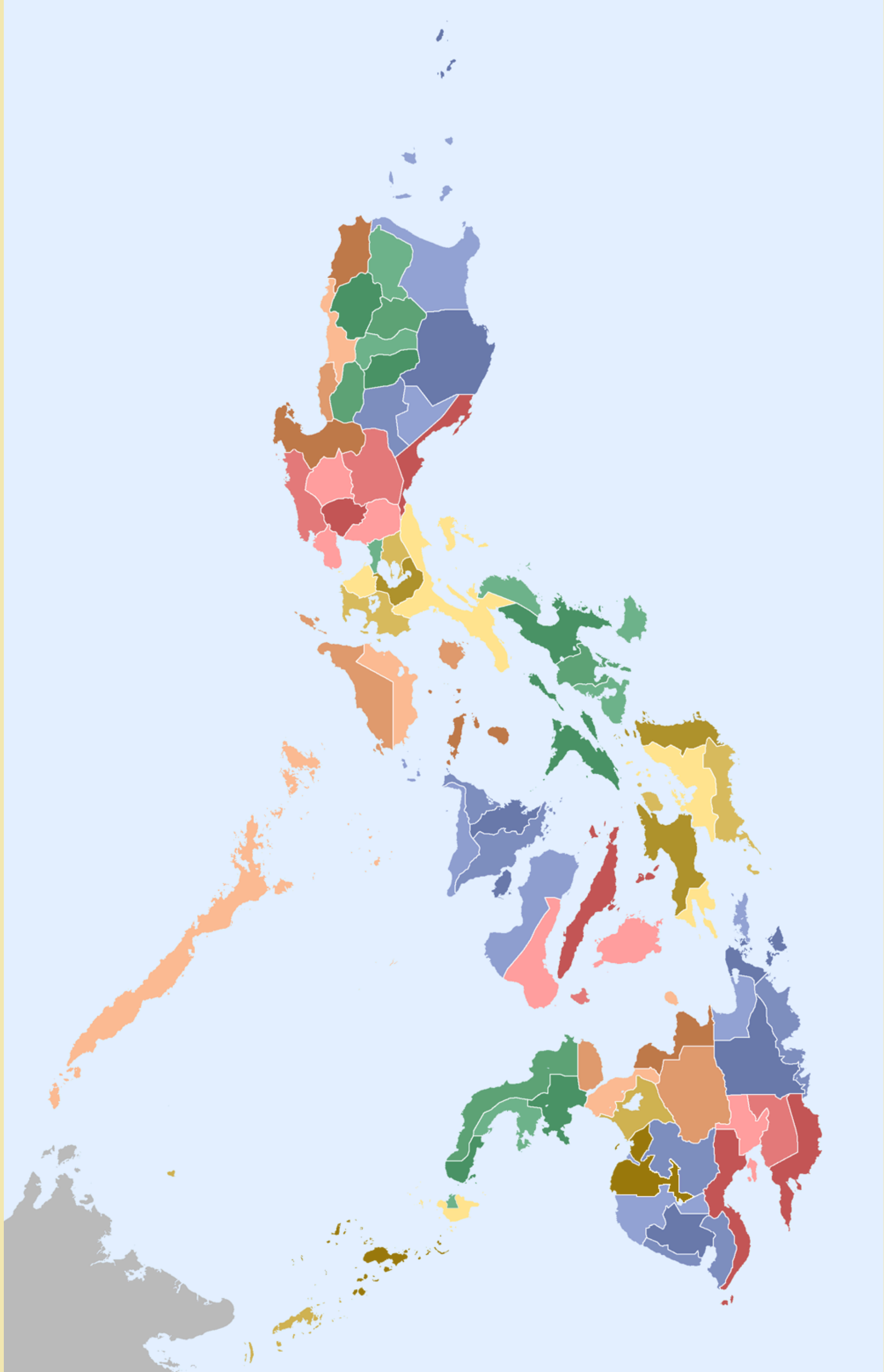
- Find the shortest paths from a specified source vertex to every other vertices in the graph.


A
B
C
D
E
5
8
1
3
2
2
2
6
1

# GRAPH COLORING

# VERTEX COLORING

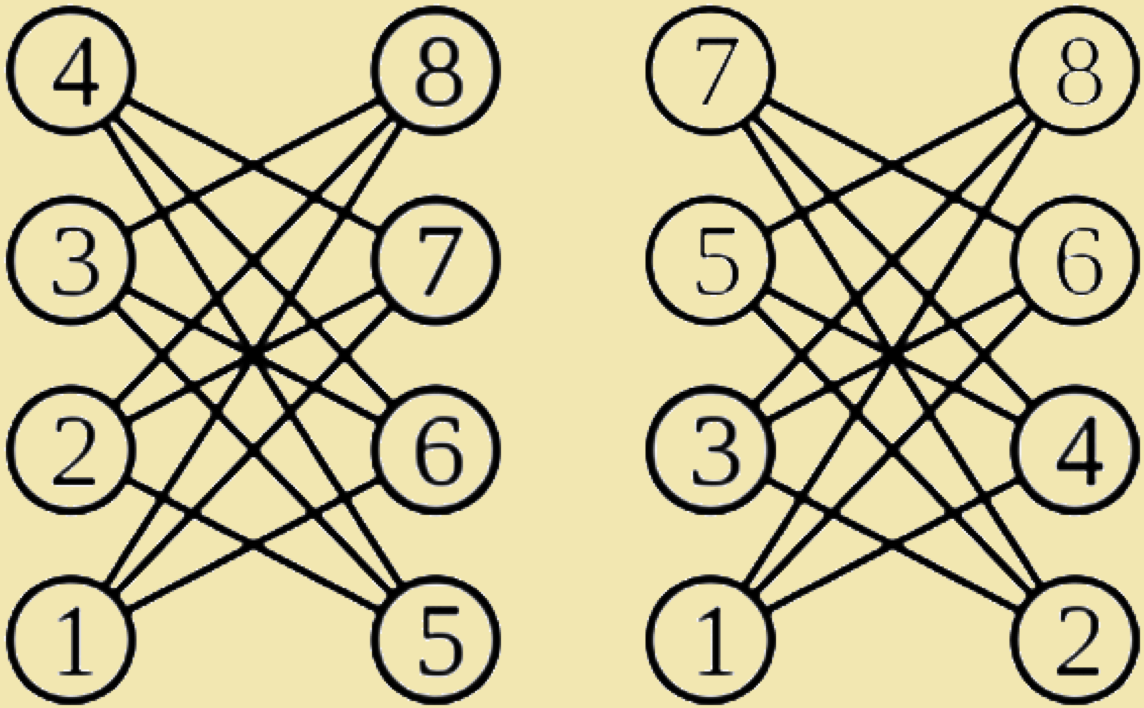
Color the vertices of a graph such that no two adjacent vertices have the same color.


N. SAMAR
E. SAMAR
W. SAMAR
BILIRAN
LEYTE
SO. LEYTE


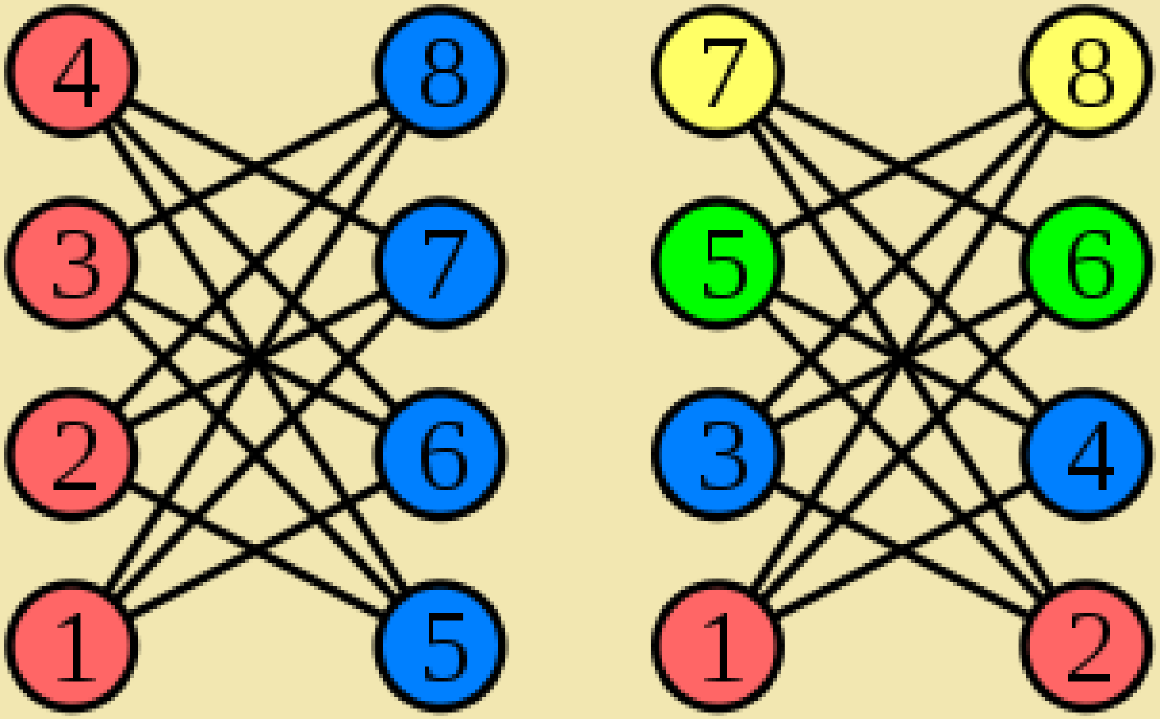
# GREEDY algorithm

- Consider the vertices in a specific order $v_1, \ldots v_n$.

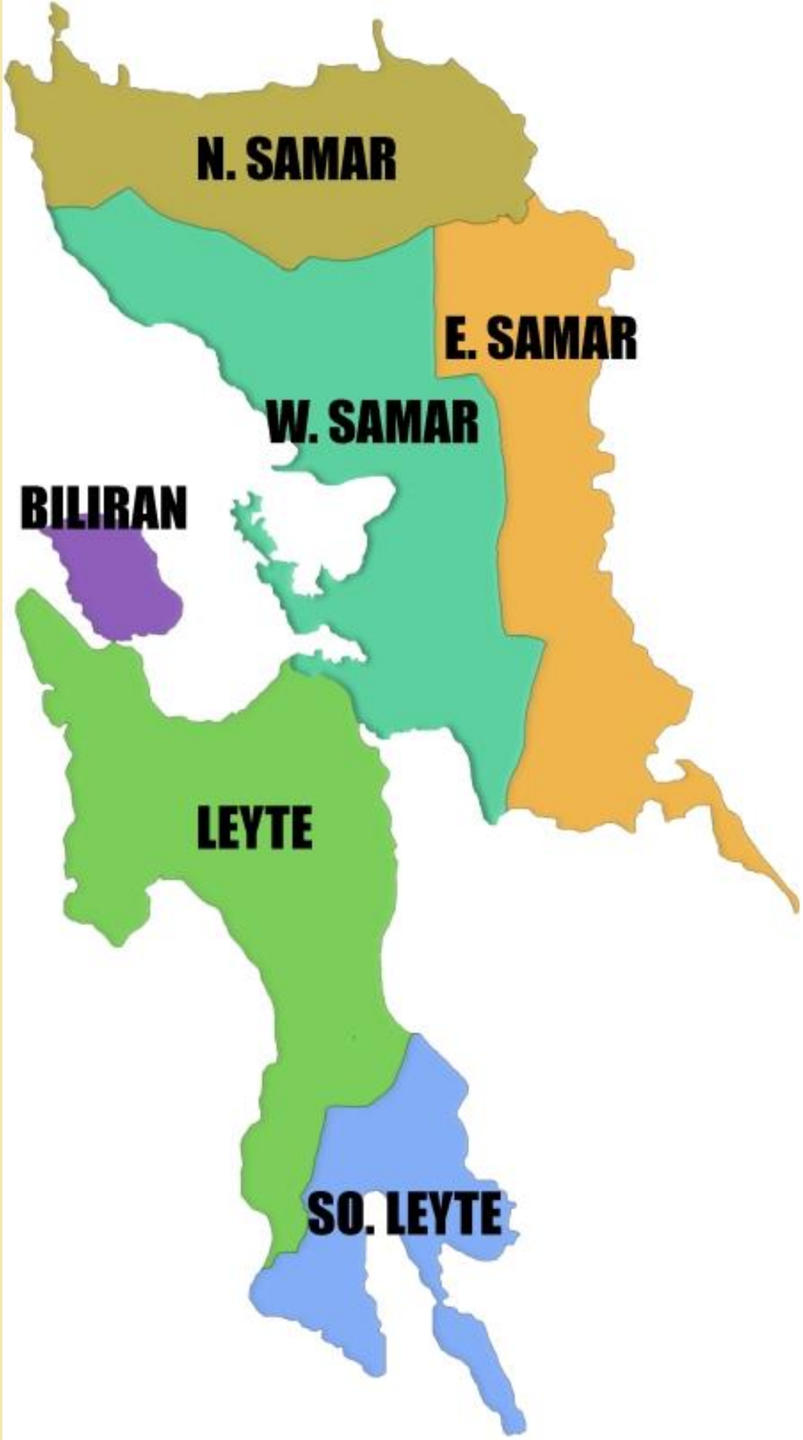
- Assign to $v_i$ the smallest available color not used by $v_i$'s neighbors (add a new color if needed).

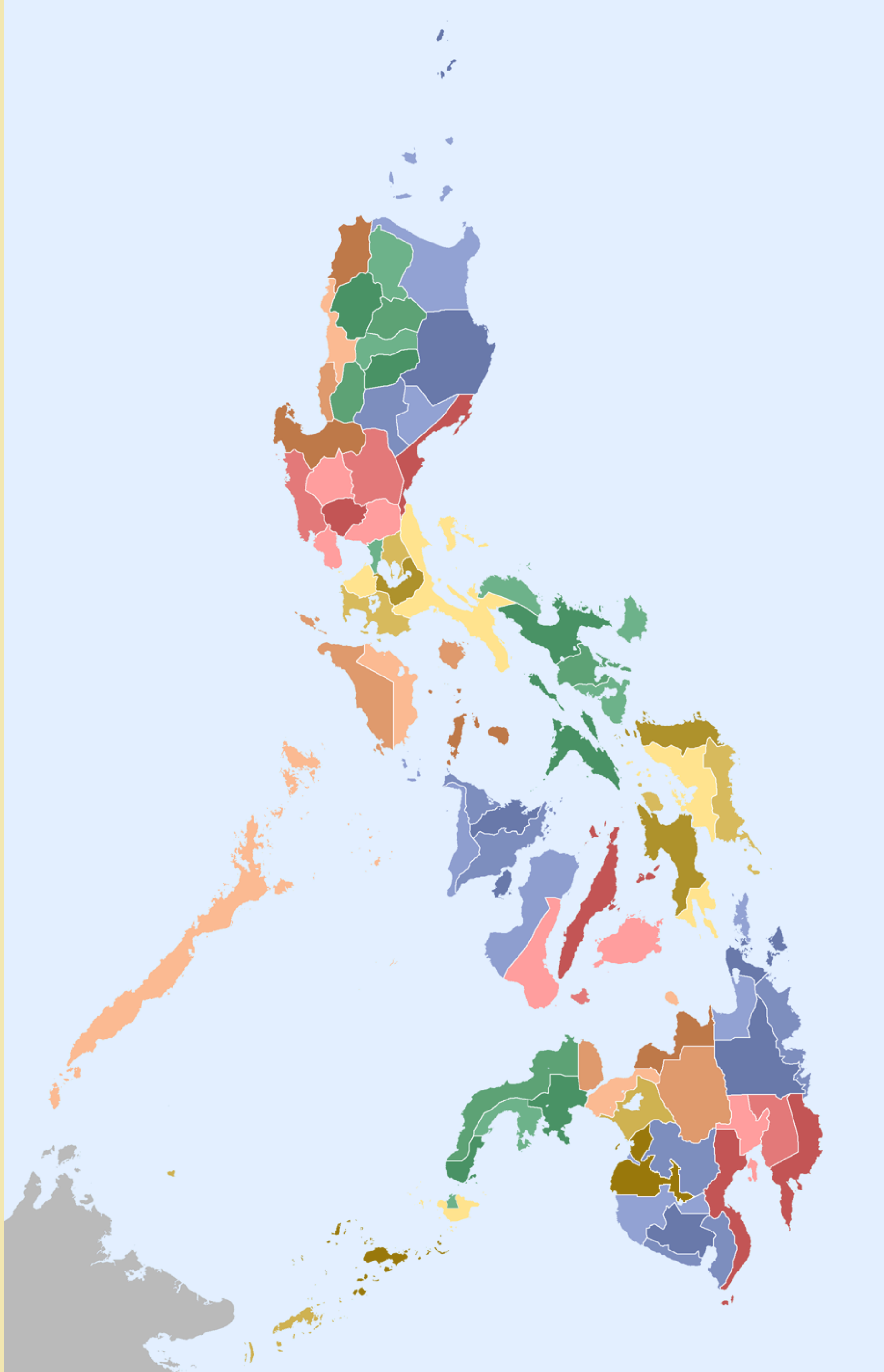



by Thore Husfeldt



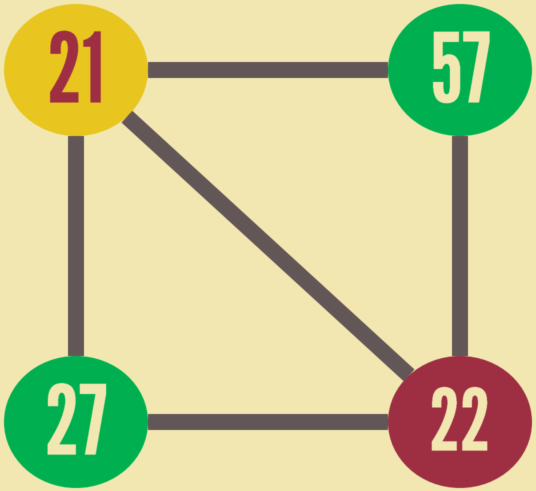

by Thore Husfeldt


N. SAMAR
E. SAMAR
W. SAMAR
BILIRAN
LEYTE
SO. LEYTE



What is the minimum number of time slots needed to schedule 4 exams given the following:

CMSC 21: Annie, Armin
CMSC 57: Annie, Mikasa, Eren
MATH 27: Armin
CMSC 22: Annie, Armin, Eren, Jean

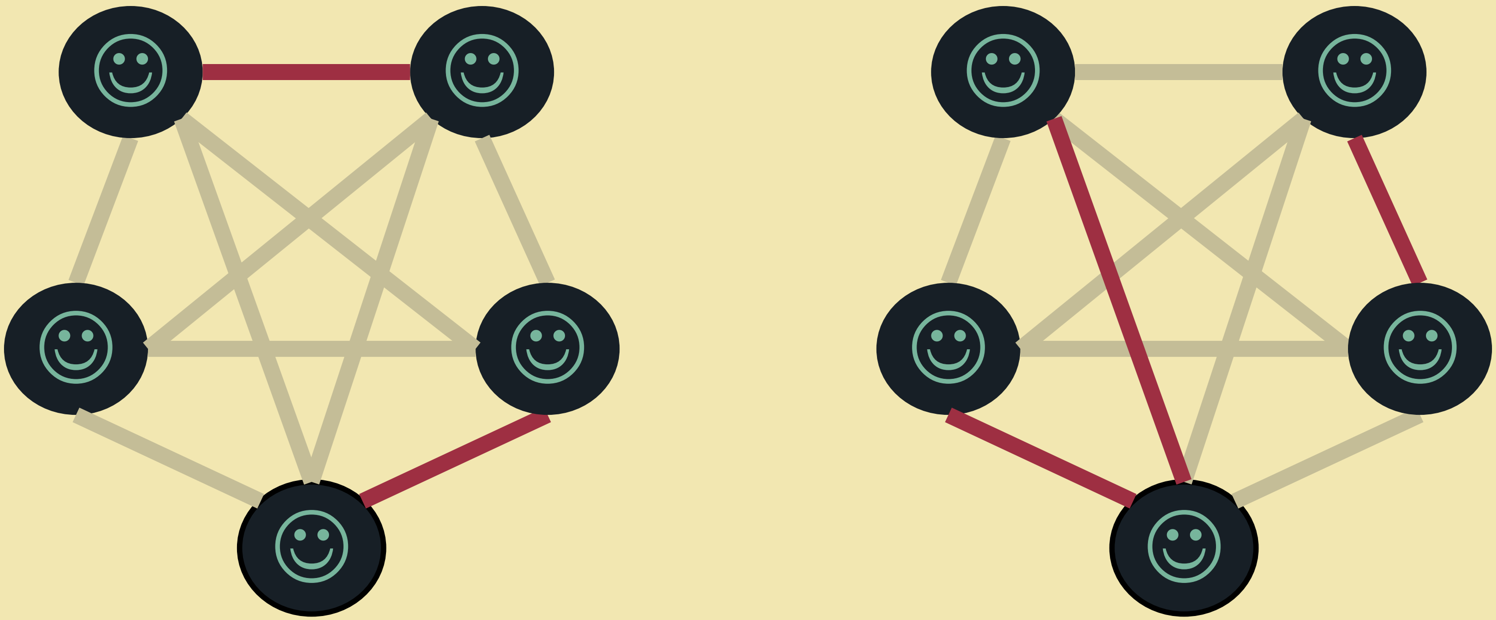

21
57
27
22

# GRAPH MATCHING

Pair off as many vertices as possible, that is, find a **MAXIMAL MATCHING** for a given graph.

# MATCHING

Set of edges where no two edges are adjacent

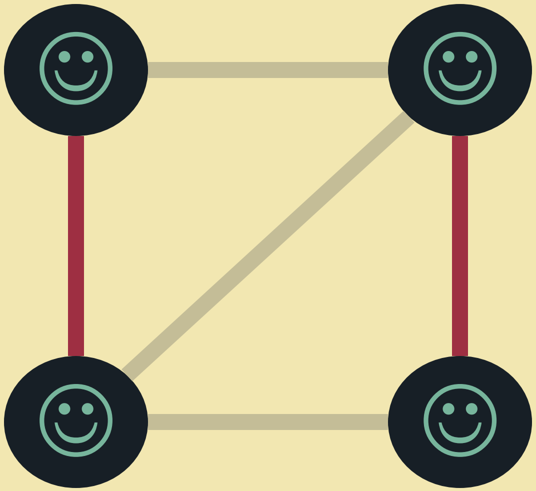


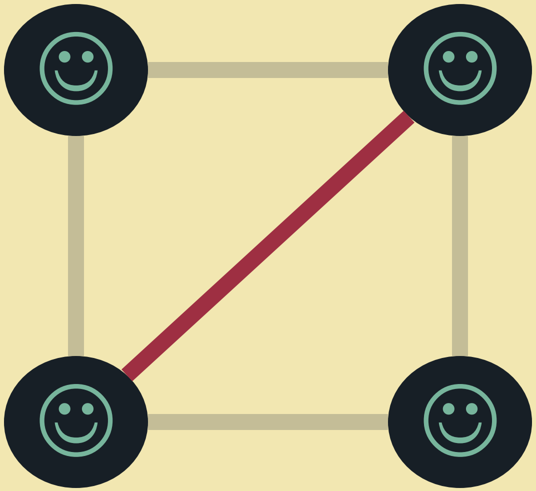
# MAXIMAL MATCHING

Matching with the largest number of edges.

# PERFECT MATCHING

A matching where every vertex in G is matched.





# AUGMENTING PATHS

method