Greedy algorithms

CMPSC 465 - Yana Safonova

Introduction

Content

- Introduction to greedy algorithms
- The minimum spanning tree of a graph
    - Applications
    - Algorithms
    - Efficient implementations
    - Connections with data clustering
- Huffman coding
- Cover set problem

What is a greedy algorithm

Greedy heuristic:

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy approach

- How to formulate a greedy heuristic for a specific problem?
- Does it provide the optimal solution?
- How fast does an algorithm work?

Greedy approach

- How to formulate a greedy heuristic for a specific problem?

Let’s look at examples

- Does it provide the optimal solution?

Not always

- How fast does an algorithm work?

Usually very fast

Example - 1

0-1 Knapsack Problem

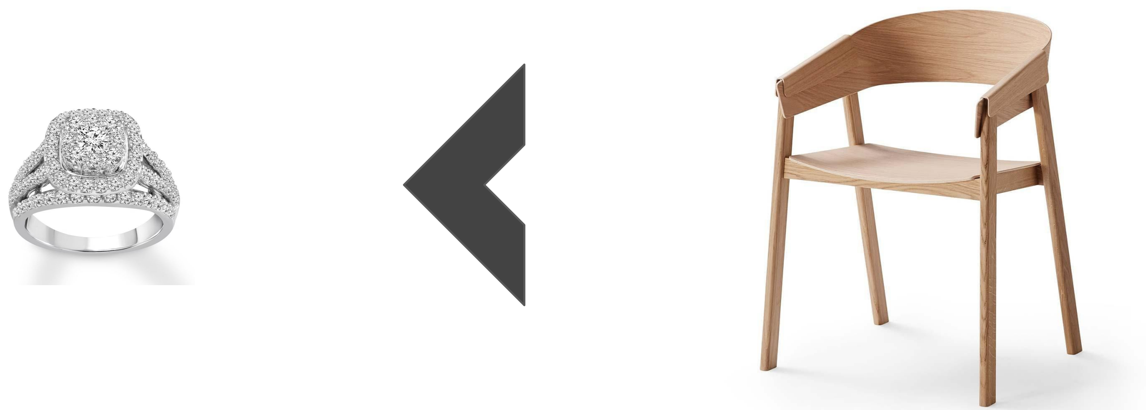
A Thief has a backpack with certain capacity. There is a set of items with certain weight and value. **Goal:** pack the backpack with the largest value

Example - 1

0-1 Knapsack Problem

A Thief has a backpack with certain capacity. There is a set of items with certain weight and value. **Goal:** pack the backpack with the largest value

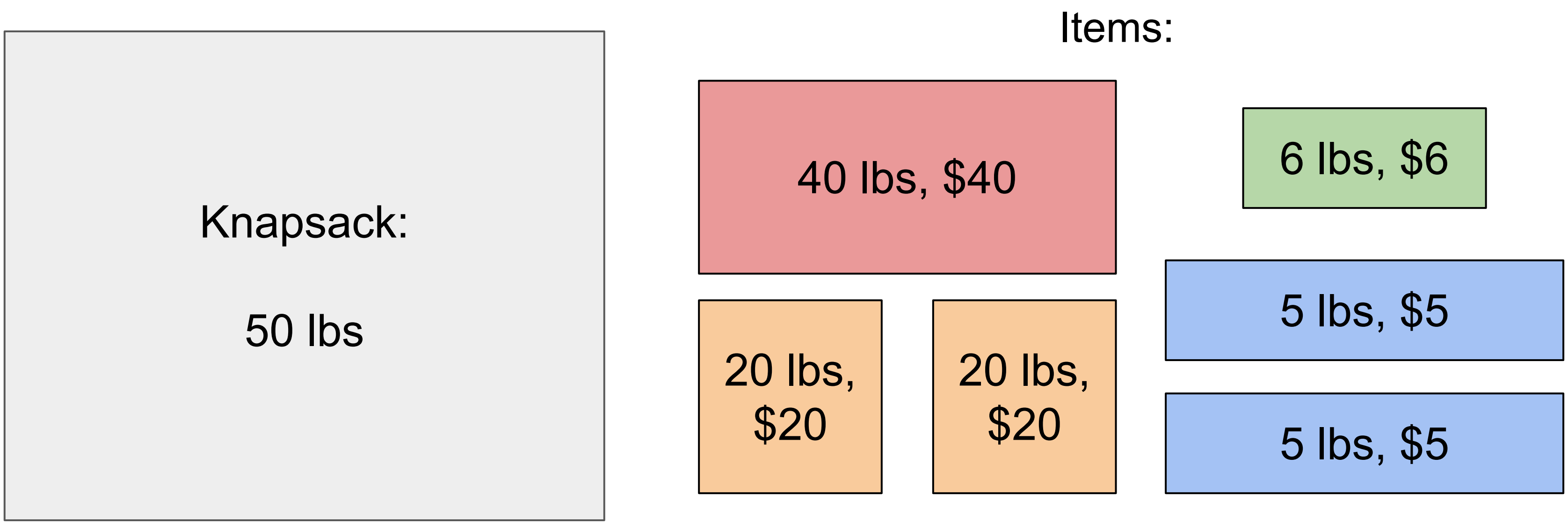
Assumption: the value is proportional to the weight



Example - 1

0-1 Knapsack Problem

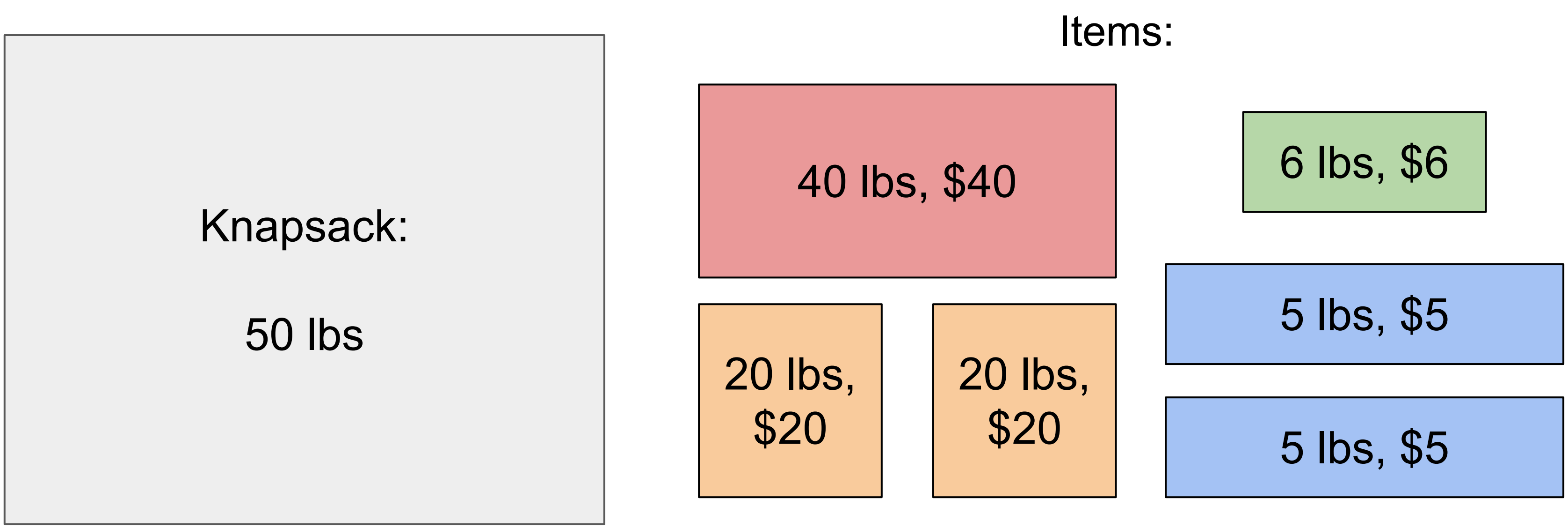
A Thief has a backpack with certain capacity. There is a set of items with certain weight and value. **Goal:** pack the backpack with the largest value



Example - 1

What is the greedy heuristic here?

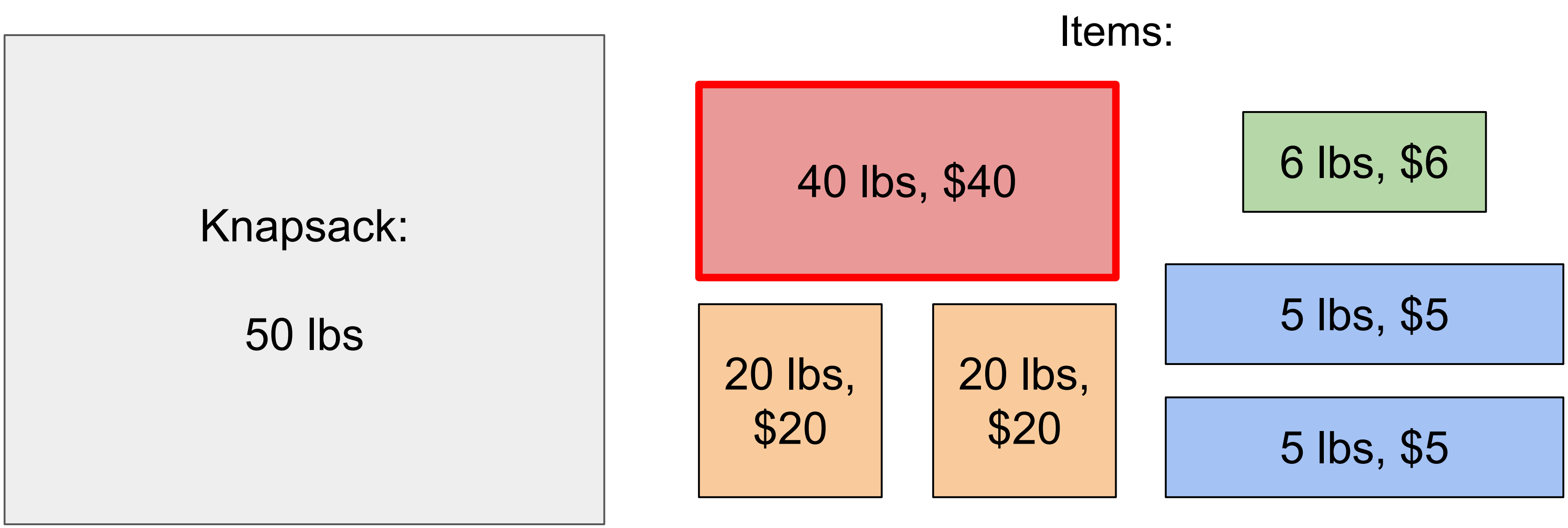
Let’s take the largest (= the most valuable item) and put it into the bag if it fits



Example - 1

What is the greedy heuristic here?

Let's take the largest (= the most valuable item) and put it into the bag if it fits

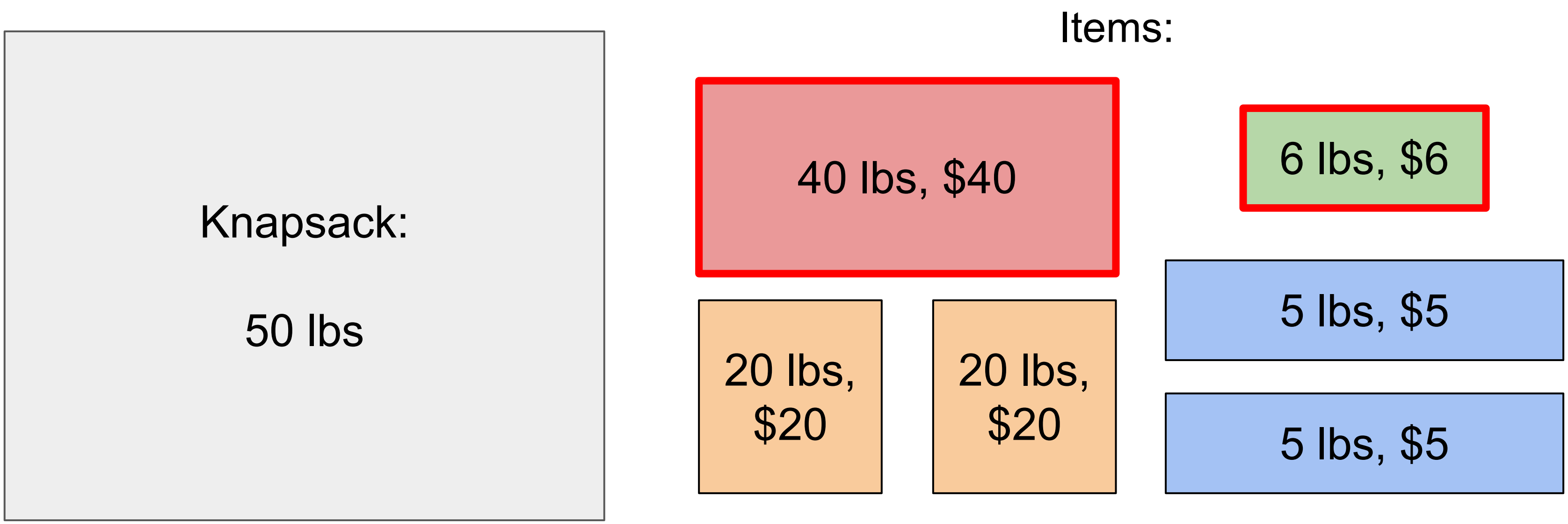


Total value: $40

Example - 1

What is the greedy heuristic here?

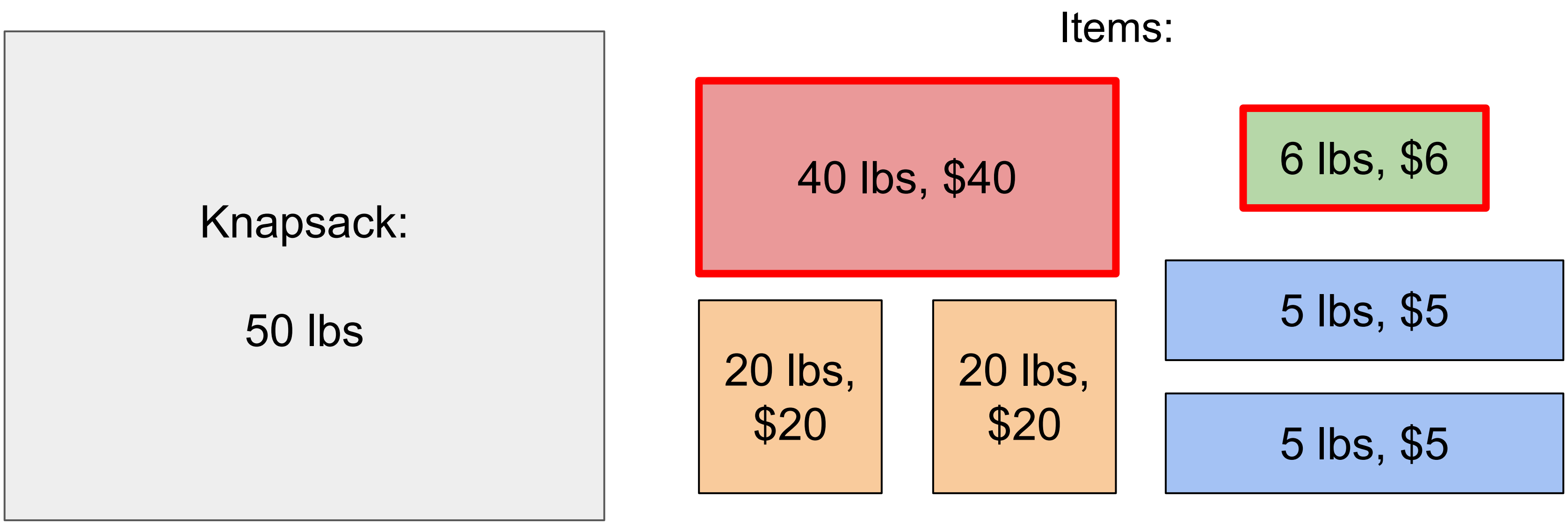
Let’s take the largest (= the most valuable item) and put it into the bag if it fits



Total value: $40 + $6 = $46

Example - 1

Is it the optimal solution?

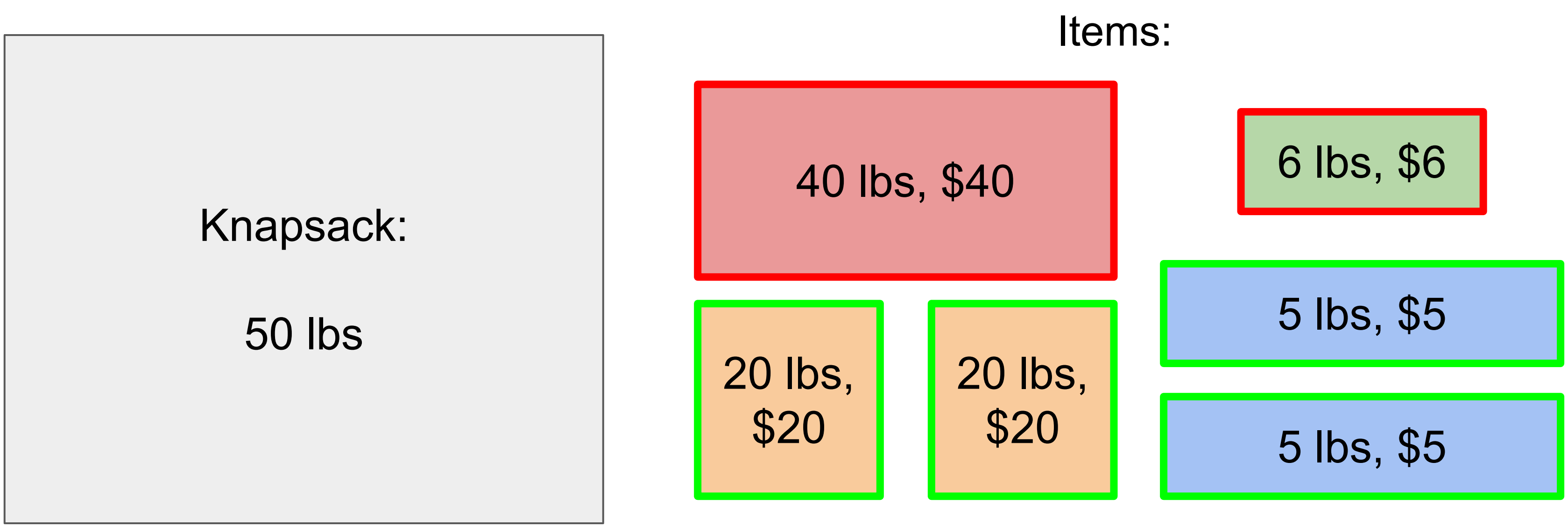


Total value: $40 + $6 = $46

Example - 1

Is it the optimal solution?

No!



Total value: \$20 + \$20 + \$5 + \$5 = \$50

Why do we need greedy algorithms?

Practical note: They are easy to think of and computationally inexpensive, and sometimes they can give a “good” suboptimal solution

“Good” could mean as approximate (the cover set problem), as heuristic - it depends on a specific problem

Theoretical note: there is a class of problem where greedy algorithms provide the optimal solution

Example - 1: implementation aspects

A greedy solution:

1. Sort items by value
2. Iterate over the sorted item list and, for each item, decide whether or not it fits

N items

1 - $O(N \log N)$

2 - $O(N)$

$O(N \log N) + O(N) = O(N \log N)$

Example - 1: implementation aspects

The optimal solution:

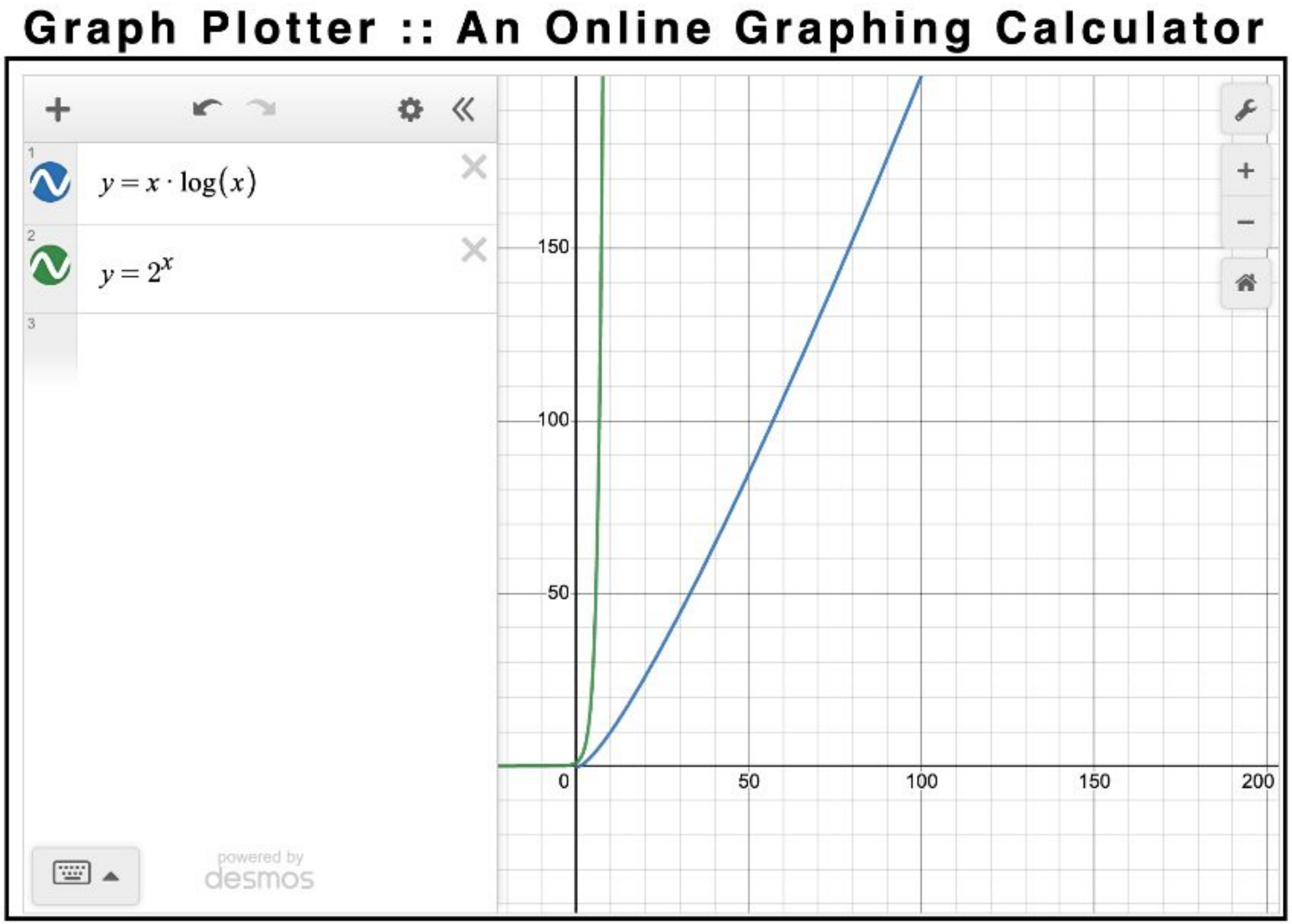
1. Create all combinations of items (all binary vectors)
2. Iterate over all combinations and choose the best one

N items

$1 + 2 = O(2^N)$

A	B	C	D	E
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
…	…	…	…	…
1	1	1	1	1

How different O(N log N) and O(2^N)?



How different O(N log N) and O(2^N)?

N = 25, 1 computation = 1 sec (hypothetical!)

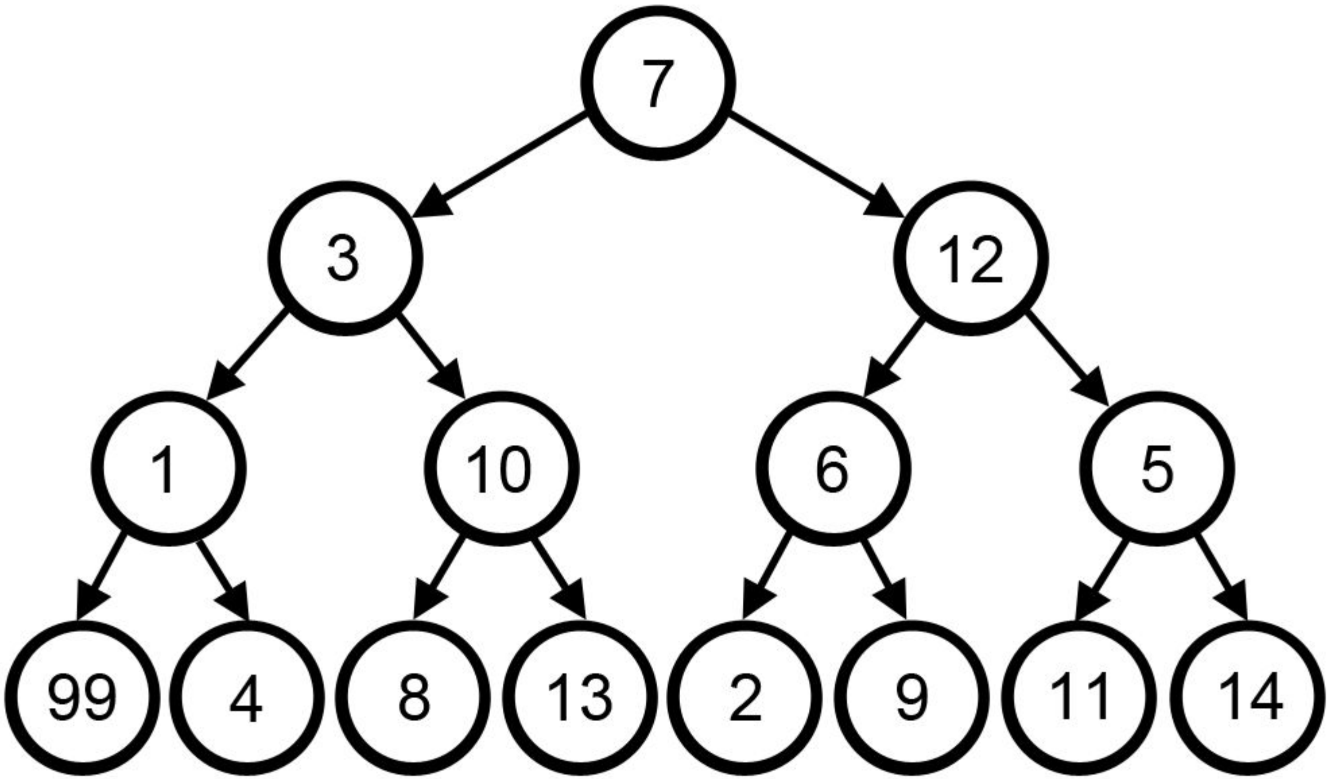
25 * log 25 = 34 sec ~ 0.5 min

2^25 = 33,554,432 s = 1.06 yr

Example - 2

Finding a path from the root to a leaf with the maximum total sum of vertex weights

The red path (7 + 12 + 6 + 9) was selected by the greedy strategy. Is it the optimal one?



When do greedy algorithms produce the optimal solution?

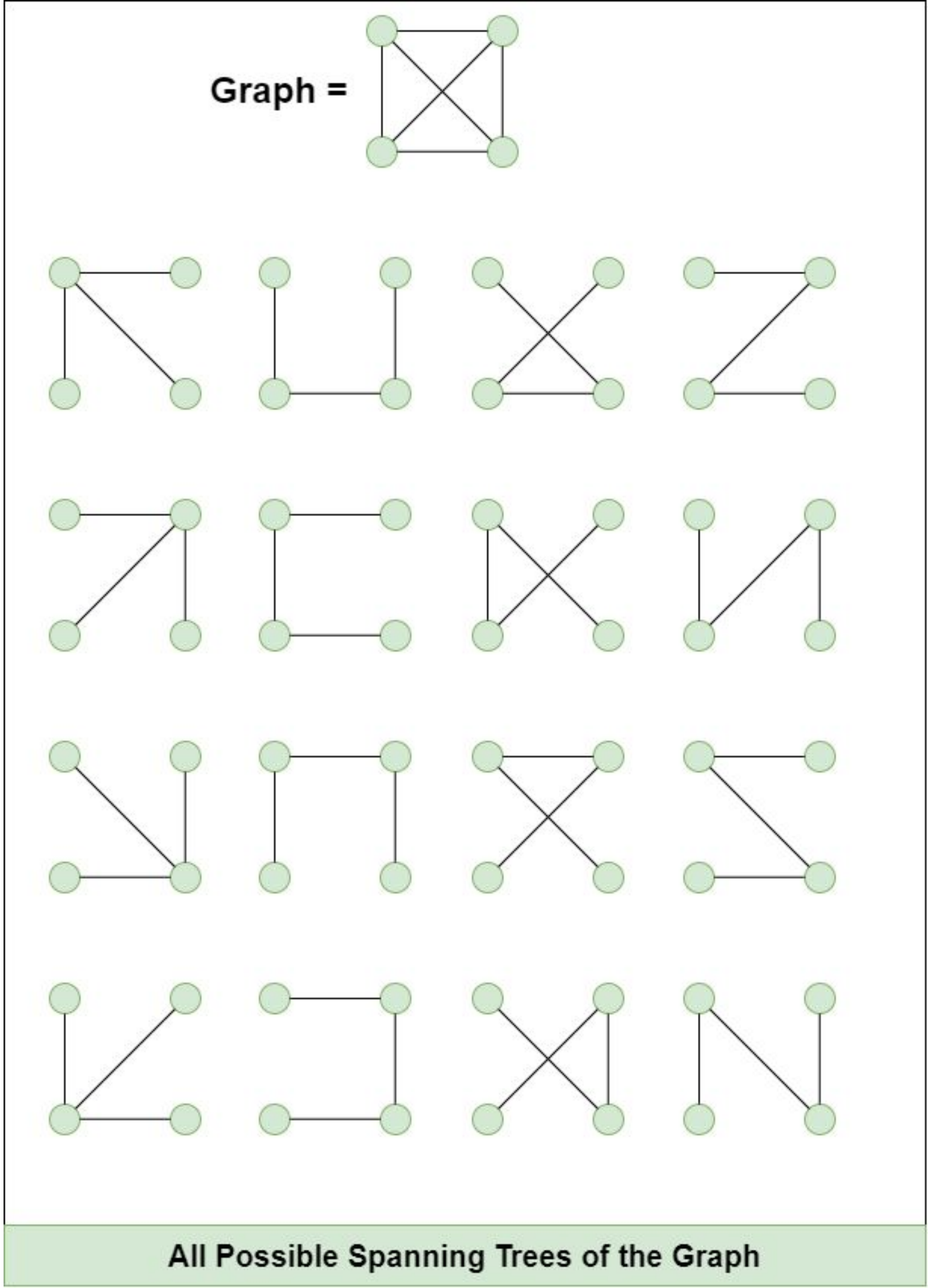
Greedy choice property: A global optimal solution can be reached by choosing the optimal choice at each step

Optimal substructure: A problem has an optimal substructure if an optimal solution to the entire problem contains the optimal solutions to the sub-problems

The minimum spanning tree

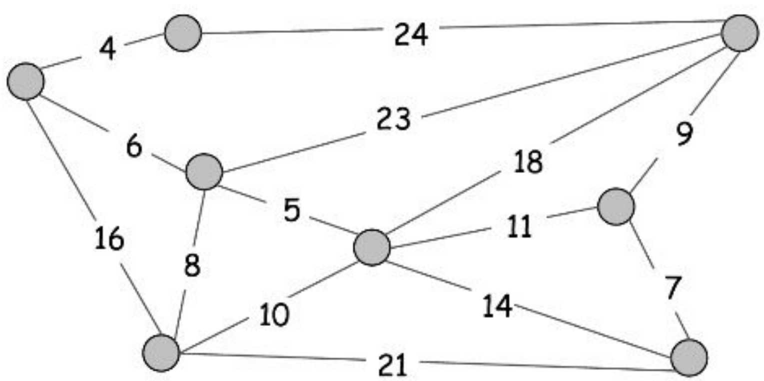
Spanning tree

A spanning tree is a subgraph of a connected graph that connects all the vertices together without any cycles.

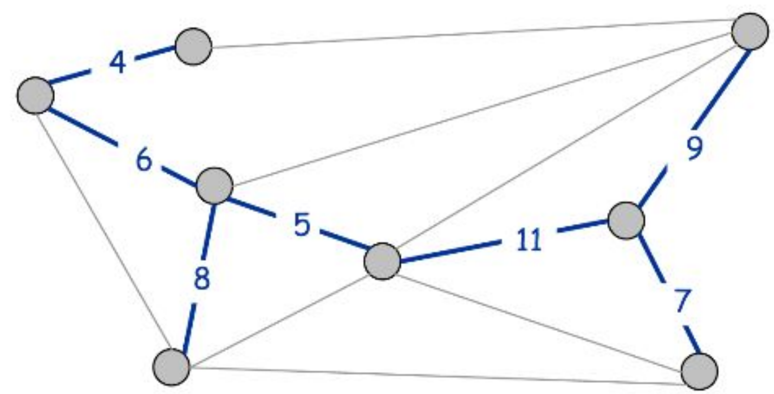


The minimum spanning tree (MST)

Minimum spanning tree. Given a connected graph $G = (V, E)$ with real-valued edge weights c_e, an MST is a subset of the edges $T \subseteq E$ such that T is a spanning tree whose sum of edge weights is minimized.



$G = (V, E)$



$T, \ \Sigma_{e \in T} c_e = 50$

MST applications

- Network design
  - Circuit, telephone, electrical, TV cable, computer, road
- Taxonomy
  - Evolutionary analysis
- Clustering
  - Single-linkage clustering of the data
- Subroutine in other algorithms

MST finding algorithms

- Kruskal’s algorithm
- Kruskal’s variation: the reverse-delete algorithm
- Prim’s algorithm

MST finding algorithms

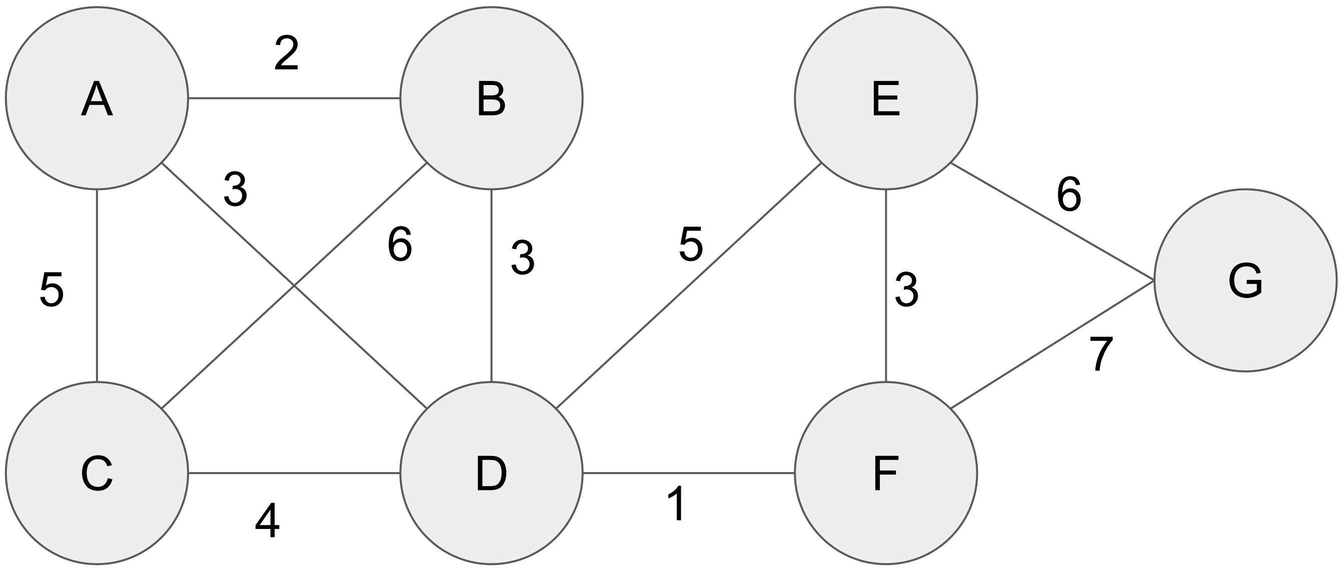
Kruskal's algorithm. Start with $T = \phi$. Consider edges in ascending order of cost. Insert edge e in T unless doing so would create a cycle.

Reverse-Delete algorithm. Start with $T = E$. Consider edges in descending order of cost. Delete edge e from T unless doing so would disconnect T.

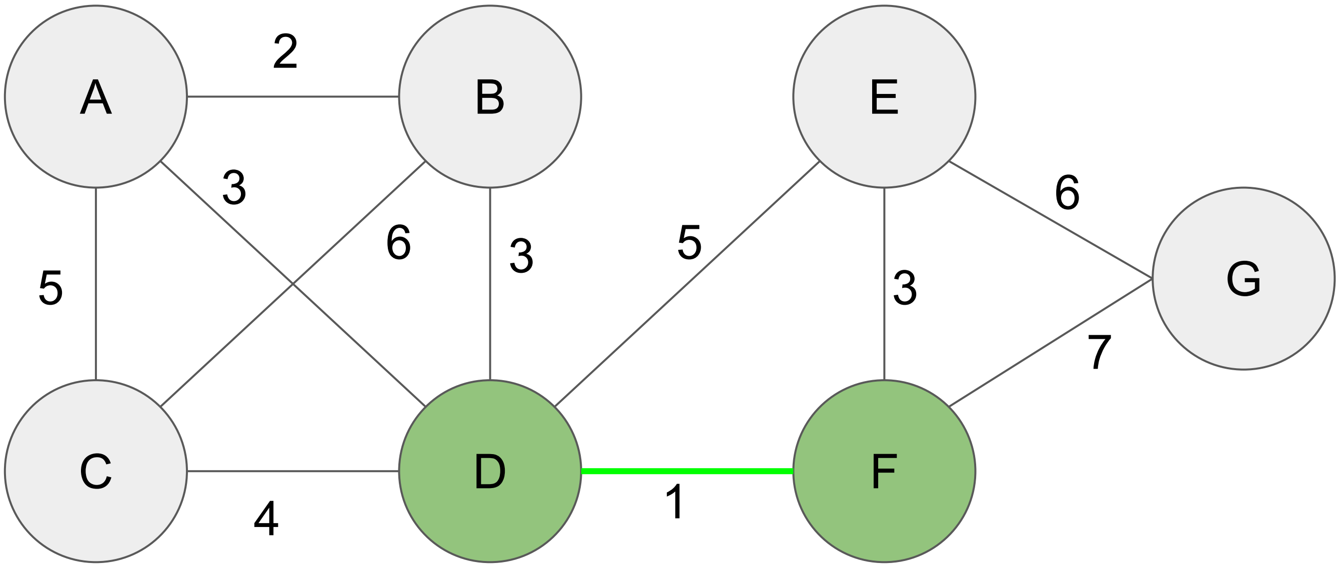
Prim's algorithm. Start with some root node s and greedily grow a tree T from s outward. At each step, add the cheapest edge e to T that has exactly one endpoint in T.

Remark. All three algorithms produce an MST.

Kruskal’s algorithm

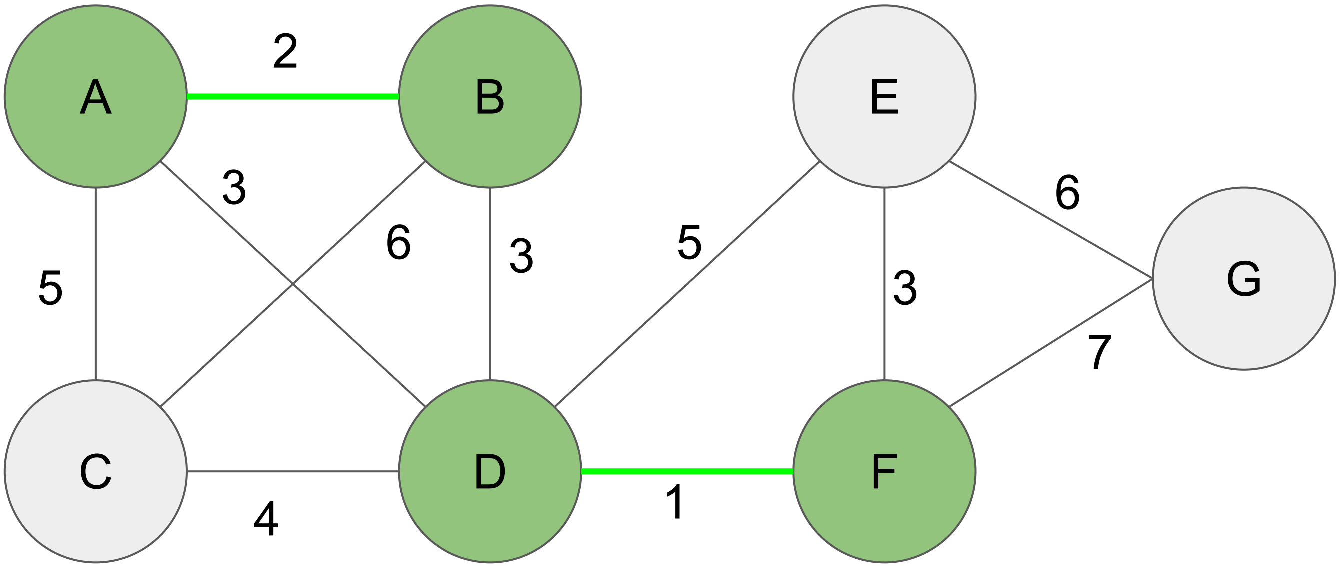


Kruskal's algorithm - weight 1



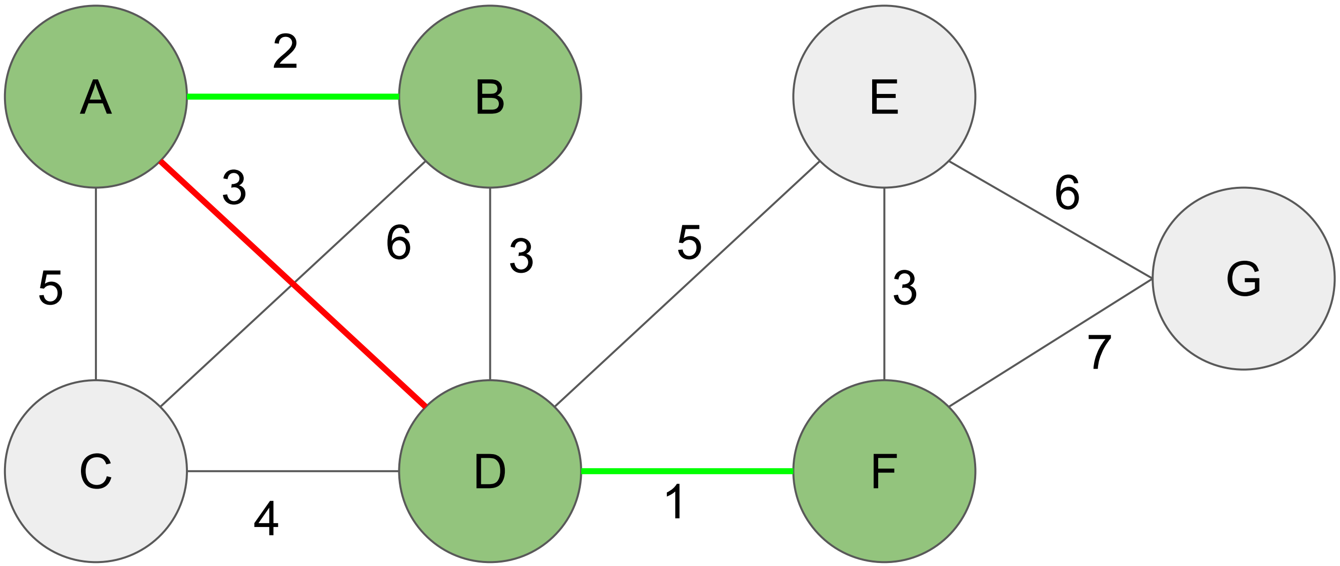
(D, F) can be added

Kruskal's algorithm - weight 2



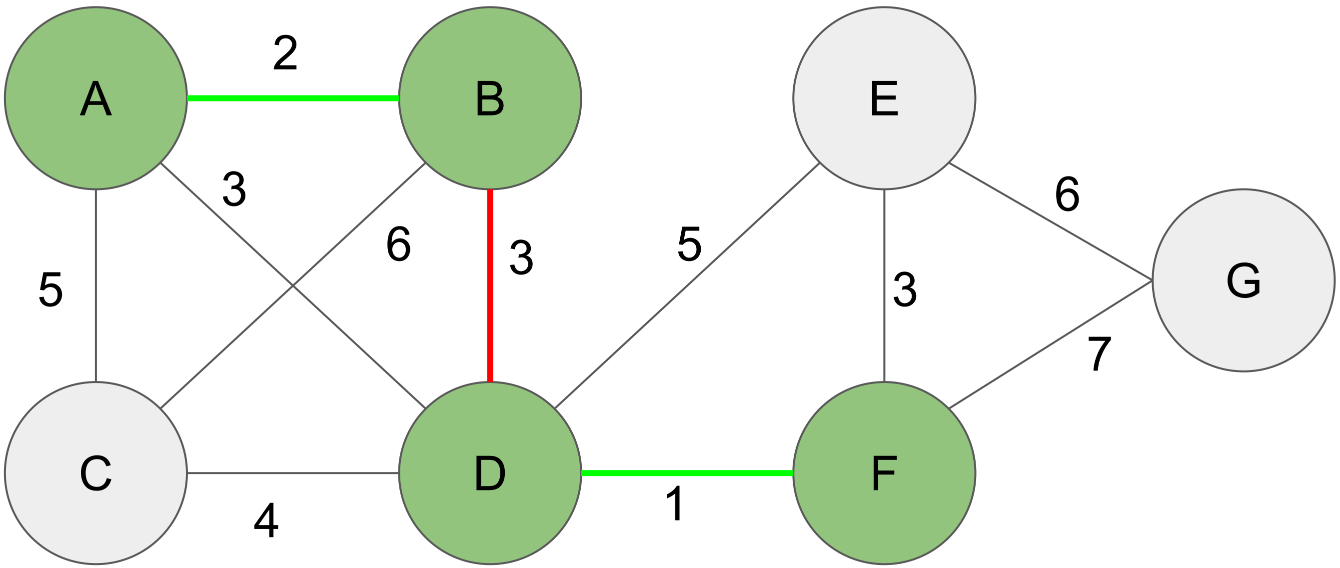
(A, B) can be added

Kruskal’s algorithm - weight 3



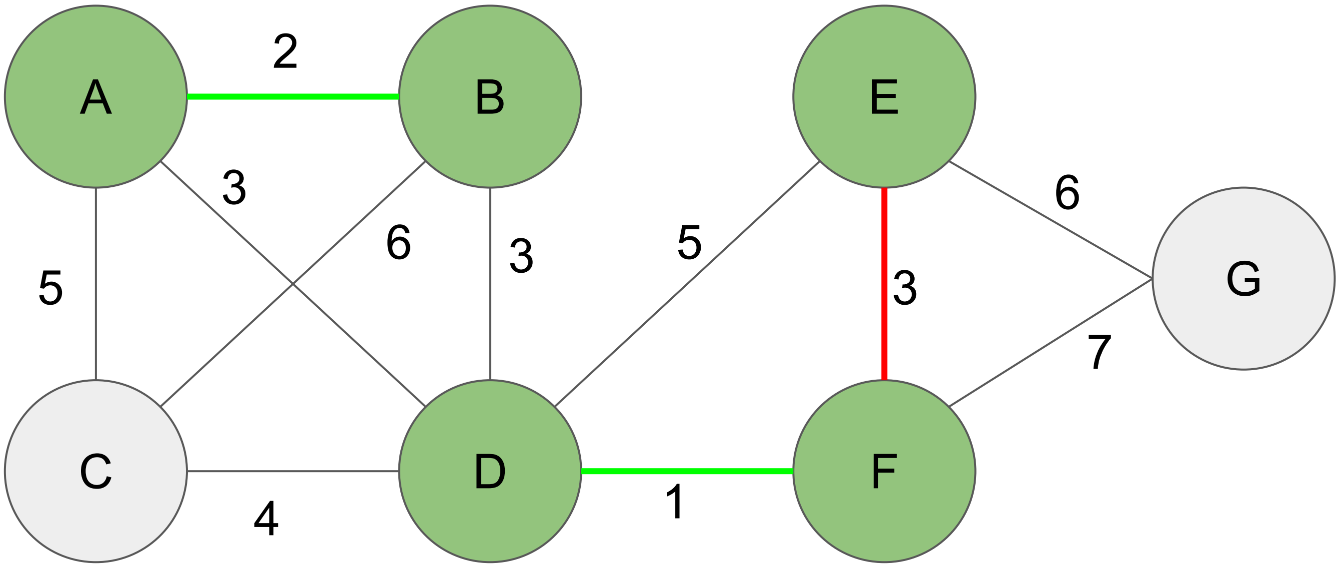
(A, D), (B, D), (E, F)

Kruskal’s algorithm - weight 3



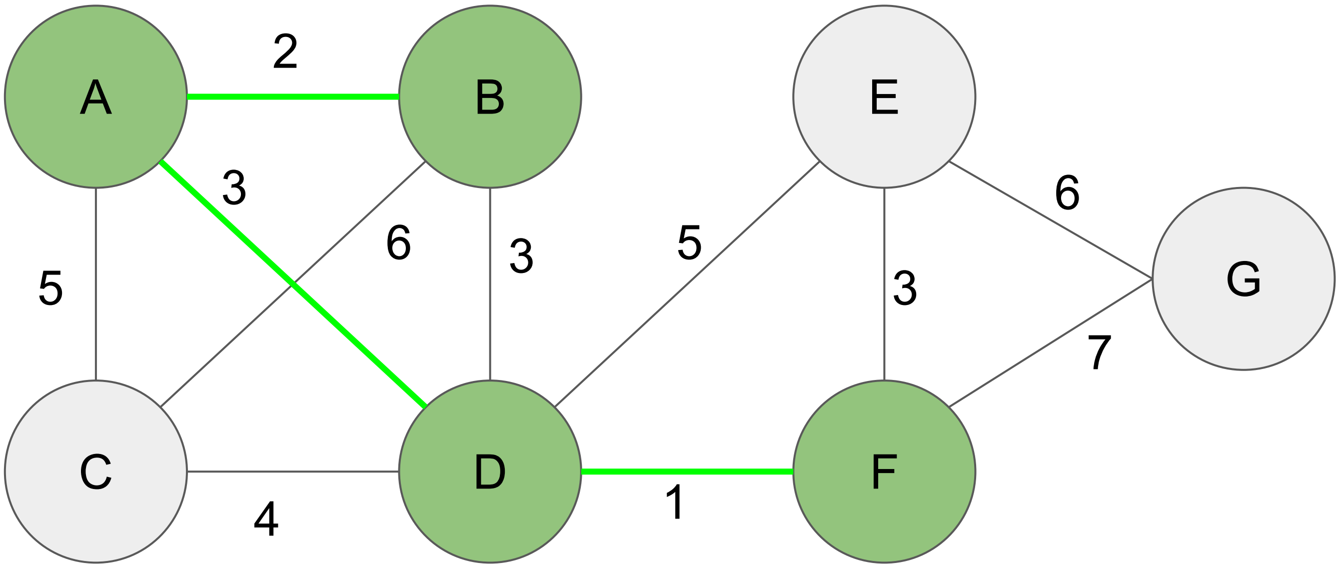
(A, D), (B, D), (E, F)

Kruskal’s algorithm - weight 3



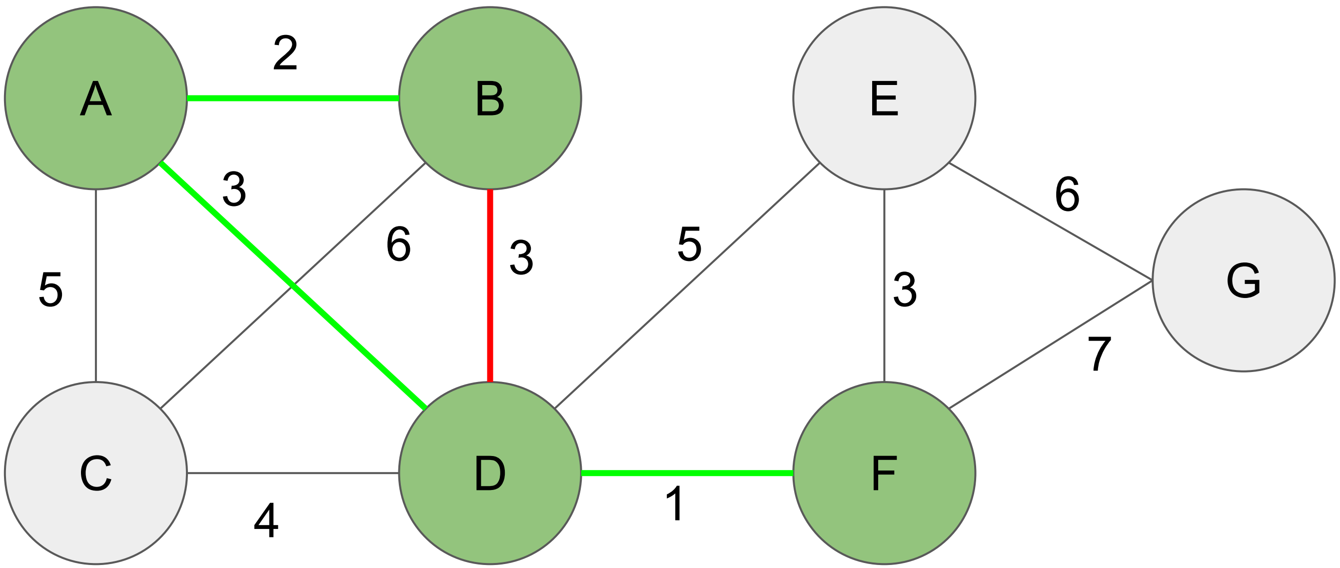
(A, D), (B, D), (E, F)

Kruskal’s algorithm - weight 3



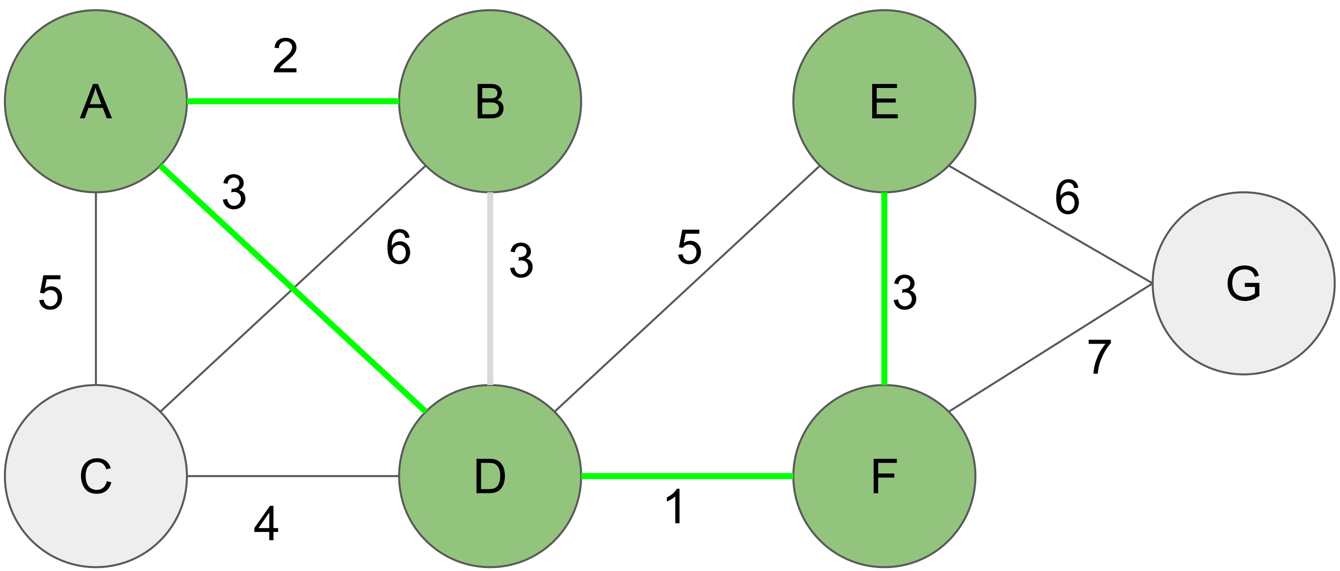
We can add (A, D)

Kruskal’s algorithm - weight 3



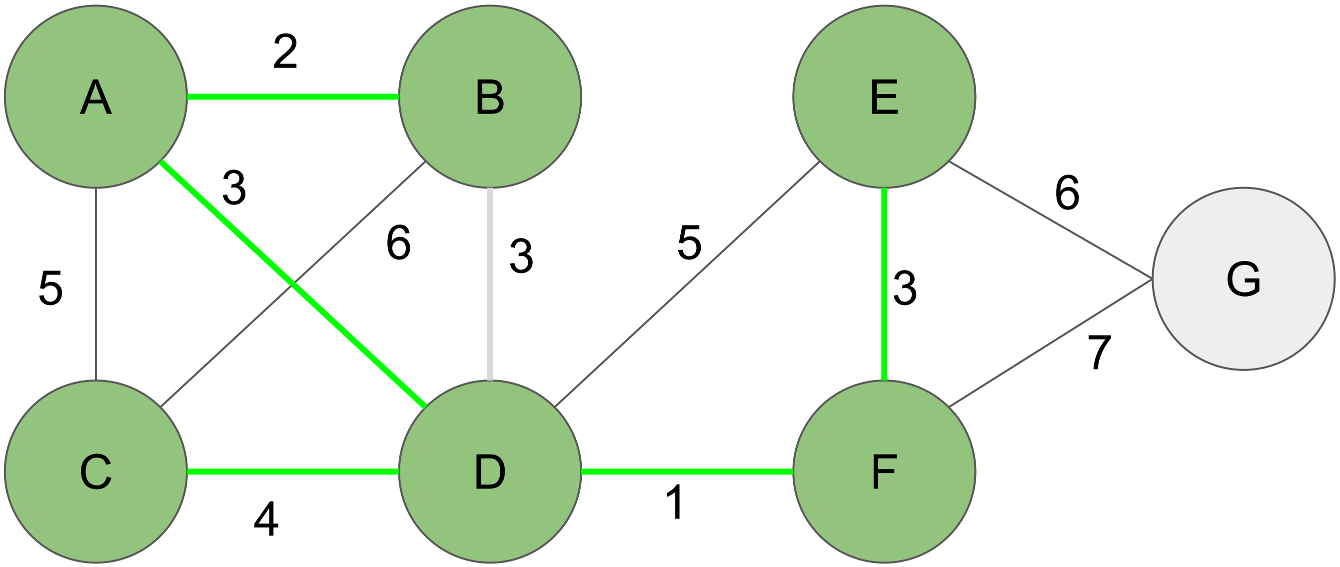
(B, D), (E, F)?
(B, D) creates a cycle A–B–D

Kruskal's algorithm - weight 3



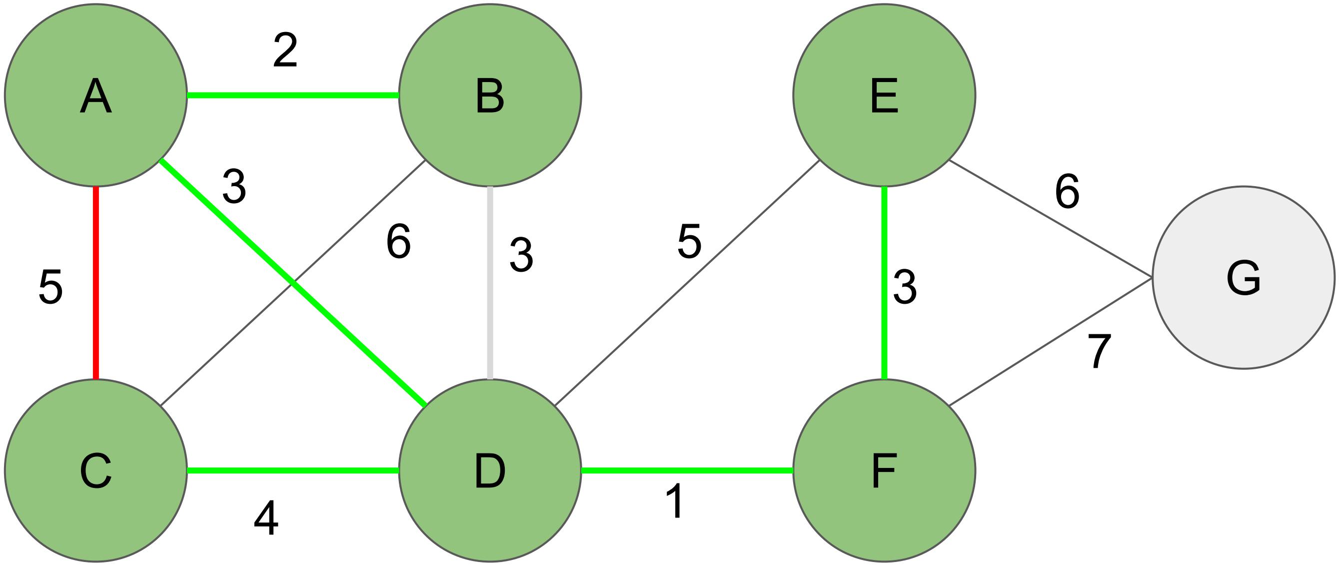
We can add (E, F)

Kruskal’s algorithm - weight 4



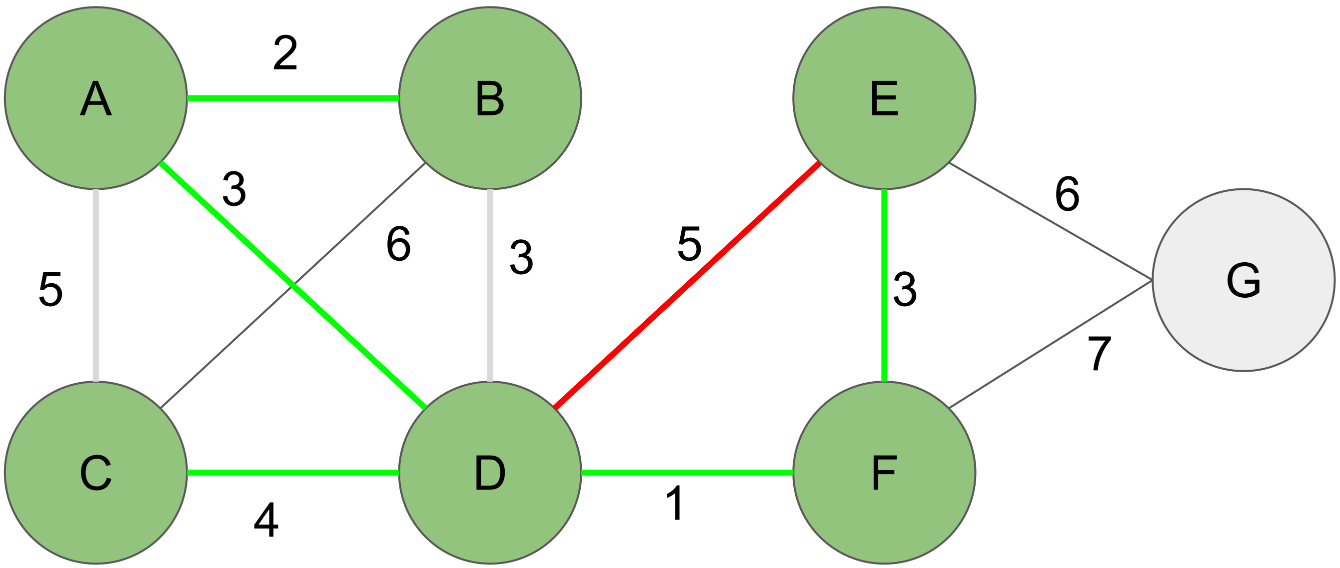
(C, D) can be added

Kruskal’s algorithm - weight 5



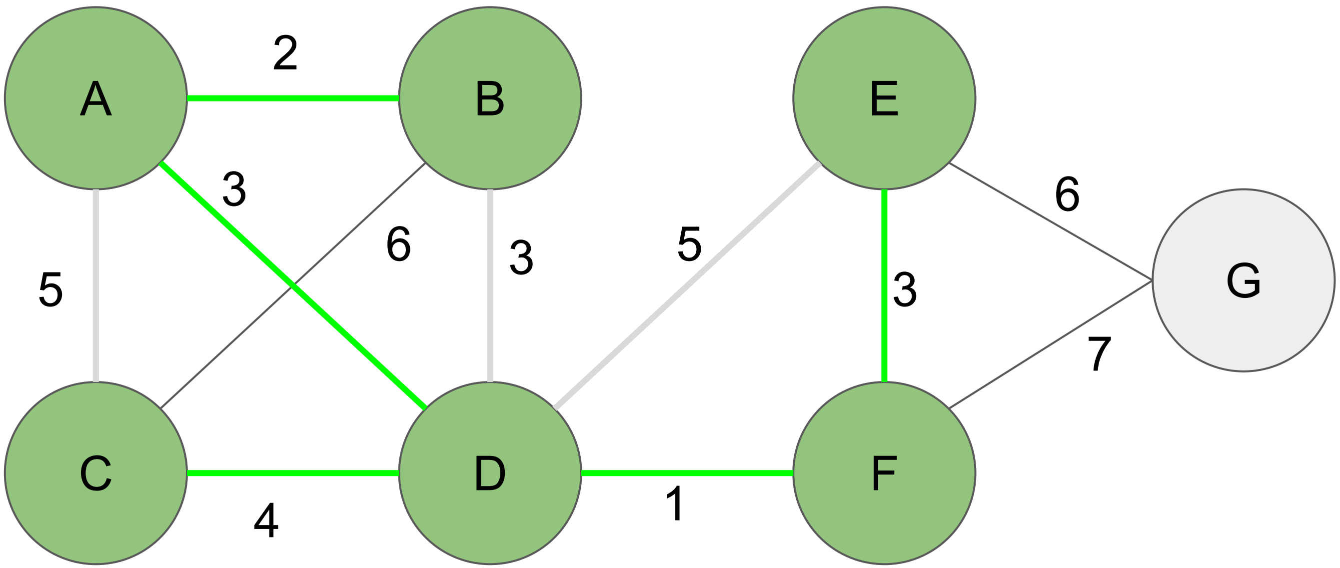
(A, C), (D, E)?

Kruskal’s algorithm - weight 5



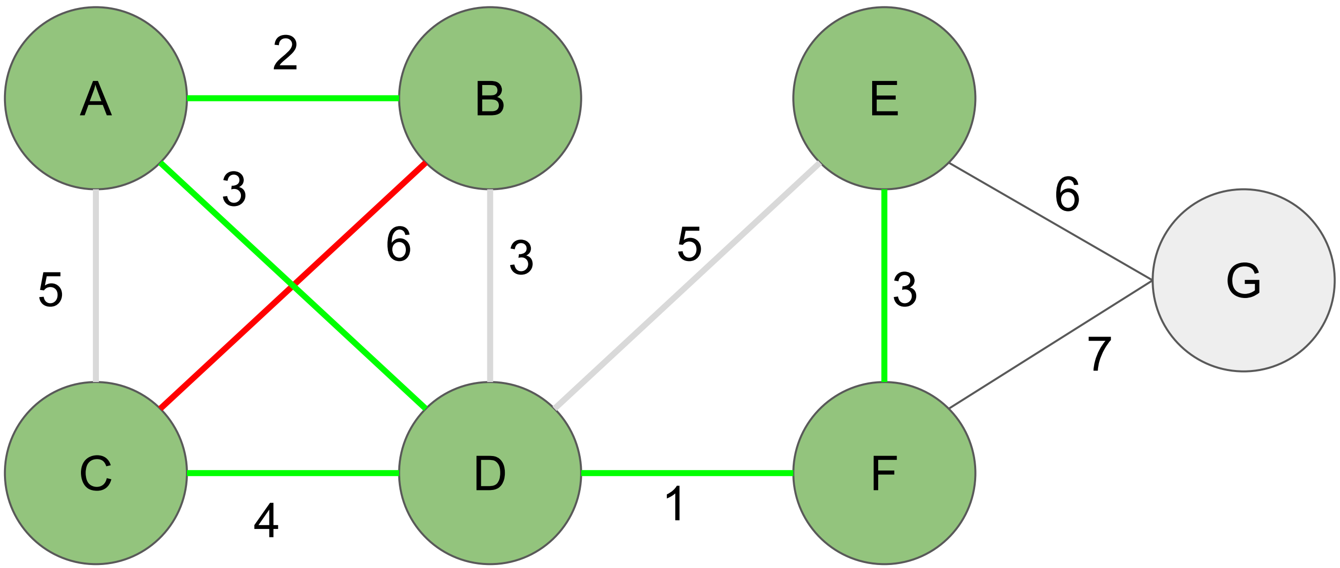
(A, C), (D, E)?

Kruskal’s algorithm - weight 5



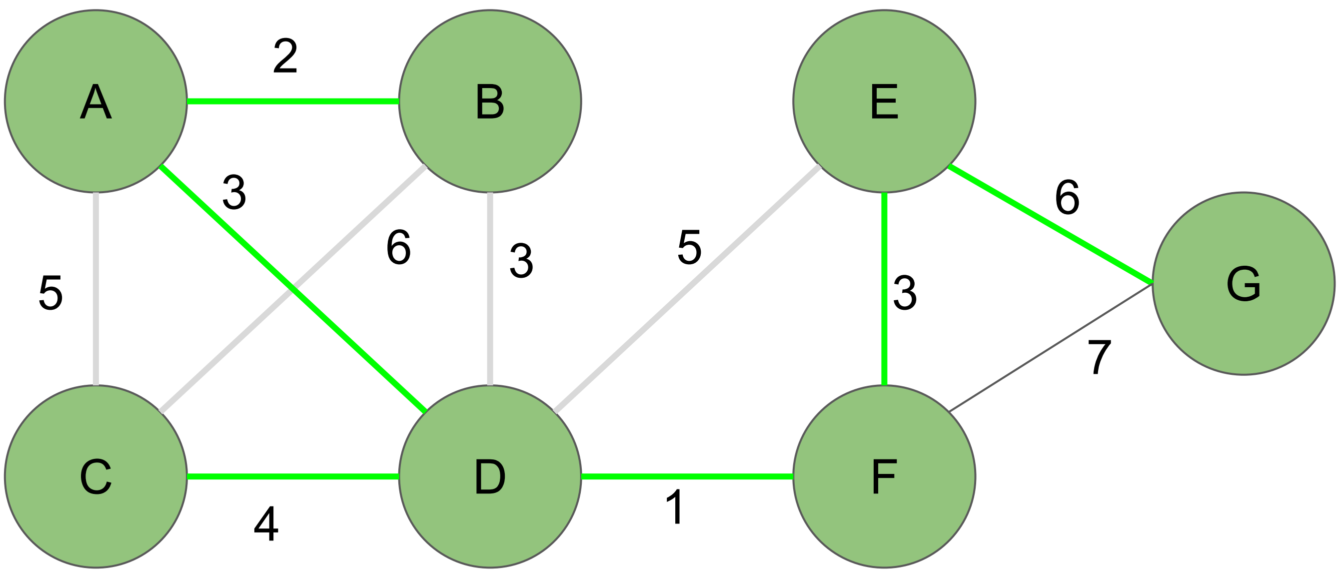
(A, C), (D, E)? Either creates a cycle

Kruskal’s algorithm - weight 6



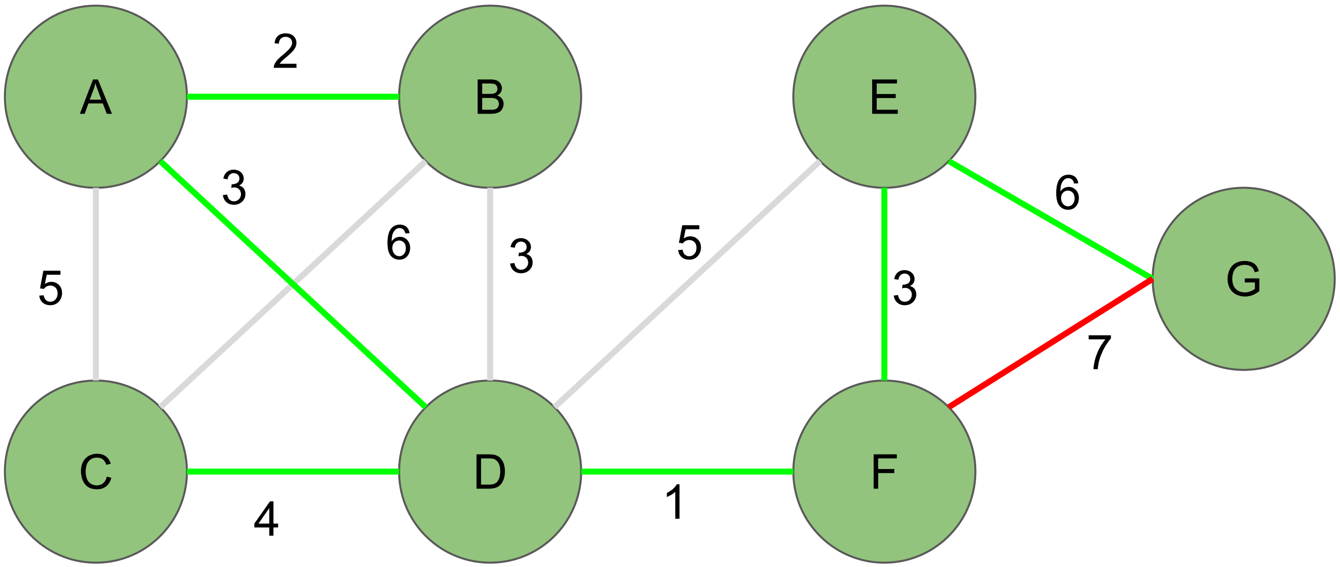
(B, C), (E, G)?

Kruskal's algorithm - weight 6

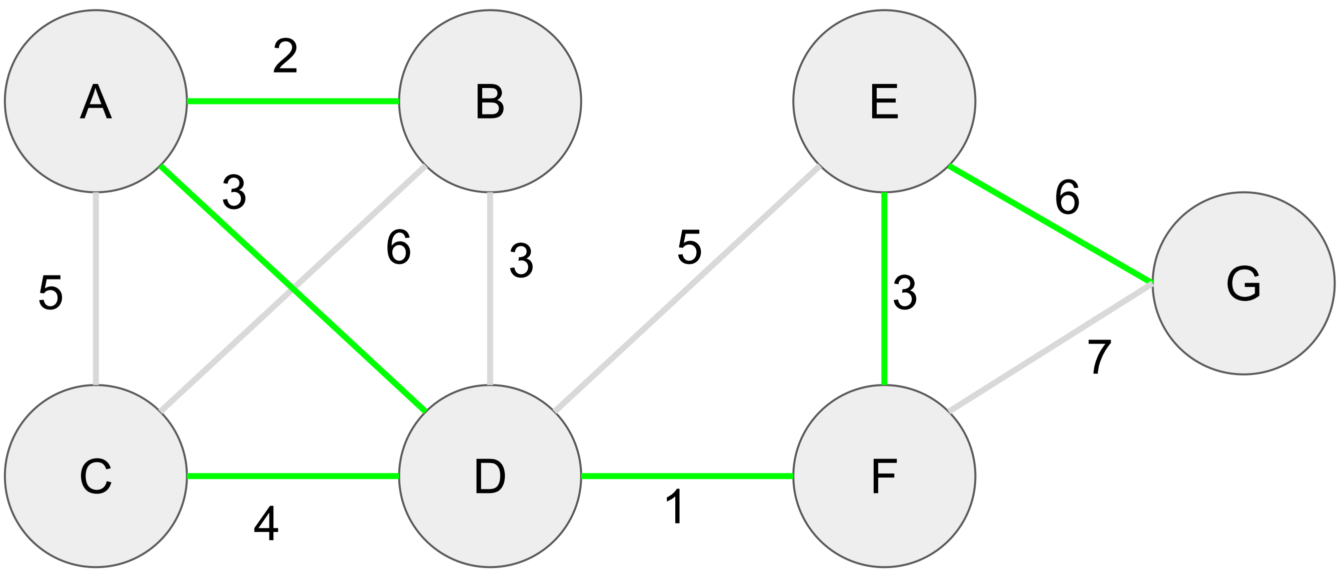


(B, C), (E, G)?
(B, C) creates a cycle
(E, G) can be added

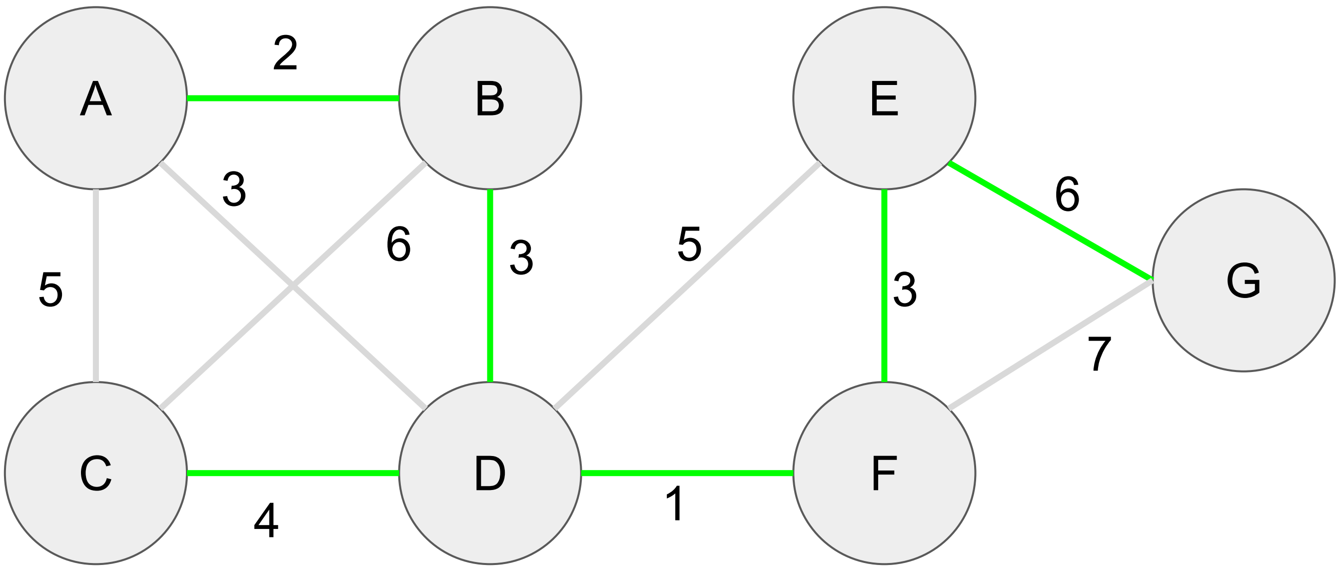
Kruskal’s algorithm - weight 7



Kruskal’s algorithm - weight 7

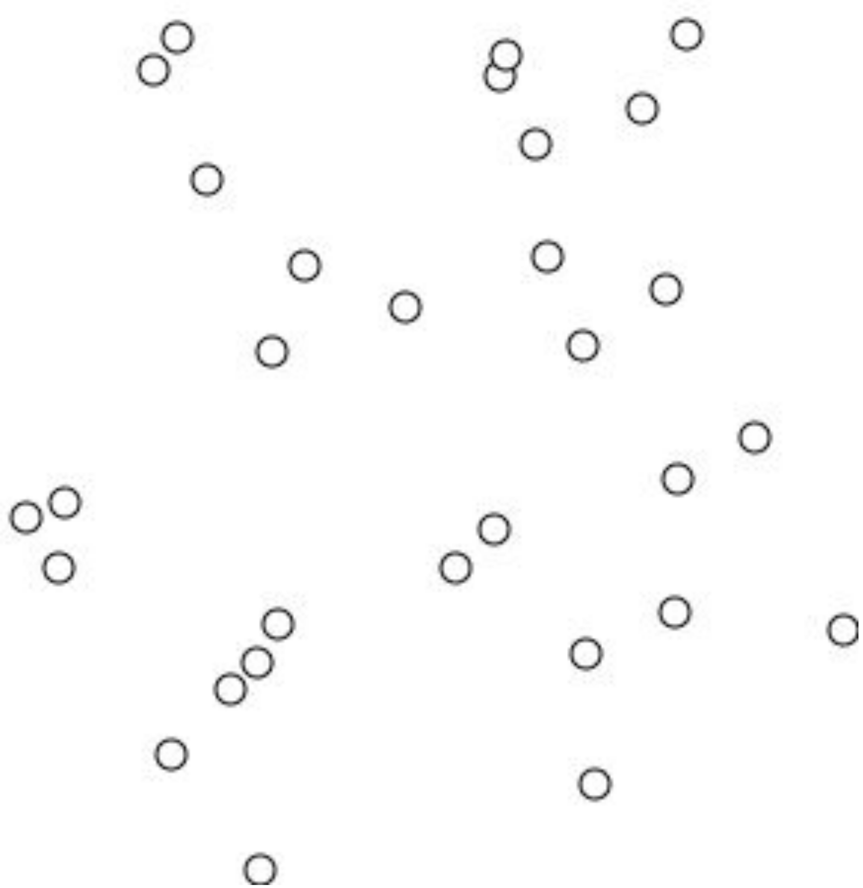


Kruskal's algorithm - it's possible for a graph to find have more than one MST



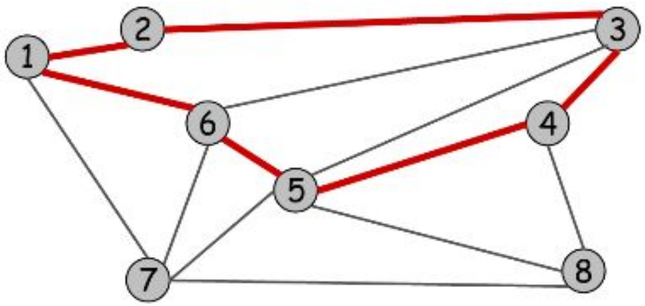
Kruskal’s algorithm - a bigger example

Kruskal's algorithm. Start with T = ϕ. Consider edges in ascending order of cost. Insert edge e in T unless doing so would create a cycle.



Kruskal’s algorithm - proof of correctness

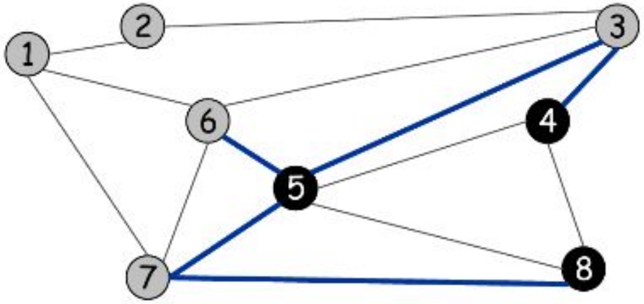
Cycle. Set of edges the form a-b, b-c, c-d, …, y-z, z-a.



Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1

Kruskal's algorithm - proof of correctness

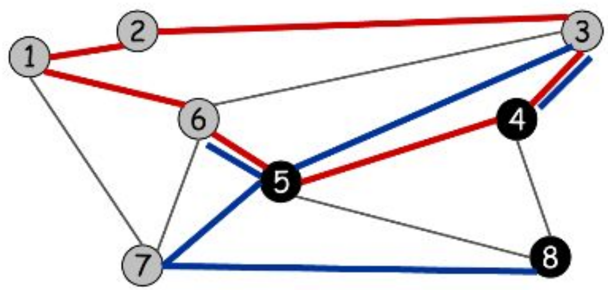
Cutset. A cut is a subset of nodes S. The corresponding cutset D is the subset of edges with exactly one endpoint in S.



Cut S = { 4, 5, 8 }
Cutset D = 5-6, 5-7, 3-4, 3-5, 7-8

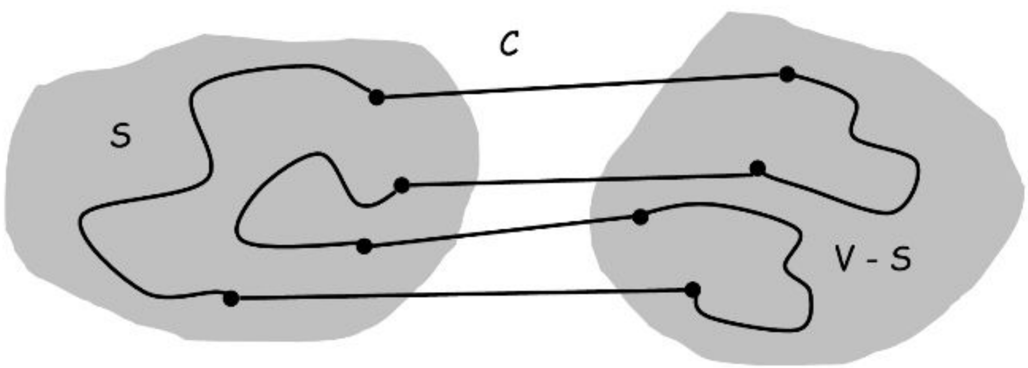
Kruskal's algorithm - proof of correctness

Claim. A cycle and a cutset intersect in an even number of edges.



Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1
Cutset D = 3-4, 3-5, 5-6, 5-7, 7-8
Intersection = 3-4, 5-6

Pf. (by picture)



A cycle has to enter and leave the cut the same amount of times