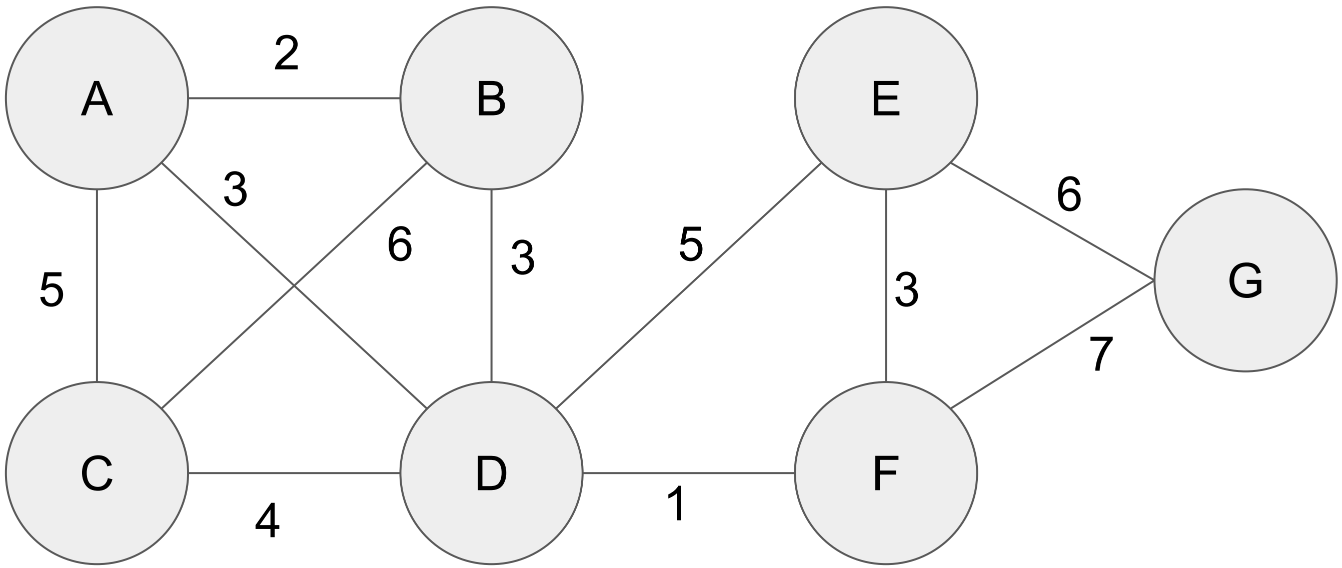
Greedy algorithms

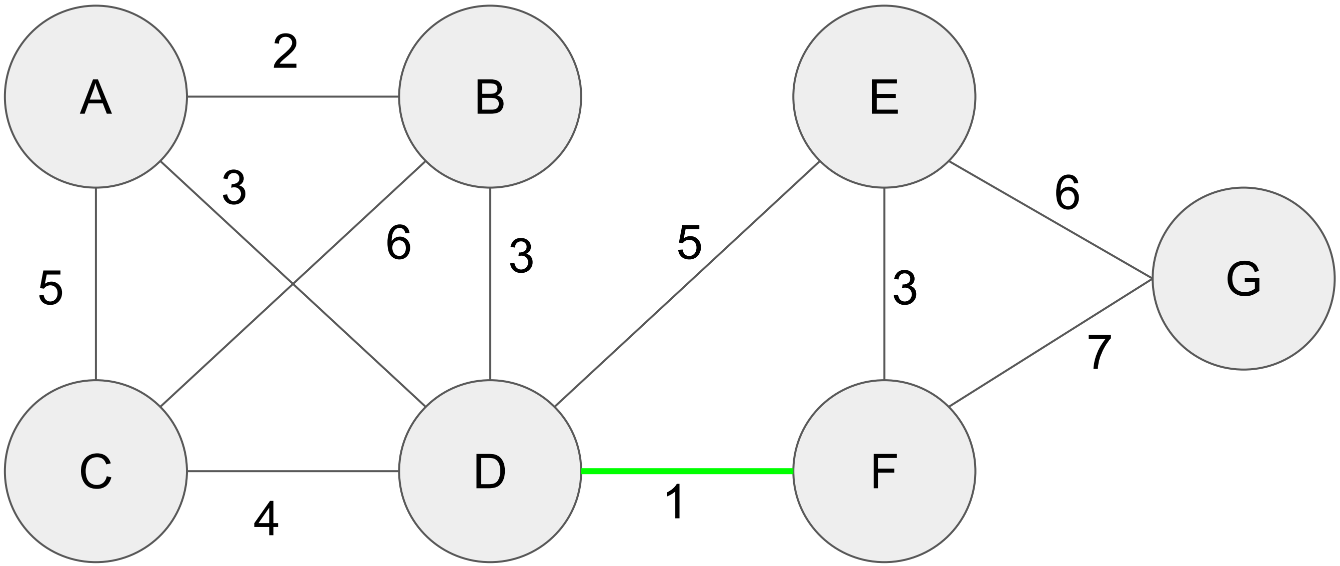
CMPSC 465 - Yana Safonova

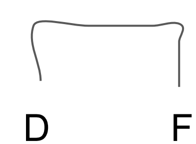
MST and clustering

Kruskal's algorithm aka single-linkage clustering

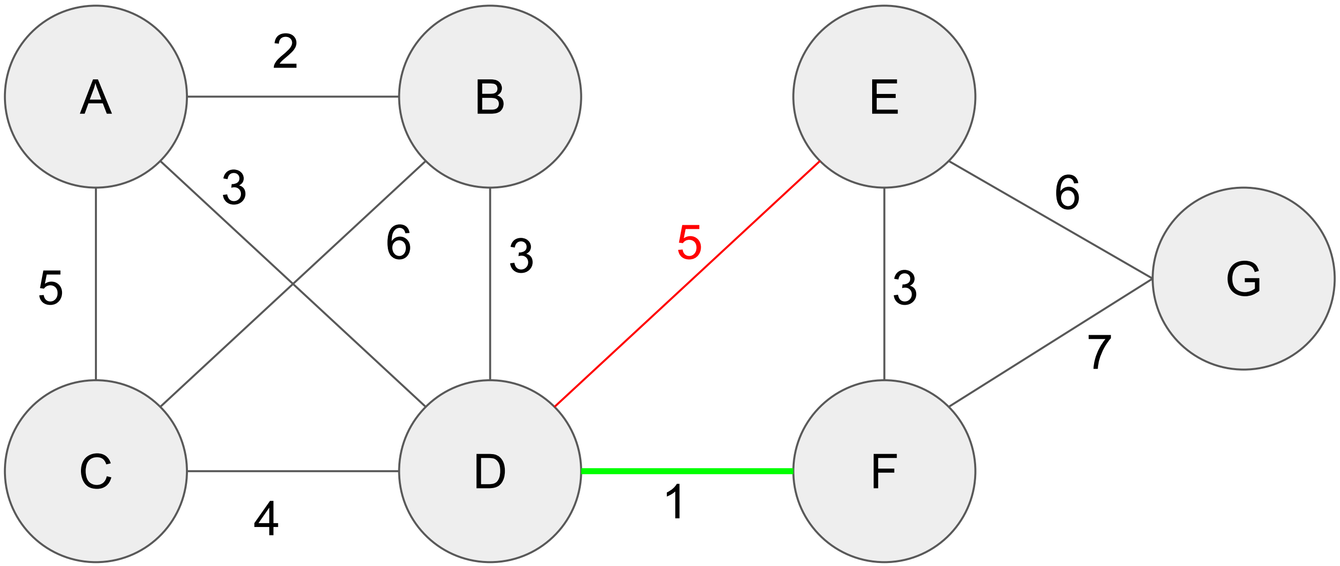


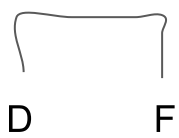
Kruskal’s algorithm - weight 1



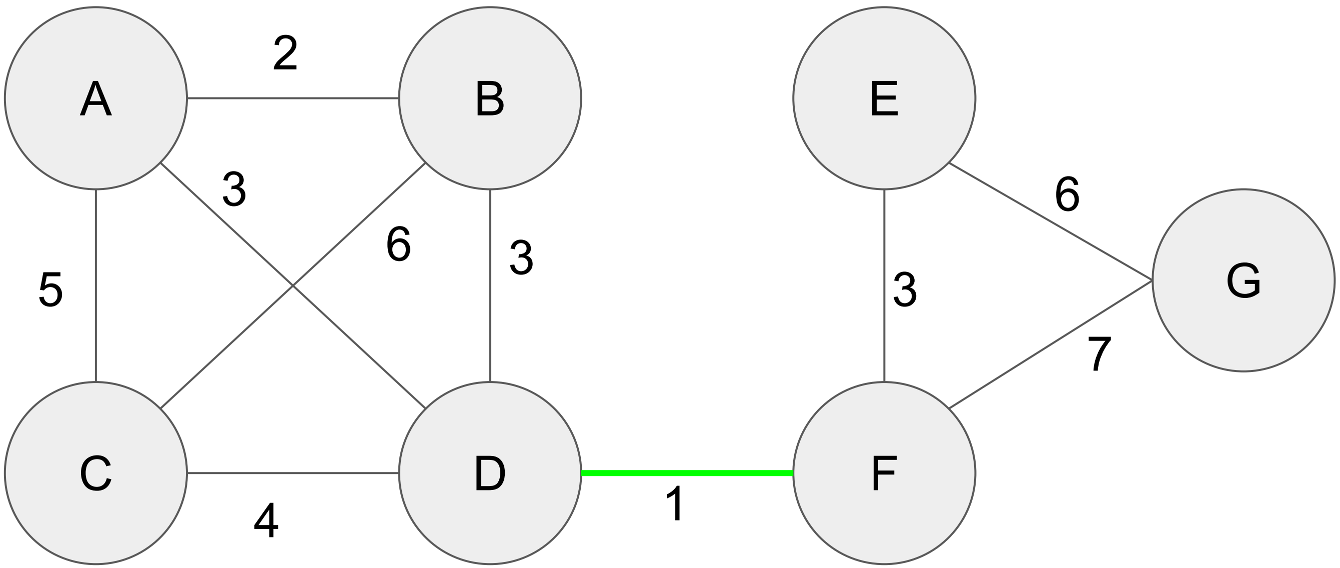


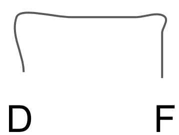
Kruskal’s algorithm - weight 1



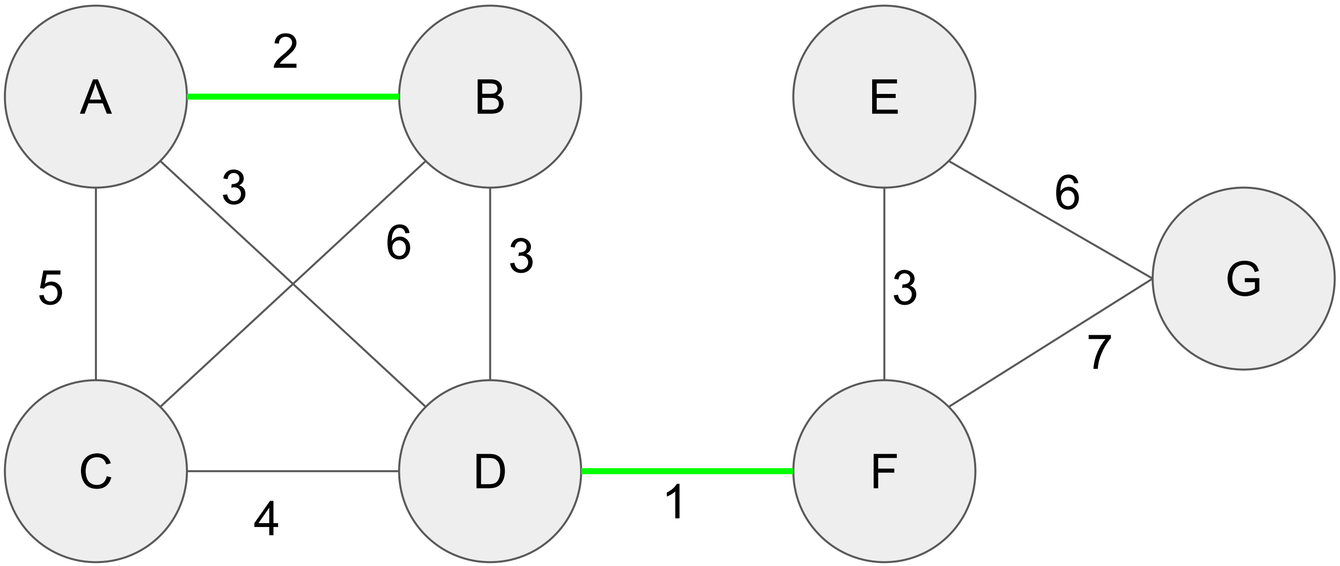


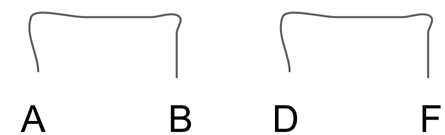
Kruskal’s algorithm - weight 1



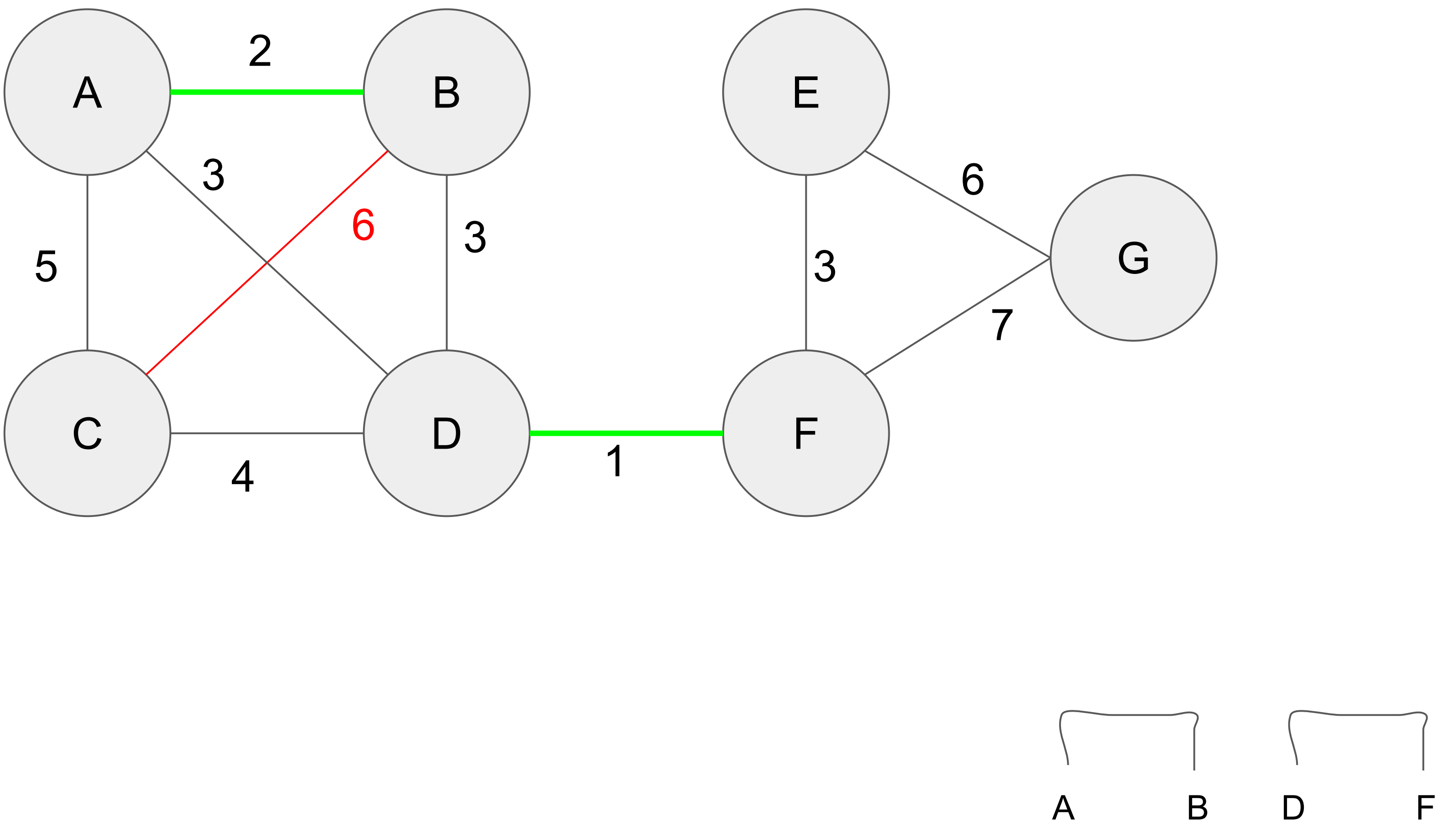


Kruskal’s algorithm - weight 2

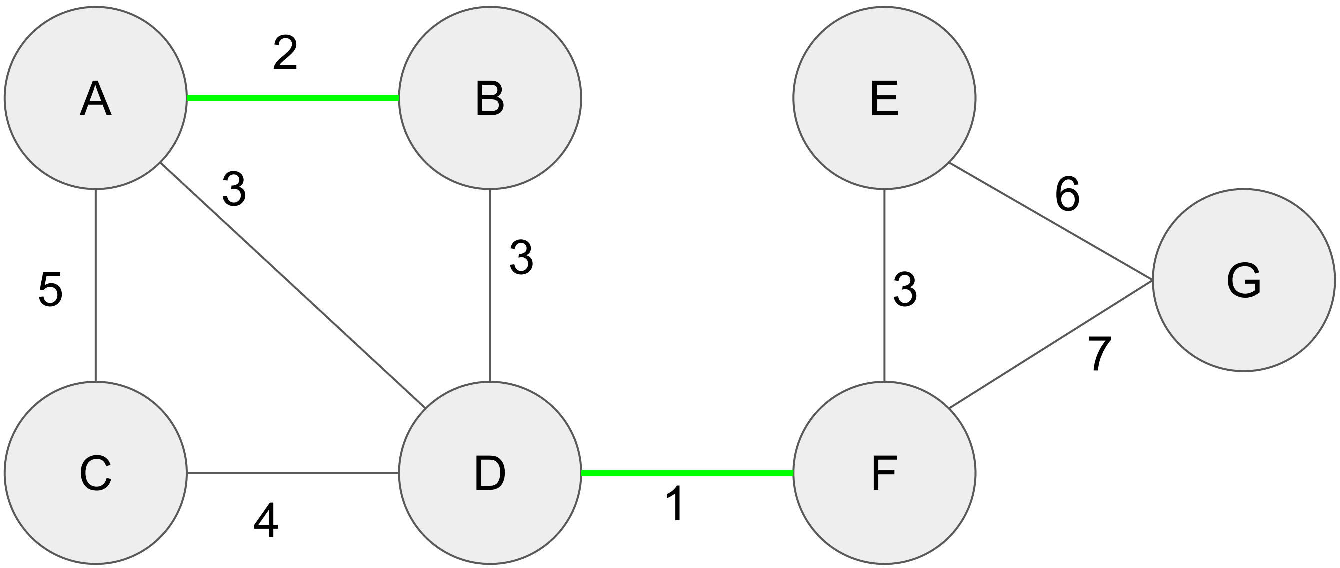


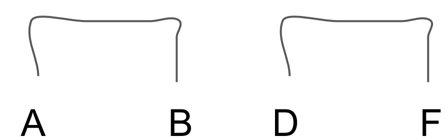


Kruskal's algorithm - weight 2

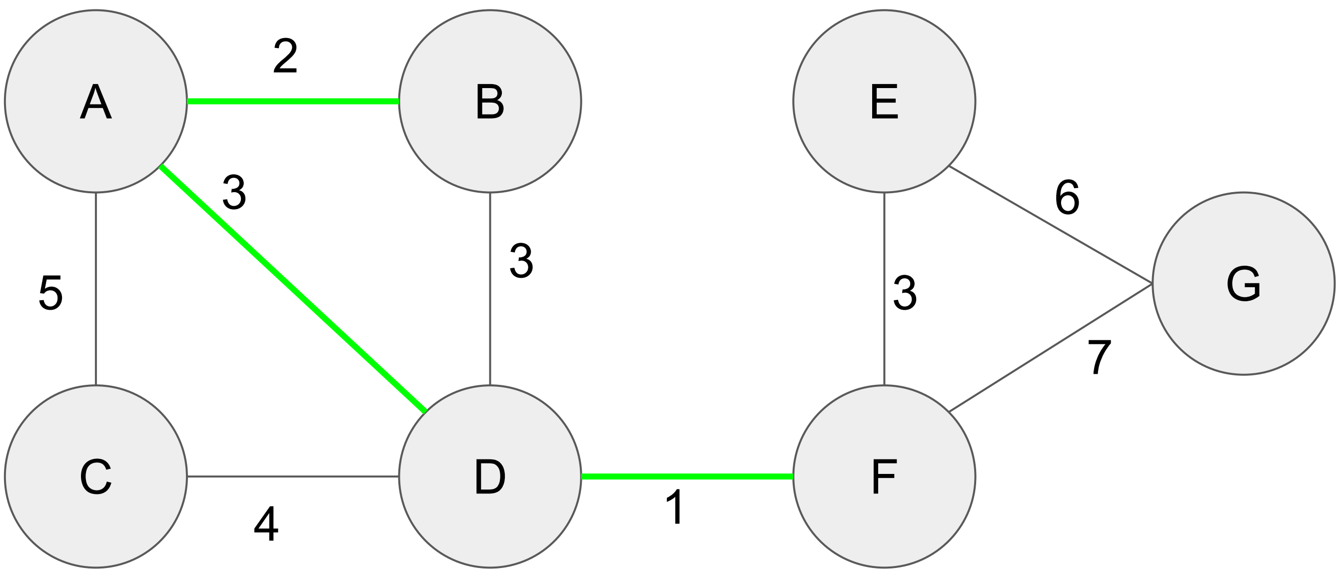


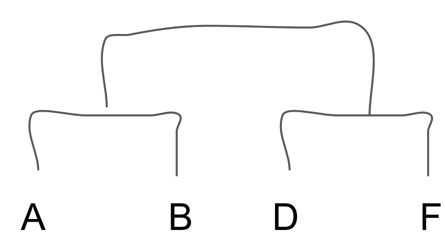
Kruskal's algorithm - weight 2



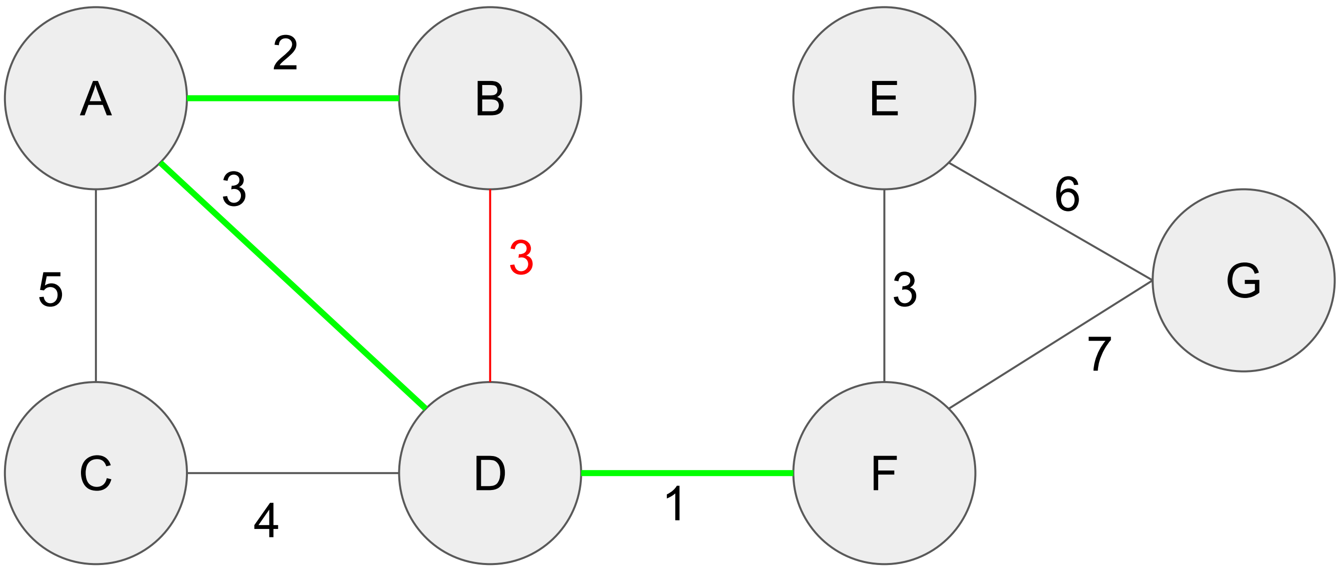


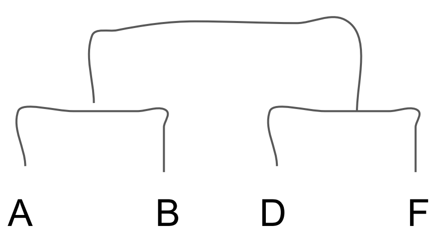
Kruskal’s algorithm - weight 3



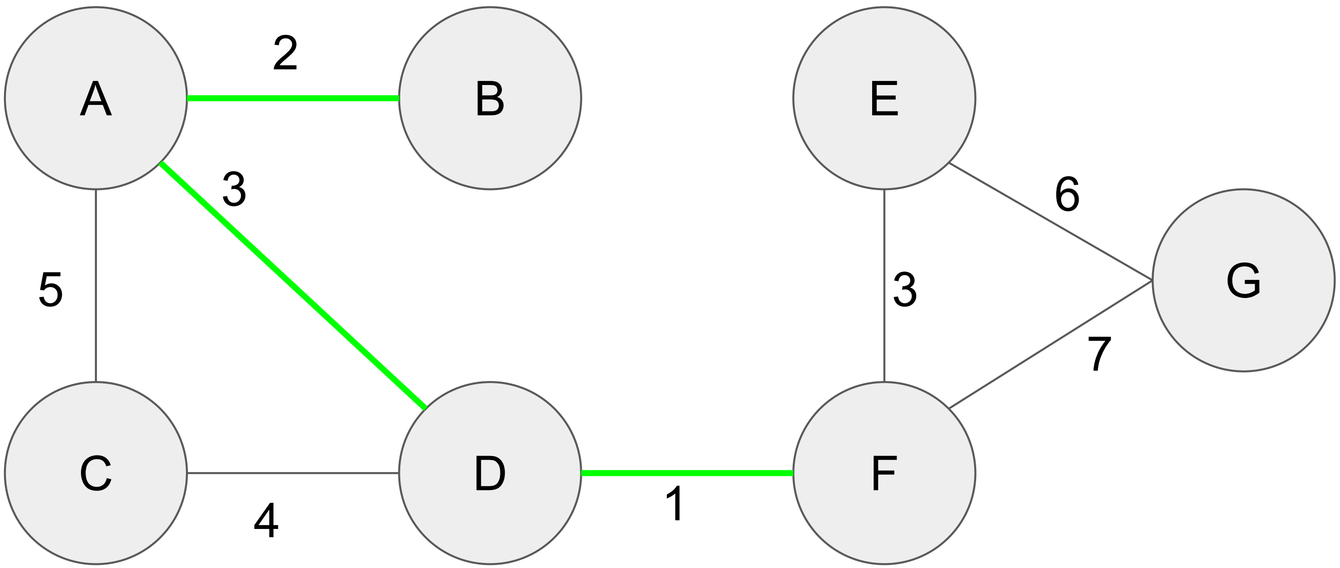


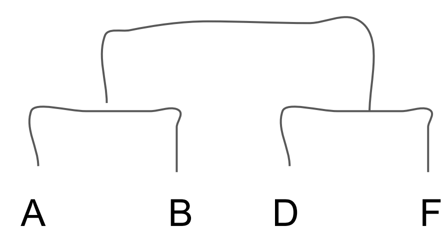
Kruskal’s algorithm - weight 3



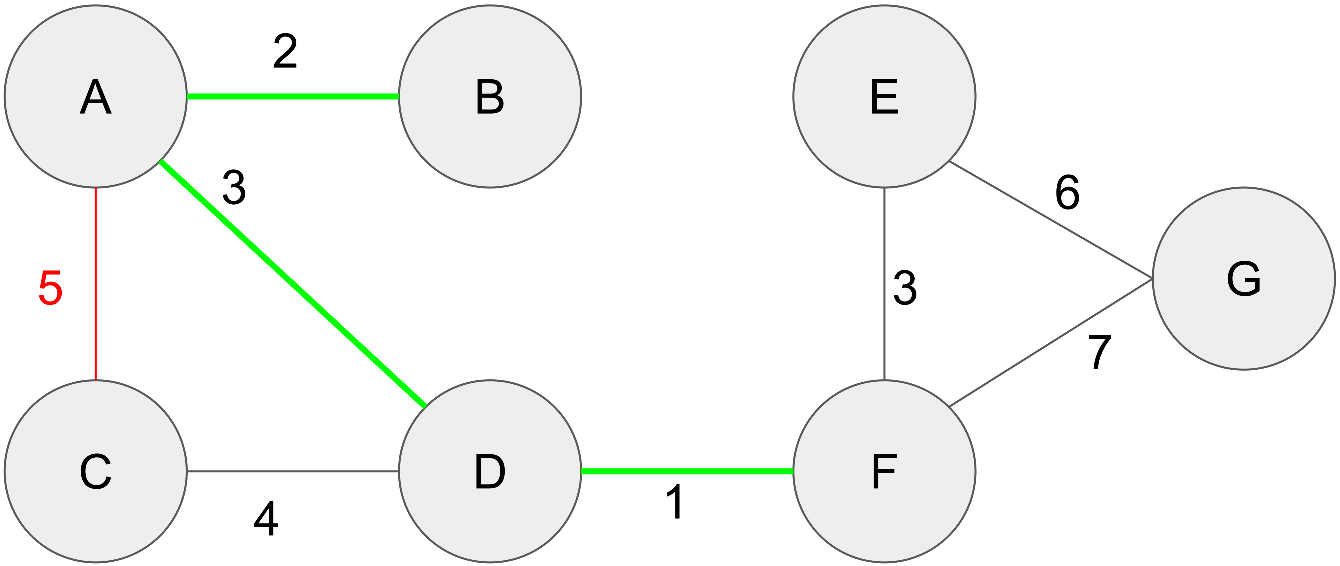


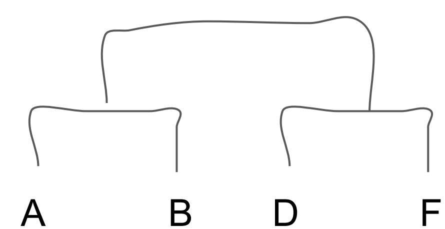
Kruskal’s algorithm - weight 3



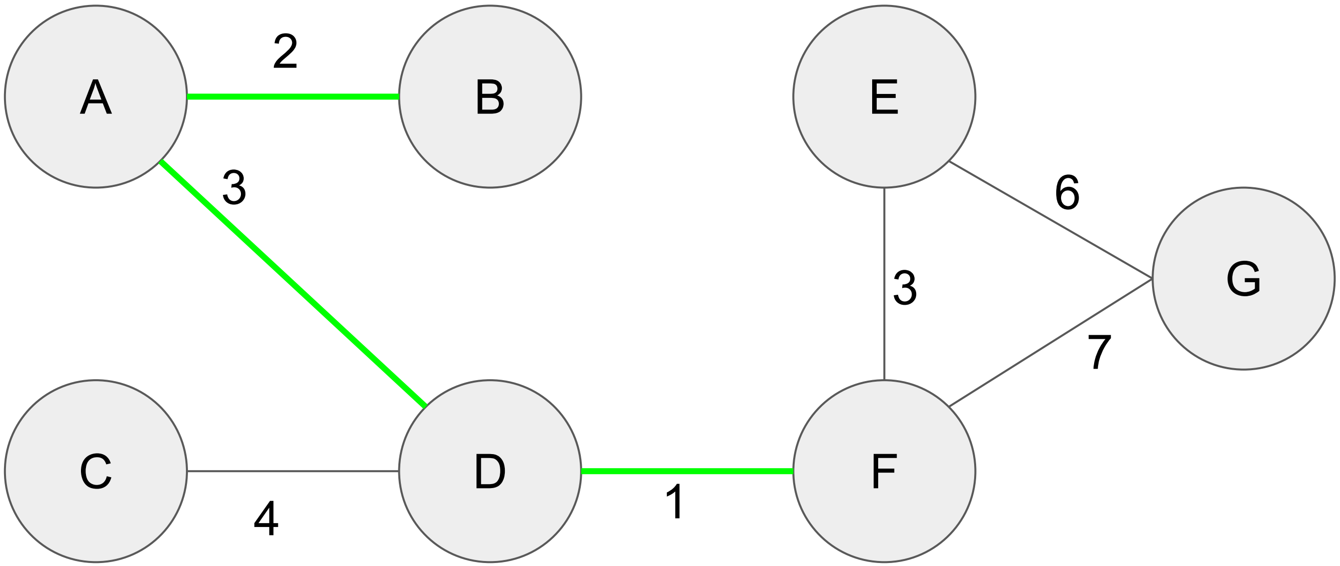


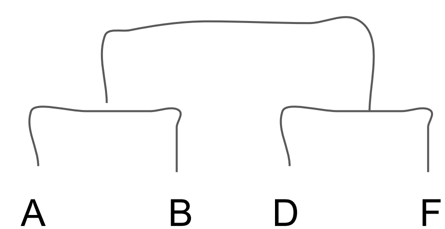
Kruskal’s algorithm - weight 3



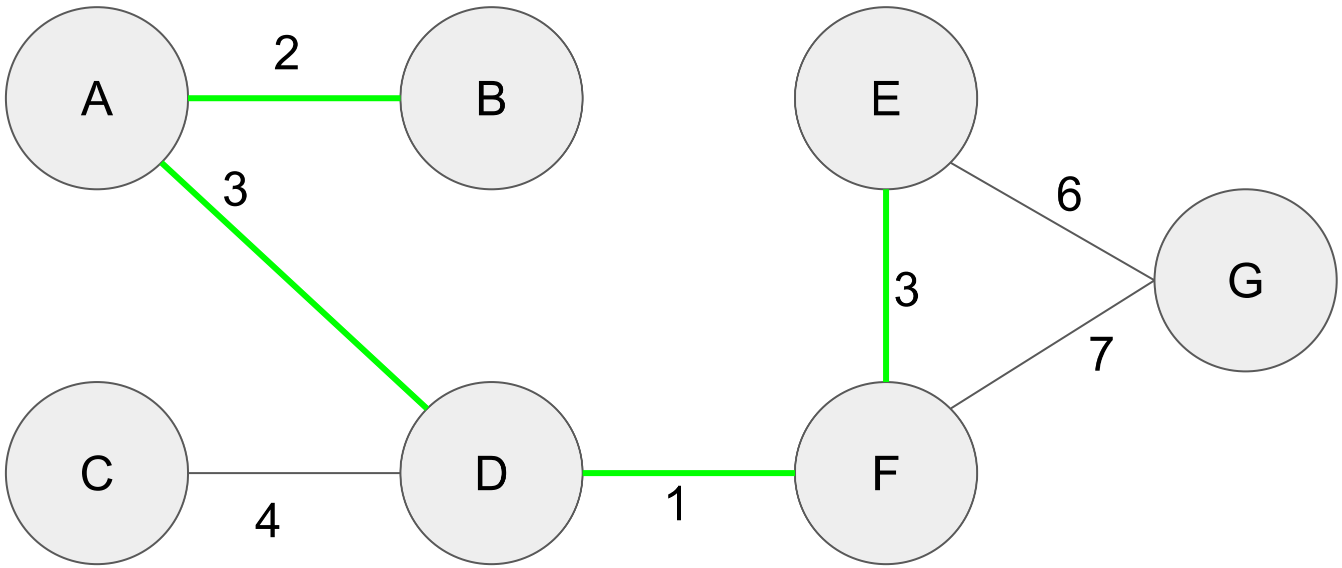


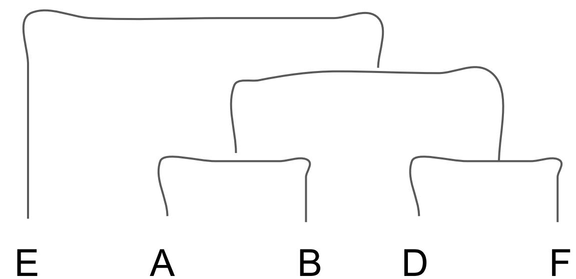
Kruskal’s algorithm - weight 3



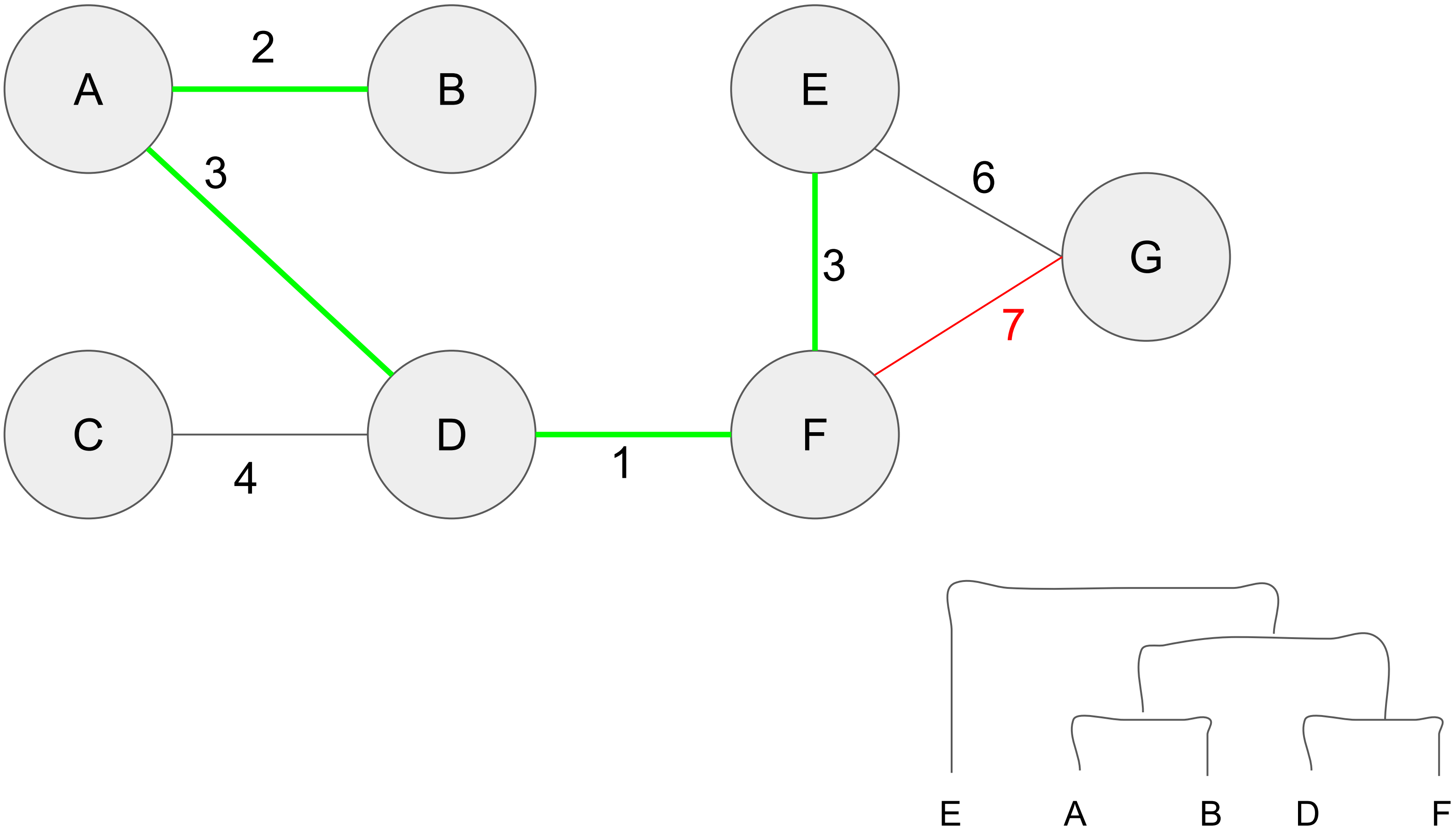


Kruskal’s algorithm - weight 3

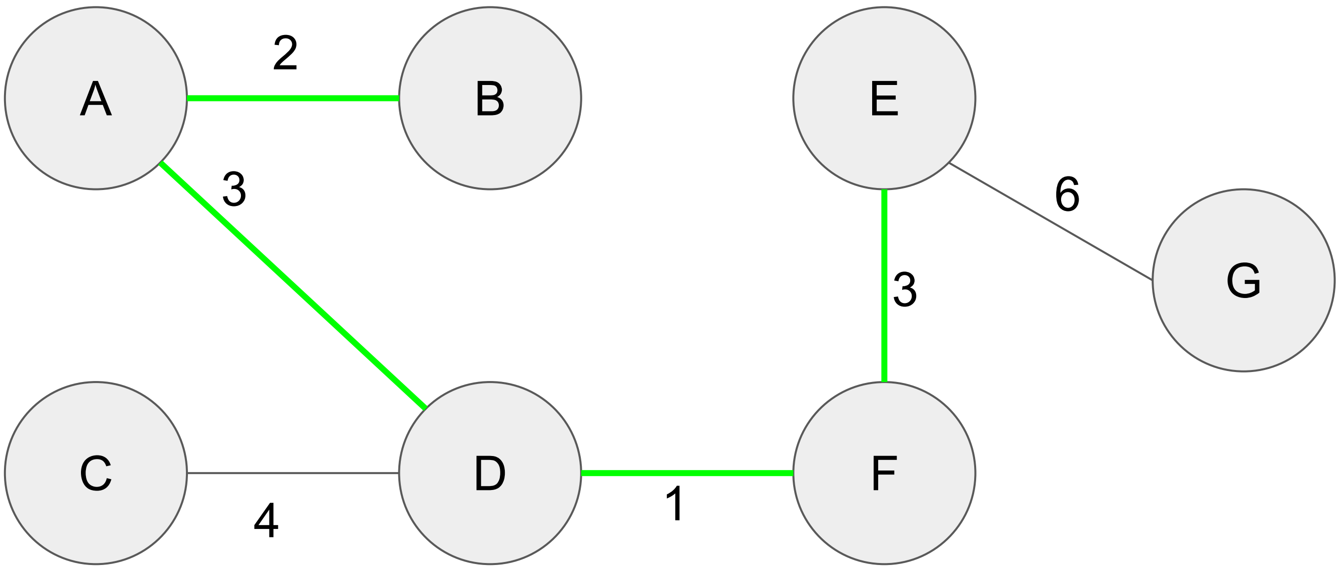


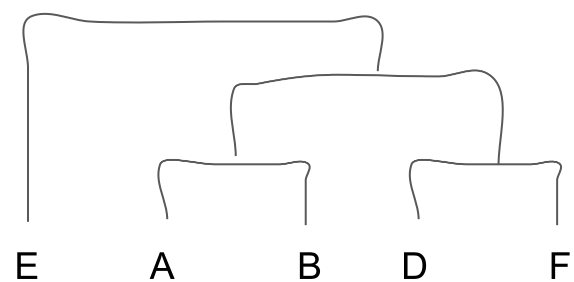


Kruskal’s algorithm - weight 3

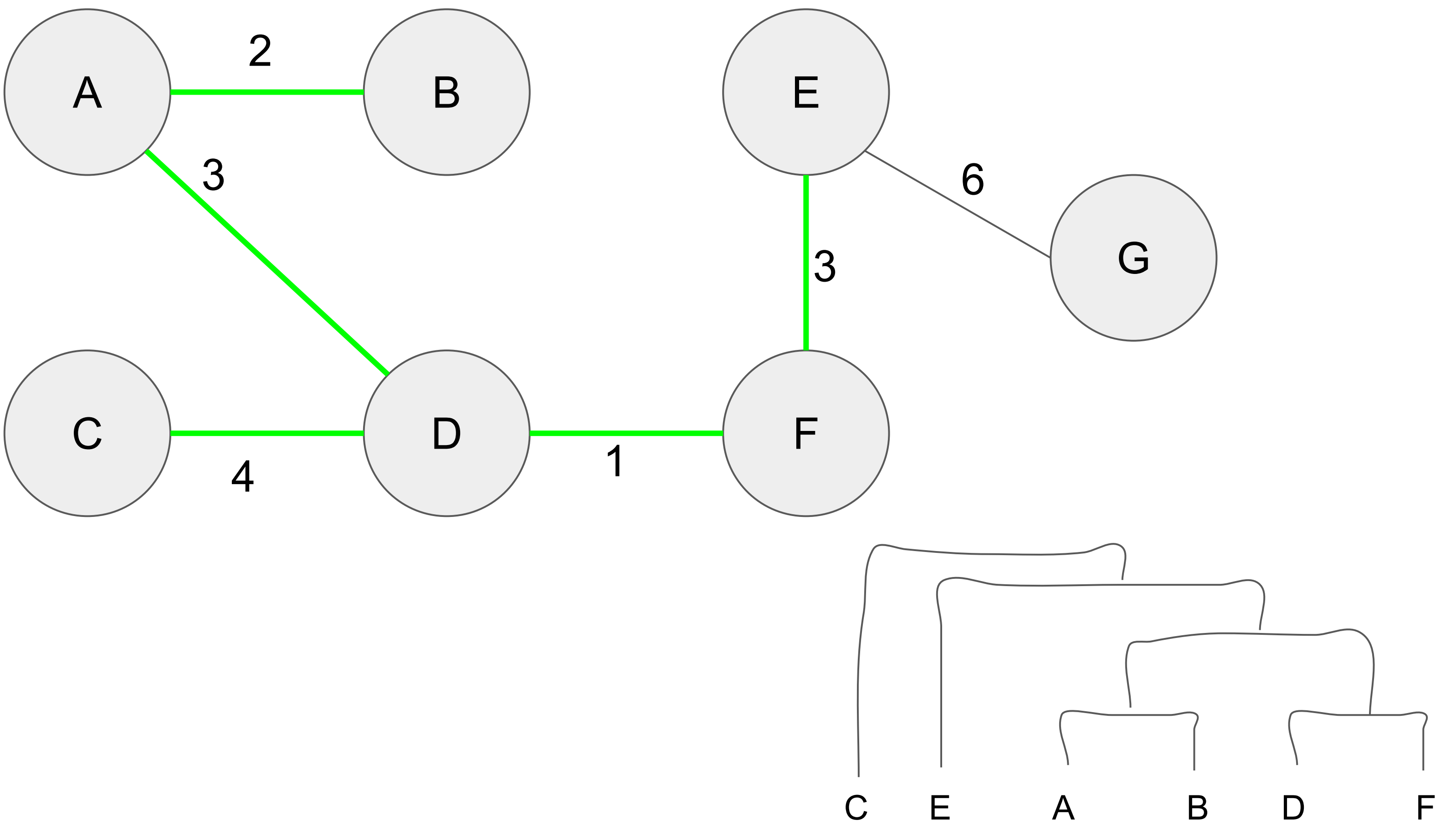


Kruskal's algorithm - weight 3

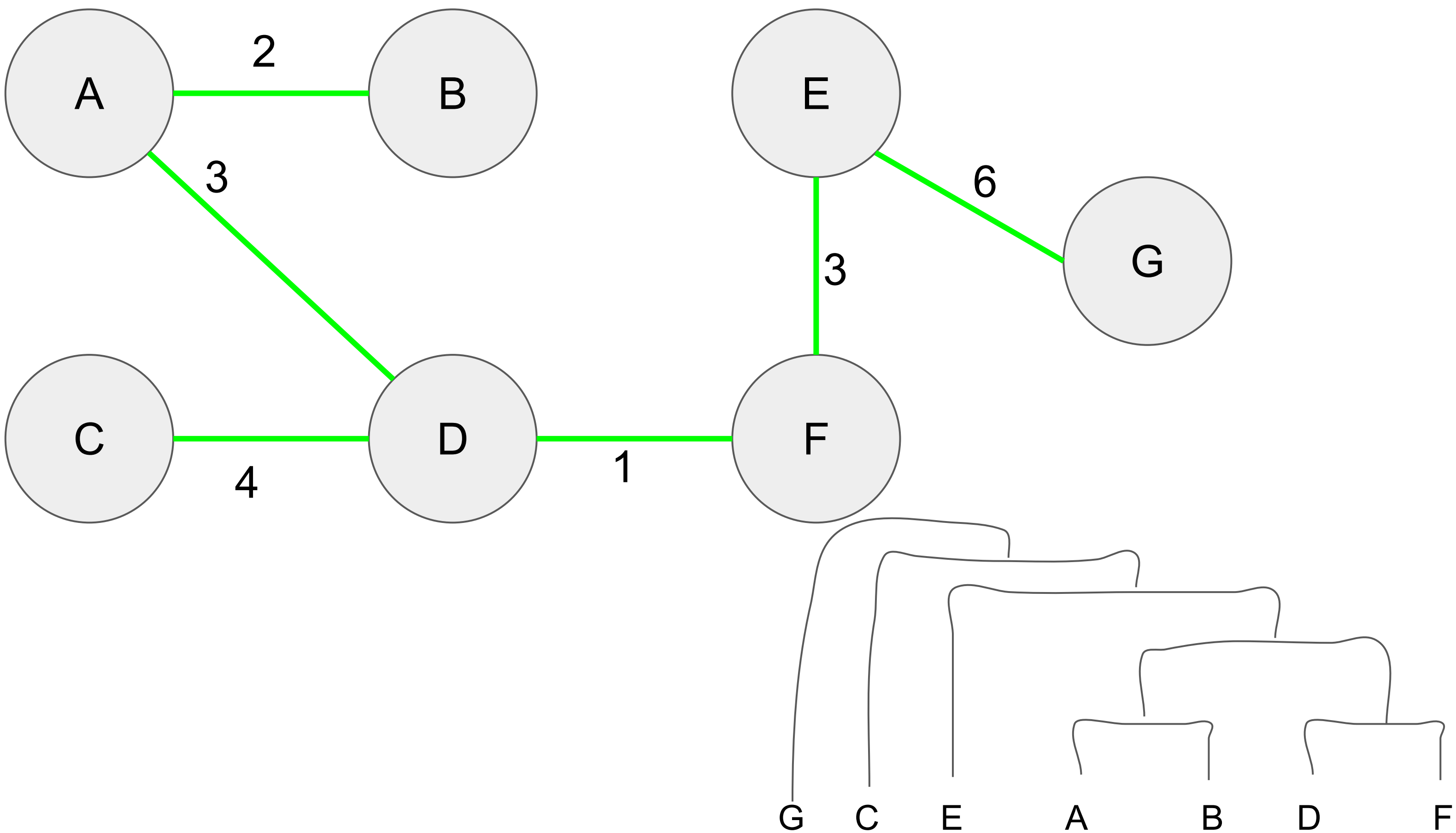




Kruskal's algorithm - weight 4



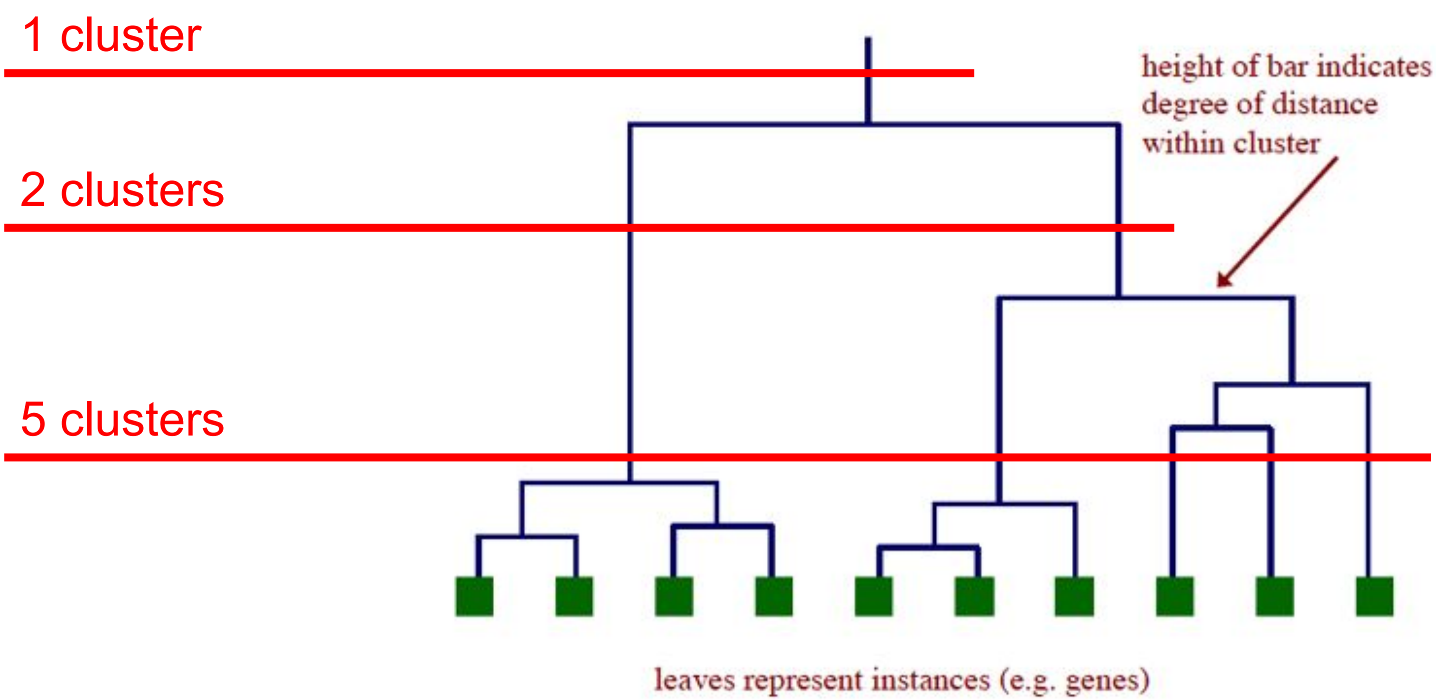
Kruskal's algorithm - weight 6



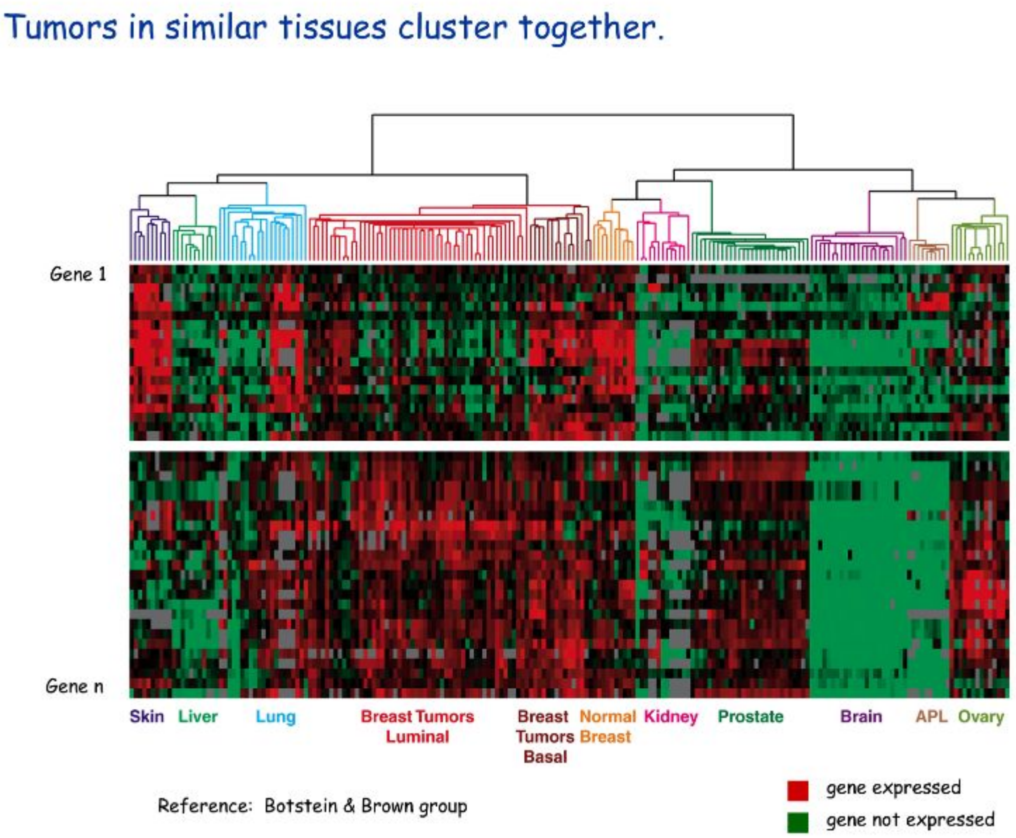
Dendrogram

Dendrogram. Scientific visualization of hypothetical sequence of evolutionary events.

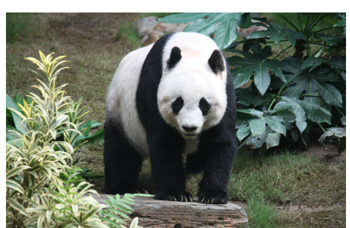
- Leaves = genes.
- Internal nodes = hypothetical ancestors.



Applications in data analysis



Applications: taxonomy



Morphological features

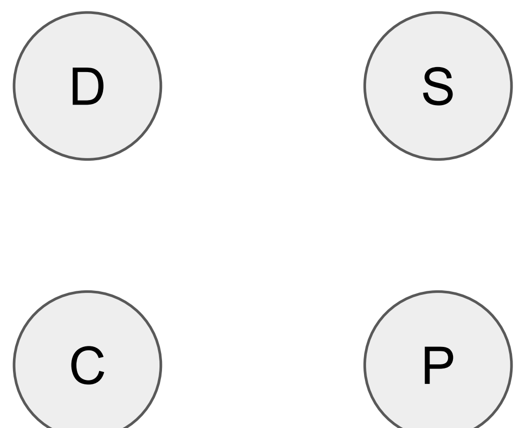
Dolphin (D): aquatic, eats fish

Seal (S): semi-aquatic, eats fish

Cow (C): terrestrial, eats grass

Panda (P): terrestrial, eats grass

Morphological MST:



Applications: taxonomy

Morphological features

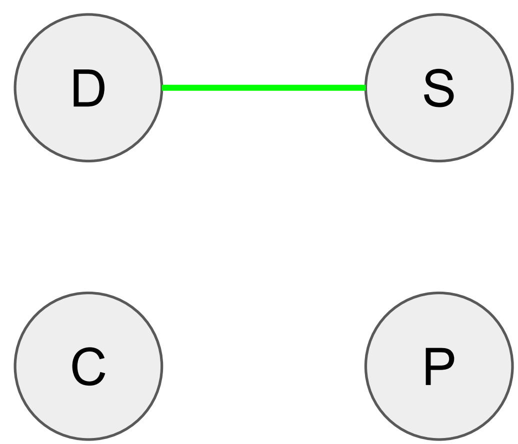
Dolphin (D): aquatic, eats fish

Seal (S): semi-aquatic, eats fish

Cow (C): terrestrial, eats grass

Panda (P): terrestrial, eats grass

Morphological MST:



Applications: taxonomy

Morphological features

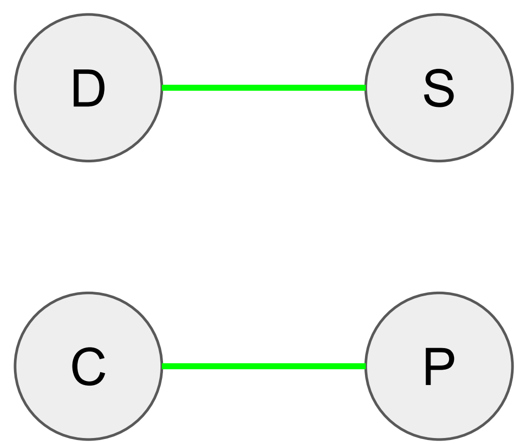
Dolphin (D): aquatic, eats fish

Seal (S): semi-aquatic, eats fish

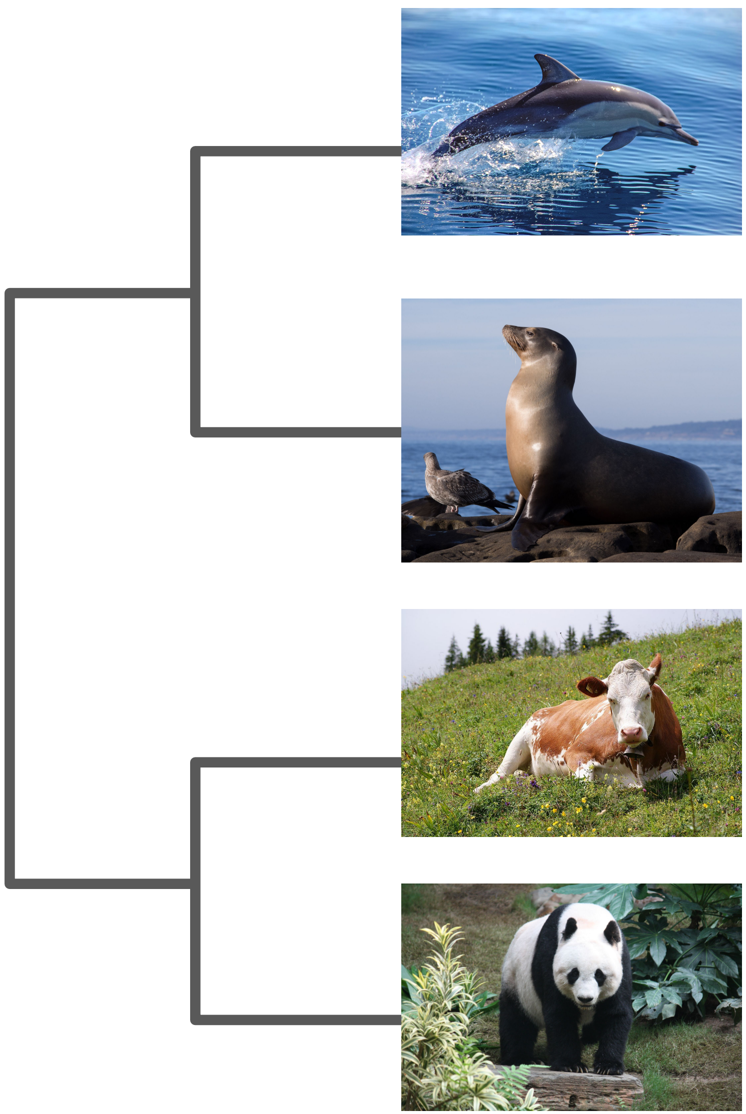
Cow (C): terrestrial, eats grass

Panda (P): terrestrial, eats grass

Morphological MST:



Applications: taxonomy



Morphological features

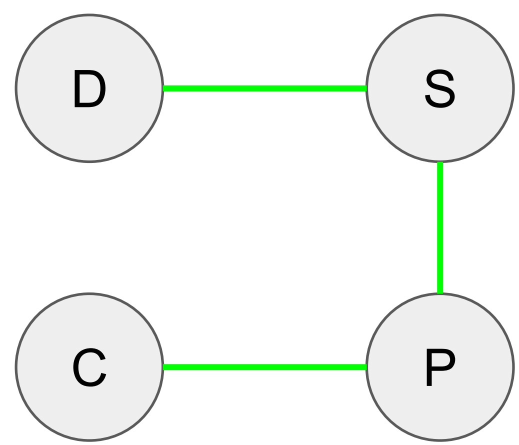
Dolphin (D): aquatic, eats fish

Seal (S): semi-aquatic, eats fish

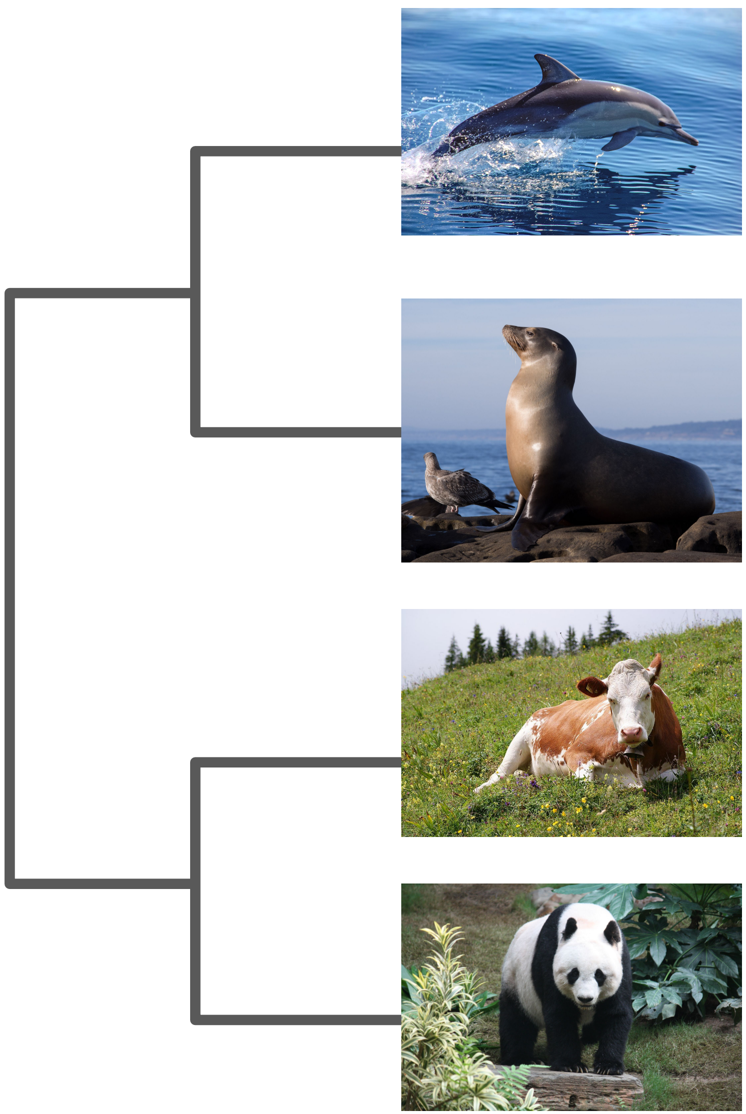
Cow (C): terrestrial, eats grass

Panda (P): terrestrial, eats grass

Morphological MST:



Applications: taxonomy



Morphological features

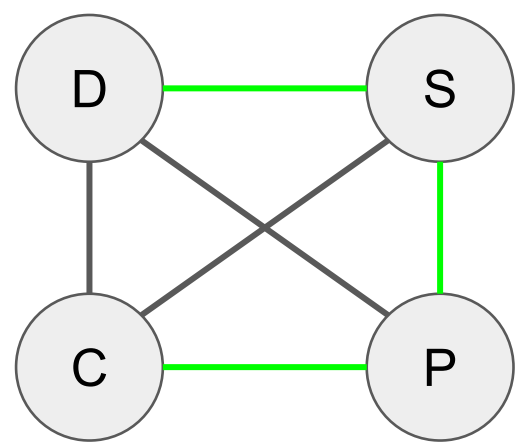
Dolphin (D): aquatic, eats fish

Seal (S): semi-aquatic, eats fish

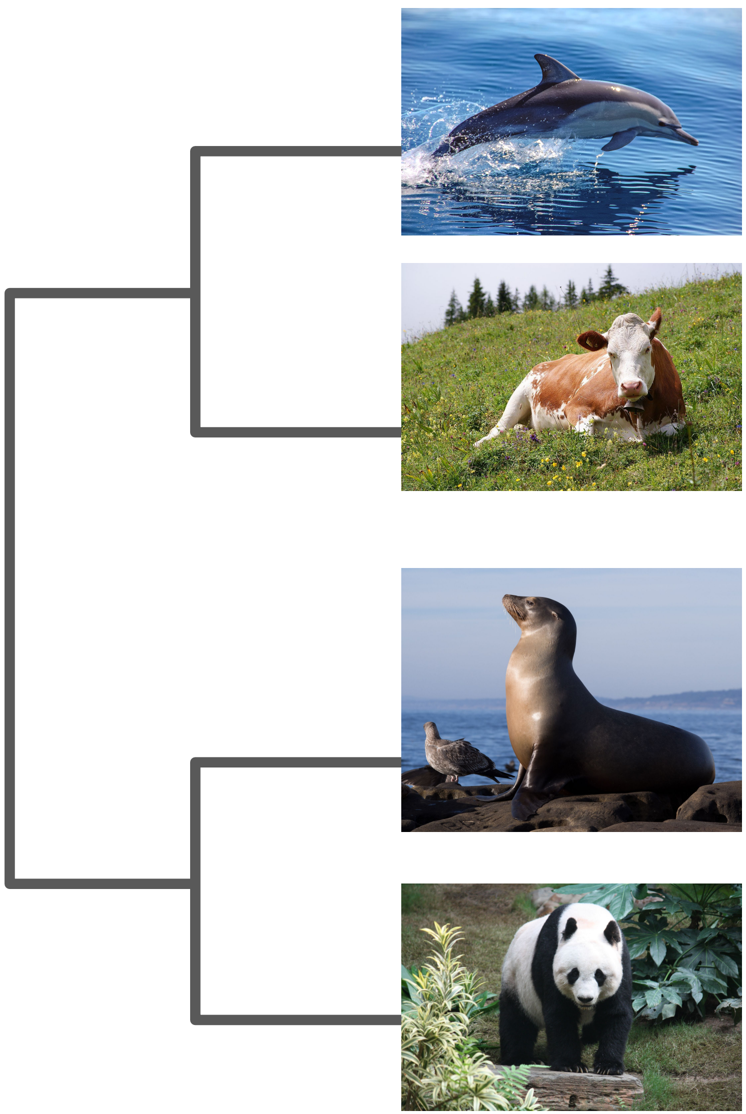
Cow (C): terrestrial, eats grass

Panda (P): terrestrial, eats grass

Morphological MST:



Applications: taxonomy



Morphological features

Dolphin (D): aquatic, eats fish

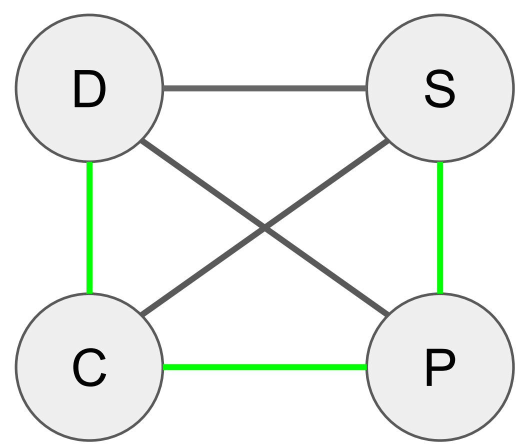
Seal (S): semi-aquatic, eats fish

Cow (C): terrestrial, eats grass

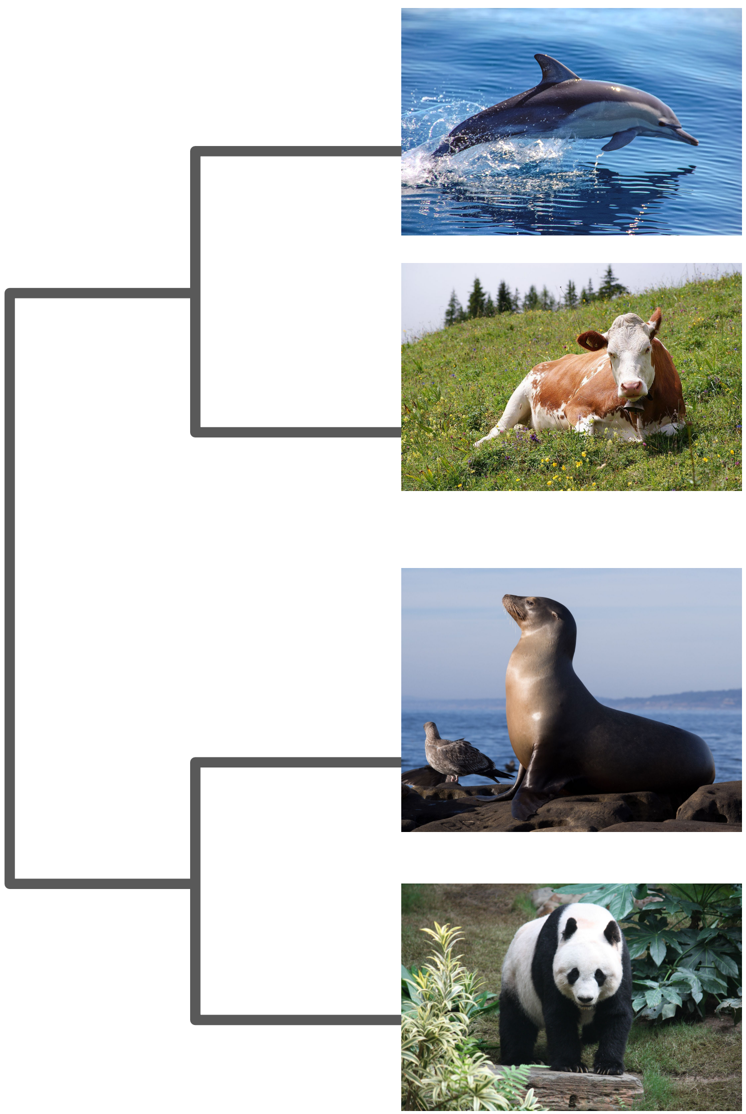
Panda (P): terrestrial, eats grass

Genetic MST

(the actual evolutionary path):



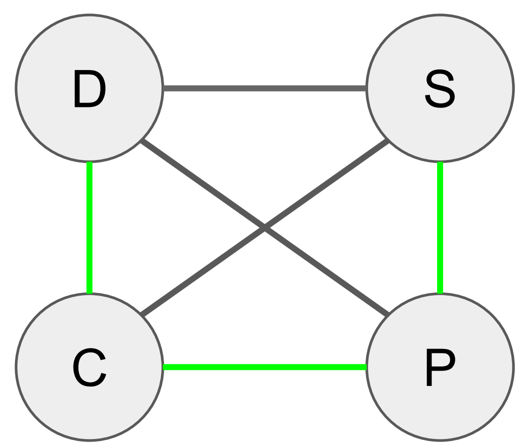
Applications: taxonomy



Computing genetic distances will be discussed in the dynamic programming section

Genetic MST

(the actual evolutionary path):



Huffman coding

Huffman encoding

An encoding scheme used in, e.g., MP3 encoding

Data: a string S of symbols over an alphabet Γ
Goal: find a binary encoding e of Γ resulting in minimum encoded length of S

Denote the encoded string by S_e

Example: ASCII encoding
$$\begin{array}{ll} a & 01100001 \\ b & 01100010 \\ \vdots & \vdots \end{array}$$

Different encodings

Consider $\Gamma = \{a, b, c\}$

Stats on S: a appears 45 times, b 16 times, and c twice

Different encodings

Consider $\Gamma = \{a, b, c\}$

Stats on S: a appears 45 times, b 16 times, and c twice

- Fixed-length encoding

$$e_1 : \begin{array}{l} a \to 00 \\ b \to 01 \\ c \to 10 \end{array} \qquad |S_{e_1}| = 45 \times 2 + 16 \times 2 + 2 \times 2 = 126$$

Different encodings

Consider $\Gamma = \{a, b, c\}$

Stats on S: a appears 45 times, b 16 times, and c twice

- Fixed-length encoding

$$e_1 : \begin{array}{l} a \to 00 \\ b \to 01 \\ c \to 10 \end{array} \qquad |S_{e_1}| = 45 \times 2 + 16 \times 2 + 2 \times 2 = 126$$

- Variable-length encoding

$$e_2 : \begin{array}{l} a \to 0 \\ b \to 10 \\ c \to 11 \end{array} \qquad |S_{e_2}| = 45 \times 1 + 16 \times 2 + 2 \times 2 = 81$$

Different encodings

Consider $\Gamma = \{a, b, c\}$

Stats on S: a appears 45 times, b 16 times, and c twice

- Fixed-length encoding

$$e_1 : \begin{array}{l} a \to 00 \\ b \to 01 \\ c \to 10 \end{array} \qquad |S_{e_1}| = 45 \times 2 + 16 \times 2 + 2 \times 2 = 126$$

- Variable-length encoding

$$e_2 : \begin{array}{l} a \to 0 \\ b \to 10 \\ c \to 11 \end{array} \qquad |S_{e_2}| = 45 \times 1 + 16 \times 2 + 2 \times 2 = 81$$

- Be careful! $e_2 : \begin{array}{l} a \to 0 \\ b \to 1 \\ c \to 01 \end{array}$ Decoding will lead to ambiguity

Prefix-free encoding

Consider the bad encoding $e_2 : \begin{array}{l} a \rightarrow 0 \\ b \rightarrow 1 \\ c \rightarrow 01 \end{array}$ How to decode 010110?

ababba?, *ccba*?, *abcba*?, or ...?

Prefix-free encoding

Consider the bad encoding e_2 : $\begin{array}{l} a \to 0 \\ b \to 1 \\ c \to 01 \end{array}$ How to decode 010110?

ababba?, *ccba*?, *abcba*?, or ...?

To avoid ambiguity, we need the encoding to be prefix-free

Definition

An encoding is **prefix-free** if no codeword is a prefix of any other codewords

Tree representation of a prefix-free encoding

Definition

A **full binary tree** is a binary tree where each node is either a leaf or it has two children

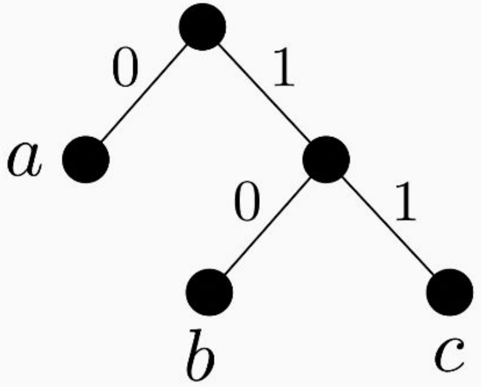
Tree representation of a prefix-free encoding

Definition

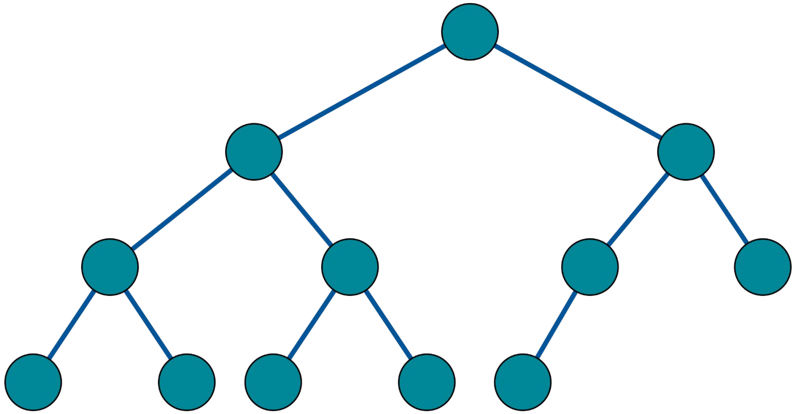
A **full binary tree** is a binary tree where each node is either a leaf or it has two children

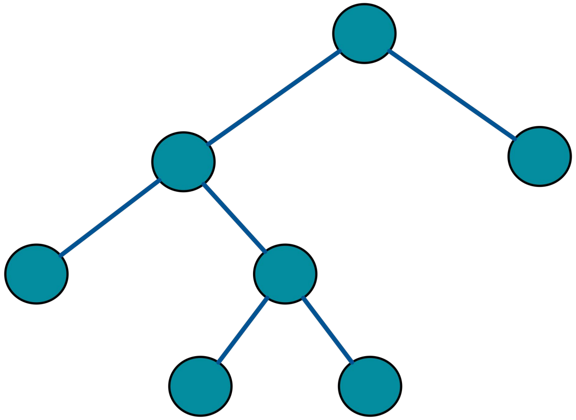
We use a full binary tree to represent a prefix-free encoding

- leaves are corresponding to symbols in Γ
- label edge to the left child with 0
- label edge to the right child with 1



Is it a full binary tree?





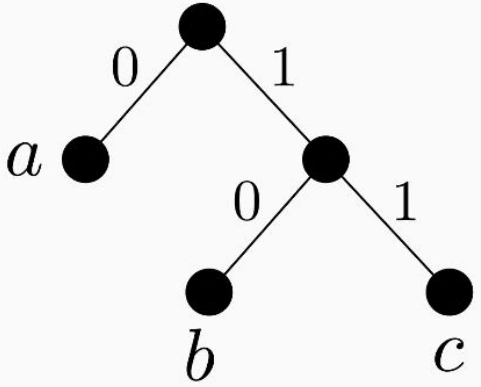
Tree representation of a prefix-free encoding

Definition

A **full binary tree** is a binary tree where each node is either a leaf or it has two children

We use a full binary tree to represent a prefix-free encoding

- leaves are corresponding to symbols in Γ
- label edge to the left child with 0
- label edge to the right child with 1



To obtain the encoding, read edge labels from root to a symbol

$a \to 0$, $b \to 10$, $c \to 11$,

Depth of a leaf $\equiv$ length of its codeword

It guarantees to be prefix-free

Tree representation of a prefix-free encoding

Let e be an encoding represented by a tree

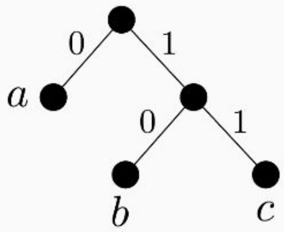
For string S, let f_v be the symbol count in S for each $v \in \Gamma$

Tree representation of a prefix-free encoding

Let e be an encoding represented by a tree

For string S, let f_v be the symbol count in S for each $v \in \Gamma$

$$|S_e| = \sum_{v \in \Gamma} f_v \cdot \mathrm{depth}(v)$$



$$|S_e| = 45 \times 1 + 16 \times 2 + 2 \times 2 = 81$$

Consider $\Gamma = \{a, b, c\}$

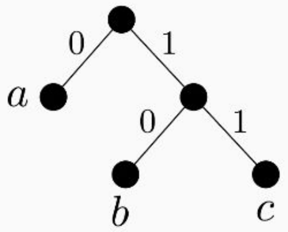
Stats on S: a appears 45 times, b 16 times, and c twice

Tree representation of a prefix-free encoding

Let e be an encoding represented by a tree

For string S, let f_v be the symbol count in S for each $v \in \Gamma$

$$|S_e| = \sum_{v \in \Gamma} f_v \cdot \mathrm{depth}(v)$$



$$|S_e| = 45 \times 1 + 16 \times 2 + 2 \times 2 = 81$$

A useful re-write: label internal nodes with counts of descendants

Consider $\Gamma = \{a, b, c\}$

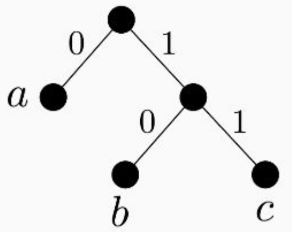
Stats on S: a appears 45 times, b 16 times, and c twice

Tree representation of a prefix-free encoding

Let e be an encoding represented by a tree

For string S, let f_v be the symbol count in S for each $v \in \Gamma$

$$|S_e| = \sum_{v \in \Gamma} f_v \cdot \mathrm{depth}(v)$$



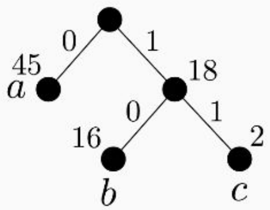
$$|S_e| = 45 \times 1 + 16 \times 2 + 2 \times 2 = 81$$

A useful re-write: label internal nodes with counts of descendants

For all non-root node v, define

$\mathrm{cost}(v) :=$ sum of leaf node counts descending from v

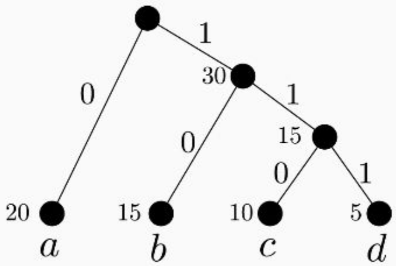
$$|S_e| = \sum_{v \in T - \{\mathrm{root}\}} \mathrm{cost}(v)$$



$$|S_e| = 45 + 16 + 2 + 18 = 81$$

Constructing the prefix-free encoding tree: examples

- $a : 20, b : 15, c : 10, d : 5$



$a \rightarrow 0$

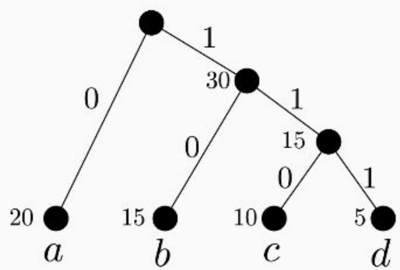
$b \rightarrow 10$

$c \rightarrow 110$

$d \rightarrow 111$

Constructing the prefix-free encoding tree: examples

- $a : 20, b : 15, c : 10, d : 5$



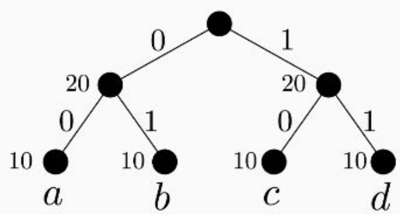
$a \to 0$

$b \to 10$

$c \to 110$

$d \to 111$

- $a : 10, b : 10, c : 10, d : 10$



$a \to 00$

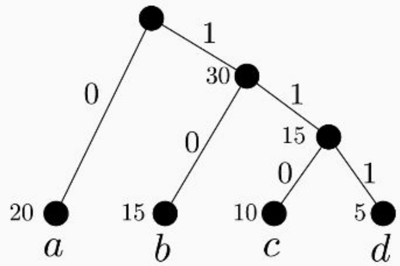
$b \to 01$

$c \to 10$

$d \to 11$

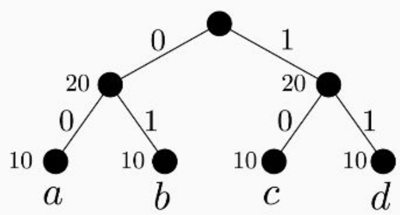
Constructing the prefix-free encoding tree: examples

- $a : 20, b : 15, c : 10, d : 5$



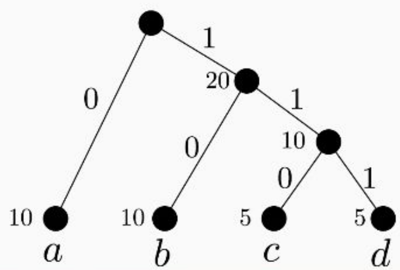
$a \to 0$
$b \to 10$
$c \to 110$
$d \to 111$

- $a : 10, b : 10, c : 10, d : 10$



$a \to 00$
$b \to 01$
$c \to 10$
$d \to 11$

- $a : 10, b : 10, c : 5, d : 5$



$a \to 0$
$b \to 10$
$c \to 110$
$d \to 111$

What are the total costs of these encodings?