

Quiz Section Week 5

April 26, 2018

Review

Topics (not guaranteed to be comprehensive!)

- Alignments
 - Reasons to align sequences
 - Needleman-Wunsch algorithm
 - Smith-Waterman algorithm
 - Effects of parameter variation (including gap penalties)
 - Testing for statistical significance of an alignment
- Phylogenetic trees
 - Rooted and unrooted topologies
 - Defining the best tree with UPGMA and Neighbor Joining
 - Concept of parsimony
 - Fitch algorithm: quantifying how parsimonious a tree is, assigning internal states
 - Finding the most parsimonious tree: Hill climbing w/ Nearest-Neighbor interchanges
 - Bootstrapping to quantify confidence in tree partitions
- Clustering
 - Defining a clustering problem
 - Hierarchical clustering
 - Impact of using single/complete/average linkage
 - K-means: Objective and algorithm
- Networks
 - Reasons to make and analyze networks
 - Basic network definitions
 - Dijkstra's Algorithm
 - Network motifs and their uses

Phylogenetic trees

UPGMA/Neighbor Joining

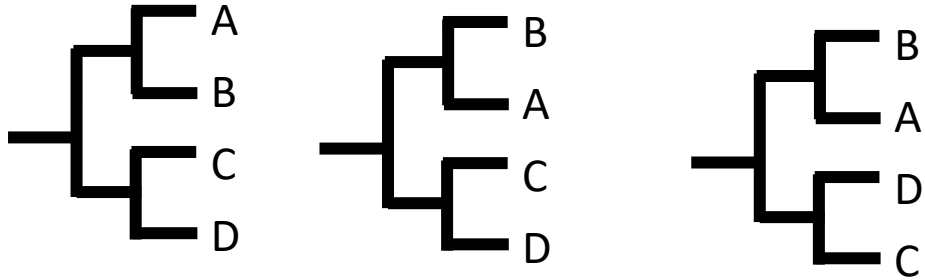
- Define the best tree: based on distance between leaves
- Find the best tree using: polynomial time algorithm to construct the best tree from a distance matrix

Parsimony approach

- Define the best tree: Minimum # of mutations required to traverse tree
- Find the best tree: by enumerating all trees (exhaustive search), or by heuristic approach like Nearest-Neighbor Interchange Hill-Climbing

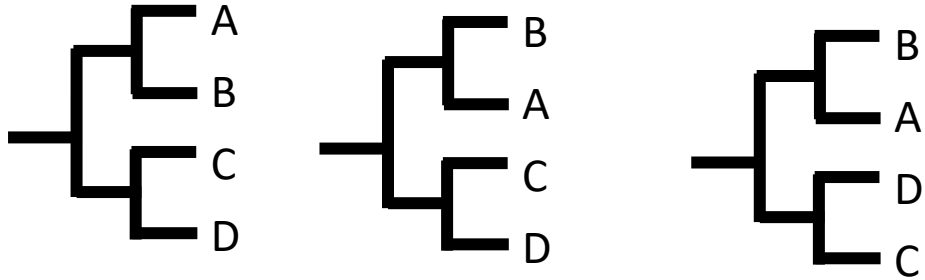
Tree topologies

Are these the same tree?

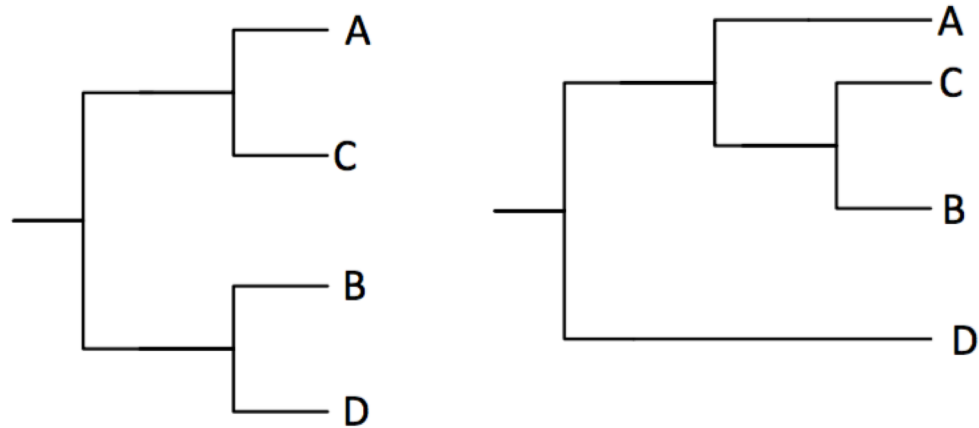


Tree topologies

Are these the same tree?



How about these?



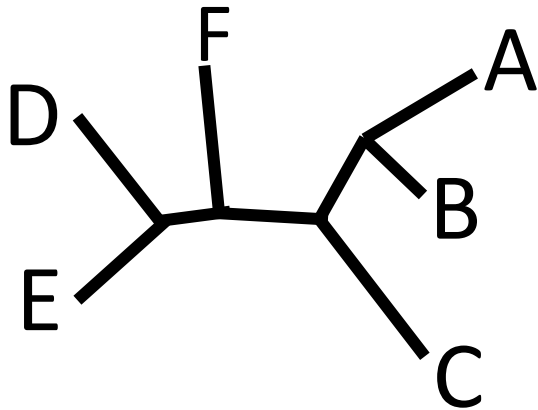
Counting tree topologies

For N leaves

of unrooted topologies = $3 * 5 * 7 * \dots * (2N-5)$

of branches = $2N-3$

E.g. an unrooted tree with 6 nodes



How many different topologies?

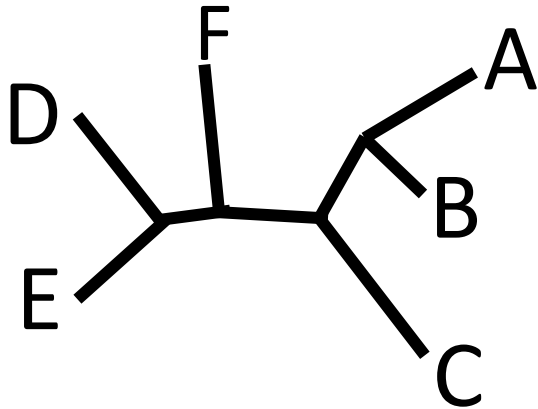
Counting tree topologies

For N leaves

of unrooted topologies = $3*5*7*\dots*(2N-5)$

of branches = $2N-3$

E.g. an unrooted tree with 6 nodes



How many different topologies?

$$3*5*7 = \mathbf{105}$$

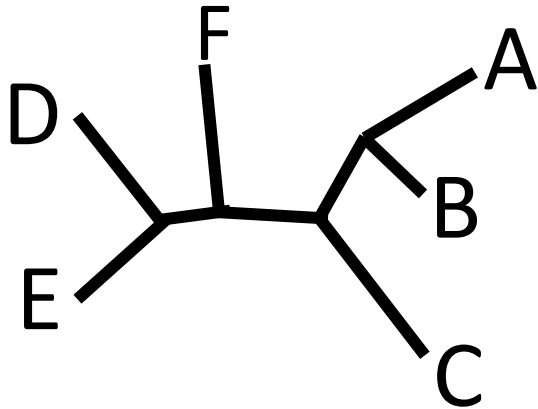
Counting tree topologies

For N leaves

of unrooted topologies = $3*5*7*\dots*(2N-5)$

of branches = $2N-3$

E.g. an unrooted tree with 6 nodes



How many different topologies?

$$3*5*7 = \mathbf{105}$$

How many branches?

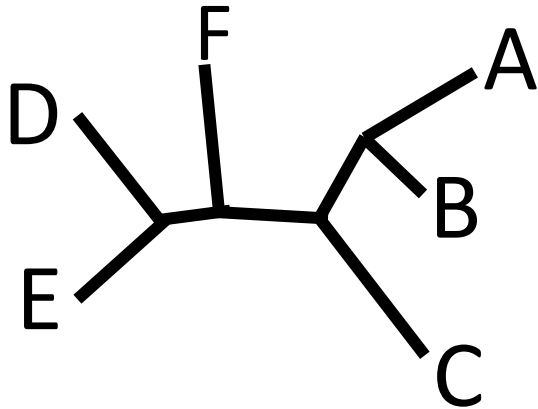
Counting tree topologies

For N leaves

of unrooted topologies = $3*5*7*\dots*(2N-5)$

of branches = $2N-3$

E.g. an unrooted tree with 6 nodes



How many different topologies?

$$3*5*7 = \mathbf{105}$$

How many branches?

$$2N-3 = \mathbf{9}$$

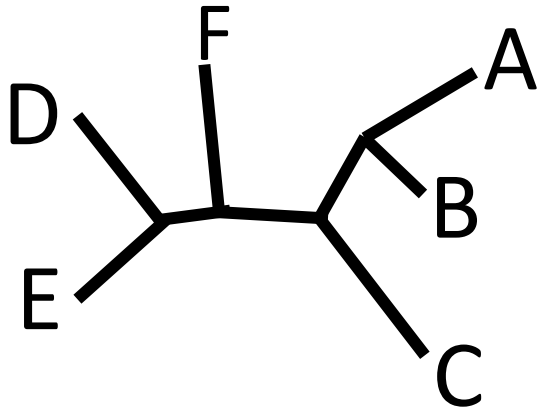
Counting tree topologies

For N leaves

of unrooted topologies = $3*5*7*\dots*(2N-5)$

of branches = $2N-3$

E.g. an unrooted tree with 6 nodes



How many different topologies?

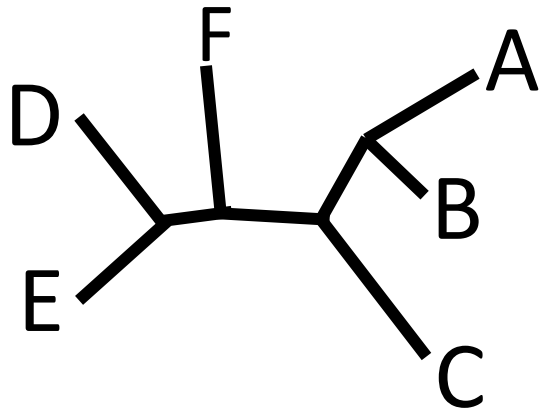
$$3*5*7 = \mathbf{105}$$

How many branches?

$$2N-3 = \mathbf{9}$$

The root could be placed on any branch

E.g. an unrooted tree with 6 nodes

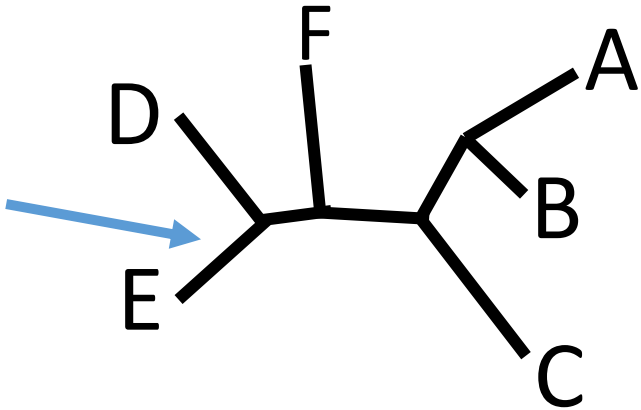


How many different topologies?

$$3 * 5 * 7 = \mathbf{105}$$

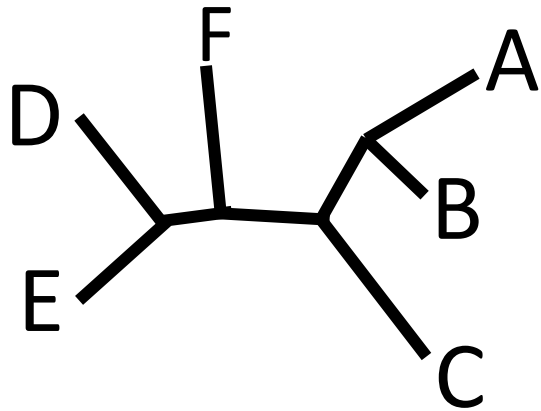
How many branches?

$$2N - 3 = \mathbf{9}$$



The root could be placed on any branch

E.g. an unrooted tree with 6 nodes

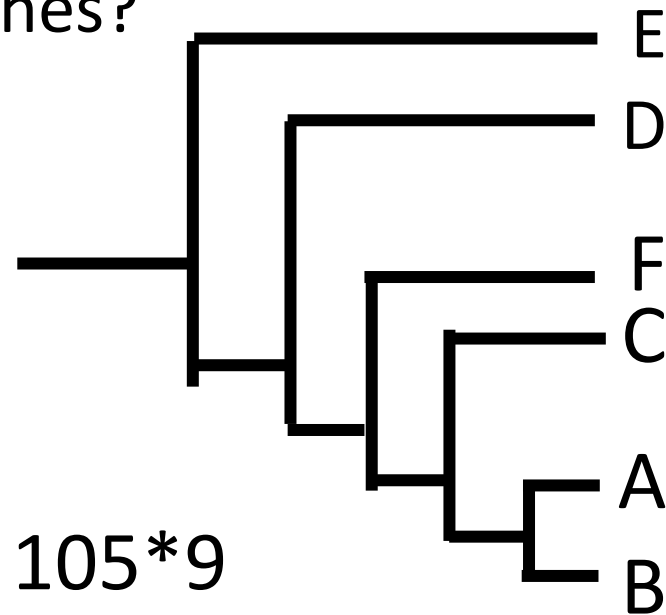
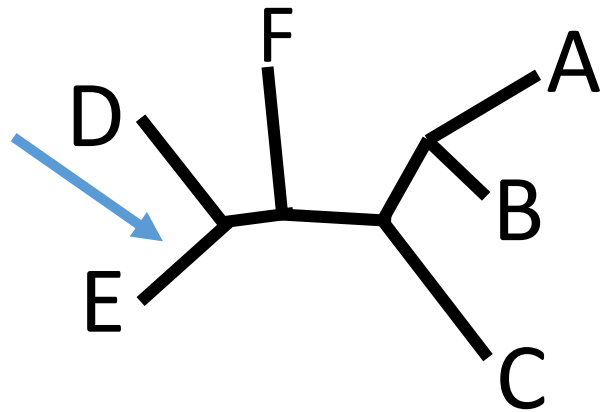


How many different topologies?

$$3*5*7 = \mathbf{105}$$

How many branches?

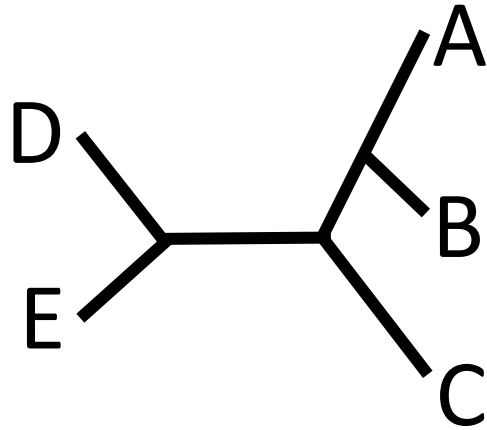
$$2N-3 = \mathbf{9}$$



$$\begin{aligned} \text{Total options} &= 105*9 \\ &= \mathbf{945} \end{aligned}$$

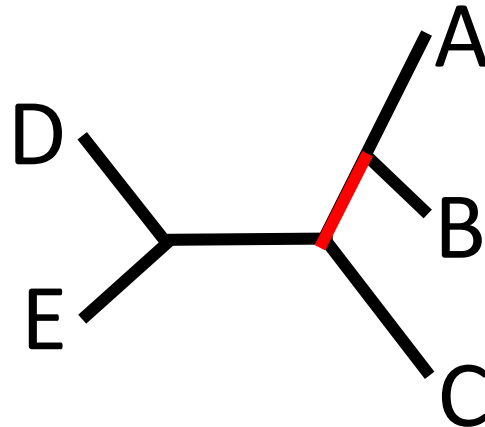
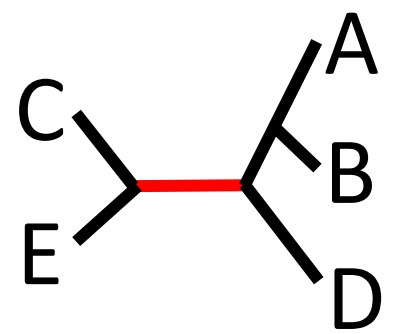
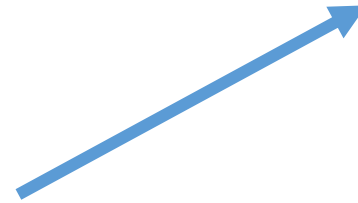
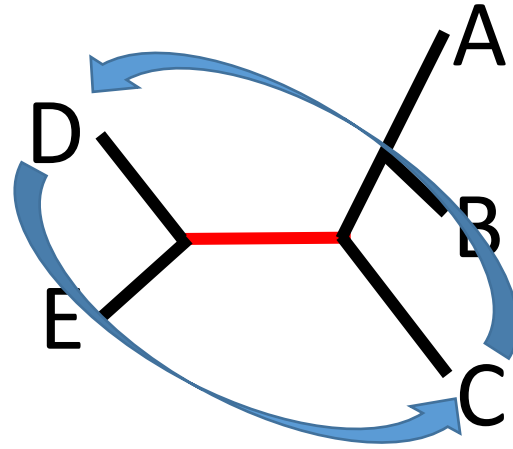
Nearest Neighbor Interchange trees

For each **internal branch**
generate two variant trees
that swap the
relationships of the four
outside branches



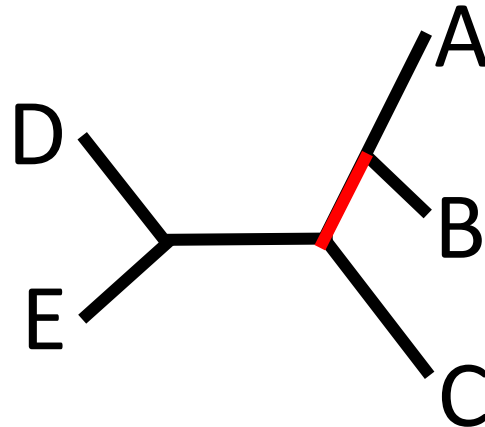
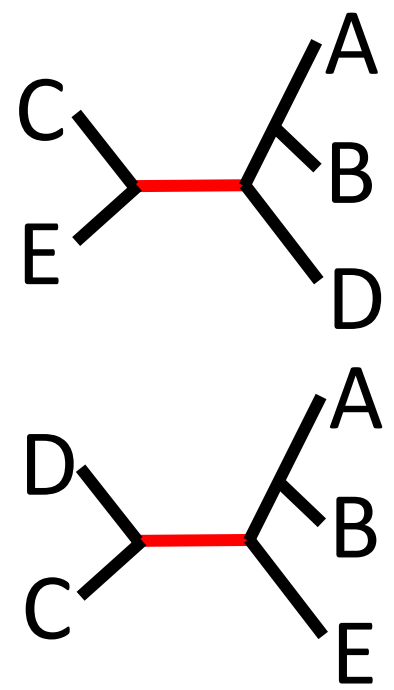
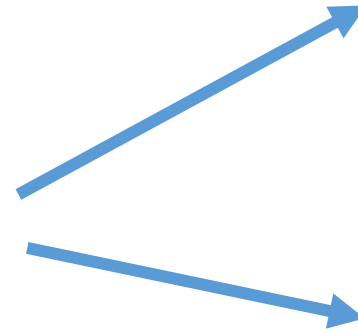
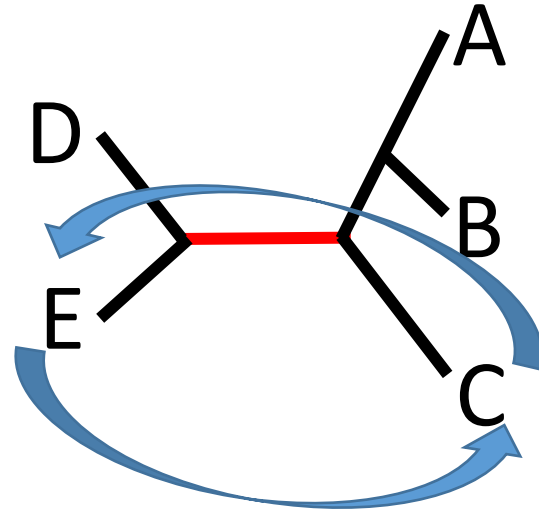
Nearest Neighbor Interchange trees

For each **internal branch**
generate two variant trees
that swap the
relationships of the four
outside branches



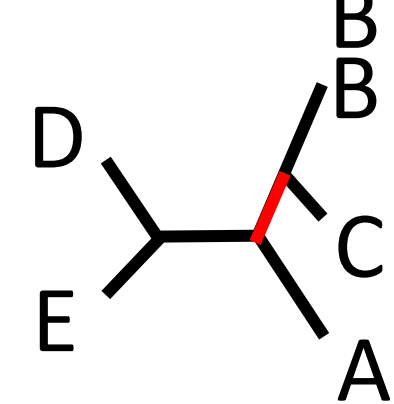
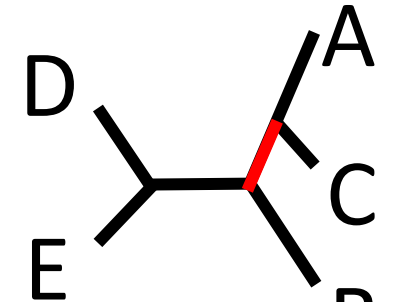
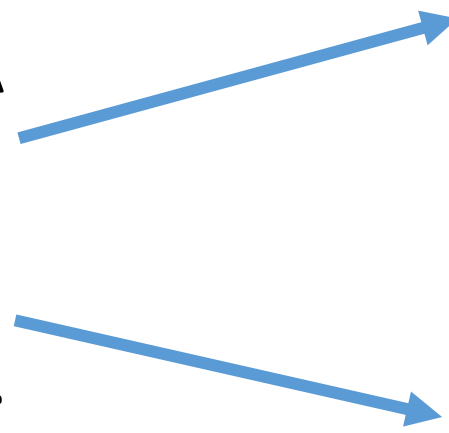
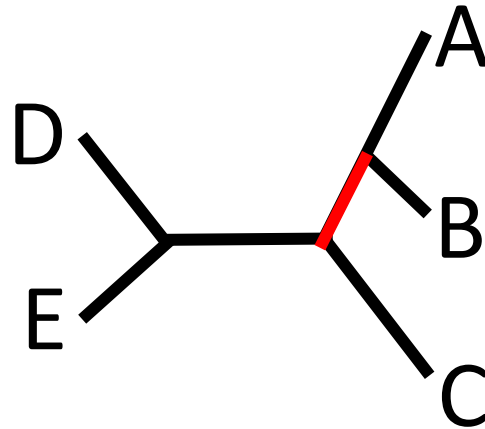
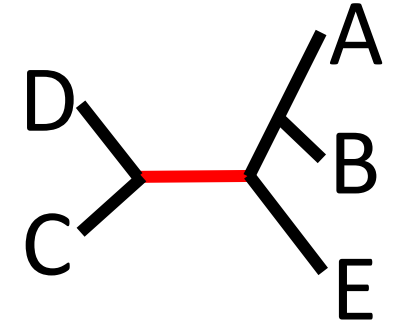
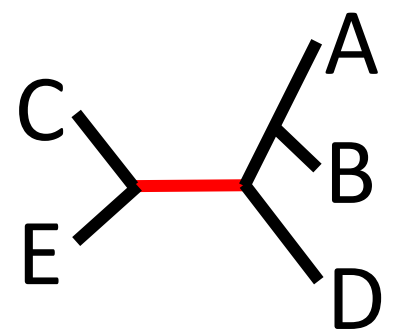
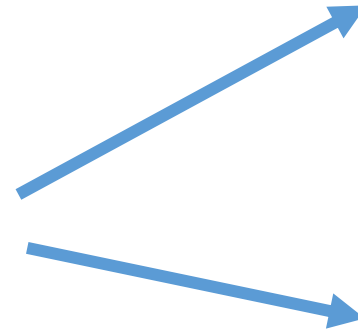
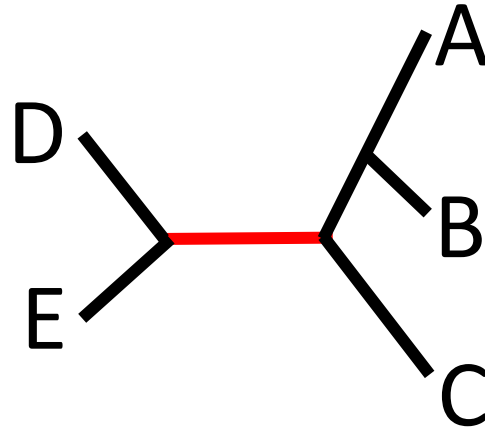
Nearest Neighbor Interchange trees

For each **internal branch**
generate two variant trees
that swap the
relationships of the four
outside branches

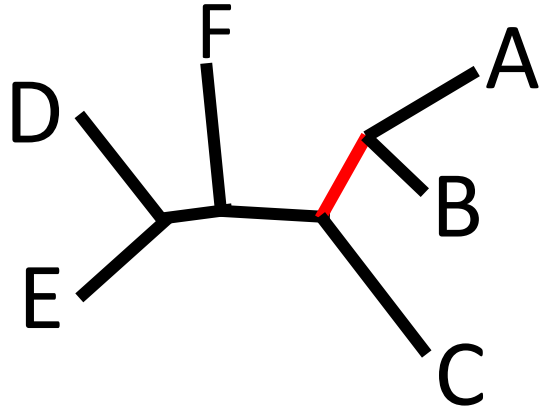


Nearest Neighbor Interchange trees

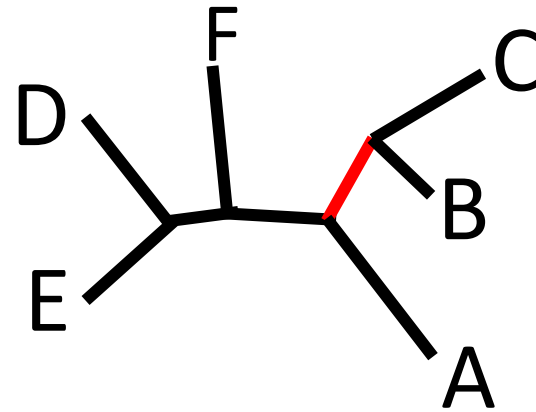
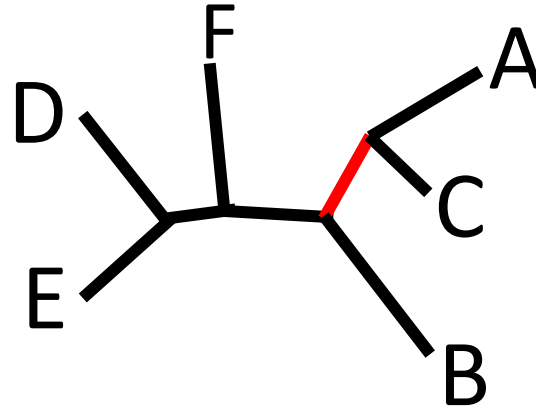
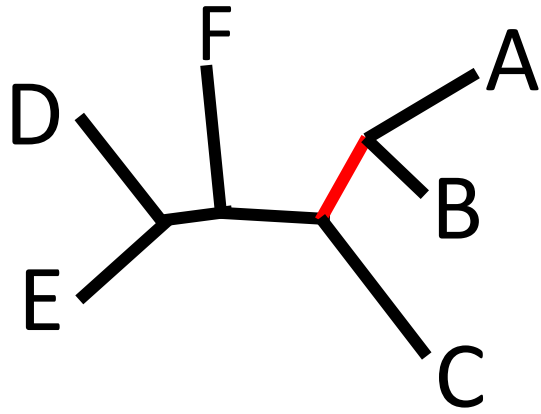
For each **internal branch**
generate two variant trees
that swap the
relationships of the four
outside branches



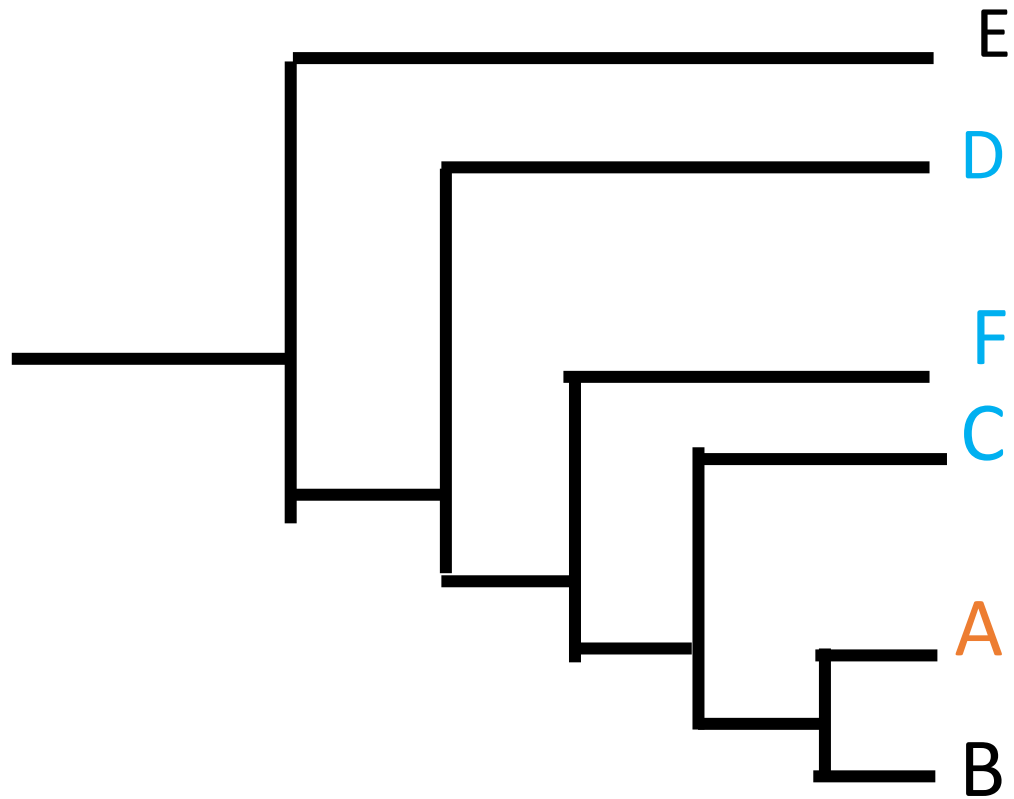
NN Practice: Draw both interchanges from swapping this branch



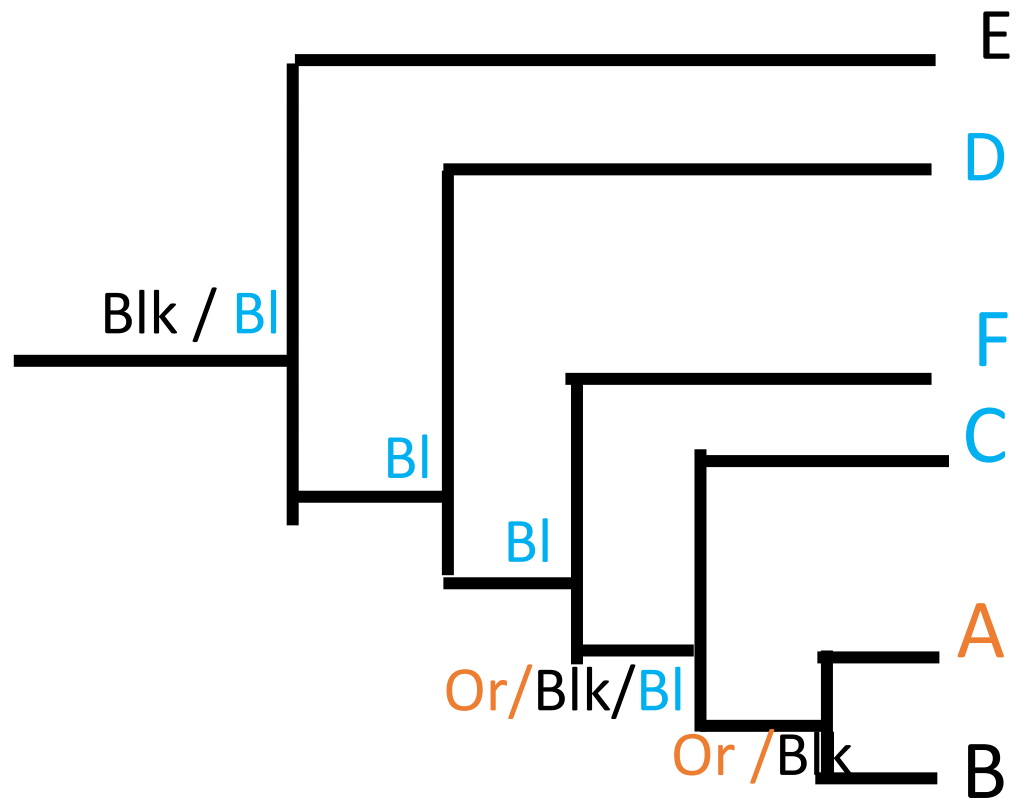
NN Practice: Draw both interchanges from swapping this branch



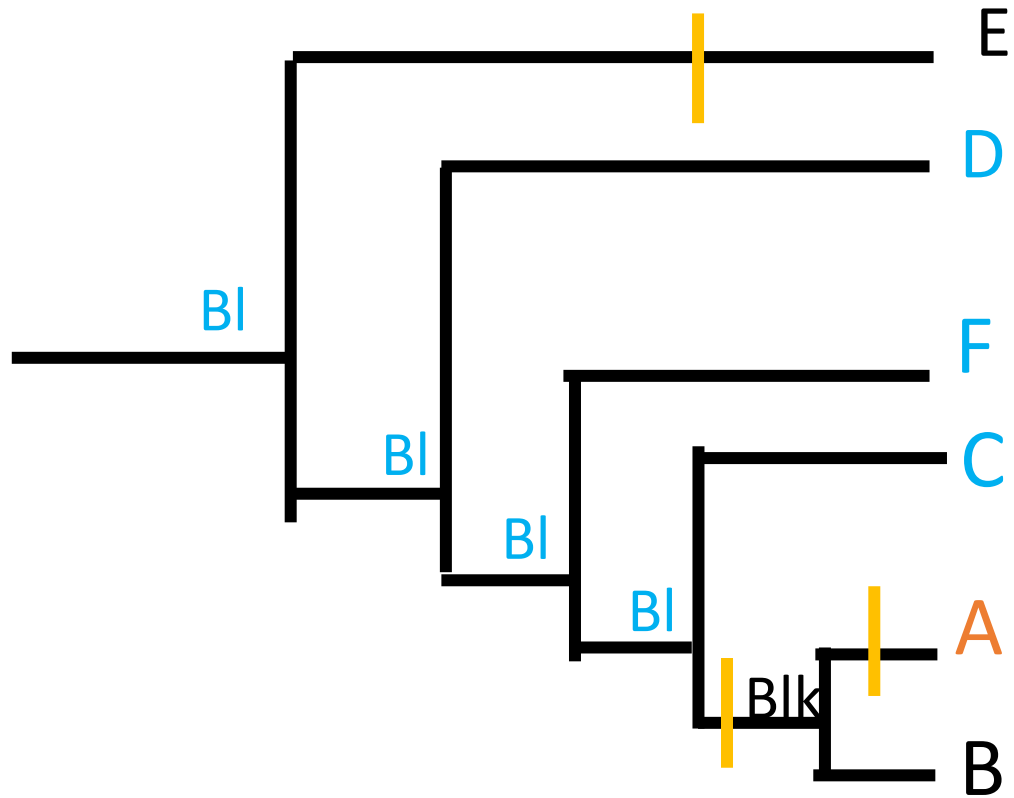
Fitch algorithm practice



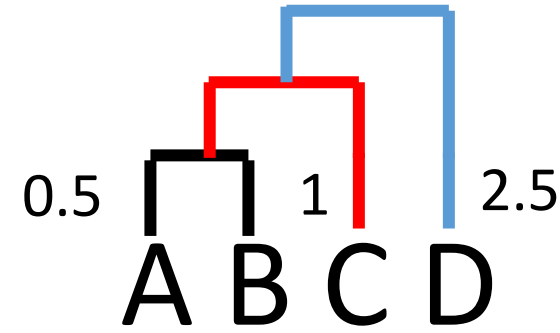
Fitch algorithm practice: bottom-up



Fitch algorithm practice: top-down



Hierarchical clustering with complete linkage example

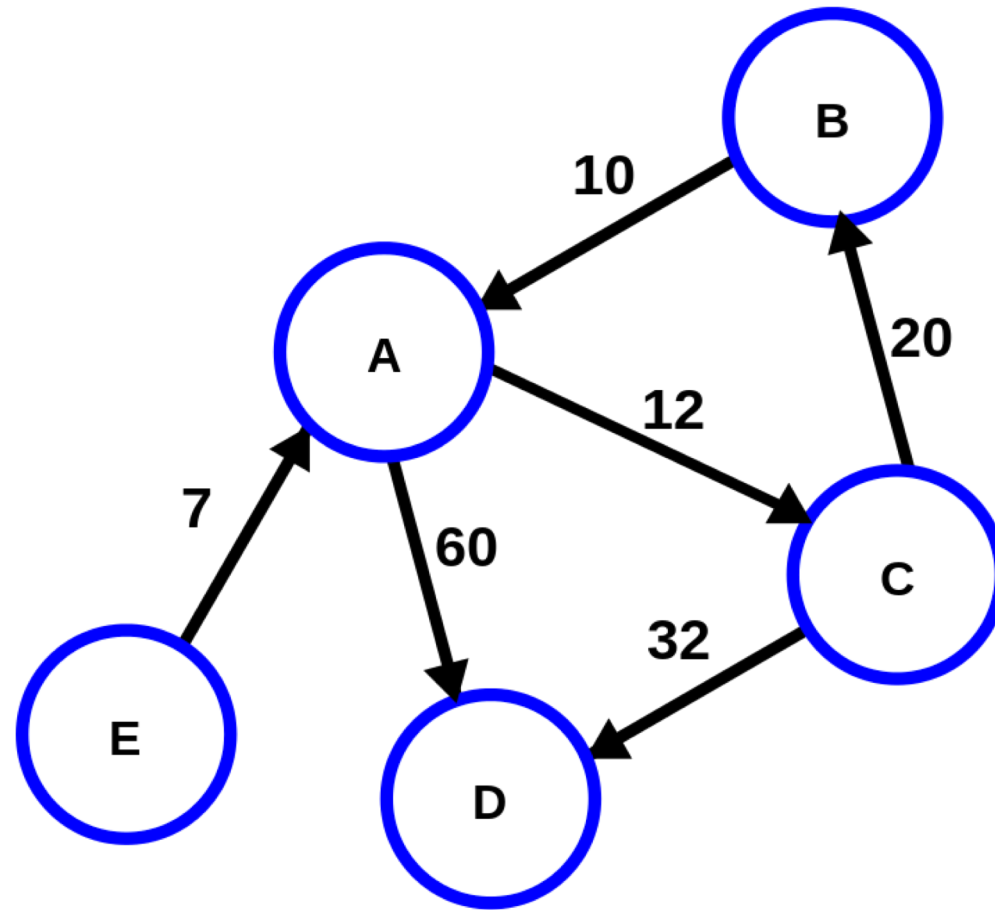


	A	B	C	D
A	0	1	2	4
B	1	0	2	5
C	2	2	0	5
D	4	5	5	0

	A,B	C	D
A,B	0	2	5
C	2	0	5
D	5	5	0

	A,B,C	D
A,B,C	0	5
D	5	0

Dijkstra's Algorithm



Another data type: Dictionaries

- a data structure that consists of an unordered set of *key: value* pairs
 - think of as *word: definition* pairs!

Q: How could we encode the entire genetic code?

Dictionaries: How could we encode the entire genetic code?

```
>>> genetic_code = {"ATG": "Start", "TGA": "Stop", "TAG":  
"Stop"}  
>>> genetic_code["TAA"] = "Stop"  
>>> genetic_code.get("TGA")  
'Stop'  
>>> genetic_code["TGA"]  
'Stop'  
>>> genetic_code.get("sss") #nothing or 'None' if not defined  
>>> genetic_code["sss"]  
KeyError: 'ttt'
```

Creating a dictionary

```
#create an empty dictionary  
myDict = {}
```

```
#create a dictionary with three entries  
myDict = {"Curly":4123, "Larry":2057, "Moe":1122}
```

```
#add another entry  
myDict["Shemp"] = 2232
```

```
#change Moe's phone number  
myDict["Moe"] = 4040
```

```
#delete Moe from dictionary  
del myDict["Moe"]
```

Rules for dictionaries

- The first item is a **key**.
- Each key can appear only once in a dict.
- A key must be an immutable object: number, string, or tuple.
- Lists cannot be keys (they are mutable).
- The key is the item you'll use to do look-ups.
- Each **key** is paired with a **value**.

Some useful dictionary methods

```
>>> genetic_code.items()
[('TAA', 'Stop'), ('TGA', 'Stop'), ('TAG', 'Stop'),
 ('ATG', 'Start')]
>>> genetic_code.keys()
['TAA', 'TGA', 'TAG', 'ATG']
>>> genetic_code.values()
['Stop', 'Stop', 'Stop', 'Start']
```

Another use of dictionaries: store counts of named elements

Example: Calculate # of each nucleotide in a sequence

```
sequence = "GACCCT"  
nuc_counts = {'A': 0, 'C': 0, 'T': 0, 'G': 0}  
for nuc in sequence:  
    #Add to the count for the given nucleotide
```

Another common use of dictionaries: store counts of named elements

Calculate # of each nucleotide in a sequence

```
sequence = "GACCCT"  
nuc_counts = {'A': 0, 'C': 0, 'T': 0, 'G': 0}  
for nuc in sequence:  
    nuc_counts[nuc] = nuc_counts[nuc] + 1
```

Traversing a dictionary by key

```
# birthdays is a dictionary with names as keys  
# and birth dates as values  
  
for person in birthdays.keys():  
    print "Send", person, "a card on", birthdays[person]
```

dictionary basics

```
D = {'dna': 'T', 'rna': 'U'} # dictionary literal assignment
D = {}                       # make an empty dictionary
D.keys()                     # get the keys as a list
D.values()                   # get the values as a list
D['dna']                      # get a value based on key
D['dna'] = 'T'               # set a key:value pair
del D['dna']                  # delete a key:value pair
'dna' in D                    # True if key 'dna' is found in D, else False
```

The keys must be immutable objects (e.g. string, int, tuple).

The values can be anything (including a list or another dictionary).

The order of elements in the list returned by **D.keys()** or **D.values()** is arbitrary (effectively random).

Each key can be stored only once in the dictionary, so if you set the value for a key for a second time it **OVERWRITES** the old value!

