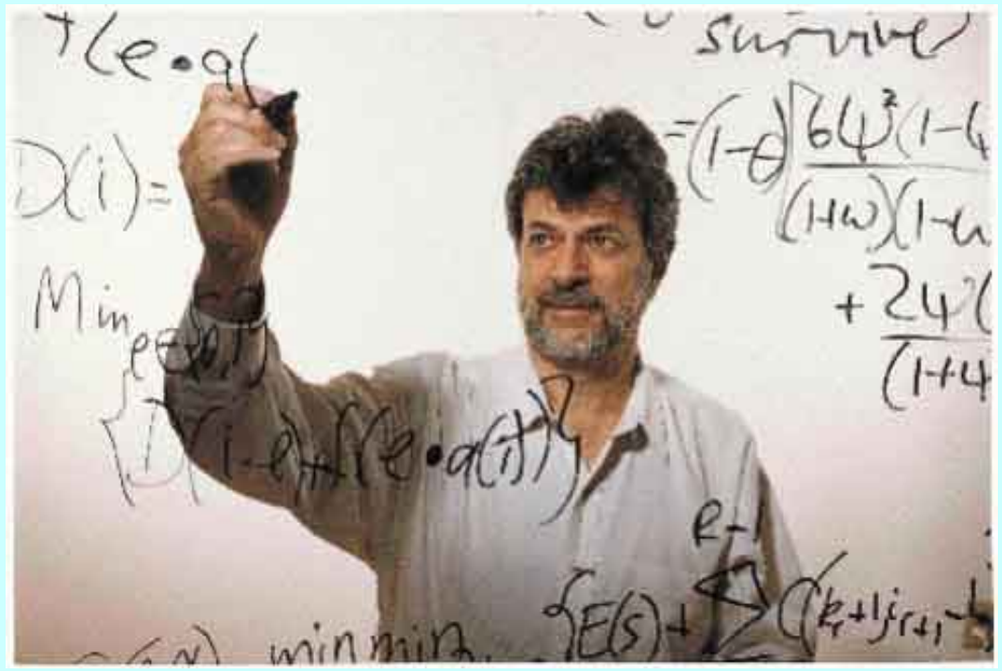
best tree

Slides from lectures of Joe Felsenstein, Jim Wilgenbusch, and Peter Beerli

David Sankoff



David Sankoff, in the 1990s, writing on a glass blackboard
(forwards? backwards?)

Sankoff's algorithm

A dynamic programming algorithm for counting the smallest number of possible (weighted) state changes needed on a given tree.
Let $\mathbf{S_j(i)}$ be the smallest (weighted) number of steps needed to evolve the subtree at or above node $\mathbf{j}$, given that node $\mathbf{j}$ is in state $\mathbf{i}$.
Suppose that $\mathbf{c_{ij}}$ is the cost of going from state $\mathbf{i}$ to state $\mathbf{j}$.

Initially, at tip (say) j

$$\mathbf{S_j(i)} = \begin{cases} 0 & \text{if node } j \text{ has (or could have) state } i \\ \infty & \text{if node } j \text{ has any other state} \end{cases}$$

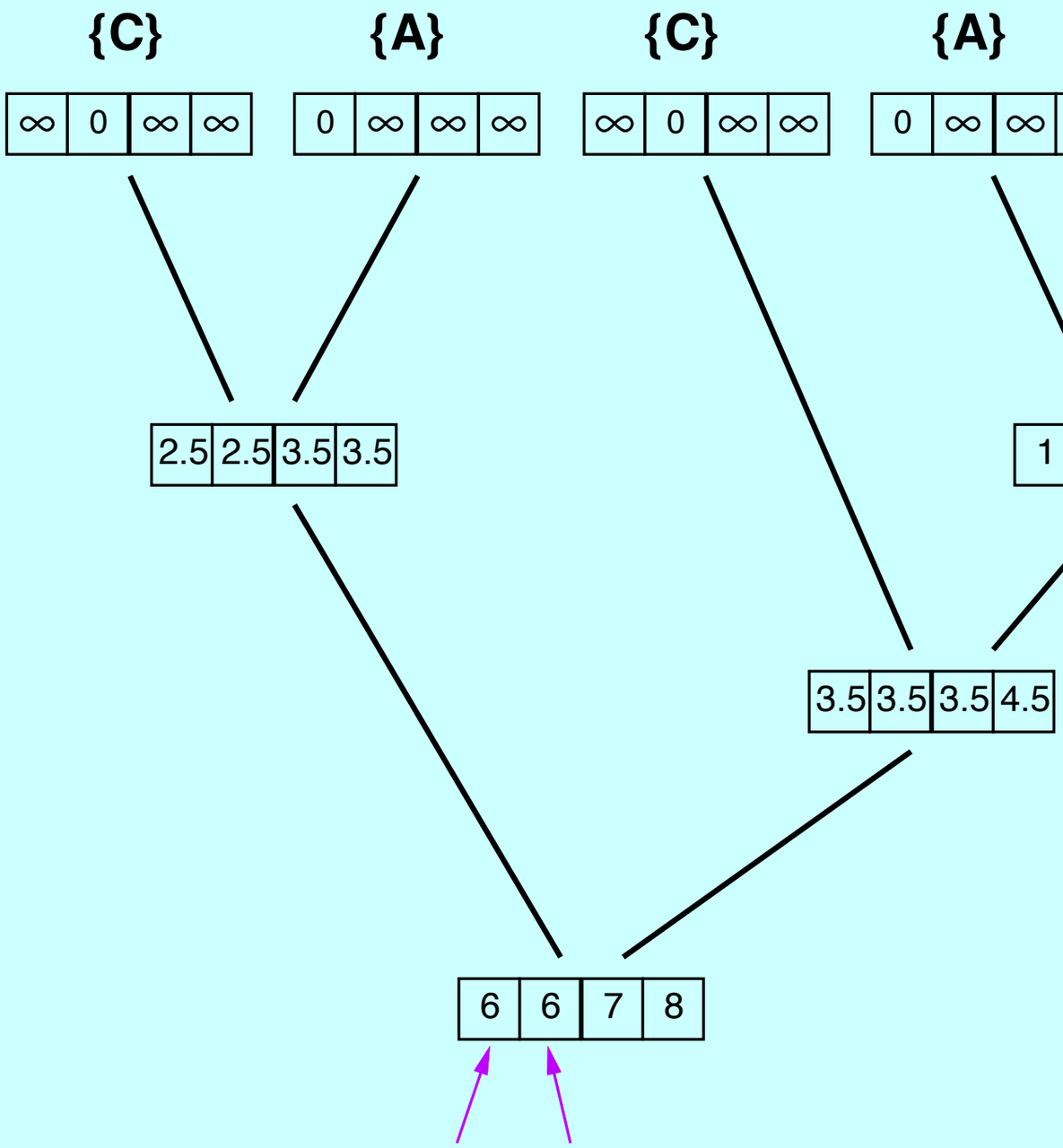
Sankoff's algorithm (continued)

Then proceeding down the tree (postorder tree traversal) for node a whose immediate descendants are ℓ and r

$$\mathbf{S_a(i)} \;=\; \min_{\mathbf{j}} \left[\mathbf{c_{ij}} + \mathbf{S}_\ell\mathbf{(j)} \right] + \min_{\mathbf{k}} \left[\mathbf{c_{ik}} + \mathbf{S_r(k)} \right]$$

The minimum number of (weighted) steps for the tree is found by computing at the bottom node (0) the $\mathbf{S_0(i)}$ and taking the smallest of these.

An example using Sankoff's algorithm



cost matrix:

from \ to	A	C	G	T
A	0	2.5	1	2.5
C	2.5	0	2.5	1
G	1	2.5	0	2.5
T	2.5	1	2.5	0

$$\mathbf{S_a(i)} \;=\; \min_{\mathbf{j}} \left[\, \mathbf{c_{ij}} + \mathbf{S}_\ell\mathbf{(j)} \,\right] + \min_{\mathbf{k}} \left[\, \mathbf{c_{ik}} + \mathbf{S_r(k)} \,\right]$$

1.1 Counting rooted trees

Cayley (1857) seems to be the first one to publish tree counting methods and many followed, it seems that many authors reinvented formulas for counting several times in the biological literature and the computer science literature. Counting trees might be futile exercise but it will give a good impression about how many things are contributing to a solution. It is also directly connected to the *exhaustive search* method.

Starting with a tree with two tips we easily recognize that there this only possible tree. We can add a third tip either into the branch from A to the root, into the branch from B to the root, or below the current root. There are 3 possible rooted bifurcating trees with 3 tips: ((a,b),c), ((a,c),b), ((c,b),a). We just added a node and and an additional branch. A fourth tip can be added at all 4 branches above the root and also 1 below the root, there are 5 possible branches to insert. this leaves us wit 3×5 possibilities for a rooted 4-tip tree.(figure 1).

We outlined an algorithm to calculate the total number of possible unrooted trees. We need only to realize that there are $2n-3$ possible branches were we can insert a new branch. The $2n-3$ we can get by recognizing that there are n tips and $n-2$ interior branches and one root branch. Once we recognize the series it is even simpler

$$1 \times 3 \times 5 \times 7... \times (2n-3) \tag{1}$$

We can express this in a more compact formula

$$\text{number of rooted trees} = \frac{(2n-3)!}{2^{n-2}(n-2)!} \tag{2}$$

$$(2n-3)!!$$

Double factorial

From Wikipedia, the free encyclopedia

In mathematics, the product of all the integers from 1 up to some non-negative integer n that have the same parity (odd or even) as n is called the **double factorial** or **semifactorial** of n and is denoted by $n!!$.[1] That is,

$$n!! = \prod_{k=0}^{\left\lceil \frac{n}{2} \right\rceil - 1} (n-2k) = n(n-2)(n-4)\cdots$$

(A consequence of this definition is that 0!! = 1, as an empty product.)

Therefore, for even n the double factorial is

$$n!! = \prod_{k=1}^{\frac{n}{2}} (2k) = n(n-2)(n-4)\cdots 4\cdot 2\,,$$

and for odd n it is

$$n!! = \prod_{k=1}^{\frac{n-1}{2}} (2k-1) = n(n-2)(n-4)\cdots 3\cdot 1\,.$$

For example, 9!! = 9 × 7 × 5 × 3 × 1 = 945.

Counting multifurcating, bifurcating rooted trees

what about unrooted trees

TABLE 1. THE NUMBERS OF ROOTED TREES WITH n LABELLED TIPS AND WITH UNLABELLED INTERIOR NODES. THE LEFT COLUMN COUNTS ALL TREES, THE RIGHT COLUMN ONLY BIFURCATING TREES.

n	All trees	Bifurcating trees
1	1	1
2	1	1
3	4	3
4	26	15
5	236	105
6	2,752	945
7	39,208	10,395
8	660,032	135,135
9	12,818,912	2,027,025
10	282,137,824	34,459,425
11	6,939,897,856	654,729,075
12	188,666,182,784	13,749,310,575
13	5,617,349,020,544	316,234,143,225
14	181,790,703,209,728	7,905,853,580,625
15	6,353,726,042,486,112	213,458,046,676,875
16	238,513,970,965,250,048	6,190,283,353,629,375
17	9,571,020,586,418,569,216	191,898,783,962,510,625
18	408,837,905,660,430,516,224	6,332,659,870,762,850,625
19	18,522,305,410,364,568,764,416	221,643,095,476,699,771,875
20	887,094,711,304,094,583,095,296	8,200,794,532,637,891,559,375
21	44,782,218,857,751,551,087,214,592	319,830,986,772,877,770,815,625
22	2,376,613,641,928,796,906,249,519,104	13,113,070,457,687,988,603,440,625

Counting labelled histories

a huge impact as they are used in the framework of the coalescence theory. On rooted labeled histories we count pairs of nodes:

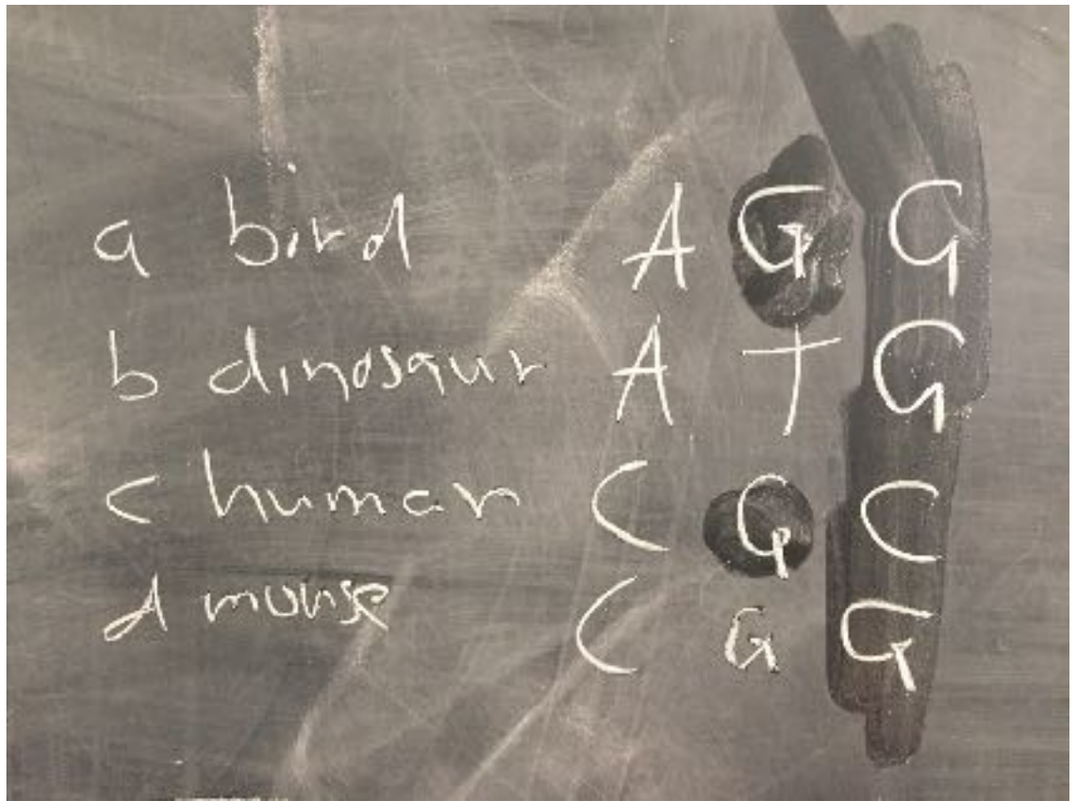
$$\frac{n(n-1)}{2} \times \frac{(n-1)(n-2)}{2} \times .... \times \frac{2(1)}{2} = \frac{n!(n-1)!}{2^{n-1}} \tag{4}$$

This numbers for labeled histories raise much faster than the rooted and even faster than the multifurcating trees.

(Draw example labelled histories)

Exercise what is the best tree using the Sankoff algorithm

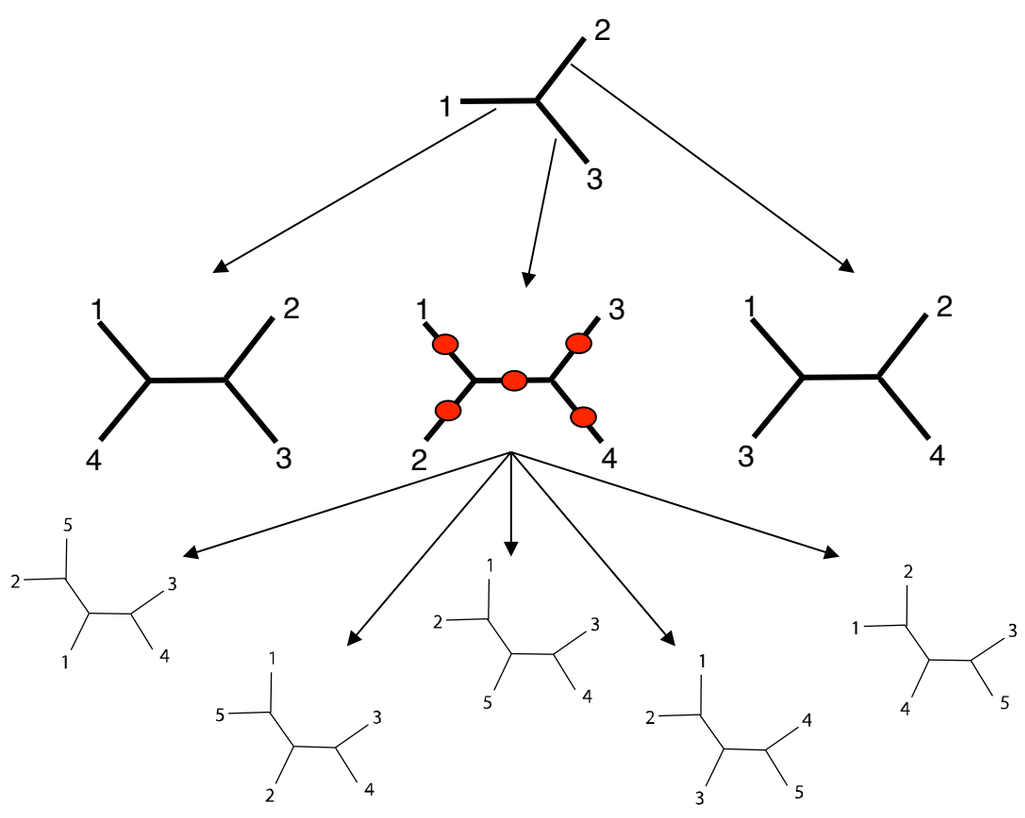
how may trees?
cost matrix?



Getting a Starting Tree

Stepwise Addition Algorithms

E.g., as is, simple, closest, further, random

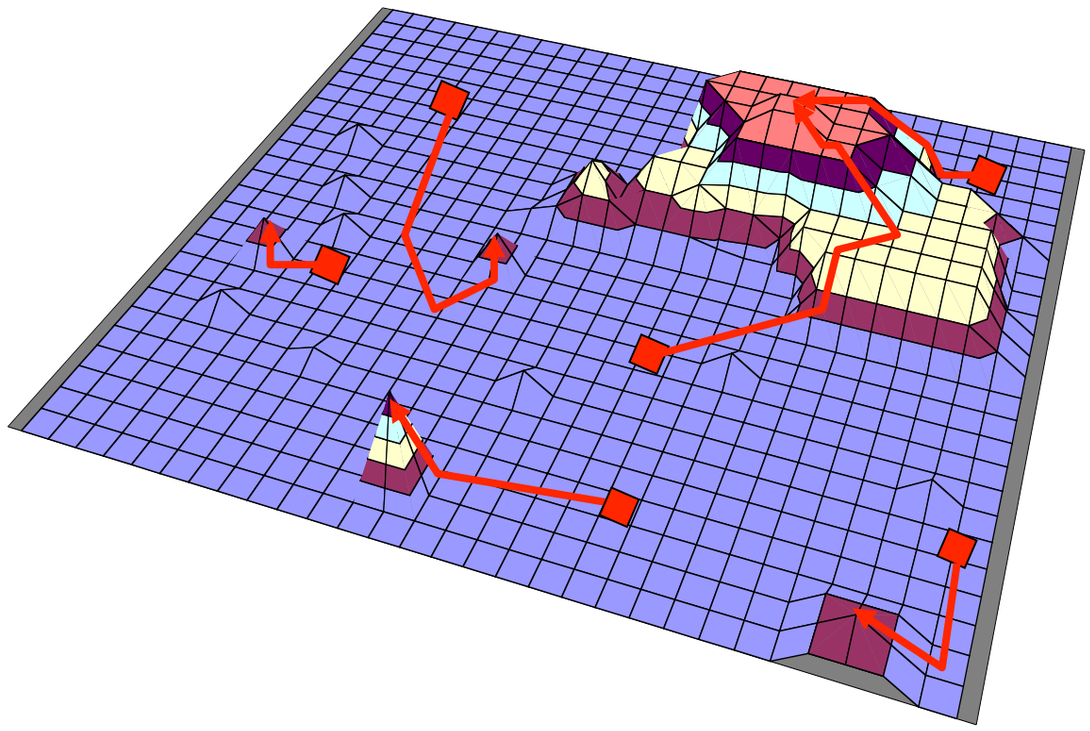


Getting a Starting Tree

Stepwise Addition Algorithms

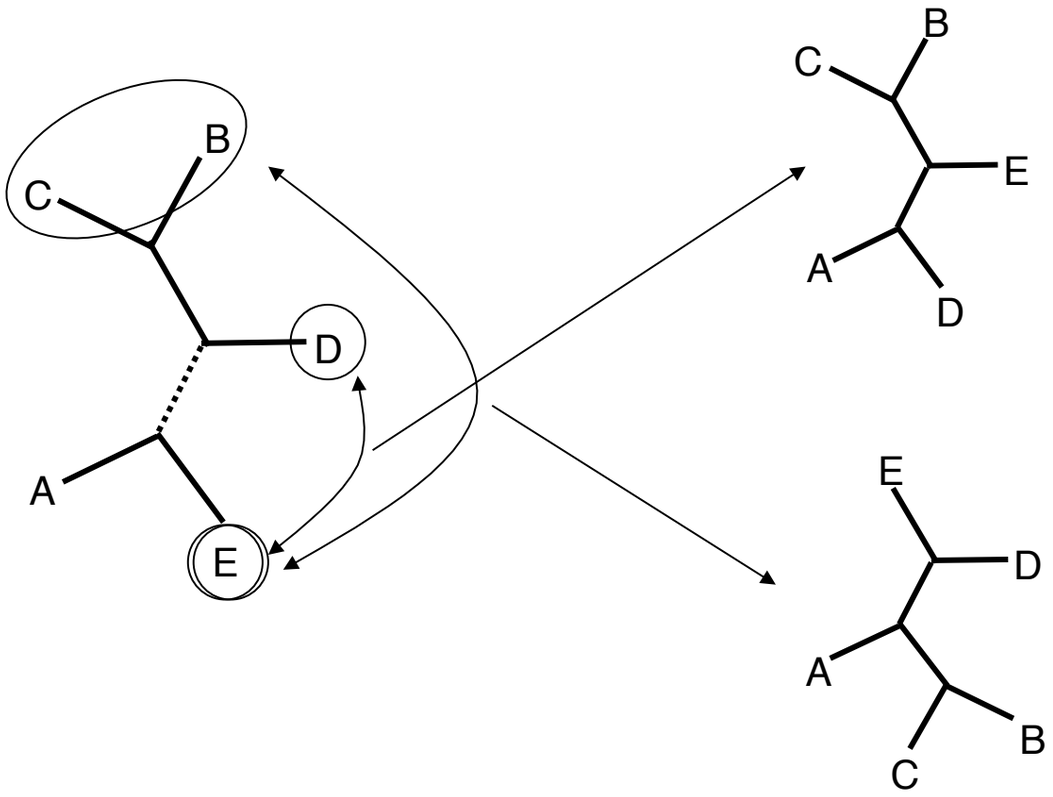
- "Greedy" algorithms
  – only make upward moves
- Prone to getting stuck on local optima

Random Addition Sequence



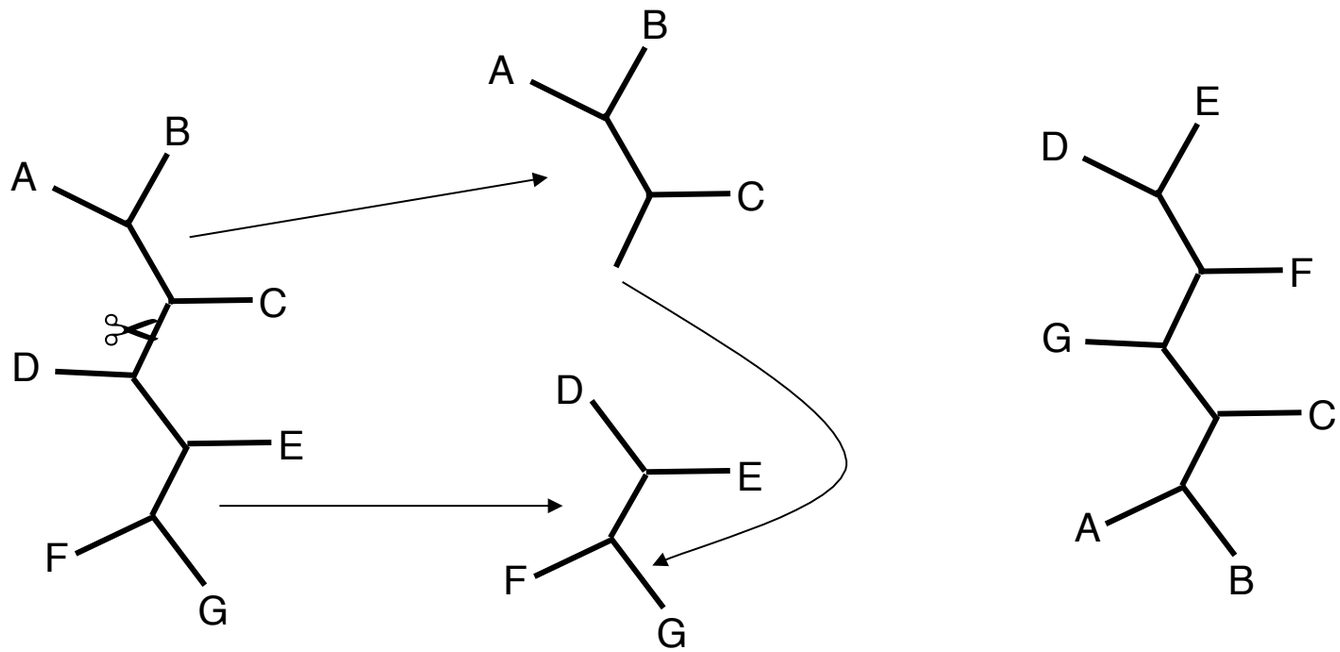
Branch swapping

Nearest Neighbor Interchange (NNI)



Branch swapping

Subtree Pruning and Regrafting (SPR)



Branch swapping

Tree Bisection and Reconnection (TBR)