Algorithmic Approaches for Biological Data, Lecture #21

Katherine St. John

City University of New York
American Museum of Natural History

25 April 2016

Outline

Guest Lecturer: Dr. Eric Ford

- Searching Graphs: Breadth-First & Depth-First Searches

Outline

Guest Lecturer: Dr. Eric Ford

- Searching Graphs: Breadth-First & Depth-First Searches
- Hill Climbing

Outline

Guest Lecturer: Dr. Eric Ford

- Searching Graphs: Breadth-First & Depth-First Searches
- Hill Climbing
- Optimality Criteria & Complexity

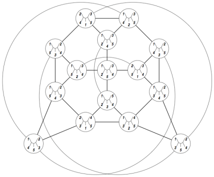
Outline

Guest Lecturer: Dr. Eric Ford

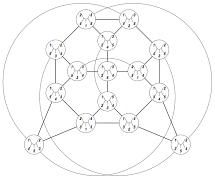
- Searching Graphs: Breadth-First & Depth-First Searches
- Hill Climbing
- Optimality Criteria & Complexity
- Parsimony Example: scoring trees in linear time

Searching Graphs

- Two common strategies:



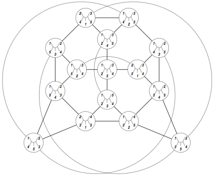
Bastert *et al.*, 2002



Bastert *et al.*, 2002

- Two common strategies:
  - Breadth First Search (BFS): visit all the neighbors, then visit all the neighbors' neighbors, etc.

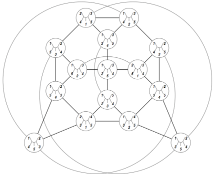
Searching Graphs



Bastert *et al.*, 2002

- Two common strategies:
  - Breadth First Search (BFS): visit all the neighbors, then visit all the neighbors' neighbors, etc.
  - Depth First Search (DFS): for each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.

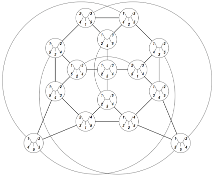
Searching Graphs



Bastert *et al.*, 2002

- Two common strategies:
  - Breadth First Search (BFS): visit all the neighbors, then visit all the neighbors' neighbors, etc.
  - Depth First Search (DFS): for each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.
- Bookkeeping is important:

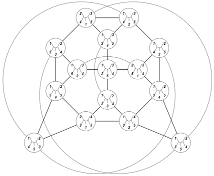
Searching Graphs



Bastert *et al.*, 2002

- Two common strategies:
  - Breadth First Search (BFS): visit all the neighbors, then visit all the neighbors' neighbors, etc.
  - Depth First Search (DFS): for each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.
- Bookkeeping is important:
  - Keep a "To Do" list (priority queue) of nodes still to visit.

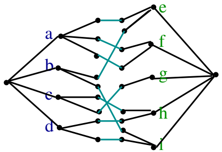
Searching Graphs

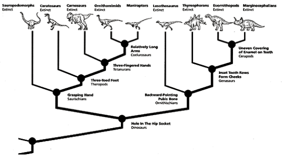


Bastert *et al.*, 2002

- Two common strategies:
  - Breadth First Search (BFS): visit all the neighbors, then visit all the neighbors' neighbors, etc.
  - Depth First Search (DFS): for each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.
- Bookkeeping is important:
  - Keep a "To Do" list (priority queue) of nodes still to visit.
  - Mark nodes as you visit them, so, you know not to visit again.

In Pairs: Searching Graphs

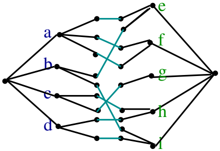


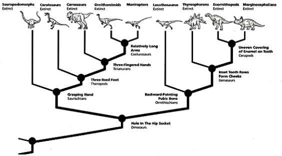


	a	b	c	d	e
a	0	1	1	0	0
b	0	0	1	1	0
c	1	0	0	1	0
d	0	0	1	0	1
e	0	0	0	0	0

1. For above, choose start and end points to be the worst for breath first search (i.e. ‘hide’ the endpoint so that it takes the most number of steps to reach it).
2. For above, choose start and end points to be the worst for depth first search
3. What is a graph on 6 vertices with the worst start/end pair for breath first search?
4. What about for depth first search?

Discussion: Searching Graphs

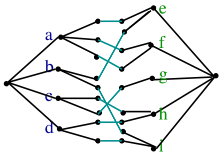


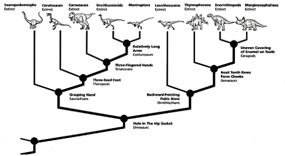


	a	b	c	d	e
a	0	1	1	0	0
b	0	0	1	1	0
c	1	0	0	1	0
d	0	0	1	0	1
e	0	0	0	0	0

- What makes a search difficult for BFS?

Discussion: Searching Graphs

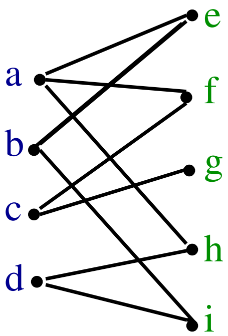




	a	b	c	d	e
a	0	1	1	0	0
b	0	0	1	1	0
c	1	0	0	1	0
d	0	0	1	0	1
e	0	0	0	0	0

- What makes a search difficult for BFS?
- What makes a search difficult for DFS?

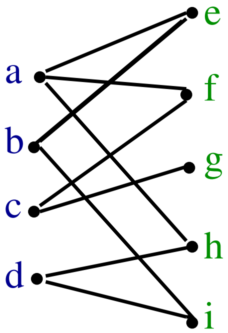
Implementing Searches



Bastert *et al.*, 2002

- How would you implement BFS?

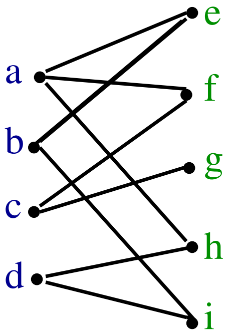
Implementing Searches



Bastert *et al.*, 2002

- How would you implement BFS?
  - Inputs: A graph `G`, a `start` and an `end`.

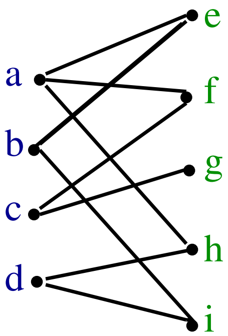
Implementing Searches



Bastert *et al.*, 2002

- How would you implement BFS?
  - Inputs: A graph `G`, a `start` and an `end`.
  - Algorithm: For each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.

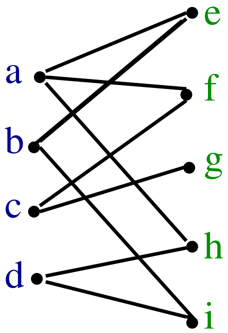
Implementing Searches



Bastert *et al.*, 2002

- How would you implement BFS?
  - Inputs: A graph `G`, a `start` and an `end`.
  - Algorithm: For each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.
- Bookkeeping is important:

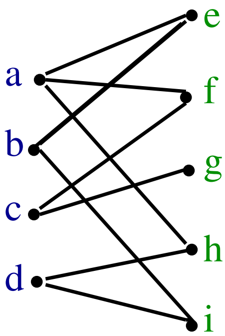
Implementing Searches



Bastert *et al.*, 2002

- How would you implement BFS?
  - Inputs: A graph `G`, a `start` and an `end`.
  - Algorithm: For each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.
- Bookkeeping is important:
  - Keep a "To Do" list to visit and a dictionary of those already visited.

Implementing Searches



Bastert *et al.*, 2002

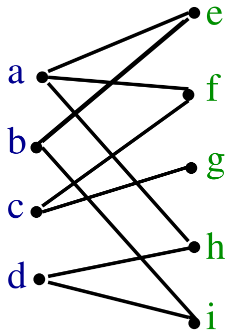
- How would you implement BFS?
  - Inputs: A graph `G`, a `start` and an `end`.
  - Algorithm: For each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.
- Bookkeeping is important:
  - Keep a "To Do" list to visit and a dictionary of those already visited.

```
toDo = [start]
```

```
visited = {}
```

  - Mark nodes as you visit them and add neighbors to the To Do list:

Implementing Searches



Bastert *et al.*, 2002

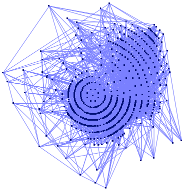
- How would you implement BFS?
  - Inputs: A graph `G`, a `start` and an `end`.
  - Algorithm: For each neighbor, visit its' neighbors, and continue as far down as possible, and repeat.
- Bookkeeping is important:
  - Keep a "To Do" list to visit and a dictionary of those already visited.

    ```
    toDo = [start]

    visited = {}
    ```
  - Mark nodes as you visit them and add neighbors to the To Do list:

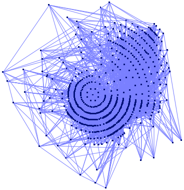
    ```
    while len(toDo) > 0:
        nextNode = toDo.pop(0)
        visited[nextNode] = 1
        For n unvisited neighbor of nextNode,

            toDo.append(n)
    ```

Application: Which Tree is Optimal?

Given a set of organisms, which tree is optimal?



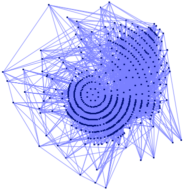
Application: Which Tree is Optimal?



Given a set of organisms, which tree is optimal?

- Two standard criteria for optimality:

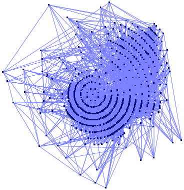
Application: Which Tree is Optimal?



Given a set of organisms, which tree is optimal?

- Two standard criteria for optimality:
  - Maximum Parsimony: find tree with fewest changes

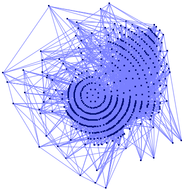
Application: Which Tree is Optimal?



Given a set of organisms, which tree is optimal?

- Two standard criteria for optimality:
  - Maximum Parsimony: find tree with fewest changes
  - Maximum Likelihood: find most likely tree (with respect to a model of evolution)

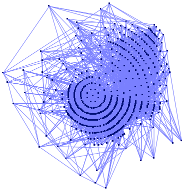
Application: Which Tree is Optimal?



Given a set of organisms, which tree is optimal?

- Two standard criteria for optimality:
  - Maximum Parsimony: find tree with fewest changes. (NP-hard, Foulds & Graham, 1982).
  - Maximum Likelihood: find most likely tree (with respect to a model of evolution) (NP-hard, Roch, 2008).

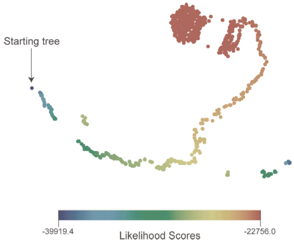
Application: Which Tree is Optimal?



Given a set of organisms, which tree is optimal?

- Two standard criteria for optimality:
  - Maximum Parsimony: find tree with fewest changes. (NP-hard, Foulds & Graham, 1982).
  - Maximum Likelihood: find most likely tree (with respect to a model of evolution) (NP-hard, Roch, 2008).
- Later this lecture, we will define (and write pseudocode) for the maximum parsimony criteria.

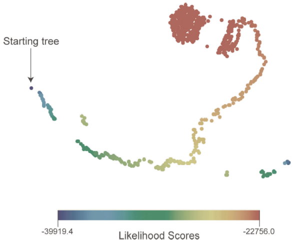
Searching for Optimal Trees



Hillis, Heath, S, 2005

- **Goal:** Find the tree with the optimal score

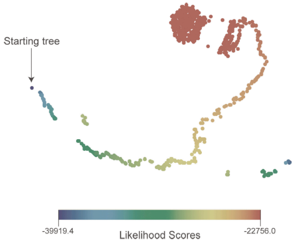
Searching for Optimal Trees



Hillis, Heath, S, 2005

- **Goal:** Find the tree with the optimal score
- Local search techniques prevail:

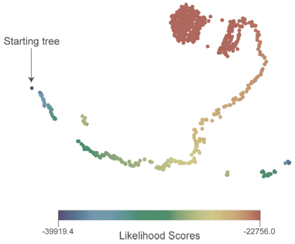
Searching for Optimal Trees



Hillis, Heath, S, 2005

- **Goal:** Find the tree with the optimal score
- Local search techniques prevail:
  - Begin with a tree

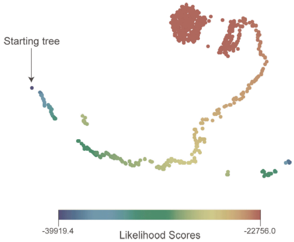
Searching for Optimal Trees



Hillis, Heath, S, 2005

- **Goal:** Find the tree with the optimal score
- Local search techniques prevail:
  - Begin with a tree
  - Choose the next tree from its neighbor (e.g. best scoring)

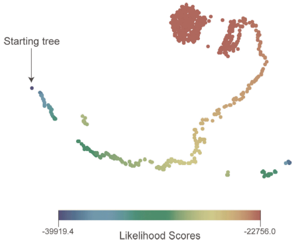
Searching for Optimal Trees



Hillis, Heath, S, 2005

- **Goal:** Find the tree with the optimal score
- Local search techniques prevail:
  - Begin with a tree
  - Choose the next tree from its neighbor (e.g. best scoring)
  - Repeat

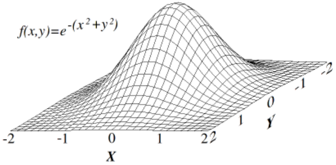
Searching for Optimal Trees



Hillis, Heath, S, 2005

- **Goal:** Find the tree with the optimal score
- Local search techniques prevail:
  - Begin with a tree
  - Choose the next tree from its neighbor (e.g. best scoring)
  - Repeat
- Many variations on the theme: branch-and-bound, MCMC, genetic algorithms,...

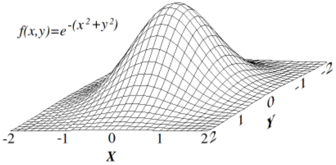
Hill Climbing



wiki

- Local Search is often Hill Climbing: at every step, you move 'up hill'.

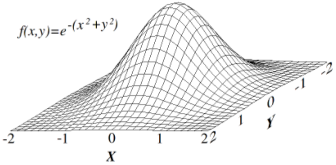
Hill Climbing



wiki

- Local Search is often Hill Climbing: at every step, you move 'up hill'.
- Works well if there is a single hill.

Hill Climbing



wiki

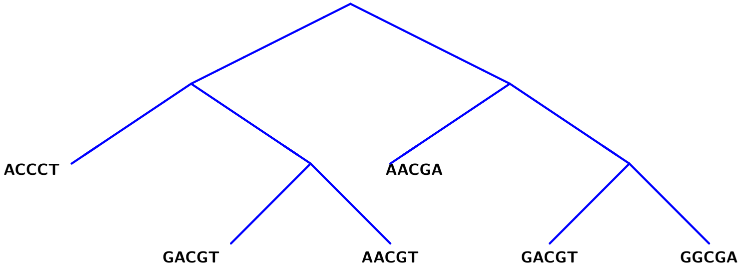
- Local Search is often Hill Climbing: at every step, you move 'up hill'.
- Works well if there is a single hill.
- If there are multiple hills, could get stuck.

Maximum Parsimony

- Find the tree that can explain the observed sequences with a minimal number of substitutions.

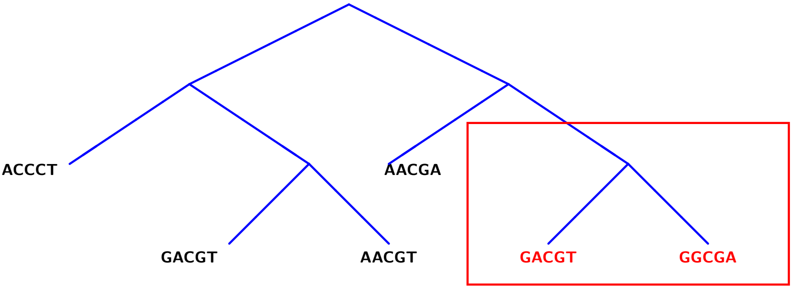
Maximum Parsimony

- Find the tree that can explain the observed sequences with a minimal number of substitutions.
- Given sequences for leaves and a tree, first measure "minimal number of substitutions."



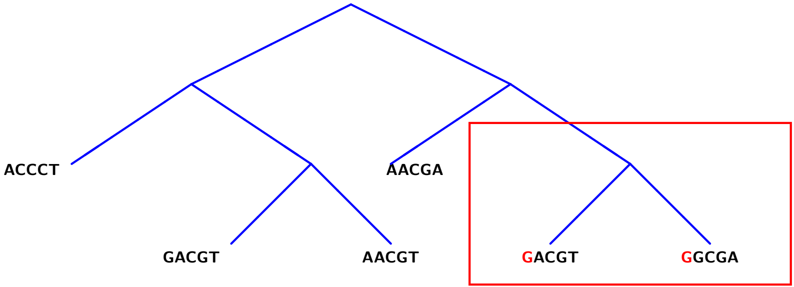
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



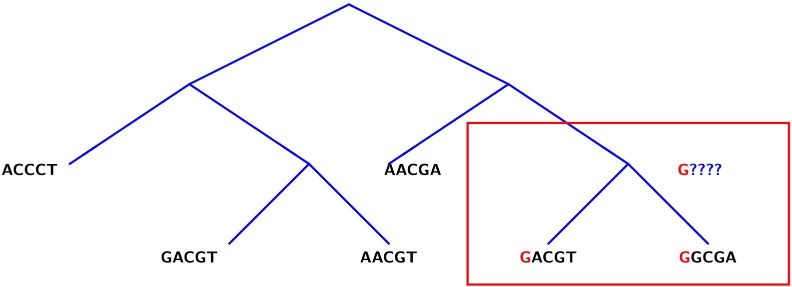
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



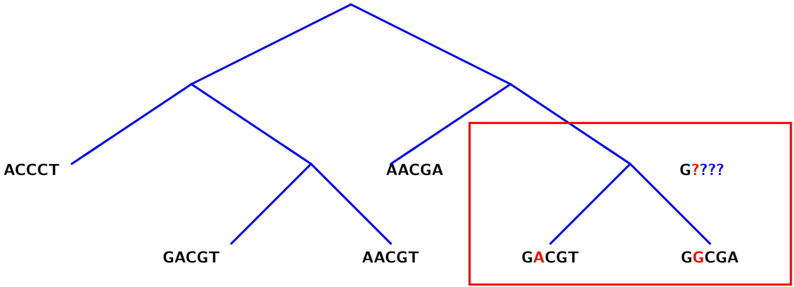
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



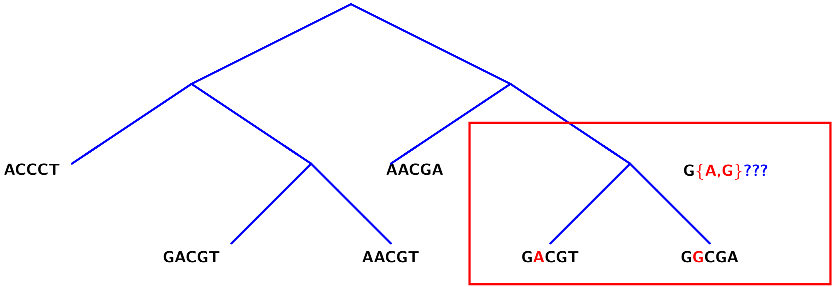
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



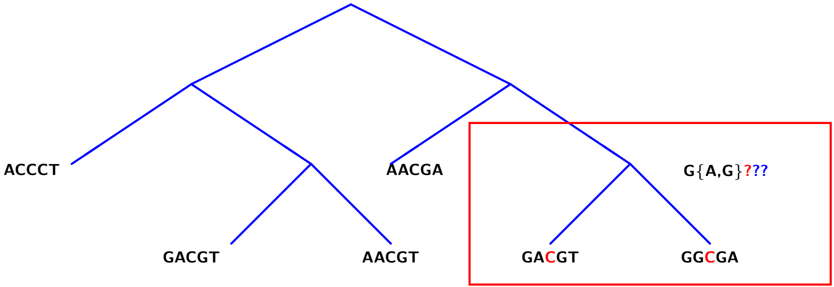
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



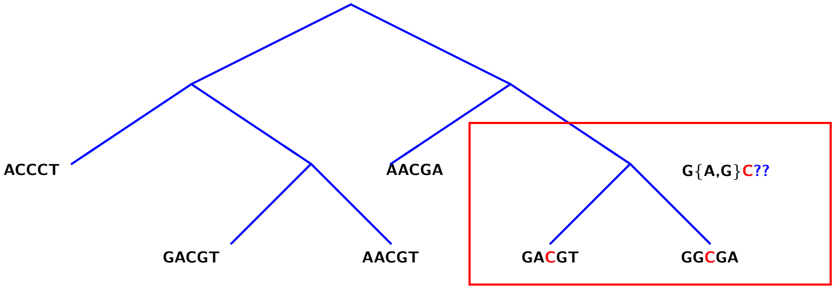
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



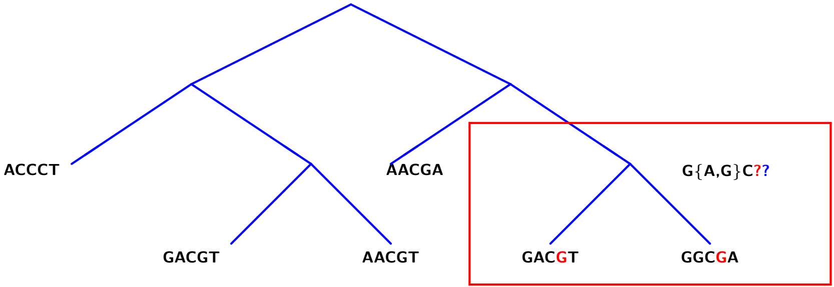
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



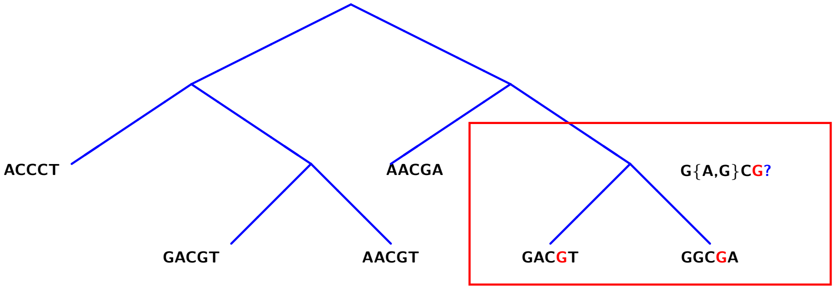
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



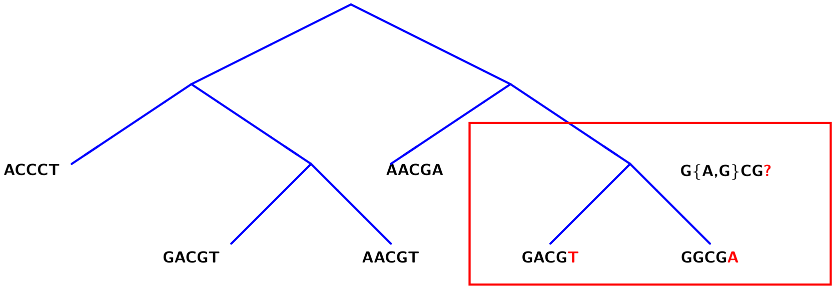
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



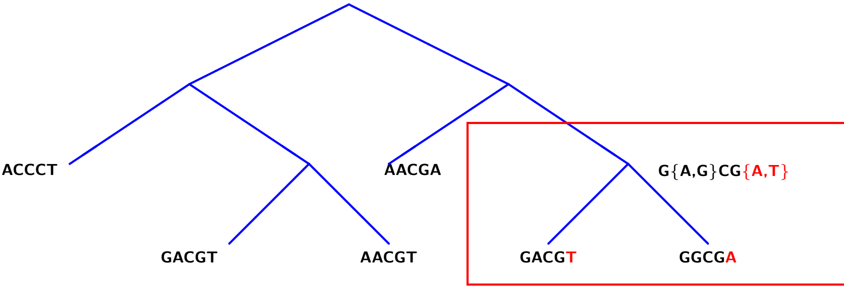
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



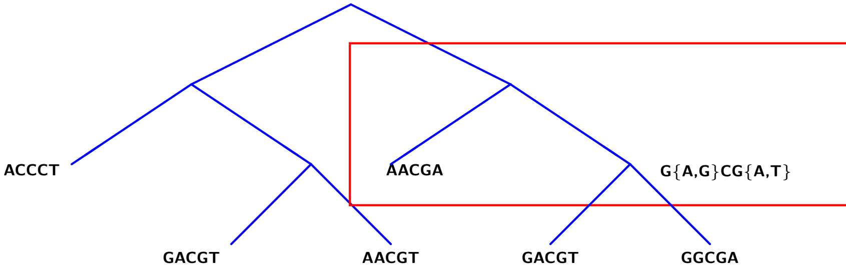
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



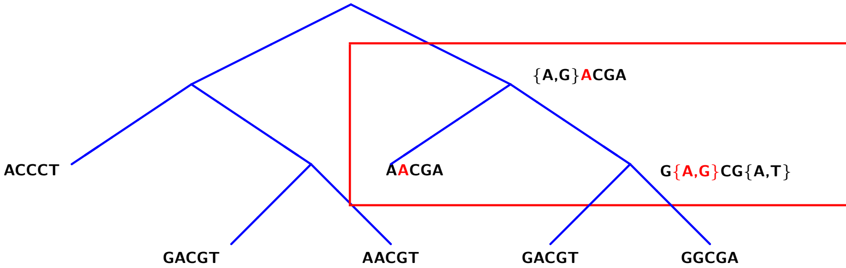
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



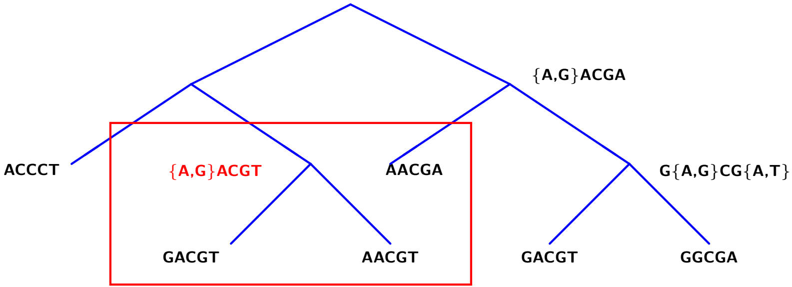
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



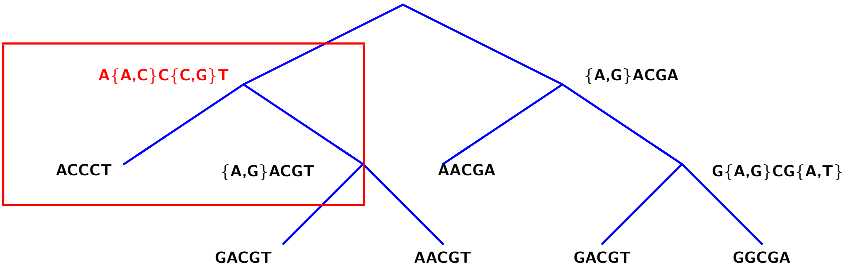
Maximum Parsimony

- Given sequences for leaves and a tree, first measure “minimal number of substitutions.”
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.

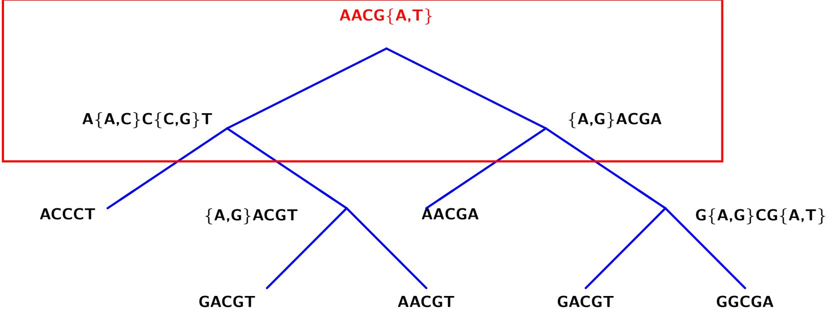


Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.

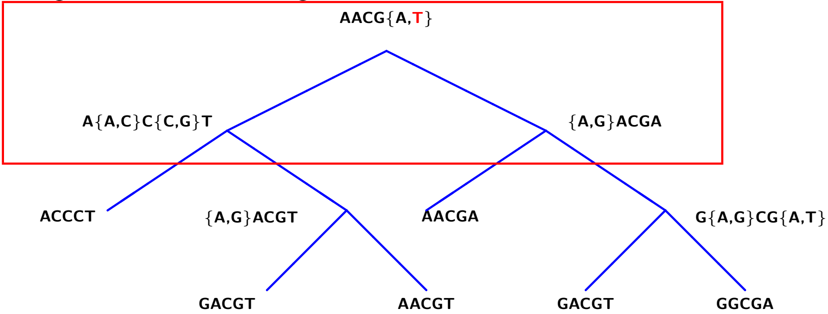


- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



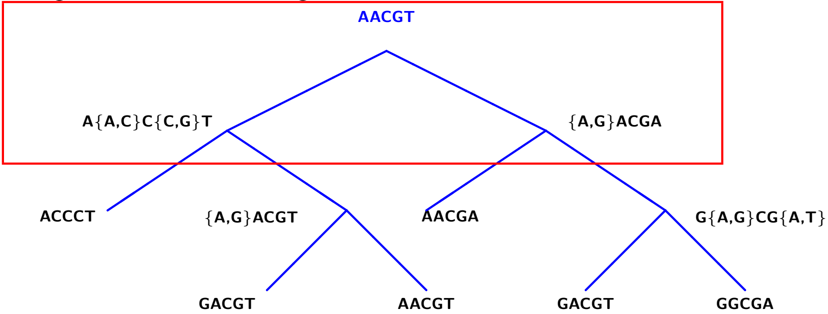
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



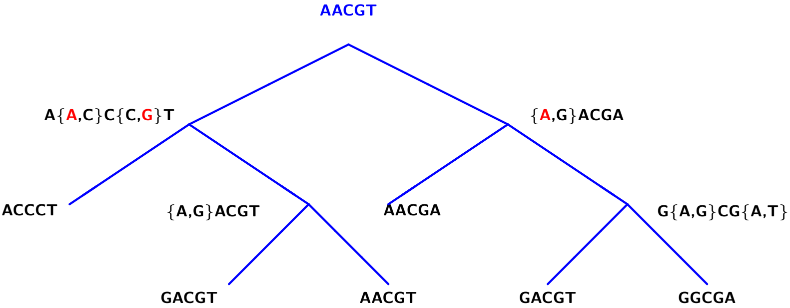
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



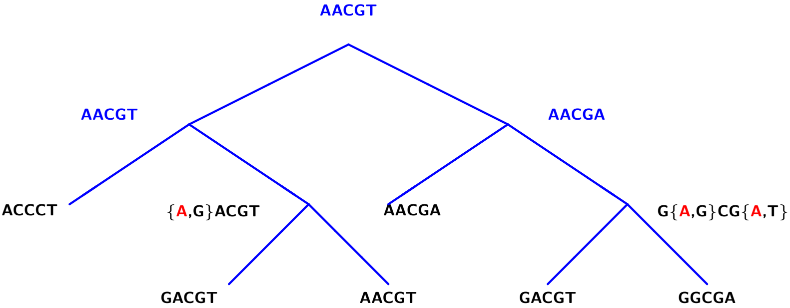
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



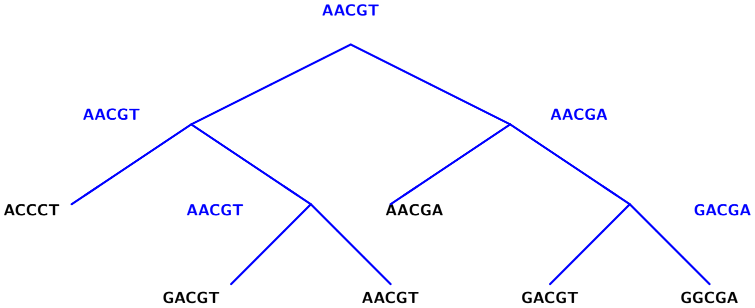
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



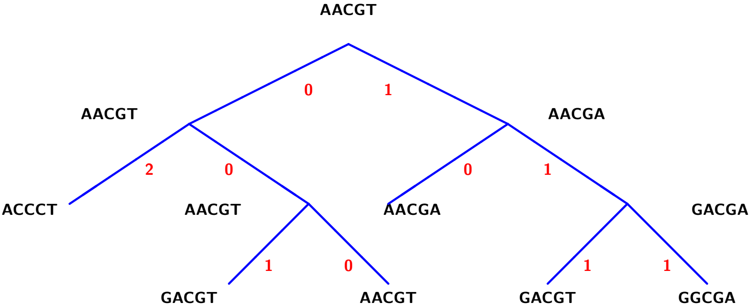
Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



Maximum Parsimony

- Given sequences for leaves and a tree, first measure "minimal number of substitutions."
- Label the internal nodes with sequences that have minimal number of changes. Then count changes.



Total change, called the parsimony score is 7.

Maximum Parsimony

- Given sequences for leaves, find tree with minimal parsimony score:

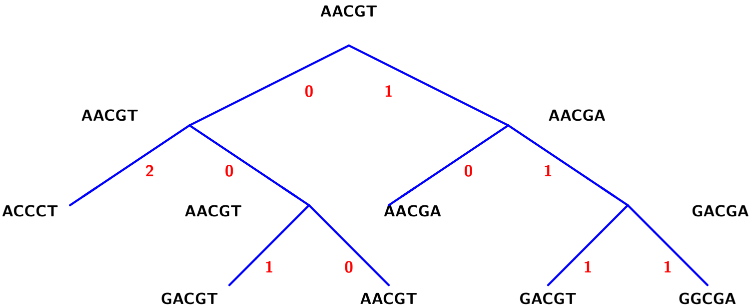
ACCCT **AACGA**

GACGT **AACGT** **GACGT** **GGCGA**

(Can you find a tree with a score better than 7?)

In Pairs

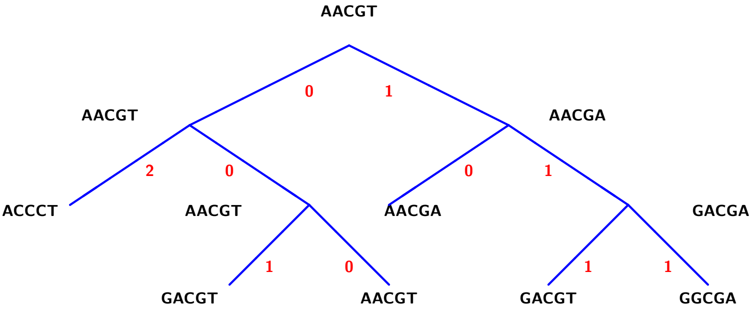
Label the internal nodes with sequences that have minimal number of changes. Then count changes.



1. Find a better scoring tree.

In Pairs

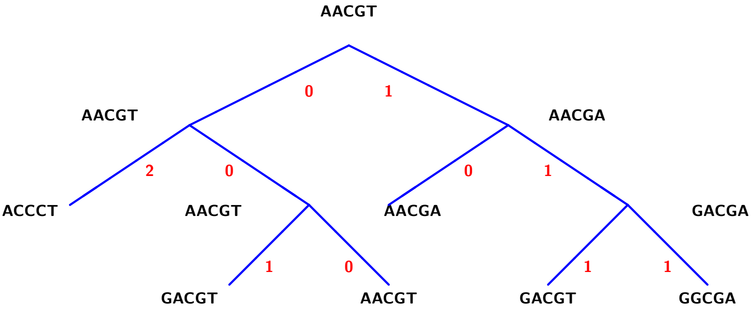
Label the internal nodes with sequences that have minimal number of changes. Then count changes.



1. Find a better scoring tree.
2. Find the best scoring tree.

In Pairs

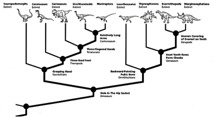
Label the internal nodes with sequences that have minimal number of changes. Then count changes.



1. Find a better scoring tree.
2. Find the best scoring tree.
3. If all the leaves (tips) were labelled by "AAAAA", what is the best scoring tree?

Algorithm Design: Scoring Trees Under Parsimony

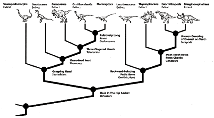
- How do you code this?



AMNH

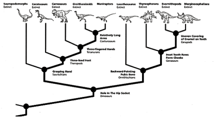
Algorithm Design: Scoring Trees Under Parsimony

- How do you code this?
  - Inputs: A tree and sequences on the leaves.



AMNH

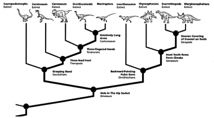
Algorithm Design: Scoring Trees Under Parsimony



AMNH

- How do you code this?
  - Inputs: A tree and sequences on the leaves.
  - Output: The parsimony score of the tree (with respect to the leaf labels).

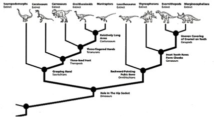
Algorithm Design: Scoring Trees Under Parsimony



AMNH

- How do you code this?
  - Inputs: A tree and sequences on the leaves.
  - Output: The parsimony score of the tree (with respect to the leaf labels).
- What data structures do you need?

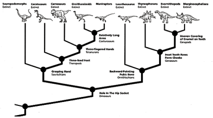
Algorithm Design: Scoring Trees Under Parsimony



AMNH

- How do you code this?
  - Inputs: A tree and sequences on the leaves.
  - Output: The parsimony score of the tree (with respect to the leaf labels).
- What data structures do you need?
  - Tree structure

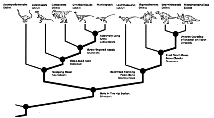
Algorithm Design: Scoring Trees Under Parsimony



AMNH

- How do you code this?
  - Inputs: A tree and sequences on the leaves.
  - Output: The parsimony score of the tree (with respect to the leaf labels).
- What data structures do you need?
  - Tree structure
  - Count of the number changes

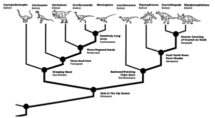
Algorithm Design: Scoring Trees Under Parsimony



AMNH

- How do you code this?
  - Inputs: A tree and sequences on the leaves.
  - Output: The parsimony score of the tree (with respect to the leaf labels).
- What data structures do you need?
  - Tree structure
  - Count of the number changes
- Algorithm:
  - First pass: Starting at the leaves, label the internal leaves (with possible multiple labels).

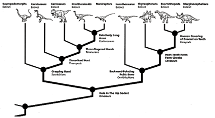
Algorithm Design: Scoring Trees Under Parsimony



AMNH

- How do you code this?
  - Inputs: A tree and sequences on the leaves.
  - Output: The parsimony score of the tree (with respect to the leaf labels).
- What data structures do you need?
  - Tree structure
  - Count of the number changes
- Algorithm:
  - First pass: Starting at the leaves, label the internal leaves (with possible multiple labels).
  - Second pass: Starting at the root, choose a labeling, then work towards the leaves minimizing the conflicts.

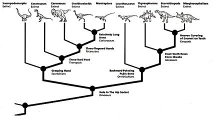
Algorithm Design: Scoring Trees Under Parsimony



AMNH

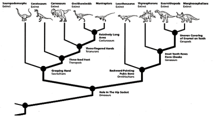
- How do you code this?
  - Inputs: A tree and sequences on the leaves.
  - Output: The parsimony score of the tree (with respect to the leaf labels).
- What data structures do you need?
  - Tree structure
  - Count of the number changes
- Algorithm:
  - First pass: Starting at the leaves, label the internal leaves (with possible multiple labels).
  - Second pass: Starting at the root, choose a labeling, then work towards the leaves minimizing the conflicts.
- *If time, write out pseudocode...*

Recap

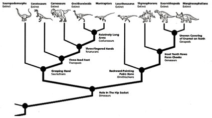


- Searching Graphs: Breadth-First & Depth-First Searches

Recap

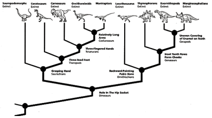


- Searching Graphs: Breadth-First & Depth-First Searches
- Hill Climbing



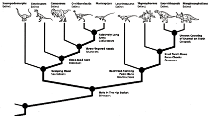
- Searching Graphs: Breadth-First & Depth-First Searches
- Hill Climbing
- Optimality Criteria: Parsimony Example

Recap



- Searching Graphs: Breadth-First & Depth-First Searches
- Hill Climbing
- Optimality Criteria: Parsimony Example
- Email lab reports to `kstjohn@amnh.org`

Recap



- Searching Graphs: Breadth-First & Depth-First Searches
- Hill Climbing
- Optimality Criteria: Parsimony Example
- Email lab reports to `kstjohn@amnh.org`
- Challenges available at `rosalind.info`