Implementing Phylogeny Trees using Binary Trees in C++

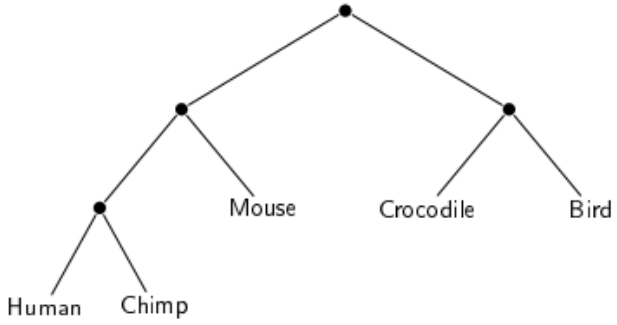
BMS 328, Spring 20

Muhammad Salah Elsadany
Computational Biology and Genomics, Biomedical Sciences Program
University of Science and Technology, Zewail City
Giza, Egypt
s-muhammadelsadany@zewailcity.edu.eg

1. Introduction

Evolution could be defined as a change in the characteristics of a species over several generations and that is based on a natural selection process. The evolution theory is based on the idea that all species gradually change over time and thus are related [1]. Evolution theory is one of the key elements of modern biological science. The life-world diversity is staggering. It has been identified and described that more than 2 million existing species of organisms and many more remain to be discovered [1]. All living organisms are related to common ancestors by descent. Human and other mammals are descended from shrewd beings that lived over 150 million years ago [1]. Lineages of organisms change over generations; diversity arises because lines coming down from common ancestors diverge over time. Evolution theory by Charles Darwin states that evolution occurs through natural selection [2]. Individuals with the features particularly well suited to their environment are more likely to survive, find food, avoid predators and resist disease [2]. Those are more likely to replicate and pass on their genes to their offspring.

Phylogenetic trees, or evolutionary trees, demonstrate the evolutionary relationships between multiple species seen as tips of the tree. Usually these would be species that live or survive. Shown below an example for a simple phylogenetic tree:



The branching pattern in a phylogenetic tree represents the evolution of organisms or other groups from a variety of common ancestors. The closer the two organisms are in the graph, the relatively close they are to each other in genetic relationships. For example in the previous figure, humans are closer to chimpanzees rather than to birds, as humans and chimps have a more recent common ancestor [4].

2. Problem Definition

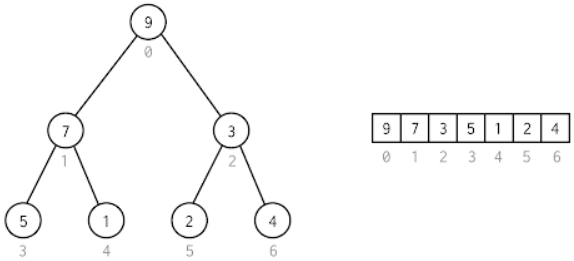
Trees provide an efficient structure for the organization of biodiversity knowledge and enable the development of an accurate, non-progressive understanding of the entire history of evolution. For building a phylogenetic tree,

scientists compare and evaluate several characteristics of the organisms or other groups involved [3]. Phylogenetics is significant, as it improves our understanding of how genes, genomes, organisms evolve. The reason for researching reconstruction of the phylogenetic tree is that because it applies to biology where it is used for the development of new vaccines, antibacterial and herbicide molecules, and for the study of microbial dynamics.

In this proposed algorithm, phylogenetic relationships will be applied using trees data structure for further analysis needed. It is important to know how computational tools can save a lot of work needed for deriving information. Thus in this program after building the user's phylogeny tree, the program will give the user a list consisting of all needed analysis for his entered tree. That analysis list includes: depth, tree root, and ancestors that do not have recently evolved generations.

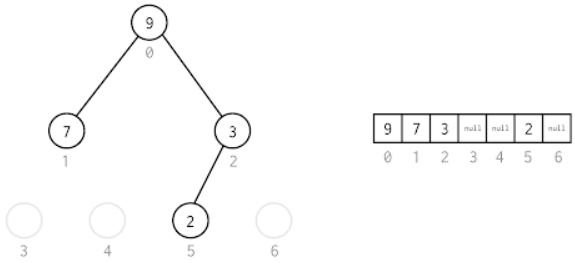
3. Related Work and Survey

Phylogeny trees have been implemented through different data structures in C++ including both linked lists and arrays. Although they can't represent a visual representation of phylogeny trees, they can be read and understood. These data structures do implement phylogeny trees using different orders for reading a tree like the 3 known traversal orders for trees: preorder, inorder, and postorder. In this project, the proposed tree algorithm will be compared with array implementation. The method used for array implementation of phylogeny trees will be following the same as described in the below figure:



The root of the tree will be at position 1 of the array and nothing will be added at position 0. The left child of any node at position n in the array is at position 2n. On the other hand, the right child of a node at position n is at position 2n + 1. By the same principle, the position of the parent of any node at position n is found at position n/2 without taking decimals into consideration.

However, if there is a node that has no children, its children positions will be empty in the array and the array elements will be fitted exactly as described in the previous part. The figure below illustrates how this could be represented:



4. Proposed Method

The proposed algorithm in this project is implementing phylogeny trees using pointer-based binary trees. This method was chosen based on some facts that include the space

chosen, and complexity analysis that will be illustrated more in the Evaluation section. The algorithm lies mainly on the entered distance matrix by the user. Distance matrix is a table that shows the distance between pairs of objects that represent species in our case. An example of a distance matrix that will be implemented in this project is as follows: [5]

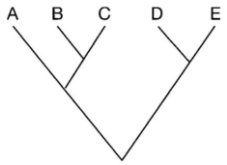
	A	B	C	D	E	F	G
A							
B	19.00						
C	27.00	31.00					
D	8.00	18.00	26.00				
E	33.00	36.00	41.00	31.00			
F	18.00	**1.00**	32.00	17.00	35.00		
G	13.00	13.00	29.00	14.00	28.00	12.00	

The first part of the algorithm is implementing the UPGMA algorithm (Unweighted Pair Group Method with Arithmetic Mean Algorithm). The UPGMA algorithm produces a rooted tree unlike other methods using distance matrix like Neighbor-Joining Method.

Following is the pseudocode for UPGMA algorithm that uses distance matrix as an input:

1. Find closest pair in distance matrix
    a. Returns indices i and j that are closest
2. Update Distance Matrix
    a. Calculate distances from new node to nodes that have parent following this formula:
    dist(AB),C = (distAC + distBC) / 2
    Assuming that A and B were the closest in the first step.
3. Update the labeling of the matrix
    a. By joining the labels of the 2-closest species in a bracketing format. e.g. (A,B).

After implementing the UPGMA algorithm, it should give a newick format for tree construction. It's a way of representing graph-theoretical trees using parentheses and commas. For example, the figure below shows a tree for the newick format, (((B,C),A),(D,E)).



(((B,C),A),(D,E))

Each parentheses represent a node and its children are separated by a comma inside.

By getting the newick format, the second algorithm part starts building a binary tree class for creating a tree data structure to build the phylogeny tree. The time complexity of binary trees will be compared with arrays complexity in the evaluation part to give advantages and disadvantages of each way.

The following is a description for all functions implemented in both algorithm parts:

- lowestcell(vector<vector<int>> table, int &x, int &y): Locates the smallest cell in the entered distance matrix
- join_labels(vector<string>&labels, int a, int b): Combines two labels in a list of labels
- join_table (vector<vector<int>>&table, int &a, int&b): Joins the entries of a table on the cell (a, b) by averaging their data entries
- UPGMA(vector<vector<int>>table, vector<string>labels): Runs the UPGMA algorithm on a labelled matrix
- Node* newNode(char data): allocates a new node with the given data and NULL left and right pointers.
- printLeafNodes(Node* root): function to print leaf nodes from left to right
- maxDepth(Node* node): Compute the "maxDepth" of a tree -- the number of

nodes along the longest path from the root node down to the farthest leaf node
- createtree(Node* tree, string word, bool left): creates a binary tree from a newick format in a recursive way
- print_tree_structure(Node *root, int spaces): to visualze the built tree structure

5. Evaluation

The data set used for experimenting the proposed algorithm was retrieved from [5]. The distance matrix from [5] was used to test the UPGMA algorithm and the output of the first part was:

```
your tree structure in a newick format is: ((((A,D),((B,F),G)),C),E)
```

The result was compared to the original results of the site and they were 100% accurate. The entered distance matrix was a 7*7 matrix of a lower triangle type. The entered labels of species were only letters [A, B, C, D, E, F, and G]. These letters represent different species as illustrated more in details in [5].

The questions the algorithm should answer are:

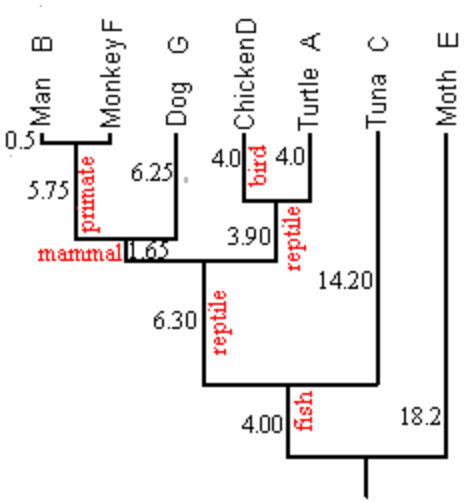
1. What is the phylogeny tree structure?
2. What is the tree depth?
3. What are the leaves?
4. What is the tree root?
5. What is the algorithm complexity?

The algorithm already answered the 4th question by the UPGMA part output. That root represents an ancestor for all entered species.

For the phylogeny tree structure, the algorithm gave the output described in the figure below:

```
the following is the visualization for your final tree
   ----
           ----E
       ----
                ----C
           ----
                          ----G
                    ----
                               ----F
                          ----
                               ----B
                ----
                          ----D
                    ----
                          ----A
```

That result was compared to the constructed tree in [5] which is:



And the results were identical in the number of nodes created, and the evolving ancestors. Although they differ in the order of nodes, but that has no effect on the evolutionary context of the tree which is exactly the same in both of them.

For the second and third questions representing tree depth and tree leaves, the results of the algorithm were exactly as expected:

```
Height of tree is 7
your tree leaves are: A D B F G C E
```

The algorithm complexity of pointer-based binary tree was compared to that of an array and the results were as follows:

	Array	Binary Tree
Space	O(n)	O(n)
Search	O(n)	O(n)
Add	O(1)	O(n)
Delete	O(n)	O(n)

Problems met during the implementation of this algorithm:

- Merging labels in the UPGMA part.
- Parsing the newick format in createtree function and doing that recursively.
- Building a function that can print the tree in a simple visualized output in print_tree_structure function.

Assumptions include:

1. The species entered are only characters not full strings.
2. The user already knows it's a binary tree that only can have 2 children at maximum for each node.
3. The user enters a lower triangle squared distance matrix

6. Conclusion and Discussion

Based on the comparison part of both algorithms in the Evaluation section, it shows that arrays could be better only in the addition of any new child. This is always the case in adding new item in any array and in our case it's much simpler as we already know where we can put a child of node at position n as described before. However, trees can prove their success in other functions and analysis that could be done on any phylogeny tree. That includes all functions implemented in the proposed algorithm like getting the tree depth, printing leaves, and simple visualization of the tree. After all, the Evaluation section proved how the proposed algorithm fulfilled its aim and answered all research questions needed.

7. Future work

In order to deal with the complexity part, we can deal with less complexity by implementing our tree with Binary Search Trees (BST). In other words, Instead of representing our species with letter symbols, we can represent them by integers and build our tree. In this way, the time complexity will definitely change because we're dealing with numbers that can be easily looked for. Following is a comparison between the 3 algorithms time complexity:

	Array	Binary Tree	Binary Search Tree
Space	O(n)	O(n)	O(n)
Search	O(n)	O(n)	O(log n)
Add	O(1)	O(n)	O(log n)
Delete	O(n)	O(n)	O(log n)

8. References

[1]"evolution | Theory, Examples, & Facts", Encyclopedia Britannica, 2020. [Online]. Available: https://www.britannica.com/science/evolution-scientific-theory.

[2]"What is evolution?", Yourgenome.org, 2020. [Online]. Available: https://www.yourgenome.org/facts/what-is-evolution.

[3]"Why is phylogenetics important?", EMBL-EBI Train online, 2020. [Online]. Available: https://www.ebi.ac.uk/training/online/course/introduction-phylogenetics/why-phylogenetics-important.

[4]"Phylogenetic Trees", Biology 1520, 2020. [Online]. Available: http://bio1520.biology.gatech.edu/biodiversity/phylogenetic-trees/.

[5]"Calculation of Phylogeny: the UPGMA Clustering Method", Nmsr.org, 2020. [Online]. Available: http://www.nmsr.org/upgma.htm.