The Application of Weighted Tree in the Construction of DNA Sequence-based Phylogenetic Tree Using Neighbour-Joining Algorithm

Naufal Muhammad Alif - 10422039[1]
Program Studi Mikrobiologi
Sekolah Ilmu dan Teknologi Hayati
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]10422039@mahasiswa.itb.ac.id, naufal.alifkun@gmail.com

Abstract*—Phylogenetic analysis is an essential part of bioinformatics. It provides a framework for understanding evolutionary relationships among organisms, genes, proteins, and other biological structures. Phylogenetic trees were originally constructed based on morphological and physiological characteristics of organisms. The discovery of DNA structure in 1953 shifted the approach to DNA-based method, allowing a more robust and quantifiable method for constructing the tree. This paper introduces the theoretical foundations of phylogenetic trees, including the mathematical concepts of graphs, trees, and weighted trees, as well as some basics of evolutionary distance models. Neighbour-joining algorithm (NJ) is one of the algorithms used to construct the DNA-based phylogenetic trees, known for its simplicity and heuristical approach to predicting phylogenetic trees. This paper explains the step-by-step process of the NJ algorithm. To deepen the understanding of NJ, a dummy program was developed in Python and tested using two distinct DNA alignment test files. The results were compared to reference trees computed from the same datasets using MEGA11. The comparison demonstrated comparable tree topologies but showed significant differences in calculated branch length (w), particularly at root child nodes. The differences are suspected to be caused by a more advanced NJ algorithm and the difference in the evolutionary distance model used by MEGA11. The results highlight areas for improvement in the dummy program, including enhancements in outgroup selection and branch length computation, to achieve greater accuracy and alignment with established tools like MEGA11.

***Keywords*—graph, weighted tree, phylogenetic tree, neighbour-joining algorithm**

I. Introduction

Phylogeny is one of the most essential bioinformatics tools. Phylogeny is defined as a model of the evolutionary relationships between organisms, as well as other structures such as genes and proteins. Phylogenetic tree is a tool to visualize phylogeny among organisims and biological structures of interest. The concept of phylogenetic trees dates back to 1809, when Lamarck presented the first evolutionary tree that described the connection between various groups of animals. In 1859, Charles Darwin utilised a hierarchical tree-like structure to illustrate the variation, selection, and common ancestry of organisms. The concept was later implemented by Ernst Haeckel to construct the “Tree of Life” in 1875 [1].

At first, morphological and physiological characters were the basis for classifying and investigating the evolutionary relationships among organisms in phylogenetics. However, this method has been proved to be unreliable in estimating evolutionary distance between organisms, since some might have a similar feature while having distant ancestors due to convergent evolution. Therefore, a more robust reference structure need to be used [2].

DNA (deoxyribonucleic acid) is a structure that stores the genetic information of a cell. DNA is a sequence of nucleic acids that stores information by acting like a string of binary code in a computer. DNA is unique from cell to cell and the divergence is proportional to the evolutionary distance between the cells. The discovery of DNA structure by Watson, Crick, and Franklin in 1953 opened the door to using DNA as a robust and quantifiable basis for phylogenetic tree construction [2].

The main idea of DNA sequence-based phylogenetic tree construction is comparing a set of homologous DNA sequences of the target organisms by calculating their pairwise Hamming or Levenshtein distances and constructing a mathematical tree so that a pair of organisms with the least divergence share a common parent node, representing a common ancestor. Some methods produce a weighted tree, in which the weight represents evolutionary distance. Several algorithms have been developed to compute phylogenetic trees from DNA sequences. One of them is the neighbour-joining algorithm [1].

This paper consists of an introduction, followed by a theoretical explanation of weighted tree that underlies the DNA-based construction of phylogenetic tree and the framework of the neighbour-joining algorithm. A dummy phylogenetic tree construction program was created in Python to deepen the understanding towards the algorithm. The development of the dummy program, as well as the comparison of its output and reference trees created using MEGA11 software is presented in the “Method” and “Result” sections.

II. Theory

A. Graph, Tree, and Weighted Tree

A *graph* $G = (V, E)$ consists of sets V and E (Fig. 1). V is a nonempty set of vertices (nodes) and E is a set of edges. Each edge has either one or two vertices *connected* to it, called its endpoints. A graph is said to be *undirected* if its edges have no direction, in other words, the pair of endpoints of each edge is unordered. Two vertices v_1 and v_2 in an undirected graph G are called *adjacent* if v_1 and v_2 are endpoints of an edge e_1 of G. In that case, the edge e_1 is called *incident with* the vertices v_1 and v_2. The *degree* of a vertex in an undirected graph in the number of edges incident with it, denoted by deg(v). A graph is *connected* if it has no isolated vertex (a vertex with degree of zero). A *simple graph* is a graph in which each of its edges connects two different vertices, where no edges connect the same pair of vertices. A *subgraph* of a graph $G = (V,E)$ is a graph $G' = (V',E')$, so that $V' \subseteq V$ and $E' \subseteq E$. *Path* in a graph is a sequence of edges that begins at a vertex and travels from one to the other along the edges of the graph. A path is a *simple circuit* if it begins and ends at the same vertex without traversing the same vertex more than once. A *spanning subgraph* of a graph G contains all vertices in G, and all edges incident with them. Two more simple graphs are isomorphic if there is a one-to-one correspondence between vertices of the graphs that preserves the adjacent relationship [3].

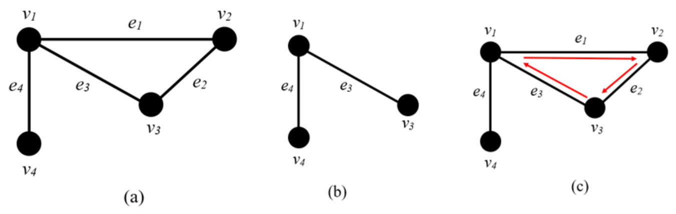


Fig 1. (a) A simple undirected graph G, (b) A subgraph of G, (c) A circuit from v_1 to v_2, v_3, and back to v_1

Tree is a connected, undirected graph that has no simple circuit. A *rooted tree* is a tree that has a vertex on the highest level which is designated as the root in and every edge is directed away from it (Fig. 2a). A *parent* vertex has an incident edge directed towards a *child* vertex in the level beneath. Vertices that share a common parent are called *siblings*. The *ancestors* of a vertex are vertices in the path from the root to the vertex. The *descendants* of a vertex v are vertices that have v as their ancestor. A vertex in that has no children is called a *leaf*, while one that has children is called an *internal vertex* (*internal node*). A *weighted tree* has values assigned to its edges that might represent distance, multiplier, bias, etc. Let a graph $G = (V,E)$, a *spanning tree* of G is a spanning subgraph of G that forms a tree. A graph can contain more than one spanning tree.

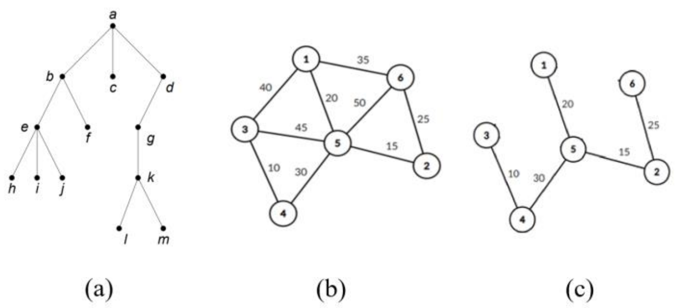


Fig 2. (a) A rooted tree with vertex a as its root, (b) A weighted graph G, and (c) The minimum spanning tree of G (b) [4], [5]

The *minimum spanning tree* of G is the spanning tree with the minimum total weight among all possible spanning trees (Fig. 2c). A rooted tree is called an *n-ary* tree if every internal vertex has no more than n children. In this case, if $n = 2$, the tree is called a *binary tree* [3].

B. Phylogenetic Tree

A Phylogenetic tree is a graphical representation of phylogeny between specified organisms, taxa, or biological structures (*operational taxonomic unit*, *OTU*). Each OTU is represented as a leaf. Each internal node represents a speciation event in the evolution or a hypothetical common ancestor of all of its descendants. Mathematically, a phylogenetic tree is a rooted weighted tree, with the weight of each edge as a measure of the evolutionary distance between an internal node or an OTU to its parent node (evolutional ancestor). The tree is based on an *outgroup*, an OTU that is the most distant from all other OTUs in the tree. Note that the term "node" is more commonly used in phylogenetics instead of "vertex".

A phylogenetic tree can either be in a radial form or a dendrogram form. On the dendrogram form, all edges from the radial tree are represented vertically, and the horizontal branches show the relantionships between OTUs. Some dendrogram shows evolutionary distance (weight of each edge) through the difference in length of each horizontal branch.

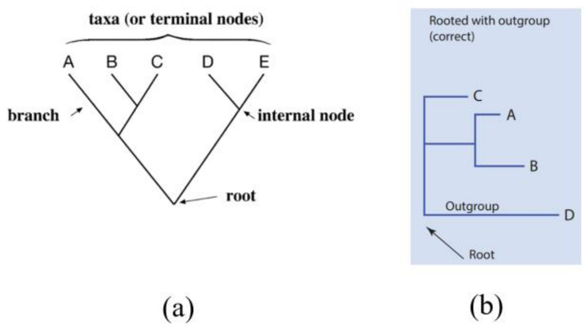


Fig 3. (a) Part of a phylogenetic tree [6], and (b) Outgroup as the base of a phylogenetic tree [1]

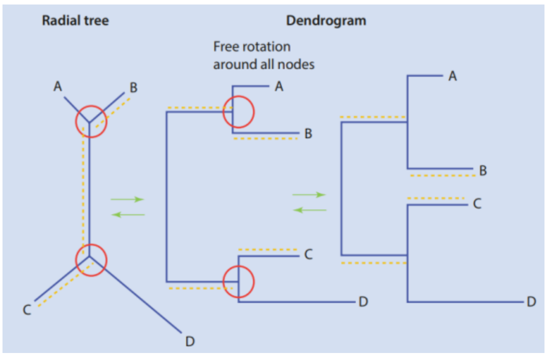


Fig 4. Radial form (left) and dendrogram form (right) of phylogenetic tree. The length of branches highlighted by yellow dashed lines represents evolutionary distances between internal nodes and OTUs. Note that all of the trees are isomorphic [1].

C. DNA Sequence Distance

From a biological perspective, DNA is a polymer of nucleotides that stores genetic information. There are four types of nucleotides, adenine (A), thymine (T), guanine (G), and cytosine (C). They are further grouped into purine (A and T) and pyrimidine (G and C) based on their chemical properties. These nucleotides are analogous to 0 and 1 in binary language used in computers. Information stored in the DNA acts as a master plan for the cell to build proteins. The structure of proteins determines the structure and functional capability of the cells. An important property of DNA is that it is conserved when a cell divides. Hence, DNA of cells in multicellular organisms or microbial cells in a culture are almost completely identical.

So if DNA is conserved when cells are dividing, how are there millions of species on Earth? Despite being conserved, DNA sequence is unique from one organism to the other. According to the theory of evolution, the variance emerges due to a random change in the nucleotide sequence called mutation. When the changes accumulate enough over time, a lineage of cells can be determined as a new species. These properties create a crucial concept in phylogenetics called *molecular clock*. It states that the evolutionary distance between two species can be determined by calculating the difference between their DNA sequences [2].

In the perspective of informatics, a DNA sequence can be portrayed as a string of characters "A", "T", "G", and "C" that represents each type of nucleotide. In aligned sequences, the character "-" is used to denote a gap in the alignment.

To measure the distance between a pair of DNA sequences, various models are available. In this study, the Kimura two-parameters model and the Jukes-Cantor model are used. The Kimura two-parameter model measures the distance based on the transition and transversion rate of characters (nucleotides) in the pair. *Transition* is the change from a purine nucleotide to another purine nucleotide (A↔T) or the change from a pyrimidine nucleotide to another pyrimidine nucleotide (G↔C). *Transversion* is the change from a purine nucleotide to a pyrimidine nucleotide, or vice versa ((A/T)↔(G/C)). In this model, the probability of transversion is assumed to be lower in the evolution based on the chemical properties and empirical observations. Hence, sequences with higher transversion rates will be considered more distantly apart. Distance (*d*) is formulated as follows, where *P* is the transition ratio, *Q* is the transversion ratio, and *l* is the sequence length (1). Note that the length of a pair of aligned sequences are equal. In this study, gap "-" will be ignored [7].

$$d = -\frac{1}{2}\ln(1 - 2P - Q)$$

$$P = \frac{\text{number of transitions}}{l}$$

$$Q = \frac{\text{number of transversions}}{l} \tag{1}$$

Jukes-Cantor model measures the distance based on mismatched characters between the pair. It assumes an equal probability of transition and transversion. Distance (*d*) is formulated as follows, where *P* is mismatch ratio and *l* is the sequence length (2) [8].

$$d = -\frac{3}{4}\ln\left(1 - \frac{4}{3}P\right)$$

$$P = \frac{\text{number of mismatches}}{l} \tag{2}$$

D. Neighbour-Joining Algorithm

Neighbour-joining algorithm (NJ) is a method for reconstructing phylogenetic trees from evolutionary distance data such as DNA sequences. The principle of NJ is to find a pair of OTUs with the minimum evolutionary distance (neighbour), cluster them together to a new parent node (joining), and repeat the process while considering the clustered OTUs as a single new OTU in each iteration. The goal is to create a tree with the least total branch length (weight) heuristically. The idea is somewhat similar to finding minimum a spanning tree from a graph of OTUs using algorithms such as Prim's algorithm and Kruskal's algorithm but with creating new nodes along the way. It is also somewhat similar to the construction of a Huffman tree whose goal is to create a maximum parsimonious dictionary of compression code while considering the constructed subtree as a new single node in the process.

The NJ algorithm starts with a star-like tree with leaves representing analysed OTUs (Fig 5a). The first step is to calculate a pairwise distance matrix (*D*) that describes the distances between each OTU, in this case, based on aligned DNA sequences. Since the matrix is symmetric to its primary diagonal and the primary diagonal will always be 0, only the half upper of the matrix needs to be calculated. In this study, the Kimura two-parameters model (1) and Jukes-Cantor model (2) are used [7], [8].

$$D_{i,j} = -\frac{1}{2}\ln(1 - 2P - Q) \tag{3}$$

$$D_{i,j} = -\frac{3}{4}\ln\left(1 - \frac{4}{3}P\right) \tag{4}$$

The next step is to calculate a total divergence vector (*R*) which consists the total divergence value of each OTUs. Total divergence of OTU *i* is the sum of distances between OTU *i* and every other OTUs *j*. Let there are *n* OTUs analysed, R_i is formulated as (3).

$$R_i = \sum_{j \neq i}^{n} D_{i,j} \tag{4}$$

Next, an adjusted distance matrix (*M*) is calculated based on the pairwise distance matrix *D* and total divergence vector *R* (4).

$$M_{i,j} = (n - 2)D_{i,j} - (R_i + R_j) \tag{5}$$

A pair of OTUs with the least adjusted distance is clustered by creating a new parent node (*u*) that only consists of the clustered pair as its children. This clustering will create a new subtree rooted in the new parent node. The branch length (weight of edge, *w*) connecting the new parent node with each of its children *i* and *j* is calculated as follows (5).

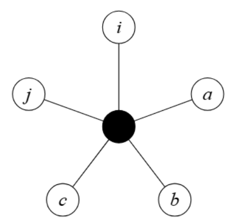
$$w_{i,u} = \frac{1}{2}(D_{i,j}) + \frac{1}{2(n-2)}(R_i - R_j)$$

$$w_{j,u} = D_{i,j} - w_{i,u} \tag{6}$$

An updated distance matrix (*D*) is calculated by considering the newly constructed subtree (cluster) as a single OTU (recall Huffman tree construction) (Fig 5b). Therefore, now there are $n-1$ OTUs to be analysed. The rest of the process is repeated until the root has only two child nodes. When calculating distance between OTUs consisting more than one organism (leaves), let *u* and *v*, the distance is determined by the average pairwise distance between the members of *u* and *v* (6) [9].

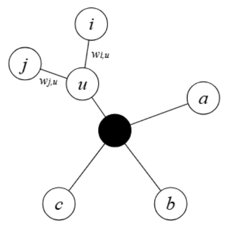
$$D_{u,v} = \frac{\sum_{i\in u}\sum_{j\in v} D_{i,j}}{n_u \cdot n_v} \tag{7}$$

(a)



	i	*j*	*a*	*b*	*c*
i	0	$D_{i,j}$	$D_{i,a}$	$D_{i,b}$	$D_{i,c}$
j	-	0	$D_{j,a}$	$D_{j,b}$	$D_{j,c}$
a	-	-	0	$D_{a,b}$	$D_{a,c}$
b	-	-	-	0	$D_{b,c}$
c	-	-	-	-	0

(b)



	u	*a*	*b*	*c*
u	0	$D_{u,a}$	$D_{u,b}$	$D_{u,c}$
a	-	0	$D_{a,b}$	$D_{a,c}$
b	-	-	0	$D_{b,c}$
c	-	-	-	0

Fig 5. (a) Initial tree in NJ algorithm (left) and its distance matrix (*D*) (right). (b) Tree after one iteration of joining (left) and its distance matrix (*D*) (right).

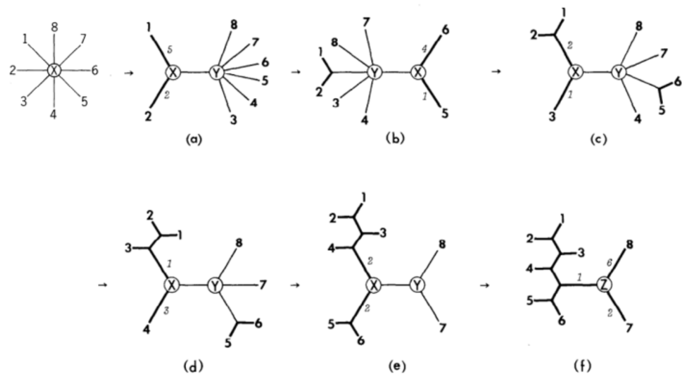


Fig 6. An example of complete NJ algorithm process [9].

III. Method

To deepen the comprehension of the construction of a phylogenetic tree based on DNA sequences using the NJ algorithm, a dummy program was developed in Python. The program receives an input of aligned DNA sequences in a FASTA format file and computes a phylogenetic tree based on it using the NJ algorithm. The distance model used is chosen between the Kimura two-parameter model and the Jukes-Cantor model by the user upon input. The program is not equipped with input validation or error-handling capability for the sake of simplicity. The alignment of DNA sequences must be performed separately using another program such as ClustalW or MEGA.

A. Object Definition

There are three object classes utilised in this program, `MainTree`, `Taxon`, and `IntNodes`. `MainTree` is an object that represent the constructed phylogenetic tree. It has attributes `taxa`, a `list` of all OTUs (organisms) analysed; `children`, a `list` of children nodes of the root; and `n_intnodes` (`int`), the number of internal nodes in the tree.

```python
# Defining object <Tree>
class MainTree:
    def __init__(self, taxa=[], children=[], n_intnodes=0):
        self.taxa = taxa
        self.children = children
        self.n_intnodes = n_intnodes
```

`Taxon` is an object that represents OTUs (organisms) on the leaves of a tree. It has attributes `seq`, a `string` of aligned DNA sequence of the organism; `id`, a `string` of NCBI identifier code; and `branch_len` (`float`), the length (weight) of the branch that is incident to the leaf and connects it to its parent node.

```python
# Defining object <Taxon>
class Taxon:
    def __init__(self, seq="", id="", branch_len=0):
        self.seq = seq
        self.id = id
        self.branch_len = branch_len
```

`IntNodes` is an object that represents internal nodes in the tree. It has attributes `children`, a `list` of its children nodes; `child_taxon`, a `list` of its descendant leaves (taxa); and `branch_len` (`float`), the length (weight) of the branch that connects it with its parent node.

```python
# Defining object <IntNodes>
class IntNodes:
    def __init__(self, children=[], child_taxon=[], branch_len=0):
        self.children = children
        self.child_taxon = child_taxon
        self.branch_len = branch_len
```

As a consequence, the MainTree and, in some cases, IntNodes will have their descendant nodes and leaves nested in their children.

B. Function and Procedure Definition

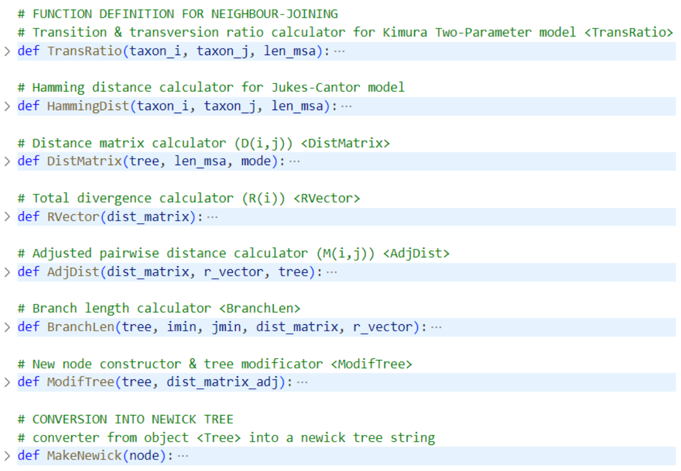
The program has eight functions defined in it.

1. `TransRatio`, a function to compute the transition ratio (*P*) and transversion ratio (*Q*) between a pair of sequences to be used later to compute distance matrix (*D*) using the Kimura two-parameters model. It receives parameters `taxon_i` and `taxon_j`, the pair of sequences in which their *P* and *Q* are computed; and `len_msa` (`int`), the length of the aligned sequence. It return a dictionary with

keys “`p`” that maps to a variable that stores *P* and “`q`” that maps to a variable that stores *Q*.

2. `HammingDist`, a function to compute the Hamming distance (*P*) between a pair of sequences to be used later in the computation of distance matrix using the Jukes-Cantor model. It receives similar parameters to the TransRatio function and returns *P*.
3. `DistMatrix`, a function to compute distance matrix (*D*) in each iteration of the NJ algorithm. It receives the parameters `tree`, the `MainTree` object which its *D* is computed; `len_msa` (`int`), the length of aligned sequence; and `mode`, a `string` that determines the distance model to be used and specified by the user upon input. If the inputted mode is “`K`”, the Kimura two-parameters model is used, and if it is “`J`”, the Jukes-Cantor model is used. It returns *D* in the form of an array.
4. `RVector`, a function that computes total divergence vector (*R*) in each iteration. It receives parameter `dist_matrix`, the distance matrix computed by function `DistMatrix`. It returns *R* as an array.
5. `AdjDist`, a function to computes adjusted pairwise distance matrix (*M*) in each iteration. It receives parameters `dist_matrix`, the distance matrix computed by the function `DistMatrix`; `r_vector`, the total divergence vector computed by the function `Rmatrix`; and `tree`, the object `MainTree`. It returns *M* as an array.
6. `BranchLen`, a function to computes the branch length of a leaf or internal node to its parent node, represented by object `Taxon` or `IntNodes`, respectively. It receives parameters `tree`, the `MainTree` object; `imin` and `jmin`, the newly clustered pair of nodes; `dist_matrix`; the distance matrix computed by `DistMatrix` function; and `r_vector`, the total divergence vector computed by `RVector` function. It returns a dictionary with keys “`i`” which maps to a variable that stores the branch length of node `imin` and “`j`” which maps to a variable that stores the branch length of node `jmin`.
7. `ModifTree`, a procedure to modify the `MainTree` object in each iteration. It consists of steps of finding a pair of OTUs with the minimum adjusted distance, computing the branch length of the pair using the `BranchLen` function, joining the neighbour, and adjusting attributes of the MainTree. It receives parameters `tree`, `the MainTree` object; and `dist_matrix_adj`, the adjusted distance matrix computed by the `AdjDist` function.
8. `MakeNewick`, a procedure to convert the `MainTree` object into a Newick tree string. Newick tree a a form of tree representation that stores every node in a tree along with their connection and weight information in a string (Fig. 7). This format will be used by the `Bio.Phylo` library to visualise the final constructed tree. This function receives a parameter tree, the `MainTree` object.

The minimized definition of all the functions is shown below.



```
# FUNCTION DEFINITION FOR NEIGHBOUR-JOINING
# Transition & transversion ratio calculator for Kimura Two-Parameter model <TransRatio>
> def TransRatio(taxon_i, taxon_j, len_msa): ···

# Hamming distance calculator for Jukes-Cantor model
> def HammingDist(taxon_i, taxon_j, len_msa): ···

# Distance matrix calculator (D(i,j)) <DistMatrix>
> def DistMatrix(tree, len_msa, mode): ···

# Total divergence calculator (R(i)) <RVector>
> def RVector(dist_matrix): ···

# Adjusted pairwise distance calculator (M(i,j)) <AdjDist>
> def AdjDist(dist_matrix, r_vector, tree): ···

# Branch length calculator <BranchLen>
> def BranchLen(tree, imin, jmin, dist_matrix, r_vector): ···

# New node constructor & tree modificator <ModifTree>
> def ModifTree(tree, dist_matrix_adj): ···

# CONVERSION INTO NEWICK TREE
# converter from object <Tree> into a newick tree string
> def MakeNewick(node): ···
```

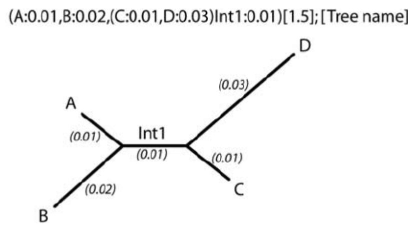
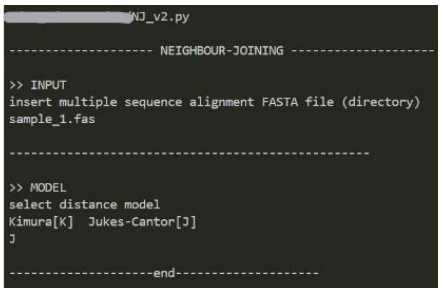


Fig 7. Newick tree string (top) representing a phylogenetic tree (bottom) [10]

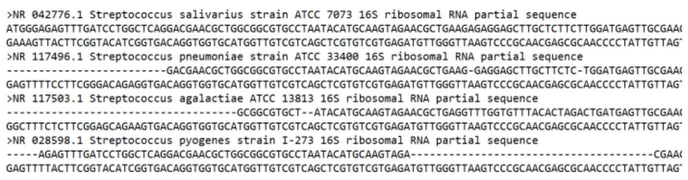
C. Input

The program receives an aligned DNA file in FASTA format. This FASTA formatted file consists of DNA sequences of targeted organisms that are aligned using multiple sequences alignment (MSA) method. FASTA is a text format with the first line starting with “>” that stores some basic information about the sequence (identifier, organism name, accession code, etc.). The following line stores the DNA sequence of the specified organism (Fig. 9). A FASTA file might consist of more than one sequence separated by “>” [1]. In this program, the input MSA FASTA file is parsed and each line is stored as an element of a `list` before being assigned to the `MainTree`.



```
[redacted]NJ_v2.py

------------------- NEIGHBOUR-JOINING -------------------

>> INPUT
insert multiple sequence alignment FASTA file (directory)
sample_1.fas

--------------------------------------------------

>> MODEL
select distance model
Kimura[K]  Jukes-Cantor[J]
J

-------------------end-------------------
```

Fig. 8 Example of input interface in Python terminal. User will be asked to input alignment file directory and distance model preference.



```
>NR 042776.1 Streptococcus salivarius strain ATCC 7073 16S ribosomal RNA partial sequence
ATGGGAGAGTTTGATCCTGGCTCAGGACGAACGCTGGCGGCGTGCCTAATACATGCAAGTAGAACGCTGAAGAGAGGAGCTTGCTCTTCTTGGATGAGTTGCGAA
GAAAGTTACTTCGGTACATCGGTGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGGGTTAAGTCCCGCAACGAGCGCAACCCCTATTGTTAG
>NR 117496.1 Streptococcus pneumoniae strain ATCC 33400 16S ribosomal RNA partial sequence
------------------------GACGAACGCTGGCGGCGTGCCTAATACATGCAAGTAGAACGCTGAAG-GAGGAGCTTGCTTCTC-TGGATGAGTTGCGAA
GAGTTTTCCTTCGGGACAGAGGTGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGGGTTAAGTCCCGCAACGAGCGCAACCCCTATTGTTAG
>NR 117503.1 Streptococcus agalactiae ATCC 13813 16S ribosomal RNA partial sequence
-----------------------------------GCGGCGTGCT--ATACATGCAAGTAGAACGCTGAGGTTTGGTGTTTACACTAGACTGATGAGTTGCGAA
GGCTTTCTCTTCGGAGCAGAAGTGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGGGTTAAGTCCCGCAACGAGCGCAACCCCTATTGTTAG
>NR 028598.1 Streptococcus pyogenes strain I-273 16S ribosomal RNA partial sequence
-----AGAGTTTGATCCTGGCTCAGGACGAACGCTGGCGGCGTGCCTAATACATGCAAGTAGA-------------------------------------CGAA
GAGTTTTACTTCGGTACATCGGTGACAGGTGGTGCATGGTTGTCGTCAGCTCGTGTCGTGAGATGTTGGGTTAAGTCCCGCAACGAGCGCAACCCCTATTGTTAG
```

Fig. 9 Example of a FASTA file storing DNA sequences aligned with MSA method.

D. Algorithm

The algorithm starts with assigning each organism/taxon to an object `Taxon` and storing them into variables. The identifier and sequence of each organism parsed from the FASTA file are stored in `id` and `seq` attributes of `Taxon`, respectively. The next step is initialising the `MainTree` object and assigning the list of organisms to the `taxa` attribute of the `MainTree`. The list is also assigned to the `children` attribute of `MainTree` as all organisms on the leaves are children of the root in the initial condition. Next, a while loop of the NJ algorithm as shown below is executed.

```
# Neighbour-Joining
while len(main_tree.children) >= 2:
    # distance matrix construction
    dist_matrix = DistMatrix(tree=main_tree, len_msa=len_msa, mode=mod)
    # R-vector construction
    r_vector = RVector(dist_matrix=dist_matrix)
    # adjusted distance matrix construction
    dist_matrix_adj = AdjDist(dist_matrix=dist_matrix, r_vector=r_vector, tree=main_tree)
    # neighbour joining, creating a new node
    ModifTree(tree=main_tree, dist_matrix_adj=dist_matrix_adj)
```

After exiting the loop, the program converts the final MainTree into a Newick tree string and uses `Bio.Phylo`, `Bio.StringIO`, and `matplotlib.pyplot` to generate and show a visualisation of the phylogenetic tree.

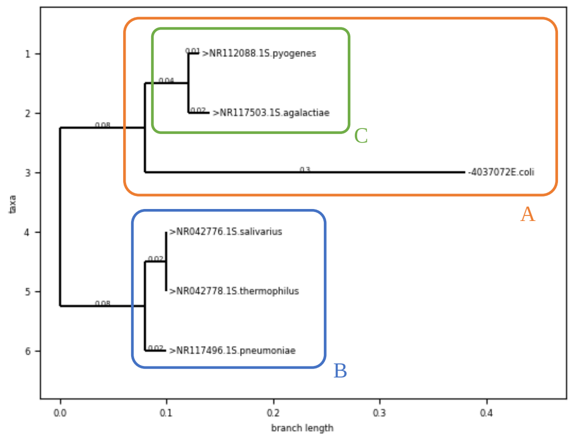
E. Testing

Table I. Bacteria used in the test

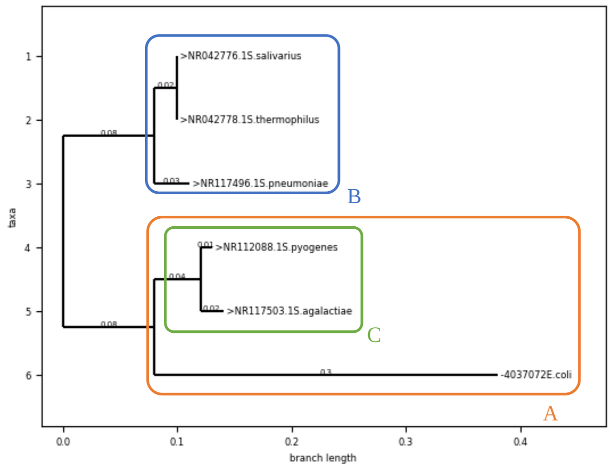
Bacteria name	NCBI accession code
Alignment 1	
Streptococcus thermophilus strain ATCC 19258	NR 042778.1
Streptococcus salivarius strain ATCC 7073	NR 042776.1
Streptococcus pneumoniae strain ATCC 33400	NR 117496.1
Streptococcus agalactiae ATCC 13813	NR 117503.1
Streptococcus pyogenes strain JCM 5674	NR 112088.1
Escherichia coli str. K-12 substr. MG1655	NC 000913.3:4035531-4037072
Alignment 2	
Pseudomonas putida strain IAM 1236	NR 043424.1
Pseudomonas protegens strain CHA0	NR 114749.1
Pseudomonas fluorescens NO7	FJ972536.1
Pseudomonas aeruginosa strain DSM 50071	NR 026078.1
Legionella pneumophila subsp. pascullei strain U8W	NR 041742.1
Escherichia coli strain U 5/41	NR 024570.1
Enterobacter cloacae strain ATCC 13047	NR 102794.2
Enterobacter asburiae strain JM-458	NR 145647.1

Testing was done by inputting two different gene sequence alignment files consisting of several bacteria in separate runs. In this test, 16S rRNA gene of the bacteria was used as the marker gene because of its universal, conserved, variable, and adequate length properties, making it the gold standard for phylogenetic analysis markers. All sequences were retrieved from the NCBI GenBank and RefSeq database. The list of DNA sequences used in this test is presented in Table I. Using the same alignment file, reference phylogenetic trees were constructed using a neighbour-joining algorithm in the MEGA11 software. The result of the dummy program and MEGA11 were compared regarding their isomorphism and branch length (edge weight, *w*) agreement.

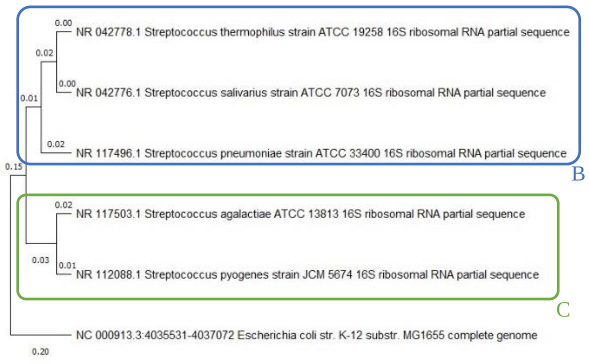
IV. Result and Discussion



(a)



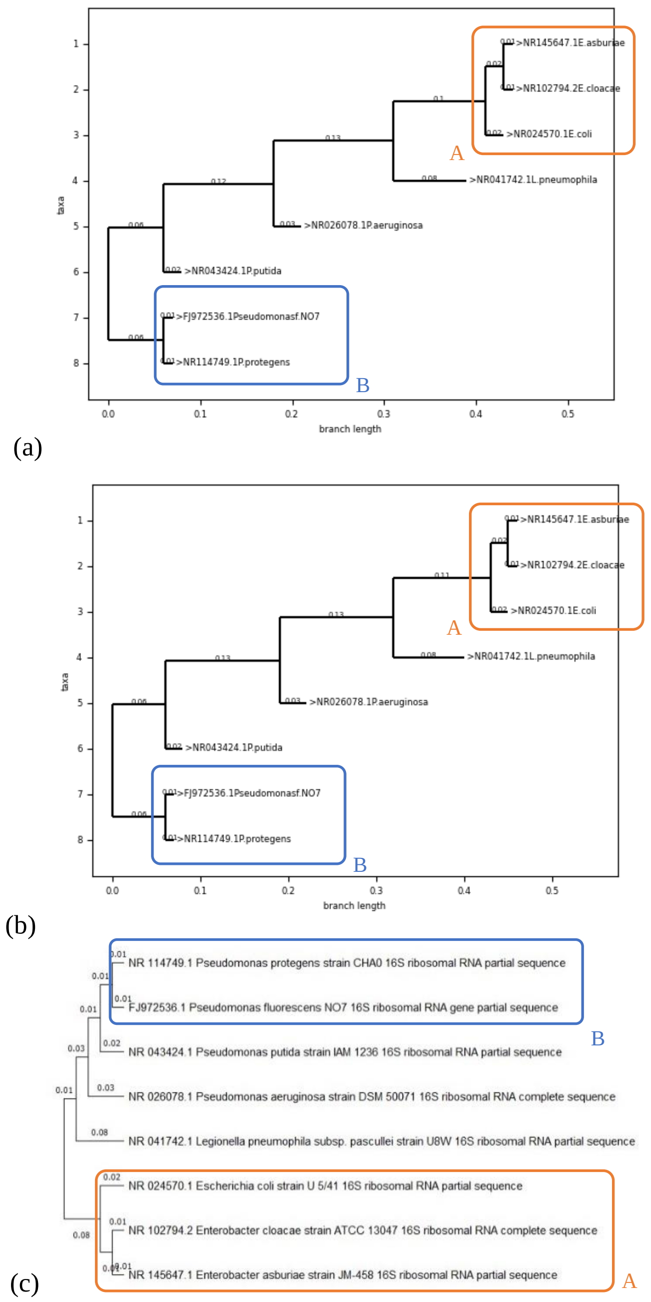
(b)



NR 042778.1 Streptococcus thermophilus strain ATCC 19258 16S ribosomal RNA partial sequence
NR 042776.1 Streptococcus salivarius strain ATCC 7073 16S ribosomal RNA partial sequence
NR 117496.1 Streptococcus pneumoniae strain ATCC 33400 16S ribosomal RNA partial sequence
NR 117503.1 Streptococcus agalactiae ATCC 13813 16S ribosomal RNA partial sequence
NR 112088.1 Streptococcus pyogenes strain JCM 5674 16S ribosomal RNA partial sequence
NC 000913.3:4035531-4037072 Escherichia coli str. K-12 substr. MG1655 complete genome
0.00
0.02
0.00
0.01
0.02
0.15
0.02
0.03
0.01
0.20
B
C

(c)

Fig. 10 Test result of Alignment 1 using (a) Kimura two-parameters model and (b) Jukes-Cantor model. (c) Reference phylogenetic tree of Alignment 1 using MEGA11. Each square A (orange), B (blue), and C (green) represents isomorphic subtrees. In this case, *w* is ignored when deciding isomorphism.



(a)
>NR145647.1E.asburiae
>NR102794.2E.cloacae
>NR024570.1E.coli
>NR041742.1L.pneumophila
>NR026078.1P.aeruginosa
>NR043424.1P.putida
>FJ972536.1Pseudomonasf.NO7
>NR114749.1P.protegens
A
B
taxa
branch length
(b)
>NR145647.1E.asburiae
>NR102794.2E.cloacae
>NR024570.1E.coli
>NR041742.1L.pneumophila
>NR026078.1P.aeruginosa
>NR043424.1P.putida
>FJ972536.1Pseudomonasf.NO7
>NR114749.1P.protegens
A
B
taxa
branch length
(c)
NR 114749.1 Pseudomonas protegens strain CHA0 16S ribosomal RNA partial sequence
FJ972536.1 Pseudomonas fluorescens NO7 16S ribosomal RNA gene partial sequence
NR 043424.1 Pseudomonas putida strain IAM 1236 16S ribosomal RNA partial sequence
NR 026078.1 Pseudomonas aeruginosa strain DSM 50071 16S ribosomal RNA complete sequence
NR 041742.1 Legionella pneumophila subsp. pascullei strain U8W 16S ribosomal RNA partial sequence
NR 024570.1 Escherichia coli strain U 5/41 16S ribosomal RNA partial sequence
NR 102794.2 Enterobacter cloacae strain ATCC 13047 16S ribosomal RNA complete sequence
NR 145647.1 Enterobacter asburiae strain JM-458 16S ribosomal RNA partial sequence
B
A

Fig. 11 Test result of Alignment 2 using (a) Kimura two-parameters model and (b) Jukes-Cantor model. (c) Reference phylogenetic tree of Alignment 2 using MEGA11

As can be seen in Fig. 10, all phylogenetic trees created using NJ algorithm are always binary. Phylogenetic trees of Alignment 1 constructed using the Kimura two-parameter model (Fig. 10a) and Jukes-Cantor model (Fig. 10b) are isomorphic when branch length (*w*) is neglected. The root of each tree has two children, subtrees A and C. The *w* of each internal node and leaf in both trees are almost identical and seem to be equal when rounded to the nearest hundredth, except for *S. pneumoniae* which has the *w* of 0.02 in Fig. 10a and 0.03 in Fig. 10b.

However, phylogenetic trees of Alignment 1 constructed using the dummy program (Fig. 10a, b) and MEGA11 (Fig. 10c) are not isomorphic even when *w* is neglected. In Fig. 10a and b, the roots have subtree A and B as their children, meanwhile, Fig. 10c has a root with *E. coli* leaf and a supertree of subtree B and C. The placement of *E. coli* is the key difference that cancesl the isomorphism between the trees. The total branch lengths (w_T) of each leaf (organism) to the root are slightly different from tree to tree, with some being equal and others differ by no more than ±0.01 when rounded to the nearest hundredth. The only organism with a w_T difference greater than 0.01 is *E. coli*, which has w_T of 0.38 in Fig. 10a, b and 0.35 in Fig. 10c.

Phylogenetic trees of Alignment 2 constructed using the Kimura two-parameter model (Fig. 11a) and the Jukes-Cantor model (Fig. 11b) are isomorphic when branch length (*w*) is neglected. However, these trees are not isomorphic with the tree constructed using MEGA11 shown in Fig. 11c. In Fig 11a, b, subtree A is on the right-most (upper-most) terminal node of the tree and subtree B is on the left-most terminal node, being the child of the root. The exact opposite placement was observed in Fig. 11c. The positions of *P. putida* and *L. pneumophila* are also inverted in Fig. 11c. The *w* between Fig. 11a, b and Fig. 11c also differ in various degrees.

Overall, phylogenetic trees constructed by the dummy program using the Kimura two-parameter model and the Jukes-Cantor model are isomorphic. However, the program failed to construct trees isomorphic to trees computed by MEGA11 from both test files, although the structure of joined neighbours is quite similar. The *w* of each corresponding internal node and leaf between the trees also differ slightly. The *w* of children of the root in trees computed by the dummy program was especially different from those of reference trees.

The differences in topology are suspected to be caused by the difference in the neighbour-joining algorithm used. MEGA11 might use a modified and more robust NJ algorithm compared to the dummy program. It is also possible that MEGA11 can handle outgroup selection and root formation better. The difference in *w* is suspected to be caused by a difference in the sequence distance calculation model used. MEGA11 does not clearly mention the distance model it uses. It might use a more advanced model with different probability distributions and assumptions of the evolutionary process. On the other hand, the dummy program used two of the earliest and simplest distance models.

V. Conclusion

This paper explains the theory of graph, tree, and weighted tree that underlies the construction of a phylogenetic tree, which is one of the most essential tools for phylogenetic analysis in bioinformatics. This paper also explained how parts of a phylogenetic tree are interpreted mathematical point of view, and covered the basics of evolutionary distance models used in phylogenetic tree construction. The step-by-step process of neighbour-joining (NJ) algorithm was elucidated. Next, an explanation of the development of the dummy program created to deepen the understanding of NJ algorithm was presented. The output of the dummy program was compared to the output from MEGA11 using the same datasets. It was observed that the dummy program successfully created phylogenetic trees with similar topology to their reference counterparts made by MEGA11. However, it failed to construct trees isomorphic with the reference trees. The dummy program was able to calculate branch lengths close to that of reference trees, but the branch lengths of child nodes of the root were significantly different. The differences are suspected by modification on the NJ algorithm and difference in the evolutionary distance model used by MEGA11. The dummy program needs some improvement in the mathematical formulae used to be able to produce more accurate trees, especially regarding its capability to select an outgroup and calculate the branch length of chid nodes of the root.

VI. Appendix

The complete repository of DNA-based phylogenetic tree construction dummy program discussed in this paper, along with DNA sequences used in the testing is accessible through https://github.com/Naufal-Alif28/NeighbourJoining.git. The explanation video of this article can be accessed through https://youtu.be/R9ceexlzVe0.

References

[1] H. Christensen. *Introduction to Bioinformatics in Microbiology*. Switzerland: Springer Nature, 2018. ISBN 978-3-319-99280-8.

[2] J.B. Reece & N.A. Campbell. *Biology*. San Francisco, CA: Pearson, 2011. 978-0-321-55823-7.

[3] K.H. Rosen. *Discrete Mathematics and Its Applications (8th ed.)*. New York, NY: McGraw-Hill Education, 2019. ISBN 978-1-259-67651-2.

[4] R. Munir. "Pohon: Bagian 1 [course material]," *Bahan Kuliah IF1220 Matematika Diskrit*. Bandung, Indonesia: Program Studi Teknik Informatika STEI-ITB, 2024. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf. [accessed 23 December 2024]

[5] R. Munir. "Pohon: Bagian 2 [course material]," *Bahan Kuliah IF1220 Matematika Diskrit*. Bandung, Indonesia: Program Studi Teknik Informatika STEI-ITB, 2024. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/24-Pohon-Bag2-2024.pdf. [accessed 23 December 2024]

[6] J. Xiong. *Essential Bioinformatics*. Cambridge, UK: Cambridge University Press, 2006. ISBN 978-0-511-16815-4.

[7] M. Kimura. "A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences," *Journal of Molecular Evolution, 16*, 1980. DOI: 10.1007/BF01731581.

[8] T.H. Jukes & C.R. Cantor. "Evolution of Protein Molecules," in: H.N. Munro. *Mammalian Protein Metabolism*. New York, NY: Academic Press, 1969, pp. 21-132. http://dx.doi.org/10.1016/B978-1-4832-3211-9.50009-7.

[9] N. Saitou & M. Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Molecular biology and evolution*, *4*(4) 1987, pp. 406-425. doi.org/10.1093/oxfordjournals.molbev.a040454.

[10] C. Creevey & J. McInerney. "Tree from Tree: Construction of Phylogenetics Supertrees Using Clann," *Methods in molecular biology*, *537*(139). http://dx.doi.org/10.1007/978-1-59745-251-9_7.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Januari 2025

[illegible]

Naufal Muhammad Alif
10422039