# test\_phylogeny\_builder.py saved under 5. Tests folder

‘’’

Tests include:

1. Verification that the phylogenetic tree is correctly built for a small set of sequences.
2. Ensure that it is rooted properly and reflects the evolutionary relationships.

 Below are further tests but they relate to edge cases:

1. Check if the tree is built for sequences that are identical, nearly identical, or completely different.
2. Test how the tree construction handles very similar or highly divergent sequences.

‘’’

import unittest

from src.phylogeny\_builder import build\_phylogenetic\_tree

from Bio import SeqIO

class TestPhylogenyBuilder(unittest.TestCase):

def test\_build\_tree(self):

"""Test the construction of a phylogenetic tree from dog breeds."""

sequences = list(SeqIO.parse("data/dog\_breeds.fa", "fasta"))

tree = build\_phylogenetic\_tree(sequences)

# Ensure the tree has the expected structure (e.g., number of nodes)

self.assertGreater(len(tree.get\_terminals()), 1) # At least two sequences should be in the tree

def test\_tree\_with\_identical\_sequences(self):

"""Test tree construction with identical sequences."""

# Creating two identical sequences

sequences = list(SeqIO.parse("data/dog\_breeds.fa", "fasta"))

identical\_sequence = sequences[0]

sequences.append(identical\_sequence) # Add a duplicate

tree = build\_phylogenetic\_tree(sequences)

self.assertGreater(len(tree.get\_terminals()), 1) # The tree should still have more than one terminal node

def test\_tree\_with\_highly\_divergent\_sequences(self):

"""Test tree construction with highly divergent sequences."""

sequences = list(SeqIO.parse("data/dog\_breeds.fa", "fasta"))

# Test with sequences that are very different

divergent\_sequences = sequences[:3] # Use first 3 for simplicity

tree = build\_phylogenetic\_tree(divergent\_sequences)

self.assertGreater(len(tree.get\_terminals()), 1) # Should still generate a tree with multiple branches

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()