Roll: 32

Jahir Sadik Monon

Fundamentals of Genomics and Proteomics Lab (Task 2)

Department of Computer Science and Engineering, University of Dhaka

Problem statement: Implement UPMGA algorithm and construct the phylogenetic tree.

Input:

S1: ACGCGTTGGGCGATGGCAAC

S2: ACGCGTTGGGCGACGGTAAT

S3: ACGCATTGAATGATGATAAT

S4: ACGCATTGAATGATGATAAT

S5: ACACATTGAGTGATAATAAT

```
from typing import List, Tuple

def pairwise_distance(str1: str, str2: str) -> int:
    len1, len2 = len(str1), len(str2)
    distance = 0
    for i in range(min(len1, len2)):
        if str1[i] != str2[i]:
            distance += 1
    distance += abs(len1 - len2)
    return distance


def print_dist(dist_entry):
    length = len(dist_entry.keys())
    keys = list(dist_entry.keys())
    mapping = {}
    for i in range(length):
        mapping[i] = keys[i]


    dist_mat = []
    for i in range(length):
        dist_mat_entry = []
        for j in range(length):
            if mapping[i] == mapping[j]:
                dist_mat_entry.append(0)
                continue
            dist_mat_entry.append(dist_entry[mapping[i]][mapping[j]])
        dist_mat.append(dist_mat_entry)

    print("\t", end = "")
    for i in range(length):
        print(f"S{mapping[i]}", end = "\t")
    print("\n-----------------------------------------")
    for i, elem in enumerate(dist_mat):
        print(f"S{mapping[i]}", end = "\t")
        for j in elem:
            print(j, end = "\t")
        print()


def upgma(dist_entry, n):
    itr = 1;
```

```
    itr = 1;
    joining_order = []
    while n > 1:
        print_dist(dist_entry)
        # print(f"keys: {dist_entry.keys()}")
        n -= 1
        itr += 1
        key_r = None
        key_c = None
        min_value = float('inf')
        for i in dist_entry.keys():
            for j in dist_entry[i].keys():
                if i != j and min_value > dist_entry[i][j]:
                    min_value = dist_entry[i][j]
                    key_r = i
                    key_c = j
        print("_____________________________________________")
        print(f"Minimum value {min_value} found in: Row S{key_r} Column S{key_c}")
        n_key = key_r + "_" + key_c
        print(f"Iteration: {itr}")
        print(f"Joined: S{key_r} and S{key_c}")
        joining_order.append(f"S{key_r},S{key_c}")
        print("_____________________________________________")
        dist_entry[n_key] = {}
        for i in dist_entry.keys():
            first = 0
            second = 0
            if i != key_r and i != key_c and i != n_key:
                if i in dist_entry[key_r]:
                    first = dist_entry[key_r][i]
                else:
                    first = dist_entry[i][key_r]
                if i in dist_entry[key_c]:
                    second = dist_entry[key_c][i]
                else:
                    second = dist_entry[i][key_c]

                dist_entry[n_key][i] = 0.5*first + 0.5*second
                dist_entry[i][n_key] = 0.5*first + 0.5*second

        dist_entry.pop(key_c)
        dist_entry.pop(key_r)

        for j in dist_entry.values():
            if key_r in j:
                j.pop(key_r)
            if key_c in j:
                j.pop(key_c)

    return joining_order


sequences = ["ACGCGTTGGGCGATGGCAAC",
             "ACGCGTTGGGCGACGGTAAT",
             "ACGCATTGAATGATGATAAT",
             "ACGCATTGAATGATGATAAT",
             "ACACATTGAGTGATAATAAT"]

sequences_test_from_slides = ["GTGCTGCACGGCTCAGTATAGCATTTACCCTTCCATCTTCAGATCCTGAA",
              "ACGCTGCACGGCTCAGTGCGGTGCTTACCCTCCCATCTTCAGATCCTGAA",
              "GTGCTGCACGGCTCGGCGCAGCATTTACCCTCCCATCTTCAGATCCTATC",
              "GTATCACACGACTCAGCGCAGCATTTGCCCTCCCGTCTTCAGATCCTAAA",
              "GTATCACATAGCTCAGCGCAGCATTTGCCCTCCCGTCTTCAGATCTAAAA"
             ]
```

```
# We start our processing here
dist_entry = {f"{r+1}": {f'{c+1}': None for c in range(len(sequences))} for r in range(len(sequences))}
for i, seq1 in enumerate(sequences):
    for j, seq2 in enumerate(sequences):
        dist_entry[f"{i+1}"][f"{j+1}"] = pairwise_distance(seq1, seq2)


print("Input sequences are:")
for i, sequence in enumerate(sequences):
    print(f"S{i+1}: {sequence}")

print("_____________________________________________")
joining_order = upgma(dist_entry, len(sequences))

print(f"Joining Order in the phylogenetic tree:")
for joining in joining_order:
    print(joining)

root = joining_order[-1].split(",")
print(f"S{root[0][1:]}_{root[1][1:]}")
```