Universidade Federal do Rio Grande do Sul - Instituto de Informática
INF05018 Biologia Computacional - Prof: Márcio Dorn

Rodrigo Okido (252745)

Resolução Lista V

Será apresentado aqui a resolução da lista V. Todos os métodos serão explicados através de pseudocódigos informando a idéia geral das funções implementadas. Caso dúvidas referente a explicação, o código se encontra disponível bastante documentado para quaisquer esclarecimentos.

Variáveis Globais:

```
# Hash contendo a distância entre as espécies
dist_species = {}

# Lista contendo a distância da espécie de Ux (onde x> = 0)
dist_calculated_nodes = []

# Lista para colocar todos os cálculos de distância para a etapa 1
# (Sx = (Sum all Dx) / (N - 2))
s_dist = []

# Contador do número de nós gerados até o final do algoritmo
node_counter = 0
```

Function create_node_and_update_matrix (matrix, labels, x, y) : Cria um "Nó" com a distância do par encontrado no passo 2 e atualiza o matriz com as novas distâncias. Os rótulos são atualizados no final do processo e a variável lista “s_dist” contendo os valores de distância calculados anteriormente são apagados para a próxima iteração (se necessário).
(Etapas 3, 4 e 5 inclusos neste método)

Pseudocode:

```
global node_counter

# Calculate the distance of the especie to the node U. (Step 3)
s_xU = matrix[x][y]/2 + ((s_dist[x] - s_dist[y])/2)
s_yU = matrix[x][y]/2 + ((s_dist[y] - s_dist[x])/2)

# (Steps 4 and 5)
```

```
# Update entire column (i, x), where i > x.
PARA i em TAMANHO(x+1, y):
    matrix[i][x] = (matrix[i][x] + matrix[y][i] - matrix[x-1][y])/2


# Rest values from row i.
PARA i em TAMANHO(y+1, len(matrix)):
    matrix[i][x] = (matrix[i][x] + matrix[i][y])/2
    # Remove the column.
    DELETA matrix[i][y]

# Remove row.
DELETA matrix[y]

# Update label.
labels[x] = "[U{0} : ".format(node_counter) + labels[x] + ": {0} || ".format(s_xU) + labels[y]
+ ": {0} ]".format(s_yU)
# Increment the node counter.
node_counter = node_counter + 1
# Add this node to an list of calculated nodes.
dist_calculated_nodes.append(labels[x])
# Delete one of the species (due to joining)
del labels[y]

# Erase the list of s_dist for calculations again in the next iteration.
del s_dist[:]
```

Function calculate_distance (matrix, labels) : Calcula para cada espécie a distância de todas as outras espécies. Utiliza as espécies de distância mais baixa e retorna suas coordenadas. (Etapa 1 e Etapa 2 são feitas nesta função)

Pseudocode:

```
lowest_pair_dist = 1000000000

# Index coordinates
x = 0
y = 0

# Built list adding all distances between species (Step 1)
```

```
PARA i em tamanho(matriz)
        Dist = 0
        PARA j em tamanho (labels)
                SE i ! = j
                        Adiciona na distância final para a espécie

result_specie = dist  / (len(matrix[ i ] - 2)
Adiciona o resultado numa lista de distancias s_dist


# Take the pair of species with the lowest distance (Step 2)
# If equals distances, choses the first
PARA i em TAMANHO(matriz)
      dist = 0
      PARA j em TAMANHO (labels)

      SE i != j:
            dist = matrix[i][j] - s_dist[i] - s_dist[j]
            SE dist MENOR QUE lowest_pair_dist:
                lowest_pair_dist = dist
                x, y = i, j


pair = "[" + labels[x] + " - " + labels[y] + "]"
dist_species[pair] = lowest_pair_dist

RETORNA x,y
```

Function neighbor_joining (matrix, assoc_label) : Realiza o método Neighbor Joining. Faz o chamamento dos métodos implementados e comentados anteriormente, e é executado enquanto houver elementos na matriz, ou seja, enquanto houverem mais de uma espécie na matriz, o método mantém realizando os cálculos.

Pseudocode:

```
Enquanto tamanho (assoc_label) maior que 1:

        x,y = calculate_distance(matrix, assoc_label)
        create_node_and_update_matrix(matrix, assoc_label, x, y)

Retorna assoc_label[0]
```

Function main ( ) : Função principal de execução. Gerado uma lista de associação (hard coded) e a matriz que será utilizada para ser aplicado método Neighbor Joining. Ao final os resultados serão impressos na tela.

Pseudocode:

```
assoc_label = ["GOR", "ORANG", "HUM", "CHI", "GIB"]

nj_matrix = [
    [0, 0.1890, 0.1100, 0.1130, 0.2150],
    [0.1890, 0, 0.1790, 0.1920, 0.2110],
    [0.1100, 0.1790, 0, 0.09405, 0.2050],
    [0.1130, 0.1920, 0.09405, 0, 0.2140],
    [0.2150, 0.2110, 0.2050, 0.2140, 0]
    ]

neighbor_joining(nj_matrix, assoc_label)
imprime dist_calculated_nodes
```

O resultado produzido será :

['[U0 : ORANG: 0.0931666666667 || GIB: 0.117833333333]', '[U1 : HUM: 0.0443583333333 || CHI: 0.0496916666667]', '[U2 : HUM: 0.0448333333333 || ORANG: 0.0396666666667]', '[U3 : HUM: 0.0839166666667 || GOR: 0.0260833333333]']

Este resultado mostra a distancia de cada par para cada U calculado. Como podemos observar, em alguns momentos a espécie é calculada novamente, e isto provavelmente pode ser algum erro entre as etapas de atualização na matriz.