Biologia Computacional 2019/2 - Lista de Exercícios 3 - Parte I

Aluno: Leonardo Oliveira Wellausen

1) Código:

```
class Node:
    def __init__(self, left, right, val):
        self.left = left
        self.right = right
        self.val = val

    def print(self, gap):
        if self.right is not None:
            print(gap + self.val)
            self.left.print(gap + self.val + "-left----")
            self.right.print(gap + self.val + "-right----")
        else:
            print(gap + "OTU:" + str(self.val))

        return
```

Estrutura utilizada para criação e armazenamento da árvore. Assim como um método para a impressão da mesma.

```
distances = {
    ('gori', 'oran'): 0.1890,
    ('gori', 'huma'): 0.1100,
    ('gori', 'chim'): 0.1130,
    ('gori', 'giba'): 0.2150,
    ('oran', 'huma'): 0.1790,
    ('oran', 'chim'): 0.1920,
    ('oran', 'giba'): 0.2110,
    ('huma', 'chim'): 0.0940,
    ('huma', 'giba'): 0.2050,
    ('chim', 'giba'): 0.2140
}
keys = [k for k in distances.keys()]
for k in keys:
    distances[(k[1], k[0])] = distances[k]

otus = {i[0] for i in distances.keys()}
tree = {i: Node(None, None, i) for i in otus}
root = Node(None, None, None)
```

Inicialização da matriz de distâncias (hard-coded) e outras variáveis do algoritmo. OTUs é a lista com todos os OTUs e grupos encontrados. Tree é a árvore montada com raiz root. Todos as entradas no dicionário de distância são espelhados.

```
while len(otus) > 1:
    new_node = get_min(distances)
    new_otu = '(' + new_node[0] + ',' + new_node[1] + ')'

    root = tree[new_otu] = Node(tree[new_node[0]],
                                tree[new_node[1]],
                                new_otu)

    otus.remove(new_node[0])
    otus.remove(new_node[1])
    distances.pop((new_node[0], new_node[1]))
    distances.pop((new_node[1], new_node[0]))

    for i in otus:
        distances[(new_otu, i)] = distances[(i, new_otu)] = (distances.pop((i, new_node[0])) + distances.pop((i, new_node[1]))) / 2
        distances.pop((new_node[0], i))
        distances.pop((new_node[1], i))

    otus.add(new_otu)

root.print('')
```

Criação da árvore. Primeiro encontramos o novo grupo a ser criado (quais OTUs devem ser unidas). Criamos o novo OTU a ser inserido na árvore e inserimos este novo nodo como raiz. Após, removemos os dois OTUs unidos da lista de OTUs e também da matriz de distâncias.

Após o novo nodo criado, atualizamos as distâncias na matriz, já removendo todas entradas referentes aos OTUs removidos. Por último, inserimos o novo grupo na lista de OTUs. Paramos quando todos os OTUs forem removidos e só sobrar um único grupo contendo todos.

Imprimimos a árvore.

```
def get_min(dic: dict):
    min_v = np.inf
    min_k = ()
    for k in dic.keys():
        if dic[k] < min_v:
            min_v = dic[k]
            min_k = k

    return min_k
```

Função que retorna o grupo com menor distância.

Resultado:

```
C:\Users\lwell\AppData\Local\Programs\Python\Python37\python.exe D:/Documents/CIC/biocomp/lista3/e1.py
((((huma,chim),gori),oran),giba)
((((huma,chim),gori),oran),giba)-left----(((huma,chim),gori),oran)
((((huma,chim),gori),oran),giba)-left----(((huma,chim),gori),oran)-left----((huma,chim),gori)
((((huma,chim),gori),oran),giba)-left----(((huma,chim),gori),oran)-left----((huma,chim),gori)-left----(huma,chim)
((((huma,chim),gori),oran),giba)-left----(((huma,chim),gori),oran)-left----((huma,chim),gori)-left----(huma,chim)-left----OTU:huma
((((huma,chim),gori),oran),giba)-left----(((huma,chim),gori),oran)-left----((huma,chim),gori)-left----(huma,chim)-right----OTU:chim
((((huma,chim),gori),oran),giba)-left----(((huma,chim),gori),oran)-left----((huma,chim),gori)-right----OTU:gori
((((huma,chim),gori),oran),giba)-left----(((huma,chim),gori),oran)-right----OTU:oran
((((huma,chim),gori),oran),giba)-right----OTU:giba

Process finished with exit code 0
```

Entendendo o resultado: Na primeira linha vemos o nodo raiz, que por si só já contém toda a informação necessária para entender os grupos gerados. Esta informação é na forma de nodos representados por pares identados de parênteses, onde “(n1, n2)” é um noso que tem n1 e n2 como filhos esquerdo e direito, respectivamente.

Em seguida montamos a topologia da árvore; em cada linha um nodo é expandido para a esquerda até chegar em uma folha (simbolizada por OTU), e em seguida expandido para a direita.

Exemplo demonstrativo:

(origem do nodo) nodo

(origem do nodo) nodo + filho à esquerda (expande o filho à esquerda)

(origem do nodo) nodo + filho à direita (expande o filho à direita)