Instituto de Informática, UFRGS
INF05018 - Biologia Computacional - Turma: U (2019/2)
Prof. Márcio Dorn
Lista de exercícios 2 parte 1
Nome: Pedro Foletto Pimenta   Nº Cartão UFRGS: 00229778

1a. A espécie que apresenta maior semelhança com a humana dentre as listadas, em termos de sequência, é a do cavalo (horse). O alinhamento entre as sequências de humano e de cavalo teve um score de 41028 e uma identidade de 121.

Código fonte da função *needleman_wunsch*:

```
# needleman_wunsch algorithm: global alignment of two sequences
def needleman_wunsch(seq1, seq2):

    # size of the grid
    size1 = len(seq1) + 1 # +1 for the extra column/row in the points table
    size2 = len(seq2) + 1

    # construct the grid
    point_matrix = np.zeros((size1, size2))
    for i in range(size1):
            for j in range(size2):
                    if(i==0):    # first row
                            point_matrix[i][j] = -j
                    elif(j==0):    # first column
                            point_matrix[i][j] = -i
                    else:        # rest of the table
                            if(seq1[i-1] == seq2[j-1]): # match
                                    match_score = 5
                            else: # mismatch
                                    match_score = -3
                            expr1 = point_matrix[i-1][j-1] + match_score
                            expr2 = point_matrix[i][j-1] -4 # -4 == +gap(seq1)
                            expr3 = point_matrix[i-1][j] -4 # -4 == +gap(seq2)
                            point_matrix[i][j] = max(expr1, expr2, expr3)

    # find the best alignment
    alignment1 = ""
    alignment2 = ""

    match_count = 0

    # start at the bottom right
    i, j = size1-1, size2-1
    alignment_score = point_matrix[i][j] # init total score

    # trace back to the upper left corner
    while (i > 0 and j > 0):
            up = point_matrix[i-1][j]
            left = point_matrix[i][j-1]
            diag = point_matrix[i-1][j-1]
            # find out best direction
            if(diag >= up and diag >= left):
                    i = i - 1
                    j = j - 1
                    alignment1 = seq1[i] + alignment1
                    alignment2 = seq2[j] + alignment2
            elif(up >= diag and up >= left):
                    i = i - 1
                    alignment1 = seq1[i] + alignment1
                    alignment2 = "-" + alignment2
            else: # (left >= diag and left >= up)
                    j = j - 1
                    alignment1 = "-" + alignment1
                    alignment2 = seq2[j] + alignment2
            # count score
            alignment_score = alignment_score + point_matrix[i][j]

    # get last remaining letters
    while(i > 0):
            i = i - 1
            alignment1 = seq1[i] + alignment1
            alignment2 = "-" + alignment2
    while(j > 0):
            j = j - 1
            alignment1 = "-" + alignment1
            alignment2 = seq2[j] + alignment2

    # identidade
    match_count = 0
    for i in range(len(alignment1)):
            if(alignment1[i] == alignment2[i]):
                    match_count += 1

    # print point matrix
    print("...printing point matrix of dimensions " + str(point_matrix.shape))
    print(point_matrix)

    # print aligned sequences
    print_alignment(alignment1, alignment2)

    # print "identidade do alinhamento" ?
    print("Identidade (?): " + str(match_count))

    # print score
    print("Alignment score: " + str(alignment_score))

    return alignment_score
```

Código fonte da *main*:

```
# qual das especies apresenta a maior semelhanca, em termos de sequencia, com a espécie humana (homo sapiens)
best_score = 0
most_similar = ""
for seq_name in seq_dict.keys(): # iterate through all sequences
   if(seq_name != "human"): # do not align human with human
           print("\nAligning the human sequence with " + seq_name + " sequence...")
           seq = seq_dict[seq_name]
           score = needleman_wunsch(seq, seq_human) # align
           if(score > best_score):
                   best_score = score
                   most_similar = seq_name

print("\n\nThe species that is most similar to humans (in sequence terms) is " + str(most_similar) + " (alignment score: 
"+str(best_score) + ")")
```

2a. Usando a matriz Blosum 62 para os valores de match/mismatch, a espécie que apresenta maior semelhança com a humana dentre as listadas, em termos de sequência, é a do cavalo (horse). O alinhamento entre as sequências de humano e de cavalo com a tabela Blosum 62 teve um score de 46116 e uma identidade de 122.

Código fonte da função *needleman_wunsch* usando a matriz Blosum 62:

```
# needleman_wunsch algorithm: global alignment of two sequences using the BLOSUM62 matrix
def needleman_wunsch(seq1, seq2):

  # size of the grid
  size1 = len(seq1) + 1 # +1 for the extra column/row in the points table
  size2 = len(seq2) + 1

  # load blosum62 matrix
  blosum62 = Lookup_table("blosum62.txt")

  # construct the grid
  point_matrix = np.zeros((size1, size2))
  for i in range(size1):
          for j in range(size2):
                  if(i==0):    # first row
                          point_matrix[i][j] = -j
                  elif(j==0):    # first column
```

```
                        point_matrix[i][j] = -i
                else:   # rest of the table
                        match_score = int(blosum62.lookup_score(seq1[i-1], seq2[j-1])) # using blosum62

                        expr1 = point_matrix[i-1][j-1] + match_score
                        expr2 = point_matrix[i][j-1] -4 # -4 == +gap(seq1) ?????
                        expr3 = point_matrix[i-1][j] -4 # -4 == +gap(seq2) ?????
                        point_matrix[i][j] = max(expr1, expr2, expr3)

    # find the best alignment
    alignment1 = ""
    alignment2 = ""

    match_count = 0

    # start at the bottom right
    i, j = size1-1, size2-1
    alignment_score = point_matrix[i][j] # init total score

    # trace back to the upper left corner
    while (i > 0 and j > 0):
        up = point_matrix[i-1][j]
        down = point_matrix[i][j-1]
        diag = point_matrix[i-1][j-1]
        # find out best direction
        if(diag >= up and diag >= down):
                i = i - 1
                j = j - 1
                alignment1 = seq1[i] + alignment1
                alignment2 = seq2[j] + alignment2
        elif(up >= diag and up >= down):
                i = i - 1
                alignment1 = seq1[i] + alignment1
                alignment2 = "-" + alignment2
        else: # (down >= diag and down >= up)
                j = j - 1
                alignment1 = "-" + alignment1
                alignment2 = seq2[j] + alignment2

        # count score
        alignment_score = alignment_score + point_matrix[i][j]

    # get last remaining letters
    while(i > 0):
        i = i - 1
        alignment1 = seq1[i] + alignment1
        alignment2 = "-" + alignment2
    while(j > 0):
        j = j - 1
        alignment1 = "-" + alignment1
        alignment2 = seq2[j] + alignment2

    # identidade
    match_count = 0
    for i in range(len(alignment1)):
        if(alignment1[i] == alignment2[i]):
                match_count += 1

    # print point matrix
    print("...printing point matrix of dimensions " + str(point_matrix.shape))
    print(point_matrix)

    # print aligned sequences
    print_alignment(alignment1, alignment2)

    # print "identidade do alinhamento" ?
    print("Identidade (?): " + str(match_count))

    # print score
    print("Alignment score: " + str(alignment_score))

    return alignment_score
```

O código fonte da *main* é o mesmo usado para o exercício 1a.