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

Rodrigo Okido (252745)

Resolução Lista II

Para cada sequencia de animal, um arquivo foi gerado. Dando um total de 8 arquivos diferentes. As funções criadas foram essas:

Variável Global:

similarity_id = [ ]

Adiciona nesta lista, todas as similaridades entre sequências encontradas.

Function main ( ) :

Pseudocode:

```
content1 = ""
Results = { } #Hash onde vai ser guardado a comparação com o seu resultado após a aplicação do método de needleman wunsch.
#Realiza este procedimento duas vezes no mínimo para realizar a comparação.
Open file ("human.txt"):
    proxima_linha(file) #Ignora primeira linha
    proxima_linha(file) #Ignora segunda linha
    para line em file: #Grava Conteudo
      line = line.ignore('\n')
      content1+= line

#Repete isso para os outros arquivos
Key_max = max(results.keys) #Retorna o indice da chave que tem o maior valor
Print results[key_max] #Retorna o valor
```

Function check_identity (string_id1 , string_id2):

Esta função é chamada somente após as duas sequências estarem alinhadas conforme o método.

Pseudocode:

```
identity1 = list(str_id1) #Transforma em lista a sequence 1
identity2 = list(str_id2) #Transforma em lista a sequence 2

final_score = 0 #Resultado final do score de identidade

para i em tamanho_identidade:
   SE identity1[i] == identity2[i]:
      final_score = final_score + 5 #Introduz match
   OU SE identity1[i] != identity2[i]:
      final_score = final_score - 3 #Introduz mismatch
   OU SE identity1[i] == "-":
      final_score = final_score - 4 #Introduz gap
   SE NÃO:
      final_score = final_score - 4 #Introduz gap


retorna final_score #Retorna o score final
```

Function check_max (sequence_list1, sequence_list2, x, y, pos_table, pos_table_x, pos_table_y):

Função usada para retornar o máximo valor na tabela. Usado no método de Needleman Wunsch para pegar o maior número entre as 3 possibilidades de caminho (cima, esquerda ou diagonal). Recebe as duas sequências desejadas, os índices x e y que será analisado da sequência, e os valores que já estão na tabela à esquerda, acima e na diagonal do valor que será calculado.

Pseudocode:

```
match = 5
mismatch = -3
gap = -4

SE sequence_list1[y] == sequence_list2[x]:
        retorna maximo(match + pos_table, gap + pos_table_x, gap + pos_table_y)
SE NÃO:
        retorna maximo(pos_table + mismatch, pos_table_x + gap, pos_table_y + gap)
```

Function needleman_wunsch(seq1, seq2):
Esta é a função principal que executa o método de Needleman_wunsch.

Pseudocode:

```
#Score
score = 0

#Transforma em lista as duas sequências recebidas
seq1_list = list(seq1) #column
seq2_list = list(seq2) #lines

#Armazena o tamanho das duas sequências (+1 devido a coluna e linha do gap)
#Será adiante descontado esse acréscimo para o correto uso nas sequências.
size_seq1 = tamanho(seq1) + 1
size_seq2 = tamanho(seq2) + 1

SE size_seq1 > 0 E size_seq2 > 0:

    #Inicializa preenchendo tudo com zeros
    needle_table = [0] * (size_seq2) #lines
    PARA i EM tamanho(size_seq2):
        needle_table[i] = [0] * (size_seq1) #columns

    #Inicializa a gap em coluna
    gap_seq1 = 0
    PARA i EM TAMANHO(size_seq1):
        SE i == 0:
            continue
        SE NÃO:
            gap_seq1 = gap_seq1 - 4
            needle_table[0][i] = gap_seq1

    #Inicializa a gap em linha
    gap_seq2 = 0
    PARA j EM TAMANHO(size_seq2):
        SE j == 0:
            continua
        SE NÃO:
            gap_seq2 = gap_seq2 - 4
            needle_table[j][0] = gap_seq2
```

```
    #Preenche o resto da tabela com os valores corretos
    PARA x EM tamanho(size_seq2):
        SE x+1 == size_seq2:
                termina
        PARA y EM tamanho(size_seq1):
            SE y+1 == size_seq1:
                 termina
            needle_table[x+1][y+1] = check_max(seq1_list, seq2_list, x, y, needle_table[x][y],
needle_table[x][y+1], needle_table[x+1][y] ) #Aqui usa o método check_max para verificar o
melhor valor a ser atribuído nesta posição da tabela.


    #Calcula pontuação e monta o alinhamento das duas sequências
    x_calc = size_seq2 - 1
    y_calc = size_seq1 - 1
    score = needle_table[x_calc][y_calc]

    id1 = ""
    id2 = ""


    while x_calc != 0 and y_calc != 0:
        path1 = needle_table[x_calc-1][y_calc-1] #Pega o valor da diagonal
        path2 = needle_table[x_calc][y_calc-1] #Pega o valor da esquerda
        path3 = needle_table[x_calc-1][y_calc] #Pega o valor acima

        #Realiza as comparações e atribui os scores e a montagem do alinhamento das duas
sequências.
        SE x_calc == 0 E y_calc != 0:
            score = score + path2
            y_calc = y_calc - 1
            id2 += "-"
        OU SE x_calc != 0 E y_calc == 0:
            score = score + path3
            x_calc = x_calc - 1
            id1 += "-"
        OU SE path1 > path2 E path1 > path3:
            score = score + path1
            x_calc = x_calc -1
            y_calc = y_calc -1
            id1 += seq1_list[y_calc]
            id2 += seq2_list[x_calc]
```

```
        OU SE path2 > path1 E path2 > path3:
            score = score + path2
            x_calc = x_calc
            y_calc = y_calc -1
            id1 += seq1_list[x_calc]
            id2 += "-"
        SE NÃO:
            score = score + path3
            x_calc = x_calc -1
            y_calc = y_calc
            id1 += "-"
            id2 += seq2_list[y_calc]


    Imprime needle_table #imprime a tabela final preenchida
    Imprime reverse( id1 ) #Imprime a sequencia 1 (Invertida pois o cálculo na tabela é feita do fim para o inicio)
    imprime reverse( id2 ) #Imprime a sequencia 2 (Invertida pois o cálculo na tabela é feita do fim para o inicio)

    global similarity_id
    Insere fim da lista similarity_id (check_identity(id1,id2)) #E por fim checa finalmente a identidade do alinhamento chamando a função check_identity e coloca na lista similarity_id.

    retorna score #Retorna o score obtido na tabela

  SE NÃO:
    Retorna Erro
```

Resultado Encontrado:

```
#Tabela Needleman preenchida (** Muito grande..)

#As duas sequências alinhadas.
VLSPA-DKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKV
ADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLD
KFLAS-VSTVLTSKY-
VLSA-DDKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHFDLSHGSAQVKAHGKK
VGDALTLAVGHLDDLPGALSNLSDLHAHKLRVDPVNFKLLSHCLLSTLAVHLPNDFTPAVHASL
DKFLS-VVSTVLTSKYR
```

41028 #Score obtido da tabela
539 #Identidade do alinhamento