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

Rodrigo Okido (252745)

Resolução Lista III

Dois arquivos foram gerados armazenando a sequência do homo sapiens e do glabrata. Baseado fortemente na lista II, onde foi utilizado o método de needleman wunsch, esta lista foi implementada com a seguinte idéia:

Variável Global:

identity_score = 0

Armazena o score de identidade final das duas sequências.

Function main ( ) :

Pseudocode:

```
#Open and read file.
   content1 = ""
   content2 = ""
   results = {}

   openfile("glabrata.txt") :
      pula_linha(file)
      para cada linha em arquivo:
         linha = ignora('\n')
         content1+= linha

   openfile("homosapiens.txt") :
      pula_linha(file2)
       para cada linha em arquivo:
         linha = ignora('\n')
         content2+= linha

   cmp1 = smith_waterman(content1,content2)
   results["Human - Glabrata"] = cmp1

   #print results
```

```
Imprime "Score table (Smith Waterman): ", score
Imprime "Score identidade: ", identity_score
```

Function check_identity (string_id1 , string_id2):

Esta função é chamada somente após as duas sequências estarem alinhadas conforme o método. A implementação segue exatamente a mesma da lista II, alterando apenas os valores de scores para este 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 + 1 #Introduz match
   OU SE identity1[i] != identity2[i]:
      final_score = final_score - 1 #Introduz mismatch
   OU SE identity1[i] == "-":
      final_score = final_score - 2 #Introduz gap
   SE NÃO:
      final_score = final_score - 2 #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 Smith Waterman 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.

Novamente, esta função foi aproveitada da lista anterior, alterando apenas os valores de match, mismatch e gap, e acrescentado 0 na comparação entre os valores para retorno do máximo valor.

Pseudocode:

```
match = 1
mismatch = -1
gap = -2

SE sequence_list1[y] == sequence_list2[x]:
        retorna maximo(0, match + pos_table, gap + pos_table_x, gap + pos_table_y)
SE NÃO:
        retorna maximo(0, pos_table + mismatch, pos_table_x + gap, pos_table_y + gap)
```

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

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
```

```
#Armazena o maior valor
maior_valor = -1
#Armazena o maior indice X e Y localizado na tabela
 x_index_maior_numero = -1
 y_index_maior_numero = -1

 #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
    smith_watertable[x+1][y+1] = check_max(seq1_list, seq2_list, x, y,
    smith_watertable[x][y], smith_watertable[x][y+1], smith_watertable[x+1][y] ) #Aqui usa o
    método check_max para verificar o melhor valor a ser atribuído nesta posição da tabela.
    SE smith_watertable[x+1][y+1] > maior_valor:
            Maior_valor =  smith_watertable[x+1][y+1]
            x_index_maior_numero = x+1
            y_index_maior_numero = y+1


 #Calcula pontuação e monta o alinhamento das duas sequências
 #Começa o score a partir do ponto onde se localiza o maior valor dentro da tabela
 score = smith_watertable[x_index_maior_numero][y_index_maior_numero]
 id1 = ""
 id2 = ""

#Pega o valor da diagonal
path1 = smith_watertable[x_index_maior_numero - 1][y_index_maior_numero - 1]
#Pega o valor da esquerda
path2 = smith_watertable[x_index_maior_numero][y_index_maior_numero - 1]
#Pega o valor acima
path3 = smith_watertable[x_index_maior_numero -1][y_index_maior_numero]

 ENQUANTO path1 != 0 OU path2 != 0 OU path3 != 0:


    #Realiza as comparações e atribui os scores e a montagem do alinhamento das duas
sequências.
    SE x_index_maior_numero == 0 E y_index_maior_numero != 0:
            Acrescenta score com o valor de path2
            Diminui 1 de y_index_maior_numero
```

Append “-” na id2.

OU SE x_index_maior_numero != 0 E y_index_maior_numero == 0:

Acrescenta score com o valor de path3

Diminui 1 de x_index_maior_numero

Append “-” na id1.

OU SE path1 >= path2 E path1 >= path3:

Acrescenta score com o valor de path1

Append da letras das sequencias nas strings id1 e id2.

Diminui 1 de x_index_maior_numero e y_index_maior_numero

OU SE path2 > path1 E path2 >= path3:

Acrescenta score com o valor de path2

Append a letra da coluna na string id1 e inserção de gap “-” na id2.

Diminui 1 de y_index_maior_numero

SE NÃO:

Acrescenta score com o valor de path3

Append de um gap “-” na string id1 e a letra da sequencia em linha em id2.

Diminui 1 de x_index_maior_numero e y_index_maior_numero

Atribui as path1, path2 e path3 do próximo elemento e verifica novamente no loop.

#imprime a tabela final preenchida

Imprime smith_watertable

#Imprime a sequencia 1 (Invertida pois o cálculo na tabela é feita do fim para o inicio)

Imprime reverse(id1)

#Imprime a sequencia 2 (Invertida pois o cálculo na tabela é feita do fim para o inicio)

imprime reverse(id2)

#E por fim checa finalmente a identidade do alinhamento chamando a função check_identity e coloca na lista similarity_id.

Atribui identity_score = check_identity(id1,id2)

#Retorna o score obtido na tabela

retorna score

SE NÃO:

Retorna Erro

Resultado Encontrado:

#Tabela Smith waterman preenchida (** Muito grande..)

#As duas sequências alinhadas.
AEL
AER
Score table (Smith Waterman): 10
Score identidade: 1

Não foi possível relatar alguma semelhança do homo sapiens com a espécie em questão.