Introdução à biologia computacional

Projeto 1: alinhamento de sequencias

Aluno: Clerton Ribeiro

craf@cin.ufpe.br

Índice

- O Algoritmo
- Ler entrada de dados
- Criar as matrizes
- Inicializar as matrizes
- Preencher as matrizes
- Computar o alinhamento
- Problemas

Algoritmo

- Ler entrada de dados;
- Criar as matrizes;
- Inicializar as matrizes;
- Preencher as matrizes;
- Computar o alinhamento;

Algoritmo

```
s,t,align_type,parameters,prior =
readInput("input.data")

m,n,matrix,traceback = createMatrix(s,t)

matrix =
initializeMatrix(matrix,m,n,parameters,align_type)

matrix,traceback =
fillMatrix(matrix,traceback,m,n,parameters,align_type)

result =
computeAlignment(matrix,traceback,m,n,align_type)

print result
```

Ler entrada de dados

- Formato de entrada:
  - Sequencia 1;
  - Sequencia 2;
  - Tipo de alinhamento (global, semiglobal,local);
  - Match, Substituição, Inserção e Remoção;
  - Prioridades;

Ler entrada de dados

- input1.data

```
ACTCG
ACAGTAG
global
1 0 -1 -1
M S I R
```

Criar as matrizes

- Matriz com os *scores*
  - Matriz nxm com todos os valores zero;
- Matriz com o traceback
  - Indicação de qual célula o *score* foi proveniente;
  - Mesma dimensão da matriz com os *scores;*

Criar as matrizes

```
def createMatrix(s,t):

    m = len(s)

    n = len(t)


    matrix = [[0 for col in range(m+1)] for row in range(n+1)]

    traceback = [[0 for col in range(m+1)] for row in
    range(n+1)]


    return m,n,matrix,traceback
```

Inicializar as matrizes

- Inicialização dos alinhamentos locais e semi-globais com primeiras linhas e colunas zeradas.
  - Como a matriz já foi criada com valores zerados...
- Inicialização do alinhamento global:
  - Multiplicação das penalidades pelo indice da primeira linha e primeira coluna.

Inicializar as matrizes

```
def initializeMatrix(matrix,m,n,parameters,align_type):

if align_type == 'global':
    for index in range(0,m+1):
                matrix[0][index] = index*int(parameters[2])

    for index in range(0,n+1):
                matrix[index][0] = index*int(parameters[3])

    return matrix
else:
    return matrix
```

Preencher as matrizes

- O preenchimento da matriz de scores é similar para os três alinhamentos;
  - M(i,j) = max {M(i,j-1)+I,M(i-1,j-1)+p(i,j),M(i-1,j)+R}
  - O alinhamento local considera um quarto fator: o zero!
- O preenchimento da matriz de *traceback* considera por proveniência:
  - Para célula de cima 1
  - Para a célula a esquerda -1
  - Para a célula na diagonal superior 0

Preencher as matrizes

```
def fillMatrix(matrix,traceback,m,n,parameters,align_type):


    for i in range(1,n+1):
                for j in range(1,m+1):
                        left = matrix[i][j-1] + int(parameters[2])
                        top = matrix[i-1][j] + int(parameters[3])
                        penalty = 0
                        if s[j-1] == t[i-1]:
                                penalty = int(parameters[0])
                        else:
                                penalty = int(parameters[1])
                        both = matrix[i-1][j-1]+penalty
```

Preencher as matrizes

```
...
best = left
path = -1
if top > best:
        best = top
        path = 1
if both > best:
        best = both
        path = 0
if align_type == 'local':
        if 0 > best:
                best = 0
matrix[i][j] = best
traceback[i][j] = path

return matrix,traceback
```

Computar o alinhamento

- Usar a matriz com o *traceback;*
- Para o local e global, procurar na matriz a posição com o maior *score*:
  - Será a posição inicial da matriz de *traceback;*
- Se vier da diagonal, repete a string da célula correspondente, caso contrário, é um gap!
  - A diferença linha/coluna com as dimensões no inicio/fim indicam gap também.

Computar o alinhamento

```
def computeAlignment(matrix,traceback,m,n,align_type):
    i = n
    j = m
    result = []
    if (align_type == 'local') or (align_type == 'semiglobal'):
                best = matrix[0][0]
                for k in range(0,n+1):
                        for l in range(0,m+1):
                                if matrix[k][l] > best:
                                        best = matrix[k][l]
                                        i = k
                                        j = l
                if i < n:
                        result.append((n-i)*'-')
                if j < m:
                        result.append((m-j)*'-')
```

Computar o alinhamento

```
while i > 0 and j > 0 :
            if traceback[i][j] == 0:
                  if align_type == 'local' and matrix[i][j] == 0:
                        break
                  result.append(s[j-1])
                  i-=1
                  j-=1
            elif(traceback[i][j] == 1):
                  result.append("-")
                  i-=1
            elif(traceback[i][j] == -1):
                  result.append("-")
                  j-=1
if i > 0:
            result.append(i*'-')
if j > 0:
            result.append(j*'-')
result = "".join(result)
return result[::-1]
```

Problemas

- Faltou inserir o sistema de prioridades;
- Verificar o algoritmo de alinhamento semiglobal;
- Testar com mais sequencias.

Introdução à biologia computacional

Projeto 1: alinhamento de sequencias

Aluno: Clerton Ribeiro

craf@cin.ufpe.br