Elementy Bioinformatyki

Projekt - sprawozdanie

Zestawienia optymalne 2 sekwencji II

Anna Cholewczyńska 143198
Kamil Gersten 143220
Dominik Stefaniak 143133

1 Opis programu

Po uruchomieniu programu, z pliku *input.txt* znajdującego się w tym samym folderze co program, pobrane zostają dwa wyrazy (sekwencje nukleotydowe).

Następnie program oblicza tablicę odległości edycyjnych (D), oraz tablice A, B, C oraz S, potrzebne do wyznaczenia podobieństwa wyrazów.

Parametry takie jak wartość metryki, wartość podobieństwa znaków czy funkcja kary za przerwy są zdefiniowane w kodzie programu.

Następnie program wypisuje na konsolę sekwencje wejściowe, wartość podobieństwa wyrazów i ich optymalne dopasowanie w algorytmie z funkcją kary.

Wywołana jest metoda wyznaczająca dopasowanie na podstawie algorytmu Hirshberga.

2 Zastosowane algorytmy

Przy wypełnianiu tablicy odległości edycyjnych D skorzystano z następujących wzorów:

1. Brzeg tabeli:
$D(0,0)=0$, $D(0,j)=\Sigma_{k=1,\ldots,j} d(\text{‘–‘},w[k])$, $D(i,0)=\Sigma_{k=1,\ldots,i}\, d(u[k],\text{‘–‘})$

2. Środek tabeli $(i,j>0)$:
$D(i,j)=\min\{D(i-1,j-1)+d(u[i],w[j]),\ D(i,j-1)+d(\text{‘–‘},w[j]),$
$D(i-1,j)+d(u[i],\text{‘–‘})\}.$

, gdzie *i* i *j* to długości słów odpowiednio pierwszego i drugiego.

Funkcja *d(a,b)* zdefiniowana jest w ten sposób, że wynosi 0, gdy *a=b*, oraz 1 w innych przypadkach. Wartość tą można zmienić w kodzie programu.

Wartość odległości edycyjnej odczytywana jest z komórki D(i,j).

Optymalne dopasowanie wyznaczono za pomocą algorytmu

- w polach tabeli ustawiamy wskaźniki:
  - na brzegu $D(0,i) \rightarrow D(0,i-1)$ i $D(i,0) \rightarrow D(i-1,0)$ dla $i>0$
  - w środku $D(i,j) \rightarrow$ te z $\{D(i-1,j-1), D(i,j-1), D(i-1,j)\}$, które realizują minimum w punkcie 2.
- odczytujemy dopasowanie „od końca" idąc po wskaźnikach wzdłuż dowolnej drogi z komórki $D(n,m)$ do $D(0,0)$.

2.1 Optymalne dopasowanie z funkcją kary

Dla wypełniania tablic A, B, C, S, skorzystano z wzorów:

$$S(0,0)=0,\ S(i,0)=S(0,i)=B(i,0)=A(0,i)=p(i) \text{ dla } i>0 \text{ oraz}$$

$$A(i,j)=\max_{k\in\{0,j-1\}}\{\max\{B(i,k),C(i,k)\}+p(j-k)\}$$
$$B(i,j)=\max_{k\in\{0,i-1\}}\{\max\{A(k,j),C(k,j)\}+p(i-k)\}$$
$$C(i,j)=S(i-1,j-1)+s(u[i],w[j])$$
$$S(i,j)=\max\{A(i,j),B(i,j),C(i,j)\}$$

– dla $i,j>0$

, gdzie *p(x)* to funkcja kary za przerwy, zdefiniowana w kodzie.

Wartość podobieństwa sekwencji odczytywana jest z pola *S(i,j)*.

Dopasowanie optymalne dla podobieństw wyrazów wyznaczane jest poprzez wyznaczanie najdłuższej ścieżki wiodącej z *S(i,j)*, do *S(0,0)I.* Wykorzystano tutaj identyczny algorytm co w przypadku odległości edycyjnych, z tą różnicą że tutaj szukano maksimum zamiast minimum.

2.2 Algorytm Hirshberga

Algorytm Hirshberga jest opisywany jako wersję „dziel i rządź" algorytmu Needlemana-Wunscha (wykorzystanego i opisanego w poprzednim przypadku). Jest to algorytm rekurencyjny, wywoływany tak długo, aż sekwencje wejściowe osiągną długość 1.

Definiując funkcję *NWScore(X,Y),* jako funkcję zwracającą ostatni wiersz macierzy S algorytmu Needlemana-Wunscha:

```
function NWScore(X,Y)
  Score(0,0) = 0
  for j=1 to length(Y)
    Score(0,j) = Score(0,j-1) + Ins(Yj)
  for i=1 to length(X)
    Score(i,0) = Score(i-1,0) + Del(Xi)
    for j=1 to length(Y)
      scoreSub = Score(i-1,j-1) + Sub(Xi, Yj)
      scoreDel = Score(i-1,j) + Del(Xi)
      scoreIns = Score(i,j-1) + Ins(Yj)
      Score(i,j) = max(scoreSub, scoreDel, scoreIns)
    end
  end
  for j=0 to length(Y)
    LastLine(j) = Score(length(X),j)
  return LastLine
```

, gdzie Sub, Del i Ins to są metryki za podmienienie, usunięcie i wstawienie znaku.

Sam algorytm Hirshberga wygląda następująco:

```
function Hirschberg(X,Y)
  Z = ""
  W = ""
  if length(X) == 0
    for i=1 to length(Y)
      Z = Z + '-'
      W = W + Y_i
    end
  else if length(Y) == 0
    for i=1 to length(X)
      Z = Z + X_i
      W = W + '-'
    end
  else if length(X) == 1 or length(Y) == 1
    (Z,W) = NeedlemanWunsch(X,Y)
  else
    xlen = length(X)
    xmid = length(X)/2
    ylen = length(Y)

    ScoreL = NWScore(X_1:xmid, Y)
    ScoreR = NWScore(rev(X_xmid+1:xlen), rev(Y))
    ymid = arg max ScoreL + rev(ScoreR)

    (Z,W) = Hirschberg(X_1:xmid, y_1:ymid) + Hirschberg(X_xmid+1:xlen, Y_ymid+1:ylen)
  end
  return (Z,W)
```

, tj.:

- Dla długości jednej z sekwencji wejściowej = 0, sekwencja wyjściowa odpowiadająca tej równej 0 będzie wypełniona znakami ‘-‘, a druga sekwencja będzie przepisana
- Dla długości jednej z sekwencji wejściowej = 1, wykonywany jest zwyczajny algorytm Needlemana-Wunscha
- W innych przypadkach:
    - Sekwencja wejściowa X, dzielona jest w połowie (rys 1,2,3)
    - Na podstawie sumy ostatnich rzędów obu połówek sekw. X, wyznaczane jest miejsce rozdzielenia sekwencji Y. (Rys. 4)
    - Na obu zestawach połówek wywoływany jest algorytm Hirshberga.
    - Sekwencjami wyjściowymi są sekwencje powstałe na skutek złączenia sekwencji wyjściowych funkcji Hirshberga na obu zestawach połówek

```
      T   A   T   G   C
  0  -2  -4  -6  -8 -10
A -2  -1   0  -2  -4  -6
G -4  -3  -2  -1   0  -2
T -6  -2  -4   0  -2  -1
A -8  -4   0  -2  -1  -3
C -10 -6  -2  -1  -3   1
G -12 -8  -4  -3   1  -1
C -14 -10 -6  -5  -1   3
A -16 -12 -8  -7  -3   1
```

Rys. 1

```
      T   A   T   G   C
  0  -2  -4  -6  -8 -10
A -2  -1   0  -2  -4  -6
G -4  -3  -2  -1   0  -2
T -6  -2  -4   0  -2  -1
A -8  -4   0  -2  -1  -3
```

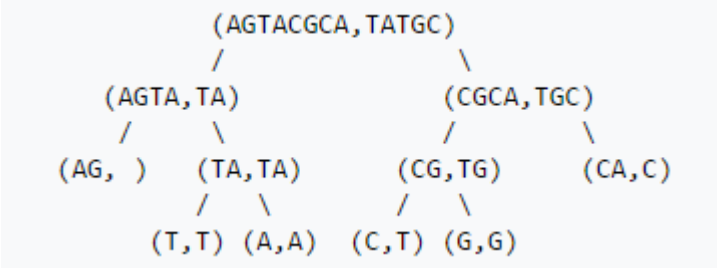
Rys. 2

```
      C   G   T   A   T
  0  -2  -4  -6  -8 -10
A -2 -1  -3  -5  -4  -6
C -4  0  -2  -4  -6  -5
G -6 -2   2   0  -2  -4
C -8 -4   0   1  -1  -3
```

Rys. 3

```
ScoreL      = [ -8 -4  0 -2 -1 -3 ]
rev(ScoreR) = [ -3 -1  1  0 -4 -8 ]
Sum         = [-11 -5  1 -2 -5 -11]
```

Rys. 4



Rys. 5

3 Bibliografia

- *Liniowe dopasowanie dwóch sekwencji*, Materiały pomocnicze do przedmiotu *„Elementy Bioinformatyki”*, Krzysztof Giaro
- *Computational Biology*, Slajdy wykładowe do przedmiotu „*Elementy Bioinformatyki”*, Krzysztof Giaro
- *Hirschberg's algorithm,* https://en.wikipedia.org/wiki/Hirschberg's_algorithm