به نام خداوند بخشنده و مهربان

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

درس مقدمه‌ای بر بیوانفورماتیک

گزارش سوال برنامه نویسی تمرین اول

استاد درس: دکتر زینعلی

نام دانشجو:

روزبه قاسمی ۹۵۳۱۴۲۴

بهار ۱۳۹۹

مقدمه

برای پیاده سازی Affine gap طبق ویدیو موجود در کانال پیاده سازی انجام گرفته است. به این صورت که ماتریس های M، L_x و L_y را از طریق فرمول های موجود در آموزش بدست می‌آوریم. سپس با دادن دو رشته ورودی و مقادیر Match و Mismatch و gap open و gap extension به کلاس Affine gap ابتدا ماتریس را تشکیل می‌دهیم سپس با بدست آوردن ماتریس از طریق متد traceback اقدام به پیدا کردن همتراز سازی می‌کنیم.

کلاس Affine Gap

در این کلاس ابتدا از تابع آماده __init__ مقادیر گرفته شد را self میکنیم تا بتوانیم در ادامه از آنها استفاده کنیم.

```
def __init__(self, seq1, seq2, match, mismatch, gap_open, gap_extension):
    self.seq1 = seq1
    self.seq2 = seq2
    self.match = match
    self.mismatch = mismatch
    self.gap_open = gap_open
    self.gap_extension = gap_extension
    self.alingment()
```

سپس یک تابع به اسم matcher تعریف می‌کینم تا در ادامه هنگامی که میخواهیم ببینیم دو Residue های موجود با هم match است یا mismatch و بر اساس آن امتیاز دهی صورت بگیرد.

```
def matchchar(self, seq1, seq2, i, j):
    if seq2[i - 1] == seq1[j - 1]:
        return self.match
    else:
        return self.mismatch
```

تابع Alignment

سپس طبق اسلاید های زیر ماتریس ها را مقدار دهی می‌کنیم:

هم‌ترازی سراسری با استفاده از affine gap

$$M(i,j)=\max\begin{cases}M(i-1,j-1)+s(x[i],y[j])\\ I_x(i-1,j-1)+s(x[i],y[j])\\ I_y(i-1,j-1)+s(x[i],y[j])\end{cases}$$

$$I_x(i,j)=\max\begin{cases}M(i-1,j)-d\\ I_x(i-1,j)-e\end{cases}$$

$$I_y(i,j)=\max\begin{cases}M(i,j-1)-d\\ I_y(i,j-1)-e\end{cases}$$

هم‌ترازی سراسری با استفاده از affine gap

- مقداردهی اولیه

$$M(0,0)=0$$
$$I_x(i,0)=-d+(i-1)e$$
$$I_y(0,j)=-d+(j-1)e$$

other cells in top row and leftmost column $=-\infty$

- ردیابی هم‌ترازی (traceback)
  - شروع از بزرگترین $M(n,m), I_x(n,m), I_y(n,m)$
  - خاتمه در هر کدام از $M(0,0), I_x(0,0), I_y(0,0)$
  - بین سه جدول می‌توان حرکت کرد.

در نهایت پیاده سازی ما به شکل زیر خواهد بود:

```
def alingment(self):
    dim_seq1 = len(self.seq1) + 1
    split_seq1 = split(self.seq1)
    dime_seq2 = len(self.seq2) + 1
    split_seq2 = split(self.seq2)
    M = make_matrix(dime_seq2, dim_seq1)
    Ix = make_matrix(dime_seq2, dim_seq1)
    Iy = make_matrix(dime_seq2, dim_seq1)
    M[0][0] = 0
    Ix[0][0] = self.gap_open - self.gap_extension
    Iy[0][0] = self.gap_open - self.gap_extension

    for i in range(1, dime_seq2):
        M[i][0] = -Infinity
        Ix[i][0] = self.gap_open + ((i - 1) * self.gap_extension)
        Iy[i][0] = -Infinity

    for j in range(1, dim_seq1):
        M[0][j] = -Infinity
        Ix[0][j] = -Infinity
        Iy[0][j] = self.gap_open + ((j - 1) * self.gap_extension)

    for i in range(1, dime_seq2):
        for j in range(1, dim_seq1):
            M[i][j] = self.matchchar(self.seq1, self.seq2, i, j) + max(
                M[i - 1][j - 1],
                Ix[i - 1][j - 1],
                Iy[i - 1][j - 1]
            )

            Ix[i][j] = max(
                self.gap_open + M[i - 1][j],
                self.gap_extension + Ix[i - 1][j],
            )

            Iy[i][j] = max(
                self.gap_open + M[i][j - 1],
                self.gap_extension + Iy[i][j - 1]
            )

    Ix_score = (Ix[i][j])
    Iy_score = (Iy[i][j])
    M_score = (M[i][j])
    optimal_score = max(Ix_score, Iy_score, M_score)
    print(optimal_score)
    self.traceback(M, Ix, Iy, self.gap_open, self.gap_extension, split_seq1, split_seq2, i, j)
```

تابع Alignment

```
def traceback(self, M, Ix, Iy, gap_open, gap_extension, seq1, seq2, row, col):
    GAFirst = ""
    GASecond = ""
    i = row
    j = col
    if M[i][j] > Ix[i][j] and M[i][j] > Ix[i][j]:
        current_matrix = 'm'
        optimalScore = M[i][j]
        print("Optimal Score for M is  = " + str(optimalScore))

    elif Iy[i][j] > Ix[i][j] and Iy[i][j] > M[i][j]:
        current_matrix = 'y'
        optimalScore = Iy[i][j]
        print("Optimal Score for Iy is = " + str(optimalScore))

    else:
        current_matrix = 'x'
        optimalScore = Ix[i][j]
        print("Optimal Score for Ix is = " + str(optimalScore))

    while i > 0 or j > 0:
        if current_matrix == 'm':
            GAFirst += seq1[j - 1]
            GASecond += seq2[i - 1]
            print(GASecond)
            if seq2[i - 1] == seq1[j - 1]:
                penalty = self.match
            else:
                penalty = self.mismatch

            if (Iy[i - 1][j - 1] + penalty) == M[i][j]:
                print("section 1")
                print(M[i - 1][j - 1])
                print(penalty)
                print(M[i][j])
                print("End section 1")

                i -= 1
                j -= 1
                current_matrix = 'y'
            elif (Ix[i - 1][j - 1] + penalty) == M[i][j]:
                print("section 2")
                print(Ix[i - 1][j - 1])
                print(penalty)
                print(M[i][j])
                print("End section 2")
                i -= 1
                j -= 1
                current_matrix = 'x'
            elif (M[i - 1][j - 1] + penalty) == M[i][j]:
                print("section 3")
                print(Iy[i - 1][j - 1])
```

```
            print(penalty)
            print(M[i][j])
            print("End section 3")
            i -= 1
            j -= 1
            current_matrix = 'm'

        elif current_matrix == 'x':
            GAFirst += "-"
            GASecond += seq2[i - 1]

            if (Ix[i - 1][j] + gap_extension) == Ix[i][j]:
                print("Sub section 1 for X: ")
                print(Ix[i - 1][j])
                print(gap_extension)
                print(Ix[i][j])
                i -= 1
                current_matrix = 'x'
            elif (M[i - 1][j] + gap_open) == Ix[i][j]:
                print("Sub section 2 for X: ")
                print(M[i - 1][j])
                print(gap_open)
                print(Ix[i][j])
                i -= 1
                current_matrix = 'm'

        elif current_matrix == 'y':
            GAFirst += seq1[j - 1]
            GASecond += "-"

            if (Iy[i][j - 1] + gap_extension) == Iy[i][j]:
                print("Sub section 1 for Y: ")
                print(Iy[i][j - 1])
                print(gap_extension)
                print(Iy[i][j])
                j -= 1
                current_matrix = 'y'
            elif (M[i][j - 1] + gap_open) == Iy[i][j]:
                print("Sub section 2 for Y: ")
                print(M[i][j - 1])
                print(gap_open)
                print(Iy[i][j])
                j -= 1
                current_matrix = 'm'
    GAFirst = GAFirst[::-1]
    GASecond = GASecond[::-1]

    print("Optimal Alignment:")
    print(GAFirst)
    print(GASecond)
```

برای پیاده سازی این قسمت باید به صورت بازگشتی از انتها به ابتدا بیایم و مطابق دو اسلایدی که در بالاتر ذکر شد پیاده سازی صورت گرفته است.

تست برنامه

برای تست دو رشته ورودی را همراه با امتیازاتی که در مقدمه ذکر شد به آن می‌دهیم:

```
input_seq = ['ACACT', 'AAT']

model = Affine_gap(input_seq[0], input_seq[1], 1, -1, -4, -1)
```

خروجی آن به شکل زیر می‌باشد:

```
Optimal Alignment:
ACACT
AA--T
```

پایان