زهرا گنجی 9531802

گزارش تمرین 5

در این برنامه قصد داریم هم ترازی سراسری بین دو رشته را با روش affine gap بدست آوریم.

ابتدا دو ورودی که به صورت دو رشته هستند را میگیریم و طول آن ها را بدست می آوریم.M طول رشته اول و N طول رشته ی دوم است.

در روش هم ترازی سراری با affine gap به جای یک ماتریس به 3 ماتریس M+1*N+1 احتیاج داریم.

- m[i,j] بهترین امتیاز برای حالتی که x[i] با y[j]هم تراز شود. در این برنامه با matrix نشان داده می شود.
- Ix[i,j] بهترین امتیاز برای حالتی که x[i] با gapهم تراز شود.
- Iy[i,j] بهترین امتیاز برای حالتی که y[j] با gap هم تراز شود.

این سه ماتریس را در برنامه با سه numpy array با مقادیر اولیه ی 0 ایجاد کرده ایم.

```
input1 = raw_input("Enter first string: ")
input2 = raw_input("Enter second string: ")

M = len(input1)
N = len(input2)

matrix = numpy.zeros(shape=(M+1,N+1))
I_x = numpy.zeros(shape=(M+1,N+1))
I_y = numpy.zeros(shape=(M+1,N+1))
```

مقدار affine gap را این گونه بدست می آوریم:

$$w(g) = -d - (g-1)e$$

هزینه شروع — طول — هزینه گسترش

که در برنامه هزینه ی شروع با `oGap_penalty` و هزینه ی گسترش با `eGap_penalty` نشان داده شده است.
به دلیل اینکه مقادیر `mismatch_penalty` و `gap_penalty` منفی تعریف شده اند، در فرمول `affine gap` در کد، از منفی استفاده نکرده ام.

مقداردهی اولیه ماتریس ها:

- مقداردهی اولیه

$$M(0,0) = 0$$
$$I_x(i,0) = -d + (i-1)e$$
$$I_y(0,j) = -d + (j-1)e$$
$$\text{other cells in top row and leftmost column } = -\infty$$

```
matrix[0,0] = 0
for i in range(M+1):
    I_x[i,0] = oGap_penalty + (i-1)*eGap_penalty
    if i !=0:
        array = []
        array.append("I_x[%d,%d]" % (i-1,0))
        thisdict["I_x[%d,%d]" % (i,0)] = array

for i in range(1,M+1):
    I_y[i, 0] = float('-inf')
    matrix[i, 0] = float('-inf')

for j in range(N+1):
    I_y[0,j] = oGap_penalty + (j-1)*eGap_penalty
    if j !=0:
        array = []
        array.append("I_y[%d,%d]" % (0,j-1))
        thisdict["I_y[%d,%d]" % (0,j)] = array


for j in range(1,N+1):
    I_x[0, j] = float('-inf')
    matrix[0, j] = float('-inf')
```

سپس سطر به سطراز سطر اول شروع به مقداردهی و پر کردن ماتریس ها می کنیم ، بدین صورت:

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

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

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

S(x[i],y[j]) یا امتیاز matchاست یا mismatch دو کاراکتر از رشته های ورودی است و در برنامه آن را با score نشان می دهیم.

با پر کردن هر خانه ی ماتریس ها، یک آرایه از خانه هایی که برای رسیدن به این خانه استفاده کردیم (یعنی خانه ی با مقدار max) نگه داری میکنیم و این آرایه و این خانه را در یک dictionary نگه داری میکنیم به نام thisdict.(مقادیر inf- را در thisdict نگه داری نمی کنیم چون در traceback استفاده نمی شوند.

```
for i in range(1,M+1):
    for j in range(1,N + 1):
        if input1[i - 1] == input2[j - 1]:
            score = match_award
        else:
            score = mismatch_penalty

        matrix[i, j] = max(matrix[i - 1, j - 1] + score, I_x[i - 1, j - 1] + score, I_y[i - 1, j - 1] + score)
        if (matrix[i,j] != float('-inf')):
            matrix_array = []
            if matrix[i, j] == matrix[i - 1, j - 1] + score:
                matrix_array.append("matrix[%d,%d]" % (i - 1, j - 1))
            if matrix[i, j] == I_x[i - 1, j - 1] + score:
                matrix_array.append("I_x[%d,%d]" % (i - 1, j - 1))
            if matrix[i, j] == I_y[i - 1, j - 1] + score:
                matrix_array.append("I_y[%d,%d]" % (i - 1, j - 1))
            thisdict["matrix[%d,%d]" % (i, j)] = matrix_array

        I_x[i, j] = max(matrix[i - 1, j] + oGap_penalty, I_x[i - 1, j] + eGap_penalty)
        if (I_x[i, j] != float('-inf')):
            I_x_array = []
            if I_x[i, j] == matrix[i - 1, j] + oGap_penalty:
                I_x_array.append("matrix[%d,%d]" % (i - 1, j))
            if I_x[i, j] == I_x[i - 1, j] + eGap_penalty:
                I_x_array.append("I_x[%d,%d]" % (i - 1, j))
            thisdict["I_x[%d,%d]" % (i, j)] = I_x_array
```

سپس اگر این هم ترازی بیشتر از 1، optimal alignment داشت یعنی هر key درthisdict بیش تر از یک مقدار در آرایه ی value خود داشت، مقادیر را در dictionary های دیگر به نام dict از هم جدا میکنیم تا با traceback در dictionary های مختلف optimal alignmet های چند گانه بدست آیند.

سپس trace back را در هر dictionary انجام می دهیم .

- 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 هر مرحله با شروع از بزرگترین و قرار دادن آن در tb ، tb را به عنوان key در dictionary سرچ می کنیم و value آن را بدست می آوریم و کاراکتر بعدی در رشته های alignment را بدست می آوریم: اگر i در key با i در value یا j در key با j در value برابر بود، - کارکتر بعدی است، در غیر این صورت یک کاراکتر از ورودی اول به رشته ی alignment اول و یک کاراکتر از رشته ی دوم به رشته ی alignment دوم اضافه می گردد.

وسپس value را به جای tb قرار می دهیم و آن را به عنوان key سرچ می کنیم و این کار را ادامه می دهیم تا به یکی از matrix[0,0] یا I_x[0,0] یا I_y[0,0] برسیم.

سپس رشته های بدست آمده را reverse میکنیم چون از آخر به اول حرکت کرده بودیم. و optimal alignmentها بدست می آیند. و آن ها را در دیکشنری answer_dict قرار می دهیم و سپس چاپ می کنیم.

برای مثال با دو ورودی aat و acact، باM=3 و N=5 سه ماتریس 4*6 ، matrix و I_x و I_y را می سازیم. با انجام مراحل فوق روی آن، جواب برنامه به این صورت می گردد: