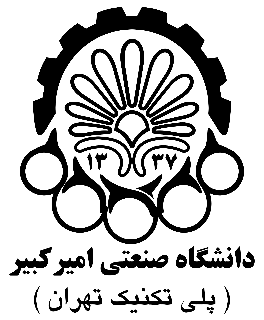
به نام خداوند بخشنده و مهربان



**درس مقدمه‌ای بر بیوانفورماتیک**

گزارش سوال برنامه نویسی تمرین اول

استاد درس: دکتر زینعلی

نام دانشجو:

روزبه قاسمی 9531424

بهار 1399

**مقدمه**

برای پیاده سازی Affine gap طبق ویدیو موجود در کانال پیاده سازی انجام گرفته است. به این صورت که ماتریس های M، Lx و Ly 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.

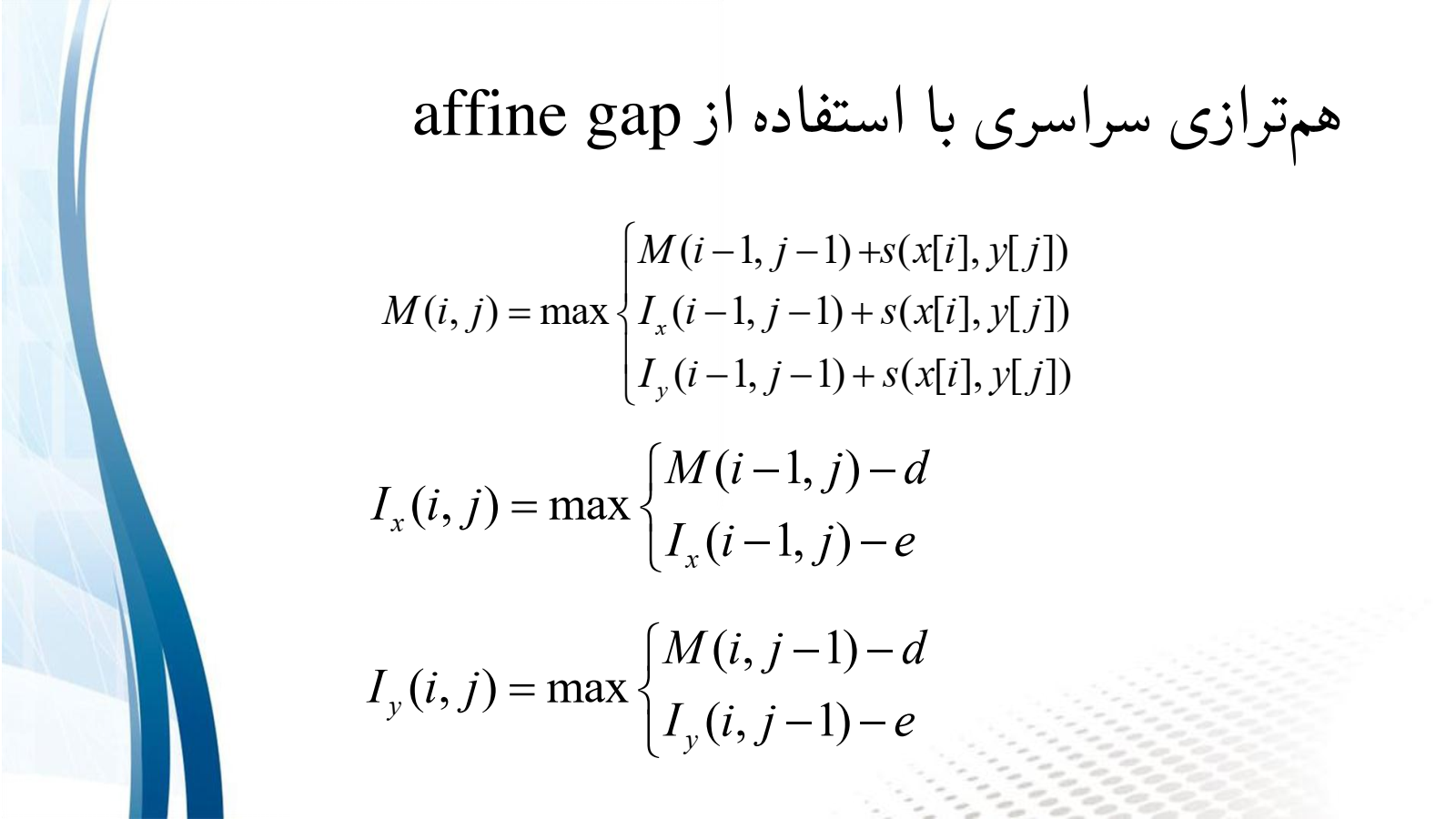
def \_\_init\_\_(self, seq1, seq2, match, mismatch, gap\_open, gap\_extension):  
 self.seq1 = seq1  
 self.seq2 = seq2  
 self.match = match  
 self.mismatch = mismatch  
 self.gap\_open = gap\_open  
 self.gap\_extension = gap\_extension  
 self.alingment()

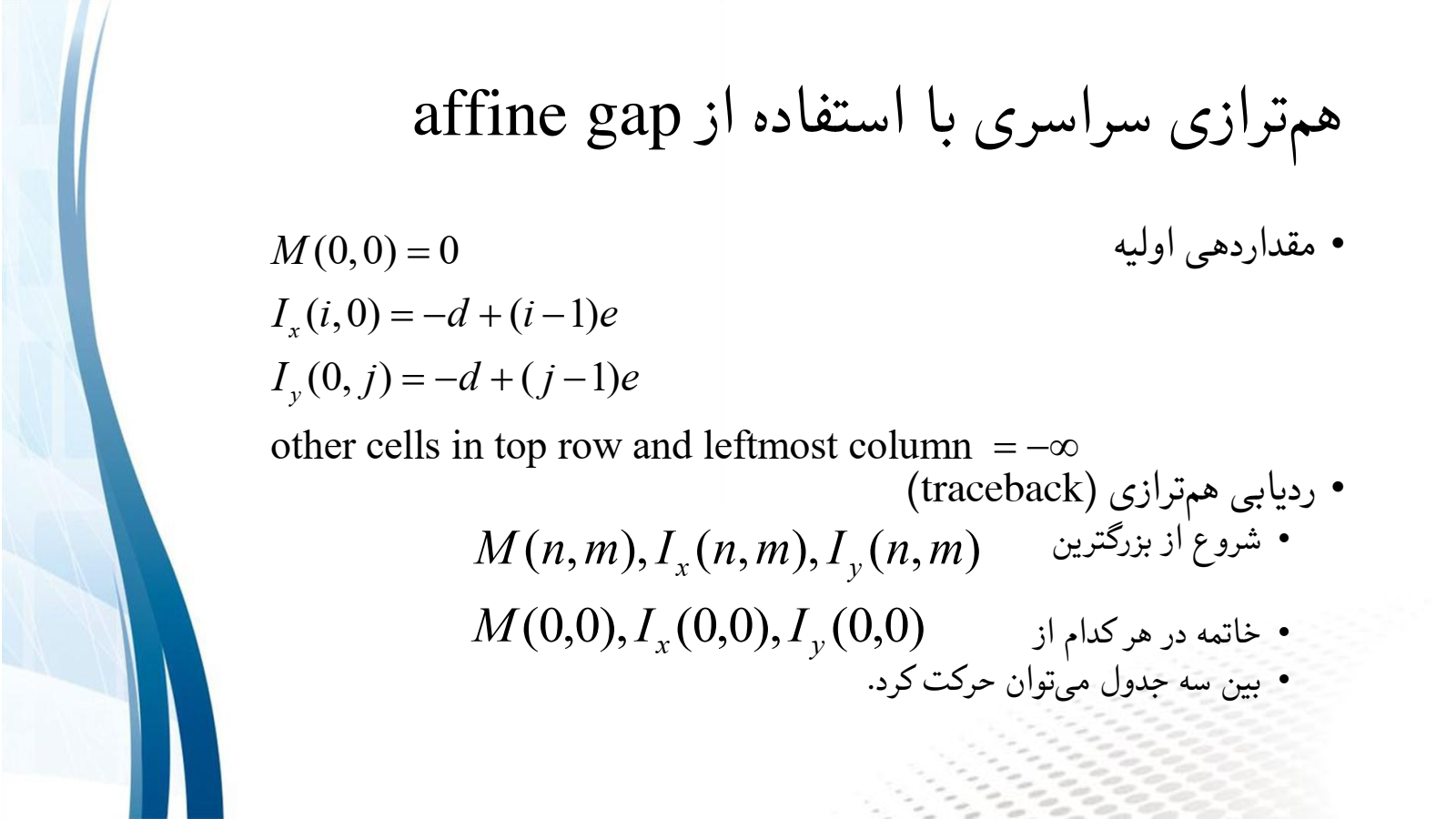
سپس یک تابع به اسم matcher تعریف می‌کینم تا در ادامه هنگامی که میخواهیم ببینیم دو Residue های موجود با هم match است یا mismatch و بر اساس آن امتیاز دهی صورت بگیرد.

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:





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**

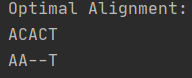
def traceback(self, M, Ix, Iy, gap\_open, gap\_extension, seq1, seq2, row, col):  
 GAFirst = ""  
 GASecond = ""  
 i = row  
 j = col  
 if M[i][j] > Ix[i][j] and M[i][j] > Ix[i][j]:  
 current\_matrix = 'm'  
 optimalScore = M[i][j]  
 print("Optimal Score for M is = " + str(optimalScore))  
  
 elif 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:

پایان