# همترازی چندگانه

- محدودیت زمان: ۵ ثانیه
- محدودیت حافظه: ۵۱۲ مگابایت

در این تمرین قرار است با زبان پایتون، همترازی چندگانه را با کمک الگوریتم Star-Alignment برای پروتئین‌ها پیاده‌سازی کنید و به صورت block-based نتایج همترازی را بهبود ببخشید.

برای راحتی مراحل مربوط به این پیاده‌سازی توضیح داده می‌شود: 1 - ابتدا همتزاری چندگانه میان دو به دوی دنباله‌ها را بدست آورید و یک ماتریس فاصله بسازید. فواصل را به کمک امتیازهای مربوطه بدست آورید (سومین خروجی تابع global_alignment امتیاز همترازی دوگانه را می‌دهد) و دنباله‌ی مرکزی را بدست آورید. پس از انتخاب دنباله‌ی مرکزی، دنباله‌ها را به با توجه به فاصله‌ی آنها با دنباله‌ی مرکزی به دنباله‌ی مرکزی align کنید. امتیاز این همترازی چندگانه را بدست آورید.

2- در مرحله‌ی بعد می‌بایست این همترازی را بهبود ببخشید. در این مرحله می‌بایست بلوک‌هایی که پتانسیل بهبود دارند را بیابید و با کمک همترازی چندگانه مرحله قبل، همترازی آنها را بدست آورید و جایگزین بلوک اولیه کنید. اگر امتیاز همترازی کلی بهبود یافت این بلوک را به صورت دائمی جایگزین کنید.

در انتخاب بلوک‌ها به موارد زیر توجه کنید:

- هر بلوک باید حداقل 2 ستون داشته باشد.
- بلوک‌هایی پتانسیل بهبود را دارند که دارای gap و یا mismatch باشند. (در هیچ بلوکی نباید یک ستون کامل از یک کاراکتر یکسان وجود داشته باشد.)

3- حال MSA و امتیاز آن را چاپ کنید. دقت کنید که MSA را با توجه به ترتیب ورودی چاپ کنید و برای محاسبه‌ی امتیاز MSA خود برای (gap,gap) امتیاز 0، برای gap هم -2 ، برای mismatch امتیاز -1 و برای match امتیاز 3 را در نظر بگیرید. برای راحتی کار لازم نیست که کد global alignment را خودتان بزنید و کافی است از کد زیر استفاده کنید: از مقدارهایی که در این کد داده شده است استفاده کنید یعنی (match = 3 و mismatch = -1 و gap = -2)

```
def global_align(x, y, s_match, s_mismatch, s_gap):
    A = []
    for i in range(len(y) + 1):
        A.append([0] * (len(x) + 1))
    for i in range(len(y) + 1):
        A[i][0] = s_gap * i
    for i in range(len(x) + 1):
        A[0][i] = s_gap * i
    for i in range(1, len(y) + 1):
        for j in range(1, len(x) + 1):
            A[i][j] = max(
                A[i][j - 1] + s_gap,
                A[i - 1][j] + s_gap,
                A[i - 1][j - 1] + (s_match if (y[i - 1] == x[j - 1] and y[i - 1] != '-
                    s_mismatch if (y[i - 1] != x[j - 1] and y[i - 1] != '-' and x[j -
                    s_gap if (y[i - 1] == '-' or x[j - 1] == '-') else 0)
            )
    align_X = ""
    align_Y = ""
    i = len(x)
```

```
j = len(y)
while i > 0 or j > 0:
    current_score = A[j][i]
    if i > 0 and j > 0 and (
            ((x[i - 1] == y[j - 1] and y[j - 1] != '-') and current_score == A[j -
            ((y[j - 1] != x[i - 1] and y[j - 1] != '-' and x[i - 1] != '-') and cu
                i - 1] + s_mismatch) or
            ((y[j - 1] == '-' or x[i - 1] == '-') and current_score == A[j - 1][i
    ):
        align_X = x[i - 1] + align_X
        align_Y = y[j - 1] + align_Y
        i = i - 1
        j = j - 1
    elif i > 0 and (current_score == A[j][i - 1] + s_gap):
        align_X = x[i - 1] + align_X
        align_Y = "-" + align_Y
        i = i - 1
    else:
        align_X = "-" + align_X
        align_Y = y[j - 1] + align_Y
```

```
            j = j - 1

    return (align_X, align_Y, A[len(y)][len(x)] )
```

دقت کنید که فقط برای global alignment از آن تابع میتوان استفاده کرد و بقیه ی برنامه را خودتان باید پیاده‌سازی کنید.

## ورودی

در خط اول عدد n را می‌گیرید که تعداد دنباله‌های ورودی را مشخص می‌کند. در n خط بعدی در هر خط یک دنباله می‌گیرید.

## خروجی

خروجی شما MSA شما است و امتیاز آن و با نگاه به مثال‌ها به راحتی قابل فهم است. (فقط دقت کنید که دنباله ها را با توجه به ترتیب ورودی چاپ کنید)

## مثال

### ورودی نمونه ۱

```
4
TYIMREAQYESAQ
TCIVMREAYE
YIMQEVQQER
WRYIAMREQYES
```

### خروجی نمونه ۱

```
51
-TYI-MREAQYESAQ
-TCIVMREA-YE---
--YI-MQEVQQER--
WRYIAMRE-QYES--
```

## ورودی نمونه ۲

```
5
TAGCTACCAGGA
CAGCTACCAGG
TAGCTACCAGT
CAGCTATCGCGGC
CAGCTACCAGGA
```

## خروجی نمونه ۲

```
240
TAGCTA-C-CAGGA
CAGCTA-C-CAGG-
TAGCTA-C-CA-GT
CAGCTATCGC-GGC
CAGCTA-C-CAGGA
```