زهرا گنجی 9531802

گزارش سوال چهارم تمرین سری دوم بیو انفورماتیک:

در این تمرین قصد داریم با استفاده از برنامه‌ای که در تمرین قبل نوشته بودیم، الگوریتم Star را پیاده‌سازی کنیم. به عنوان ورودی، به الگوریتم 4 توالی دلخواه با حداقل طول 6 بدهیم و خروجی را که یک همترازی چندگانه هست، چاپ می کنیم.

در این برنامه ، کد نوشته شده در تمرین قبل را در تابع alignment نگه داری می کنیم. که به عنوان ورودی 2 رشته را می گیرد و در خروجی، align شده ی رشته ی اول و دوم و score محاسبه شده برای این alignment را باز می گرداند.

تفاوتی که در این تمرین روی کد تمرین قبل دادیم به این شکل است که از بین چند حالت alignment با بیش ترین score ای که در خروجی آن برنامه داشتیم، تنها باید alignment ای در این تمرین در خروجی این تابع بازگردانده شود که در رشته های align شده، gap ها در سمت راست ترین حالت ممکن نسبت به بقیه ی رشته ها باشند.

برای هندل کردن این موضوع در تابع alignment:

1. index گپ ها در رشته های اول جفت رشته های خروجی الگوریتم را در دیکشنری gap_index_state و index گپ ها در رشته های دوم را در gap_index_state2 قرار می دهیم.

```
gap_index_state = {}
gap_index_state2 = {}
for state in answer_dict:
    gap_index_state[state] = []
    for i in range(len(state)):
        if state[i] == "-":
            gap_index_state[state].append(i)

    for i in range(len(answer_dict[state])):
        gap_index_state2[answer_dict[state][i]] = []
        for j in range(len(answer_dict[state][i])):
            if answer_dict[state][i][j] == "-":
                gap_index_state2[answer_dict[state][i]].append(j)
```

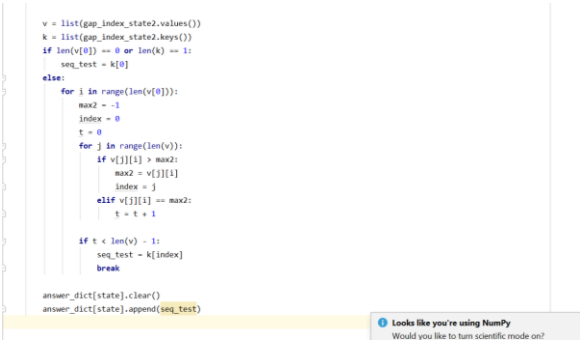
برای مثال برای دو رشته ی ATCGTTGAT و ATCGGAAT ، تمام alignmentهای با بیش ترین امتیاز به این صورت است:

```
ATCGTTGAT
ATCG-GAAT
****
ATCGTTGAT
ATCGG-AAT
****
ATCGTTGAT
ATCGGA-AT
****
```

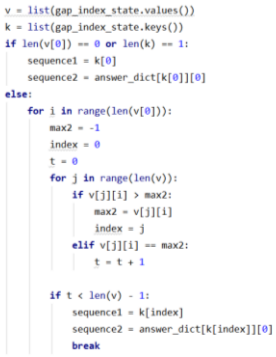
و gap_index_state و gap_index_state2 به این صورت هستند:

```
gap_index_state:
{'ATCGTTGAT': []}
gap_index_state2:
{'ATCG-GAAT': [4], 'ATCGG-AAT': [5], 'ATCGGA-AT': [6]}
```

2. سپس اول از بین رشته های اول در جفت رشته های با بیش ترین امتیاز alignment، رشته ای که index گپ در آن بیش تر باشد را نگه می داریم .



3. سپس از بین رشته های دوم متناظر با آن رشته ی اول انتخاب شده، رشته ای که index گپ بالاتری داشته باشد را انتخاب می کنیم. و رشته ی اول و دوم انتخاب شده را در sequence1 و sequence2 قرار می دهیم و تنها امتیاز این همترازی را بدست می آوریم .



خروجی alignment دو رشته ی فوق:

در برنامه اصلی ابتدا 4 رشته ی ورودی از کانسول دریافت می کنیم و در آرایه ی seq قرار می دهیم.

برای مثال فرض می کنیم این 4 رشته عبارت اند از:

- S[0] = ATCGGAAT
- S[1] = ATCGTTGAT
- S[2] = ATAGGTTCAT
- S[3] = AGGAACCT

الگوریتم star:

1. برای الگوریتم star ابتدا باید رشته ی مرکزی را تعیین کنیم. برای این کار ابتدا برای رشته ی اول و تمام رشته های دیگر غیر از آن ،امتیاز هم ترازی را با تابع alignment بدست می آوریم و با هم جمع می کنیم و در sumscore قرار می دهیم، سپس برای رشته ی دوم و سوم و چهارم نیز همین کار را انجام می دهیم . هر کدام sumscore بیش تری داشتند به عنوان رشته ی مرکزی انتخاب می شود و اندیس آن رشته در آرایه ی seq در i_max_sum نگه داری می شود.

```
354
355  #  Select Center sequence
356  max_sum = float('-inf')
357  i_max_sum = 0
358  for i in range(len(seq)):
359      sumscore = 0
360      for j in range(len(seq)):
361          if i != j:
362              score,align_str1,align_str2 =alignment(seq[i], seq[j])
363              sumscore = sumscore + score
364
365      if sumscore > max_sum:
366          i_max_sum = i
367          max_sum = sumscore
368
```

امتیاز های محاسبه شده در هر مرحله برای رشته های مثال زده شده:

Alignment رشته ی seq[0] و seq[1]: 6-

Alignment رشته ی seq[0] و seq[2]: 7-

Alignment رشته ی seq[0] و seq[3]: 4-

Sumscore برای seq[0]: 17-

Alignment رشته ی seq[1] و seq[0]: 6-

Alignment رشته ی seq[1] و seq[2]: 5-

Alignment رشته ی seq[1] و seq[3]: 12-

Sumscore برای seq[1]: 23-

Alignment رشته ی seq[2] و seq[0]: 7-

Alignment رشته ی seq[2] و seq[1]: 5-

Alignment رشته ی seq[2] و seq[3]: 9-

Sumscore برای seq[2]: 21-

Alignment رشته ی seq[3] و seq[0]: 4-

Alignment رشته ی seq[3] و seq[1]: 12-

Alignment رشته ی seq[3] و seq[2]: 9-

Sumscore برای seq[3]: 25-

در آخر رشته ی seq[0] یا 'ATCGGAAT' با sumscore، maximum یعنی 17- به عنوان رشته ی مرکزی انتخاب می شود.

2. در قدم بعدی در الگوریتم star، pairwise alignment بین رشته ی مرکزی و تمام رشته های دیگر را بدست می آوریم. و در هر alignment، رشته ی align شده ی متناظر رشته ی مرکزی را در دیکشنری center_ alignments و رشته ی align شده ی متناظر رشته ی دیگر را در seq_alignments قرار می دهیم. برای مثال اگر رشته ی مرکزی را رشته ی دوم با اندیس 1 فرض کنیم، با alignکردن رشته ی مرکزی و رشته ی اول با اندیس 0،

مقدار زخیره شده در seq_alignments: ['seq[0]']: align_str2 for seq[0]

مقدار زخیره شده در center_alignments: ['seq[0]']: align_str1 for Sc

و در هر مرحله index رشته ای که با رشته ی مرکزی هم تراز می شود در index_array ذخیره می شود.

```
# pairwise alignment between each sequence with center
center_alignments = {}
seq_alignments = {}
score_array = []
for x in range(len(seq)):
    if seq[x] != seq[i_max_sum]:
        score, align_str1, align_str2 = alignment(seq[i_max_sum], seq[x])
        center_alignments[x] = align_str1
        seq_alignments[x]=align_str2
        score_array.append(x)
```

برای مثال برای رشته های مثال زده شده این گام به این صورت اعمال می شود:

```
pairwise alignment between Sc and S1
ATCGGA-AT
ATCGTTGAT
pairwise alignment between Sc and S2
ATCGGA--AT
ATAGGTTCAT
pairwise alignment between Sc and S3
ATCGGAAT
AGGAACCT
```

و دیکشنری های center_alignments و seq_alignments به این صورت می شوند:

```
center_alignments dictionary:
{1: 'ATCGGA-AT', 2: 'ATCGGA--AT', 3: 'ATCGGAAT'}
seq_alignments dictionary:
{1: 'ATCGTTGAT', 2: 'ATAGGTTCAT', 3: 'AGGAACCT'}
```

3. در گام بعد، تمام هم ترازی ها را با یک دیگر ادغام می کنیم. به ترتیب شروع می کنیم هم ترازی بین رشته ی اول در آرایه ی seq به غیر از رشته ی مرکزی و رشته ی مرکزی را که از مرحله ی قبل بدست آورده بودیم می

نویسیم سپس رشته های بعدی را به ترتیب براساس هم ترازی آن ها با رشته ی مرکزی که در مرحله ی قبل داشتیم اضافه می کنیم و رشته ی مرکزی را آپدیت می کنیم در هر مرحله.

الگوریتم starاز قانون once a gap,always a gap پیروی می کند به این معنی که زمانی که گپی اضافه می شود این گپ ها در طول الگوریتم هیچ گاه حذف نمی شوند.

کد این قسمت به این صورت است که ابتدا یک for روی index_array داریم و اندیس رشته ای که در هر مرحله اضافه می شود را با index نمایش می دهیم.

در این قسمت در آرایه ی final_array در اندیس 0، رشته ی مرکزی در هر alignment که در مرحله قبل داشتیم را قرار می دهیم و با اضافه کردن رشته های هم تراز شده ی رشته ی مرکزی و رشته ی بعدی، final_array[0] را با رشته ی مرکزی align شده با رشته ی جدید به این صورت آپدیت می کنیم:

1. باید گپ های رشته ی مرکزی قبلی که در final_array[0] قرار داشت حفظ گردد. برای این کاراندیس گپ هایی که final_array[0] قبلی داشت را در old_dash_indices قرار می دهیم و گپ های رشته ی مرکزی جدید را در new_dash_indices قرار می دهیم.

```
# Add sequences in decreasing order of similarity score between sequence and center
final_array = []
for index in index_array:
    old_dash_Indices = []
    new_dash_Indices = []
    dash_Indices = []


    if len(final_array) == 0:
        final_array.append(center_alignments[index])

    else:
        for i in range(len(final_array[0])):
            if final_array[0][i] == "-":
                old_dash_Indices.append(i)
        for i in range(len(center_alignments[index])):
            if center_alignments[index][i] == "-":
                new_dash_Indices.append(i)
```

2. سپس برای درست کردن جایگاه گپ های جدید و قدیمی، زمانی که می خواهیم آن ها را با هم در یک رشته قرار بدهیم، روی این دو آرایه for میزنیم تا اگر index گپی از یک آرایه بعد از index گپی در آرایه ی دیگر بود، به ازای هر گپی که پیش از آن قرار داشت index با 1 جمع می شود و اگر indexها با هم برابر شدند، آن ها را از هر دو آرایه حذف و در dd قرار می دهیم.

```
while i != new_length:
    while j != old_length:
        if len(new_dash_Indices) == 0 or len(old_dash_Indices) == 0:
            break

        if new_dash_Indices[i] < old_dash_Indices[j]:
            old_dash_Indices[j] = old_dash_Indices[j] + 1
            j = j+1

        elif new_dash_Indices[i] > old_dash_Indices[j]:
            new_dash_Indices[i] = new_dash_Indices[i] + 1
            j = j+1

        elif new_dash_Indices[i] == old_dash_Indices[j]:
            dd.append(new_dash_Indices[i])
            new_dash_Indices.remove(new_dash_Indices[i])
            old_dash_Indices.remove(old_dash_Indices[j])
            j = len(old_dash_Indices)
```

3. سپس جمع old_dash_indices و new_dash_indices و dd را در آرایه ی dash_indices قرار می دهیم وآن را مرتب می کنیم .سپس تمام گپ های با اندیس های در این آرایه را به ترتیب، به رشته ی مرکزی اضافه می کنیم تا رشته ی مرکزی align شده ی جدید در هر مرحله بدست آید و آن را جایگزین final_array[0] می کنیم.

```
dash_Indices = new_dash_Indices + old_dash_Indices+ dd
dash_Indices.sort()
test_seq = seq[i_max_sum]
for i in dash_Indices:
    test_seq = test_seq[:i] + "-" + test_seq[i:]
final_array[0] = test_seq
```

بعد از آپدیت کردن final_array[0]، رشته ی align شده با رشته ی مرکزی که در هر مرحله اضافه می شود و آن را در final_arrayقرار می دهیم نیز باید چک شود که اگه طول کم تر از final_array[0] جدید داشت، گپ های قدیمی رشته ی مرکزی به آن اضافه گردد و رشته های قبلی ای که در final_array موجود بودند باید چک شوند که اگر طول کم تر از final_array[0] جدید داشتند باید گپ های جدید در رشته ی مرکزی به آن ها اضافه گردد.

```
for i in range(1, len(final_array)):
    if len(final_array[i]) < len(final_array[0]):
        for j in new_dash_Indices:
            final_array[i] = final_array[i][:j] + "-" + final_array[i][j:]
if len(seq_alignments[index]) < len(final_array[0]):
    t = seq_alignments[index]
    for j in old_dash_Indices:
        t = t[:j] + "-" + t[j:]
    final_array.append(t)
else:
    final_array.append(seq_alignments[index])
```

برای رشته های مثال زده شده، در هر مرحله با اضافه کردن هر کدام از رشته ها به ترتیب به MSA، old_dash_indices و new_dash_indices و dash_indices و final_array به این صورت می شود:

```
final_array by adding seq[1]
['ATCGGA-AT', 'ATCGTTGAT']
****
old dash indices by adding seq[2]
[]
new dash indices by adding seq[2]
[7]
dash indices by adding seq[2]
[6, 7]
final_array by adding seq[2]
['ATCGGA--AT', 'ATCGTTG-AT', 'ATAGGTTCAT']
****
old dash indices by adding seq[3]
[6, 7]
new dash indices by adding seq[3]
[]
dash indices by adding seq[3]
[6, 7]
final_array by adding seq[3]
['ATCGGA--AT', 'ATCGTTG-AT', 'ATAGGTTCAT', 'AGGAAC--CT']
****
```

Looks like you're using NumPy

4. در آخر برای چاپ کردن رشته های خروجی چک می کنیم اگر رشته ی مرکزی رشته ی با اندیس 0 نبود، پس باید ترتیب المان های final_array را عوض کنیم چون align شده ی رشته ی مرکزی در final_array[0]قرار دارد.

و سپس MSA نهایی که در final_arrayقرار داشت را در خروجی چاپ می کنیم.

```
if i_max_sum != 0:
    for i in range(i_max_sum):
        final_array[i],final_array[i+1] = final_array[i+1], final_array[i]
for i in final_array:
    print(''.join(i))
```

برای مثال خروجی رشته های فوق به این صورت می گردد:

```
ATCGGA--AT
ATCGTTG-AT
ATAGGTTCAT
AGGAAC--CT
```