الواجب المنزلي 3

مقدمة

هذا الواجب المنزلي سيعرّفك على استخدام الدوال والعوديّة وأيضاً سيعرفك على عمليات السلاسل في البايثون.

التعاون

يمكنك التعاون مع طلاب آخرين، إنما يجب على كل طالب أن يكتب ويسلم واجبه بشكل منفصل. قم بذكر أسماء الطلاب الذين عملت معهم على حل الواجب. من أجل تفاصيل أخرى، قم بمراجعة سياسة التعاون الموجودة في مخطط المساق.

عملية التسليم:

سيتم تقييم في هذا الواجب، والواجبات اللاحقة، بواسطة مهمات اختبارية. برنامج المهمات الاختبارية سيتوقع أن يكون داخل ملفاتك فقط تعريف دوال من دون نص برمجي قابل للتنفيذ خارج تعريف الدوال (بالإضافة لما هو موجود مسبقاً في النموذج). فتذكر أن تقوم بكتابة النص البرمجي الذي يقوم بالاختبار كتعليمات. (وقم باختبار برنامجك بشكل جيد).

السلاسل والبحث السلسلي/في السلسلة

كما رأينا في المحاضرة، السلاسل هو نمط بيانات شائع في العديد من لغات البرمجة، وهي مستخدمة لتمثل المعلومات النصيّة. لقد رأيتم سابقاً أمثلة شائعة عن البحث في السلسلة. على سبيل المثال، إيجاد كلمات أو عبارات في ملف نصي يتضمن البحث في تسلسل ما من المحارف (أي الملف النصي) لإيجاد حالات لتسلسلات أخرى من المحارف (الكلمة أو العبارة التي نقوم بالبحث عنها). بشكل مماثل، محركات البحث في الشبكة العنكبوتية، مثل "غوغل"، يجب حساب عدد المرات التي يتكرر بها ذكر كلمات البحث المفتاحية في الملف النصي للصفحات، لكي يتم عرض الصفحات بشكل مرتب حسب علاقتها بتلك الكلمات.

مقارنة تطابق السلاسل: وجهة نظر بيولوجية

مقارنة السلاسل هي أيضاً قيِّمة ومهمة في حالات أقل وضوحاً، مثل البيولوجيا (علم الأحياء). هناك مسألة شائعة في علم الأحياء وهي فهم بنية جزيئات الحمض النووي (DNA)، ودور بنىً محددة في تحديد "الأدنين" (A) أو "السيتوزين" (B) أو "الجوانين" (G) أو "الثيمين" (T) - فجزيئات الحمض النووي أو الجدائل الممثلة بـ AAACAACTTCGTAAGTATA تمثل جديلة محددة من الحمض النووي.

إحدى طرق فهم وظيفة جديلة محددة من الحمض النووي (أو حتى جديلة جزئية من الحمض النووي) هي بمطابقة هذه الجديلة مع مجموعة من سلاسل الحمض النووي المعروفة -أي السلاسل ذات الوظائف والبنى الخاصة المعروفة- وذلك بالاعتماد على أن البنى المتشابهة تميل إلى أن تتضمن وظائف متشابهة. الكائنات الحية البسيطة مثل الجراثيم يمكن أن يكون لها 246 مليون بنية، فأي آلية مقارنة يجب أن تكون فعالة لدرجة كبيرة لتكون مفيدة.

في هذا الواجب المنزلي نحن لا نطلب منك أن تقوم ببناء أداة فعالة عملياً، ولكن نأمل أن نقوم بتعريفك على بعض المشاكل الموجودة باستعراض بعض آليات المقارنة البسيطة.

للبدء سنستخدم بعض الدوال الداخلية في البايثون. لاستخدام هذه الدوال، قم بكتابة التعليمة التلية في بداية النص البرمجي

```
from string import *
```

هذا سيمكنك من استخدام دوال البايثون الخاصة بالسلاسل. وبالتحديد، إذا أردت إيجاد نقطة البداية للمقارنة الأولى لسلسلة كلمة مفتاحية (مفتاح) في سلسلة مستهدفة (الهدف) يمكنك استخدام دالة البحث find.

حاول أن تنفذ find على بعض الأمثلة، مثل find("atgacatgcacaagtatgcat","atgc")

لاحظ كيف تعيد فهرس أول نسخة للمفتاح في الهدف. أيضاً لاحظ أنّه إذا لم يتم إيجاد أي نسخة من المفتاح في الهدف، مثلاً find("atgacatgcacaagtatgcat","ggcc") فهو يعيد القيمة 1-

سوف نقوم باستعراض بعض الأفكار حول مقارنة السلاسل عن طريق التطرق إلى المهام المعقدة واحدة تلو الأخرى.

ملاحظة: إن الحلول التي ستكتبها سيتم اختبارها على أمثلة من سلاسل الحمض النووي، ولكن لا تفترض أن حلولك سيتم تطبيقها فقط على سلاسل الحمض النووي، بعبارة أخرى، لا تفترض أن السلاسل تتألف من 4 محارف مختلفة فقط، بل أنها يمكن أن تحتوي على عدد اعتباطي من المحارف المختلفة.

لنبدأ بمسألة بسيطة إلى حد ما. افترض أننا نريد أن نقوم بحساب عدد مرات ظهور سلسلة المفتاح في سلسلة الهدف. سنقوم بإنشاء دالتين مختلفتين للقيام بهذه المهمة: إحداهما دالّة تكرارية والأخرى عوديّة. من أجل كلتا الدالتين يمكنك الاعتماد على دالة البايثون find – يجب عليك أن تقرأ توصيفاتها لتعلم كيفية تزويدها بمعاملات اختيارية لتبدأ عملية البحث عن تطابق في أماكن غير بداية السلسلة. على سبيل المثال،

```
find("atgacatgcacaagtatgcat","atgc")
```

إنه يعيد القيمة 5

بينما find("atgacatgcacaagtatgcat","atgc",6) يعيد القيمة 15، ذلك يعني أنه ببدء البحث عند الفهرس 6 التطابق التالي موجود في المكان 15.

من أجل النسخة العوديّة، سيتوجب عليك أن تفكر بكيفية استخدام الدالّة على نسخة أصغر من نفس المسألة (مثلاً على سلسلة هدف أصغر) وكيفية دمج نتائج تلك العمليّة الحوسبية لحل المسألة الأساسية. على سبيل المثال، بفرض أنه يمكنك إيجاد النسخة الأولى من سلسلة المفتاح في سلسلة الهدف، كيف يمكنك أن تدمج تلك النتائج مع استدعاء نفس الدالّة على سلسلة هدف أصغر. قد تجد أن عملية تقسيم السلسلة إلى أجزاء هي عملية مفيدة للحصول على السلاسل الجزئية لسلسلة ما.

المسألة 1

قم بكتابة دالتين تُدعيان countSubStringMatch و countSubStringMatchRecursive واللتان تأخذان معاملين هما سلسلة المفتاح وسلسلة الهدف. هاتان الدالتان تقومان بشكل تكراري وعودي بحساب عدد نسخ المفتاح في سلسلة الهدف. يجب عليك أن تقوم بتعريف

```
def countSubStringMatch(target,key):
```

و:

```
def countSubStringMatchRecursive (target, key):
```

قم بكتابة إجابتك في ملف يحمل الاسم ps3a.py

من الآن وحتى نهاية هذا الواجب المنزلي سوف نقوم باستعراض أفكار أخرى لمقارنة سلاسل جزئية. هذه المسائل يمكن أن يتم حلها إما باستخدام دالّة تكرارية أو دالّة عوديّة. يمكنك استخدام كلا الطريقتين، كما أنه من الممكن أن تجد الطرق التكرارية أكثر بداهة في هذه الحالات من مقارنة البنى الخطيّة.

الأمر التالي الذي يجب أن نقوم به هو أن نقوم بكتابة دالّة تقوم بتعميم find بحيث تقوم بإعادة tuple مؤلف من كل نقاط البداية لتطابق سلسلة المفتاح مع سلسلة الهدف، وليس فقط النسخة الأولى.

المسألة 2

اكتب الدالّة subStringMatchExact. هذه الدالّة تأخذ معاملين هما سلسلة الهدف وسلسلة المفتاح. يجب أن تعيد tuple مؤلف من نقاط البداية لحالات تطابق سلسلة الهدف وسلسلة المفتاح، عندما نبدأ الفهرسة عند 0. قم بتعريف :def subStringMatchExact(target,key)

على سبيل المثال،

subStringMatchExact("atgacatgcacaagtatgcat","atgc")

ستعيد الـ tuple(5,15). الملف ps3_template.py يتضمن بعض حالات اختبار يمكنك استخدامها لاختبار الدالة التي قمت بكتابتها. بالتحديد، فيه سلسلتي هدف:

target1 = 'atgacatgcacaagtatgcat'

target2 = 'atgaatgcatggatgtaaatgcag'

ومن أجل سلاسل المفتاح

key10 = 'a'
key11 = 'atg'
key12 = 'atgc'
key13 = 'atgca'

قم باختبار الدالّة التي قمت بكتابتها على كل التركيب لسلاسل الهدف وسلاسل المفتاح السابقة، بالإضافة إلى أمثلة أخرى تقوم بإنشائها.

قم بكتابة إجابتك في ملف يحمل الاسم ps3b.py

الدالّة التي كتبتها في المسألة 2 ستقوم بإيجاد التطابقات التامة لسلسلة المفتاح في سلسلة الهدف. كثيراً ما يكون من المفيد إيجاد النتائج القريبة، على سبيل المثال، إيجاد تطابق سلسلة المفتاح مع سلسلة الهدف عندما يكون أحد عناصر سلسلة المفتاح مستبدلاً بعنصر آخر. على سبيل المثال، إذا أردنا مطابقة

ATGC مع ATGACATGCACAAGTATGCAT ، نحن نعلم أن هناك تطابقاً تاماً يبدأ عند 5 و آخر يبدأ عند 15

على أية حال، هناك تطابقاً يبدأ عند 0 بحيث أن العنصر A مستبدل بـ C في المفتاح حيث أننا نطابق ATGC مع الهدف. بشكل مشابه، المفتاح ATTA يطابق هذا الهدف بدءً من 0، إذا سمحنا باستبدال G مع الـ T الثانية في سلسلة المفتاح.

يمكن أن نقوم بالبناء على الدالّة من المسألة 2 لحل هذه المسألة. بالتحديد، قم بأخذ الخطوات التالية بعين الاعتبار.

أولاً، قم بتقسيم سلسلة المفتاح إلى جزئين (حيث يمكن أن يكون أحد الأجزاء سلسلة فارغة). لندعوهما key1 و key2 . من أجل كل جزء، استخدم الدالّة التي قمت بكتابتها في المسألة 2 لإيجاد نقاط البداية للتطابقات المحتملة التي تستدعي

```
starts1 = subStringMatchExact(target,key1)
```

و:

```
starts2 = subStringMatchExact(target,key2)
```

نتيجة هذين الاستدعاءين يجب أن يكون هناك اثنين من الـ tuples، كل واحد منهما يقوم بتحديد نقاط البداية لتطابقات الجزئين (key1 و key2) لسلسلة المفتاح مع الهدف. على سبيل المثال، إذا أخذنا المفتاح ATGC يمكننا الأخذ بعين الاعتبار مطابقة A و GC مع الهدف مثل ATGACATGCA (في كلتا الحالتين سنحصل على أماكن التطابقات لـ A على شكل tuple (0, 3, 5, 9) وأماكن التطابقات لـ GC على شكل tuple (7). بالطبع، يجب علينا أن نبحث عن كل الخيارات الممكنة للسلاسل الجزئية التي ينقصها عنصر: السلسلة الفارغة و TGC؛ A و GC؛ AT و C؛ و ATG والسلسلة الفارغة. لاحظ أنه يمكنك استخدام الحل من المسألة 2 لإيجاد هذه القيم.

عندما نحصل على أماكن نقاط البداية للتطابقات للسلسلتين الجزئيتين، يجب أن نقرر أي تركيبات للتطابق مع السلسلة الجزئية الأولى وللتطابق مع السلسلة الجزئية الثانية صحيح. هناك اختبار سهل لذلك. افترض أنّ الفهرس لنقاط البداية للتطابق مع السلسلة الجزئية الأولى هو n (والذي سيكون عنصراً من starts1)، وأنّ طول السلسلة الجزئية الأولى هو m. فعندها إذا كان k هو عنصر من starts2 يدل على فهرس نقطة البداية لتطابق للسلسلة الجزئية الثانية، هناك نتيجة محققة مع استبدال وحيد يبدأ عند n، إذا كان $k = n+m+1$ ، لأن ذلك يعني أن تطابقاً مع السلسلة الجزئية الثانية يبدأ عند العنصر الذي يلي نهاية السلسلة الجزئية الأولى.

المسألة 3

اكتب دالّة اسمها constrainedMatchPair والتي تأخذ ثلاث معاملات: tuple يمثل نقاط البداية للسلسلة الجزئية الأولى و tuple يمثل نقاط البداية للسلسلة الجزئية الثانية وطول السلسلة الجزئية الأولى. يجب أن تعيد الدالّة tuple فيه كل الأعضاء (سميه n) للـ tuple الأول بحيث يكون هناك عنصر في الـ tuple الثاني (لنسميه k) بحيث يكون k = n+m+1 حيث m هو طول السلسلة الجزئية الأولى. أكمل التعريف

def constrainedMatchPair(firstMatch,secondMatch,length):

لاختبار هذه الدالّة، قمنا بإعطائكم دالّة تدعى subStringMatchOneSub والتي تأخذ معاملين: سلسلة الهدف وسلسلة المفتاح. هذه الدالّة ستعيد tuple فيه كل نقاط البداية للتطابقات بين المفتاح والهدف، بحيث يكون على الأكثر عنصر وحيد من عناصر المفتاح غير متطابق بشكل صحيح مع الهدف. الدالّة موجودة في الملف ps3_template.py وتستدعي الدالّة التي ستكتبها.

قم بكتابة إجابتك في ملف يحمل الاسم ps3c.py

سوف تلاحظ أثناء اختباراتك للمسألة 3، أنّ هذه الطريقة سوف تجد التطابقات مع استبدال وحيد ولكنها أيضاً سوف تجد التطابقات بدون استبدالات، وهي التطابقات التامة للمفتاح مع الهدف. افترض أننا نريد إيجاد هذه التطابقات فقط والتي هي باستبدال واحد. إحدى الطرق السهلة للقيام بذلك هي باستخدام الدوال من المسألة 2 والمسألة 3. هذه الدوال سوف تعطيك tuple يمثل نقاط البداية للتطابقات التامّة والتطابقات التي هي باستبدال واحد على الأكثر بالترتيب. إذا أبقينا لدينا فقط هذه العناصر من الـ tuple الثاني غير الموجودة في الـ tuple الأول، سنحصل على التطابقات التي فيها استبدال وحيد بالضبط.

المسألة 4

اكتب دالّة تدعى subStringMatchExactlyOneSub والتي تأخذ معاملين: سلسلة هدف وسلسلة مفتاح. يجب أن تعيد هذه الدالّة tuple فيه كل نقاط البداية للتطابقات بين المفتاح والهدف، بحيث يكون هناك عنصر وحيد بالضبط من المفتاح غير متطابق بشكل صحيح مع الهدف. أكمل التعريف

def subStringMatchExactlyOneSub(target,key):

قم بحفظ إجاباتك في ملف وسميه ps3d.py

عملية التسليم:

1. **الحفظ**

قم بحفظ إجابتك في ملف وقم بتسميته باسم كل مسألة، لا تقم بتجاهل هذه الخطوة ولا تسمي الملف بأسماء أخرى.

2. **التوقيت ومعلومات عن التعاون**

في بداية كل ملف، قم بكتابة تعليق واذكر فيه عدد الساعات -بشكل تقريبي- التي أمضيتها على المسائل في هذا الجزء، واذكر أيضاً أسماء الأشخاص الذين تعاونت معهم. على سبيل المثال:

```
# Problem Set   3 (Part I)
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
#
... اكتب النص البرمجي هنا ...
```