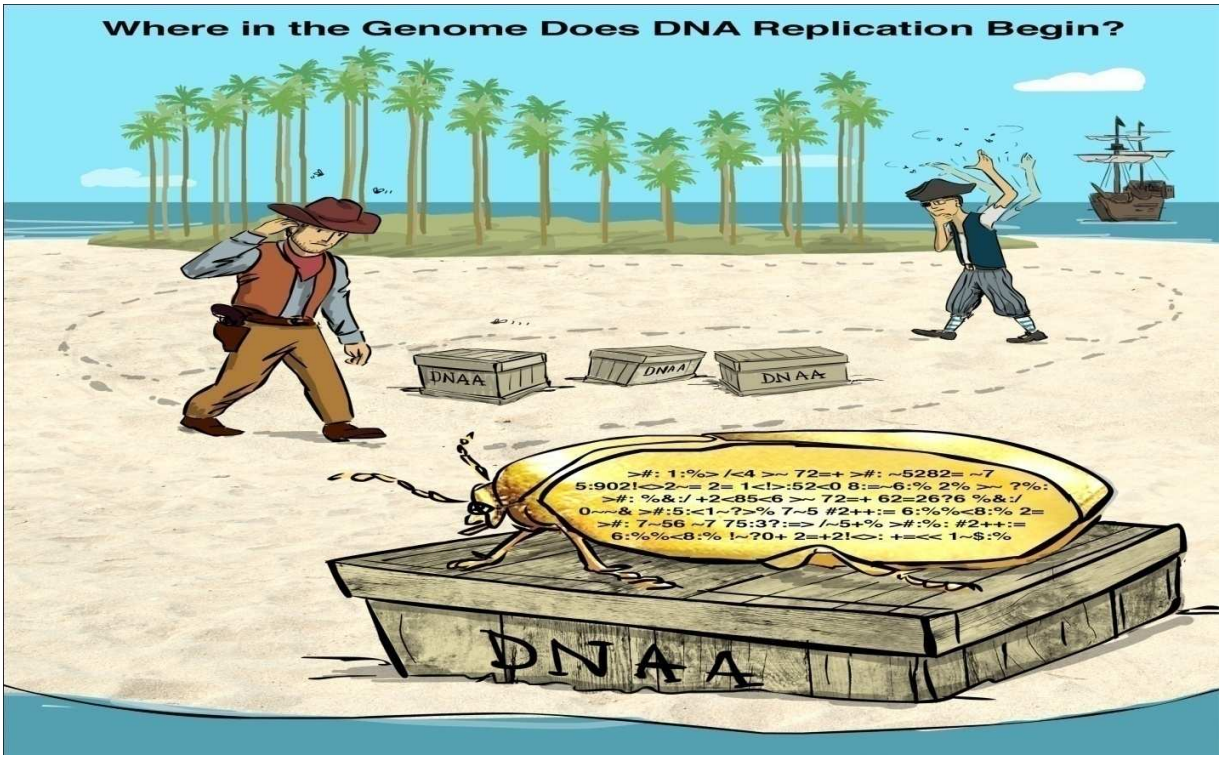
بسم الله الرحمن الرحيم

والصلاة والسلام على اشرف المرسلين سيدنا محمد صلى الله علية وسلم

Bioinformatics Lap

الفصل الاول: أين يبدأ نســـــــــــــخ الجينوم(الجزء الاول)



الجينات هي الوحدة الأساسية للمعلومات الوراثية في كل الكائنات الحية، فهي تحمل المعلومات اللازمة لبناء الخلايا والحفاظ عليها والقيام بكافة الوظائف الحيوية ومن ثم بناء أجسام الكائنات وإعطائها صفاتها المميزة.

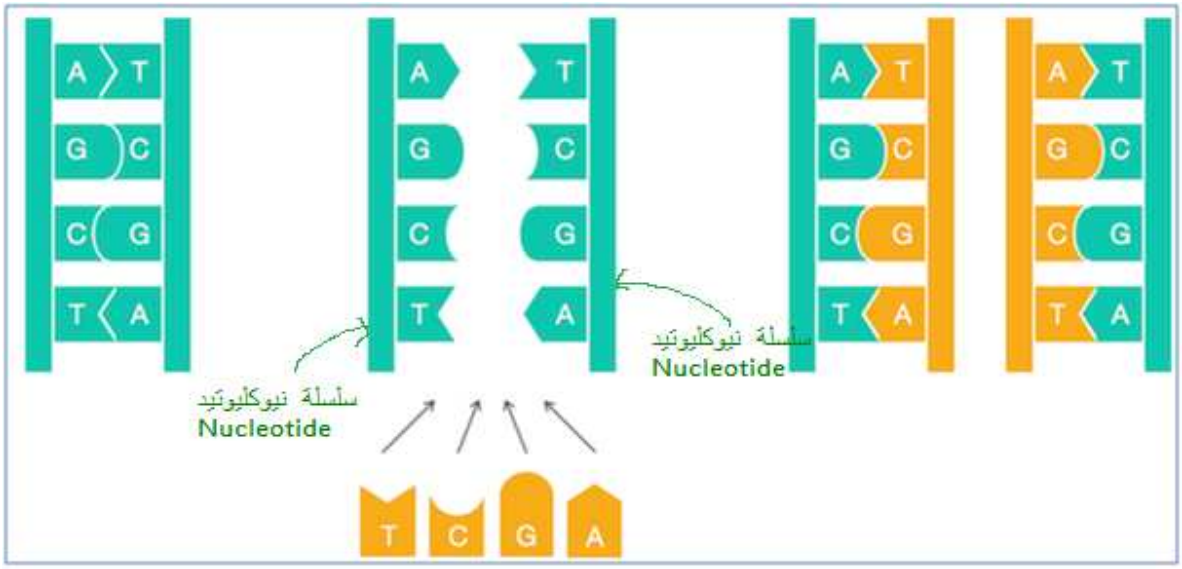
التعبير عن الجينات

تبدأ عملية التعبير عن الجين بنسخ جزء من ال (DNA) الذى هو اختصار الى (الحمض النووي الرايبوزي منقوص الأكسجين) وهو يتكون من سلسلتين من النيوكليوتيداتNucleotides تلتفان حول بعضهما بشكل حلزوني وكل سلسلة نيوكليوتيد Nucleotide تتكون من سكر خماسى وهو الديزوكسى ريبوز Deoxyribose وفوسفات phosphate وقاعدة نيتروجينية Nitrogen base

والقواعد النيتروجينية تتضمن مجموعتان البيورين Purines وتتضمن الأدينين (A) Adenine والجوانين (G) Guanine

أما المجموعة الثانية فهى البيرميدين Pyrimidines وتتضمن الثيمين (T) Thymine والسيتوزين (C) Cytosine

ويلاحظ أن القاعدة النيتروجينية أدنين A تكون في أحد السلاسل تكون متقابلة مع القاعدة النيتروجينية ثايمين(T)في السلسلة الثانية، بينما تكون القاعدة النيتروجينية غوانين G متقابلة مع القاعدة النيتروجينية سايتوسينC كما بالشكل .



ويمكن كتابة الجين على انة تتابع من القواعد النيتروجينية

TTCGTTAAGTAACTTCACTGCCCGTAGTGTACCGGCATTCGCTAGCAAGAGTCTTTCTG

اذن عملية النسخ المتماثل للجينات **Genome replication** سهلة هل يمكن ان نقوم بها بطريقة حسابية ؟ **كمبرمج نعم** كيف؟؟

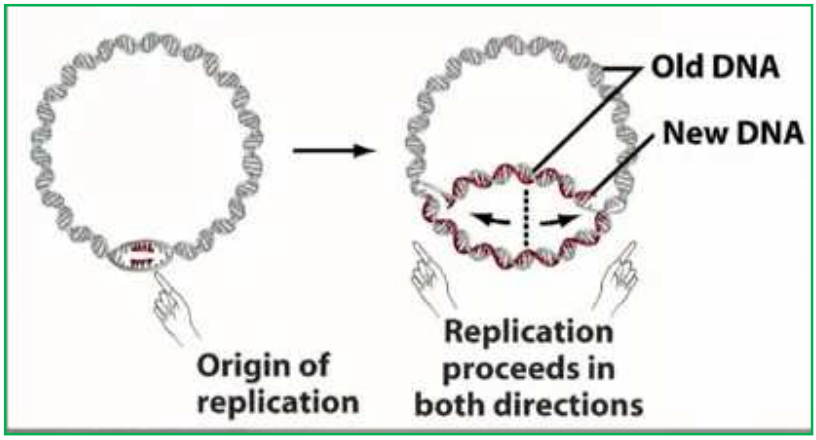
المشكلة يمكن صياغتها بالشكل الاتى : **اكتب برنامج يأخذ ملف نصى فية** DNA string *Genome* **ويقوم بعمل نسخ لة**

Replication Problem:

Input: A DNA string *Genome*.

Output: copy of DNA string *Genome*.

ولكن حقيقة عملية النسخ المتماثل للجينات **Genome replication** ليست بهذة السهولة فلا يتم نسخ الجين من اى جزء فية بل يبدأ النسخ من جزء معين فى الجين يسمى **Replication origin ويطلق علية** oriC وتتم عملية النسخ من خلال جزيء يسمى **DNA polymerases .**



وتحديد مكان بداية النسخ oriC مهم جدا جدا ليس فقط لمعرفة كيف تتم عملية النسخ ولكن ايضا مهم فى حل مشاكل كثيرة فى مجال البيوانفورميتك مثل العلاج الجينى **gene therapy** فى الرابط التالى :مثال على اهمية معرفة منطقة بداية النسخ

اذن الان هل عملية نسخ الجينات gene **Replication** سهلة هل يمكن ان نقوم بها بطريقة حسابية ؟

المشكلة يمكن صياغتها بالشكل الاتى :

***اكتب برنامج يقوم بأخذ ملف نصى ويحدد لى اين منطقة بداية النسخ oriC ؟**

Finding *oriC* Problem:

Input: A DNA string *Genome*.

Output: The location of *oriC* in *Genome*.

هل كمبرمج تستطيع ان تكتب برنامج لحل هذة المسألة لا طبعا؟فالمسألة كأنة لغز فتخيل ان لديك النص الاتى :

...TTCGTTAAGTAACTTCACTGCCCGTAGTGTACCGGCATTCGCTAGCAAGAGTCTTTCTG

GGCAAGCTTCACTTGTGATCGCGGCCTGTGCCCCCGGAATGAAACAACCACGTCCCTGCT

AACAACGACGGGAAAAGGGAAGTGATCCGTCGGCAGACCCAGACTAGTGCCCTTCTCCGG

CTTCCAACACCAACGAGTCGGACCGAATTGAGCACTCGAATGCACGGCGCTTTTTGCCGG

CCGAAACGGCGCCTCCGCATTGATCGACGCACGGCCTCTTTTGGCTACAGCGCATGGCTT

TACACTCGGCATGCATTTCCAGTGCTAATCAAACAGAATTCCTTGTAAAGTCCTTCAACC

GTGACAGACTATCGCTAAGGAGCCTTTCCAGTCGTGCCTGCAATCACTCGCGAAATCAAC

AAATCTACATCTAAGCACGCTCGTGGTTCGGAGTCCCGCCCTCATGTGGACCATAGCCGG

TTCGCCCGAGTCCTAGGCACGATCAGAGGACCTATCTTTCGCCACTCAACTCTTCTGAGT

GAAACAATATCGACCGAAACCTTGCTCGGTTTTGTCCACAACAACGTCAGGCCCATAAGC

AGACGACATTAGTCCGCTGTTGTCGCGGGCGTCCCATAGCCGTACGATGTCCCGTCGGA...

وتريد ان تعلم اين تبدأ نسخ النص دون ادنى معلومات عن ماهية هذا النص -هذا حقيقة لغز هل من حل لهذا الغز ؟؟؟؟

حتى نعمل بشكل عملى نفترض ان لدينا ال gene لبكتريا ما والهدف نريد ان نعلم منطقة بداية النسخ لهذا الجين finding *oriC* in bacterial genomes فننظر ونحلل هذا الجين جيدا لنعلم ما الذى يميز مناطق عن مناطق اخرى فية **وما الذى يميز منطقة بداية النسخ *oriC* خاصة** *ثم نطبق ما تعلمناة على بكتريا من انواع اخرى (يسمى هذا النوع من التعلم* approach *in silico*)

التطبيق على بكتريا *Vibrio cholerae*, (التى تسبب مرض الكوليرا)

تعتبر الكوليرا من أحد الأمراض التي تصيب الجهاز الهضمي، يأتي المرض في صورة إسهال حاد مرضي مسبباً عدوى في الأمعاء ببكتريا "بكتريم فيبريو كوليرا."Bacterium Vibrio Cholera - " وفي بعض الحالات تكون الأعراض بسيطة وفي البعض الآخر تكون خطيرة وتهدد حياة الإنسان وتأتي الأعراض في صورة:

1-إسهال بكميات كبيرة جداً مع ازدياد حدته.

2- قئ.

3- تقلص في عضلات الأرجل.

4-فقد سريع لسوائل الجسم مما يؤدى إلي حدوث الجفاف.

5-تعرض الإنسان لصدمة وموته في خلال ساعات إذا لم يتلق العلاج

شكل الجين لهذة البكتريا كالاتى ويتكون من1,108,250 نيوكليوتيدة :

https://beta.stepic.org/media/attachments/lessons/3/Vibrio_cholerae.txt

تم ملاحظة ان منطقة بداية النسخ *oriC* فى هذة البكتريا هى عبارة عن جزء من الجين يتكون من 500 نيوكليوتيد Nucleotideالسؤال الان كيف تعلم هذة البكتريا اى مكان فى هذا الجين الكبير والذى يتكون من 1,108,250 نيوكليوتيدة هو منطقة بداية النسخ *oriC* ؟؟فيجب ان تكون هناك رسالة مخفية فى هذة المنطقة المكونة من 500 *نيوكليوتيدة تقول انها منطقة بداية النسخ* oriC ومن البيولوجى نحن نعلم ان بداية عملية النسخ تتم من بروتين يسمى

dnaA وهو جزء صغير يربط داخل منطقة بداية النسخ ويعرف بـdnaA boxs فيمكننا النظر الى dnaA box على انها رسالة تقول للبروتين dnaA اربط هنا.

مما سبق المسألة الان تحولت الى ما يعرف بالرسالة المخفية حيث نريد الحصول على الرسالة المخفية dna Aboxفي هذة المنطقة التى تتكون من 500 نيوكليوتيد والتى تقول للخلية من خلالها انها المنطقة التى يبدأمنها نسخ الجينوم

كمبرج هل يمكن صياغة المشكلة ؟ نــعم

***نصيغ المشكلة بالشكل الاتى:اكتب برنامج يأخذ رسالة نصية كبيرة تعبر عن منطقة بداية النسخ representing *oriC* in a genome ويعطينا كمخرجات الجزء المخفى من هذا النص**

Hidden Message Problem: *Find a "hidden message" in a text.*

Input: A string *Text* (representing *oriC* in a genome).

Output: A hidden message in *Text*.

كمبرج استطعنا صياغة المشكلة لكن هل يمكن حلها وكتابة برنامج لها

قبـــــــــــــــــــــــــــــــــــــــل الاجابة هل تعلم ما هو لغز **The Gold-Bug** **وطريقة حلة؟ اذا لا فأتبع الرابط**

الان نرجع لمشكلة الرسالة المخفية هل يمكن نطبق ذلك على منطقة بداية النسخ *oriC* لبكتريا الكوليرا Bacterium Vibrio Cholera الاجابة نعم

في لغة ال DNA :سوف نبحث عن اكثر كلمة(pattern)مكررة فمثلا ACTAT هى كلمة متكونة من 5 حروف مكررة فى النص الاتى:

ACAACTATGCATACTATCGGGAACTATCCT

للتوضيح نضع القاعدة التالية:

فنرمز الى طول الكلمة(pattern) بـــــــــــ *k*-mer

ونرمز لعدد مرات حدوث الكلمة(pattern) فى النص text بأنها *Count*(*Text, Pattern*)

فمثلا:

Count(ACAACTATGCATACTATCGGGAACTATCCT, ACTAT) = 3

اخيرا نطلق على اكثر الكلمات(pattern s) تكراراً **Most Frequent *k*-mer** ونحددة بأكبر قيمة لل **count**

مثال 1:

ACTAT هو **5-mer** Most Frequent في النص ACAACTATGCATACTATCGGGAACTATCCT حيث تكرر ثلاث مرات

مثال 2:

ATA هو **3-mer** Most Frequent في النص CGATATATCCATAG. حيث تكرر ثلاث مرات

الان هل يمكن صياغة المشكلة بشكل رياضى كمبرمج نعم فيمكن صياغة المشكلة بالشكل الاتى:

***اكتب برنامج يأخذ نص ورقم K يعبر عن طول الكلمة ويخرج لى اكثر الكلمات المكررة (pattern) فى هذا النص ويكون طولها k**

Frequent Words Problem: *Find the most frequent k-mers in a string.*

Input: A string *Text* and an integer *k*.

Output: All most frequent *k*-mers in *Text*.

للتوضيح عند كتابة البرنامج تذكر المدخلات والمخرجات تكون كما بالشكل الاتى:

Sample Input:

ACGTTGCATGTCGCATGATGCATGAGAGCT

4

Sample Output:

CATG GCAT

البرنامج بلغة البايثون:

```
def  FrequentWords (text,k):
  """
  >>> Frequent Words ('ACGTTGCATGTCGCATGATGCATGAGAGCT',4)
  ('the most frequent k-mer are ', ' GCAT CATG')
  ('with frequent count =', 3)
  """

  i=0
  str=''
  list=[]
  kmer=k-1
  while i!=len(text)-kmer:
    j=i
    pattern=''
    m=0
    while m<=kmer:
      pattern=pattern+text[j]
      j=j+1
      m=m+1
    count=text.count(pattern)
    list.append([pattern, count])
    i=i+1

  max=0
  for x in xrange(len(list)):
      if list[x][1]>max:
       max=list[x][1]
       most_frqcount=max
  for y in xrange(len(list)):
    if list[y][1]==most_frqcount and str.find(list[y][0])==-1:
       str=str+' '+list[y][0]
```

```
print('the most frequent', k,'-mer are ', str)
print("with frquent count =",most_frqcount)
```

بعد التطبيق هذة الطريقة على الجين لبكتريا الكوليرا Bacterium Vibrio Cholera نحصل على الجدول الاتى:

k	3	4	5	6	7	8	9
count	26	12	8	8	5	4	3
***k*-mers**	tga	atga	gatca	tgatca	atgatca	atgatcaa	atgatcaag
		tgat	tgatc				cttgatcat
							tcttgatca
							ctcttgatc

وبما اننا نعلم انة يتم النسخ من خلال 9 nucleotides فى الجين الخاص لبكتريا الكوليرا Bacterium Vibrio Cholera

فأننا سوف ننظر فقط الى patterns المكررين ولهم طول 9-mer وهم

ATGATCAAG, CTTGATCAT, TCTTGATCA, and CTCTTGATC.

ولكن السؤال الان من منهم dnaA box؟؟ هل فى اى pattern منهم ميزة عن الاخر؟؟؟؟

فى بداية الشرح اوضحنا ان القواعد النتيروجينية كل نيوكليوتيد لها ما يقابلها فمثلا:

A تقابلها T

C تقابلها G

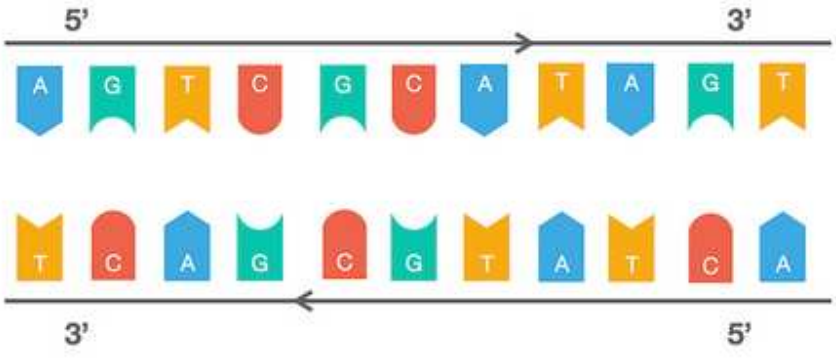
فمثلا:

AGTCGCATAGT

يكون المكمل لة

ACTATGCGACT

كما بالشكل التالى



Complementary strands run in opposite directions. The beginning and end of a DNA strand are denoted 5' (pronounced "five prime") and 3' (pronounced "three prime"), respectively.

هل يمكننا كتابة برنامج يأخذ نص dna stringوتكون مخرجاتة الcomplement لهذا النص؟

* يمكن صياغة مشكلة complement بشكل بسيط حيث المدخلات DNA string Pattern و المخرج المكمل لها

Reverse Complement Problem: *Reverse complement a nucleotide pattern.*

Input: A DNA string *Pattern*.

Output: *Pattern*, the reverse complement of *Pattern*

Sample Input:

AAAACCCGGT

Sample Output:

ACCGGGTTTT

البرنامج بلغة البايثون

```
def reverse(word):
  reverse_word=''
  i=len(word)-1
  while i>=0:
```

```
    reverse_word=reverse_word+word[i]
    i=i-1
  return reverse_word

def complement(fileName):
  with open (fileName, "r") as myfile:
    pattern=myfile.read().replace('\n', '')
  myfile.close()
  rPattern=reverse(pattern)
  cPattern=''
  for I in range(len(rPattern)):
    if rPattern[i]=='A':
      cPattern=cPattern+'T'
    elif rPattern[i]=='T':
      cPattern=cPattern+'A'
    elif rPattern[i]=='C':
      cPattern=cPattern+'G'
    elif rPattern[i]=='G':
      cPattern=cPattern+'C'

 return cPattern
```

طبعا صعوبة هذة البرامج فى التعامل مع بيانات كبيرة جدا ليس كما فى المثال لذا جرب على الملف هنا

https://beta.stepic.org/Bioinformatics-Algorithms-2/Some-Hidden-Messages-are-More-Surprising-than-Others-3/#step-2

الان نتوقف قليلا ونرجع ننظر فى ل 9-mer فى منطقة بداية النسخ oriC لبكتريا الكوليرا Bacterium Vibrio Cholera وهم

ATGATCAAG, CTTGATCAT, TCTTGATCA, and CTCTTGATC.

فنجد ان ل 9-mer ATGATCAAG هى مكملة CTTGATCAT وانة تم تكرارهم فى منطقة بداية النسخ oriC لبكتريا الكوليرا Bacterium Vibrio Cholera(التى هى 500 نيوكليوتيدة) 6 مرات

atcaatgatcaacgtaagcttctaagcATGATCAAGgtgctcacacagtttatccacaac

ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca

cggaaagATGATCAAGagaggatgatttcttggccatatcgcaatgaatacttgtgactt

gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt

acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga

tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat

tgataatgaatttacatgcttccgcgacgatttacctCTTGATCATcgatccgattgaag

atcttcaattgttaattctcttgcctcgactcatagccatgatgagctCTTGATCATgtt

tccttaaccctctatttttacggaagaATGATCAAGctgctgctCTTGATCATcgtttc

طبعا اننا نعثر على 9-mer يتم تكرارها 6 مرات(سواء هى او مكملتها) فى قطعة من DNA مكونة من 500 نيوكلوتيدة اكيد احصائيا افضل من اننا نحصل على 9-mer مكررة ثلاث مرات هذة النتيجة الاحصائية تجعلنا نفترض ان ATGATCAAG ومكملتها CTTGATCAT هى *DnaA* boxes فى البكتريا الكوليرا Bacterium Vibrio Cholera

لكن لننتظر قبل ان نجزم اننا وجدنا *DnaA* boxes فى بكتريا Bacterium Vibrio Cholera يجب ان نختبر هل هناك اماكن اخرى فى الجينوم لبكتريا Vibrio Cholera مكرر فيهم (CTTGATCAT or ATGATCAAG؟؟

هل يمكننا صياغة هذة المشكلة :

***اكتب برنامج يحدد مواقع تواجد او تكرار كلمة ما فى نص كبير**

Pattern Matching Problem: *Find all occurrences of a pattern in a string.*

Input: Two strings, *Pattern* and *Text*.

Output: All starting positions where *Pattern* appears as a substring of *Text*.

Sample Input:

ATAT

GATATATGCATATACTT

Sample Output:

1 3 9

البرنامج بلغة البايثون:

```
def find_ALLoccurrences(pattern,text):
  listOfIndex=''
  i=0
  while True:
    index =text.find(pattern,i)
    if index==-1:
      break
    listOfIndex=listOfIndex+str(index)+' '
    i=index+1
  return  listOfIndex
```

طبعا للتطبيق على بكتريا Bacterium Vibrio Cholera نحتاج لقراءة ملف الجينوم لها من هنا

https://beta.stepic.org/media/attachments/lessons/3/Vibrio_cholerae.txt

ونحاول ان نجد ال patternالاتى CTTGATCAT

البرنامج بلغة البايثون فقط اضفنا القراءة من ملف :

```
def find_ALLoccurrences(pattern,fileName):
  with open (fileName, "r") as myfile:
    text=myfile.read().replace('\n', '')
  myfile.close()
  listOfIndex=''
  i=0
  while True:
    index =text.find(pattern,i)
```

```
    if index==-1:
      break
    listOfIndex=listOfIndex+str(index)+' '
    i=index+1
  return listOfIndex
```

نلاحظ اننا عندما بحثنا فى الجينوم للبكتريا كلها عن مواقع حدوث ATGATCAAG وجدنا انها متواجدة فى 17 موقع

```
>>> find_ALLoccurrences('ATGATCAAG','Vibrio_cholerae.txt')
'116556 149355 151913 152013 152394 186189 194276 200076 224527 307692 479770 610980
653338 679985 768828 878903 985368 '
>>>
```

واقرب مواقع هم 151913 152013 152394

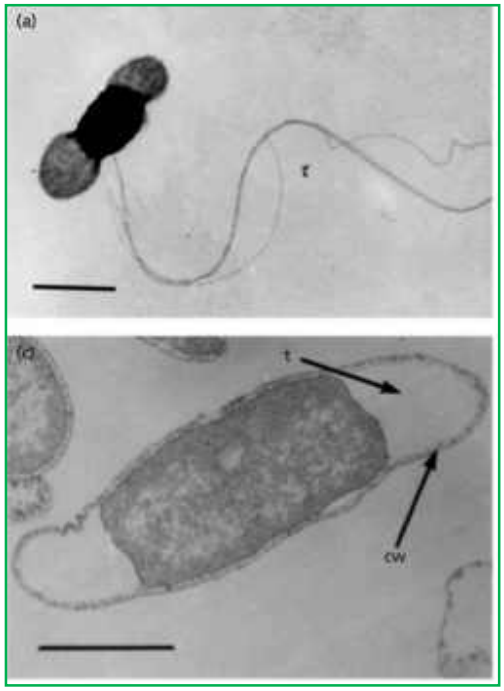
نفس النتيجة نحصل عليها عند البحث عن المكمل CTTGATCAT

```
>>> find_ALLoccurrences('CTTGATCAT','Vibrio_cholerae.txt')
'60039 98409 129189 152283 152354 152411 163207 197028 200160 357976 376771 392723 532935
600085 622755 1065555 '
```

نستنتج من ذلك ان CTTGATCAT/ ATGATCAAG هى الرسالة المخفية dnaAbox فى *DnaA* لجينوم البكتريا التى تقول عندها ابدأ النسخ

***سؤال اخير –هل هذا يعنى ان CTTGATCAT/ ATGATCAAG هى الرسالة المخفية لبقية انواع البكتريـــــا؟**

لا يمكن ان نجزم بأن CTTGATCAT/ ATGATCAAG هى الرسالة المخفية لبقية انواع البكتريـــــا الا اذا جربنا ان نجدها فى بكتريا مختلفة اذا لنأخذ منطقة بداية النسخ *oriC* لاى بكتريا جديدة ولتكن بكتريا *Thermotoga petrophila*



وهى نوع من البكتيريا التي تعيش في بيئات ساخنة للغاية؛ اسمها مستمد من اكتشافها في الماء تحت مكامن النفط، حيث درجة الحرارة يمكن أن تتجاوز 80درجة مئوية 176درجة فهرنهايت-منطقة بداية النسخ *oriC* هى كالاتى:

aactctatacctccttttgtcgaatttgtgtgatttatagagaaaatcttattaactga

aactaaaatggtaggtttggtggtaggttttgtgtacattttgtagtatctgatttttaa

ttacataccgtatattgtattaaattgacgaacaattgcatggaattgaatatatgcaaa

acaaacctaccaccaaactctgtattgaccattttaggacaacttcagggtggtaggttt

ctgaagctctcatcaatagactattttagtctttacaaacaatattaccgttcagattca

agattctacaacgctgttttaatgggcgttgcagaaaacttaccacctaaaatccagtat

ccaagccgatttcagagaaacctaccacttacctaccacttacctaccacccgggtggta

agttgcagacattattaaaaacctcatcagaagcttgttcaaaaatttcaatactcgaaa

cctaccacctgcgtcccctattatttactactactaataatagcagtataattgatctga

لو استخدمنا برنامجنا find_ALLoccurrences لنجد ATGATCAAG /CTTGATCAT بها فالنتيجة سوف تكون [ ] اى لا يوجد وهذا يجعلنا نستنتج ان الرسالة المخفية تختلف من جينوم بكتريا عن اخرى

Different bacteria may use different *DnaA* boxes as "hidden messages" to the *DnaA* protein.

اذا لنحاول ان نجد الرسالة المخفية فى منطقة بداية النسخ oriC لبكتريا *Thermotoga petrophila* *باستخدام برنامجنا* Frequent Words لنحصل على ال pattern التى لها اعلى تكرار "3" او اكثر فى منطقة بداية النسخ oriC لنجد النتيجة الاتى حيث pattern 6 لهم الطول 9 تم تكرارهم فى هذة المنطقة الصغيرة:

AACCTACCA AAACCTACC ACCTACCAC
CCTACCACC GGTAGGTTT TGGTAGGTT

سوف نستخدم برنامج Ori-Finder لتحديد ايهم هو الرسالة المخفية http://tubic.tju.edu.cn/Ori-Finder

النتيجة ان CCTACCACC والمكمل لها GGTGGTAGG هى الرسالة المخفية

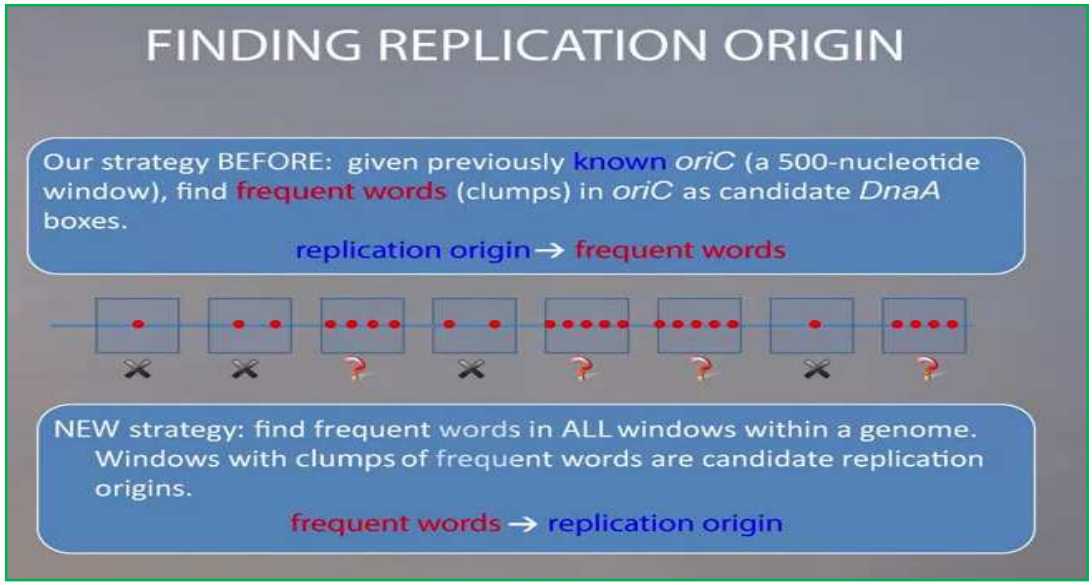
aactctatacctccttttgtcgaatttgtgtgatttatagagaaaatcttattaactga
aactaaaatggtaggtttGGTGGTAGGttttgtgtacattttgtagtatctgatttttaa
ttacataccgtatattgtattaaattgacgaacaattgcatggaattgaatatatgcaaa
acaaaCCTACCACCaaactctgtattgaccattttaggacaacttcagGGTGGTAGGttt
ctgaagctctcatcaatagactattttagtctttacaaacaatattaccgttcagattca
agattctacaacgctgttttaatgggcgttgcagaaaacttaccacctaaaatccagtat
ccaagccgatttcagagaaacctaccacttacctaccacttaCCTACCACCcgggtggta
agttgcagacattattaaaaacctcatcagaagcttgttcaaaaatttcaatactcgaaa
CCTACCACCtgcgtcccctattatttactactactaataatagcagtataattgatctga

من هنا نؤكد استنتاجنا ان الرسالة المخفية لمنطقة بداية النسخ oriC تختلف من جينوم بكتريا عن بكتريا اخرى.

لذا اذا لدينا جينوم لبكتريا جديدة ونريد معرفة منطقة بداية النسخ oriC فلن يفيدنا معرفة dnaA box (الرسائل المخفية او patterns) للبكتريا تم معرفتها سابقا

***قبل ان نفقد كل الامل لنعيد صياغة المشكلة حتى نحصل على منطقة بداية النسخ oriC –ففى السابق كنا نتبع استراتيجية اننا لدينا منطقة بداية النسخ oriC (500 نيوكليوتيدة) ونبحث فيها عن ال pattern الاكثر تكرار**

الان لنصيغ المشكلة بشكل اخر اذا كان لدينا الجينوم الكلى اعثر على المناطق التى فيها تكرار لل pattern ؟



لنقوم بذلك يجب ان نجزء الجينوم الى مناطق windows كل منها لها طول length يرمز لة بالرمز L ونعرف ال pattern الذى هو k-mer اى طولة k على انة clumps اذا تكرر فى هذا الجزء لعدد من المرات وليكن t. كمثال للتوضيح ال pattern الذى طولة 4 (k=4) هوTGCA تم تكرارة 3 مرات (t=3) فى منطقة windows طولها (L=25)

gatcagcataagggtcccTGCAaTGCAtgacaagccTGCAgttgttttac

*الان يمكن صياغة المشكلة بالشكل الاتى:

Clump Finding Problem: *Find patterns forming clumps in a string.*

Input: A string *Genome*, and integers *k*, *L*, and *t*.

Output: All distinct *k*-mers forming (*L*, *t*)-clumps in *Genome*.

كمثال للمدخلات والمخرجات للبرنامج:

Sample Input:

CGGACTCGACAGATGTGAAGAACGACAATGTGAAGACTCGACACGACAGAG TGAAGAGAAGAGGAAACATTGTAA

5 50 4

Sample Output:

CGACA GAAGA

البرنامج بلغة البايثون: طبعا من الممكن ان حل مشكلة Clump Finding بأن نطبق Frequent Words على كل جزء windows طولة L فى الجينوم الكلى

```
def ClumpFinding (text,k,l,t):
  """
  (str,int)->nonType

>>>ClumpFinding('CGGACTCGACAGATGTGAAGAACGACAATGTGAAGACTCGACACGACAGAGTGAAG
AGAAGAGGAAACATTGTAA',5, 50, 4)
   Sample Output:
   CGACA GAAGA
  """
  i=0
  str=''
  list=[]
  kmer=k-1
  while i!=len(text)-kmer:
    j=i
    pattern=''
    m=0
    while m<=kmer:
      pattern=pattern+text[j]
      j=j+1
      m=m+1
    count=text.count(pattern)
    list.append([pattern, count])
    i=i+1
  #print list

  for y in xrange(len(list)):
    if list[y][1]>=t and str.find(list[y][0])==-1:
        str=str+' '+list[y][0]

  print('the most frequent', k,'-mer are ', str)
```

نحن وجدنا ان هناك مئات من ال pattern المختلفة لها طول 9 (kmer=9) موجودة فى اجزاء windows طولها 500 نيوكليوتيدة ومكررة اكثر من 3 مرات لذا لا نعلم ايهم يعتبر DnaA box الرسالة المخفية hidden message

حتى هنا يتوقف الباحث الذى يؤمن بالفشل ولكن نحن سوف ندرس مرة اخرى عملية النسخ وكيف تتم بشكل بيولوجى.

I can accept failure, everyone fails at something. But I cannot accept not trying

رَبِّ ابْنِ لِي عِنْدَكَ بَيْتًا فِي الْجَنَّةِ

نسألكم بالله الدعاء لنا بالجنـــة وأن يتقبل الله عز وجل العمل خالصـــا لوجة- اميـــــــــن

الحمد الله رب العالميين –اللهم دومها نعمة واحفظها من الزوال –امين -اللهم ارزقنا بكل حرف مغفرة ورضوان امين يا رب

إِنَّ اللَّهَ وَمَلَائِكَتَهُ يُصَلُّونَ عَلَى النَّبِيِّ يَاأَيُّهَا الَّذِينَ آمَنُوا صَلُّوا عَلَيْهِ وَسَلِّمُوا تَسْلِيمًا

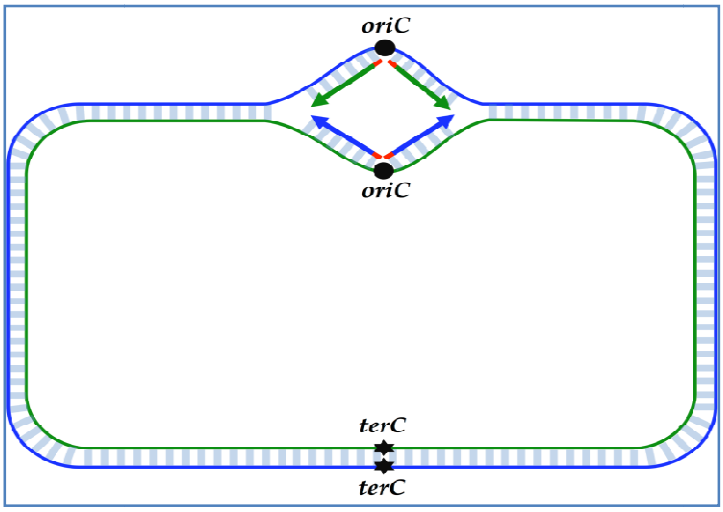
بسم الله الرحمن الرحيم

والصلاة والسلام على اشرف المرسلين سيدنا محمد صلى الله علية وسلم

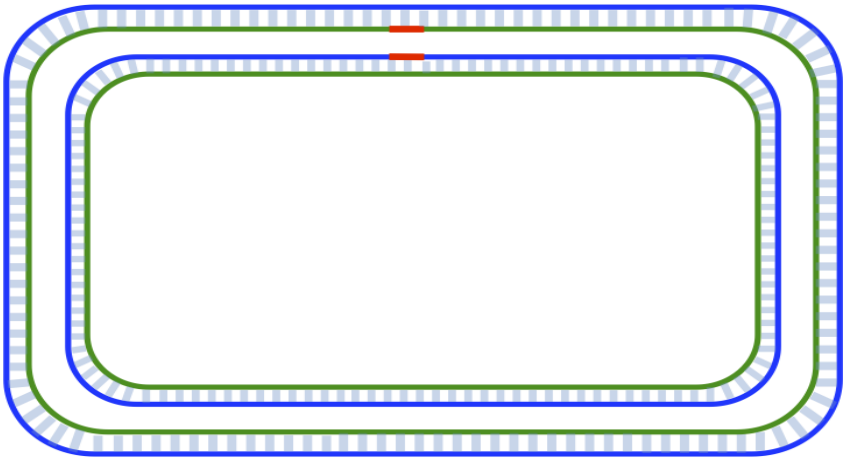
Bioinformatics Lap

الفصل الاول: أين يبدأ نســـــــــــــخ الجينوم (الجزء الثانى)

نحن الآن على استعداد لمناقشة عملية النسخ المتماثل بمزيد من التفاصيل .كما هو موضح في الشكل أدناه، نجد ان شريطان الحمض النووي ومكملة يعملان في اتجاهين متعاكسين حول كروموسوم دائري ، ابتداء من منطقة بداية النسخ oriC يتم التفرع الى فرعيين التي تتمدد في كل الاتجاهات حول كروموسوم حتى تصل الفروع المنفصلة الى نهاية خط النسخ المتماثل ويرمز لة بالرمز TERC

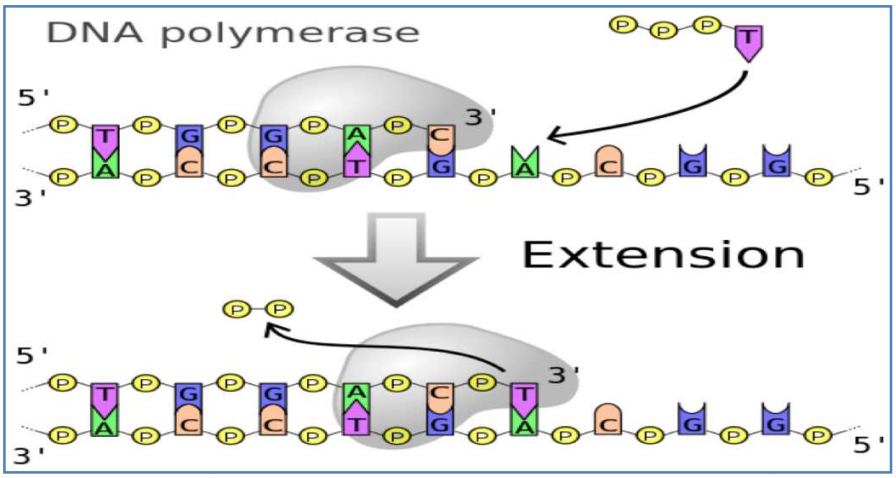


ويجب معرفة ان DNA polymerase لا ينتظر الى ان ينفصل الشريطين الابويين حتى يبدأ عملية النسخ بل كما موضوح بالشكل اعلاة حيث يبدأ النسخ بمجرد ان يتم بدأ انفصال الشريطين ولتتم عملية النسخ نحتاج اربعة DNA polymerase حيث كل واحد مسئول عن نسخ نصف شريط كما موضح بالشكل وكل واحد من DNA polymerase يحتاج **primer** وهو يعتبر مؤشر لبداية النسخ فى كل نصف شريط ويبدأ النسخ باضافة نيوكليوتيدة الى كل نصف شريط الى ان يتم النسخ كامل ونصل من oriC الى terC عندما يصل الاربع DNA polymerase الوصول الى terC فتكون انتهت عملية النسخ فنحصل على شريطيين متماثليين كما بالشكل التالى

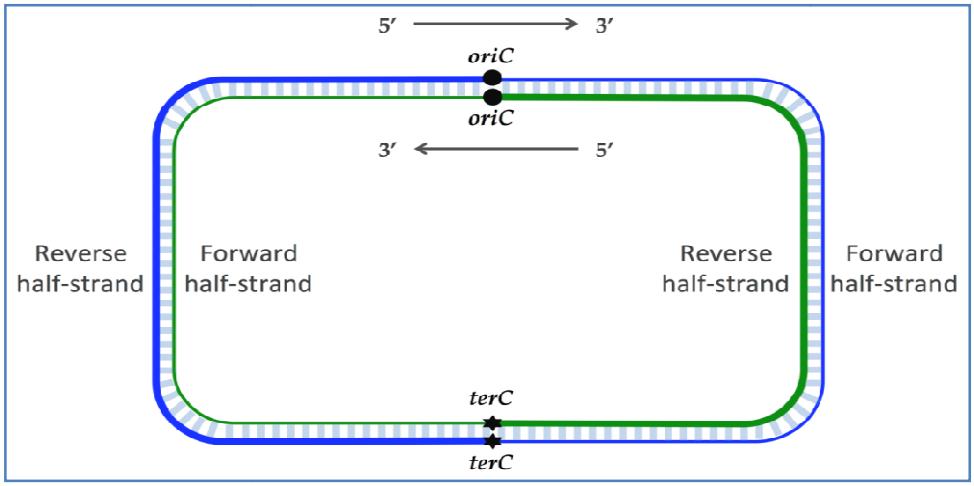


وعندما اكتب هذا الشرح فأن دكاترة البيولوجى يكتبون طلبا بفصلى لان ما تم شرحة سابقا يفتقد قاعدة مهمة جدا وهى أننا افترضنا ان DNA polymerase يقوم بعملية النسخ من الاتجاهيين (5′ ⟶ 3′ and 3′ ⟶ 5′ i.e.,)

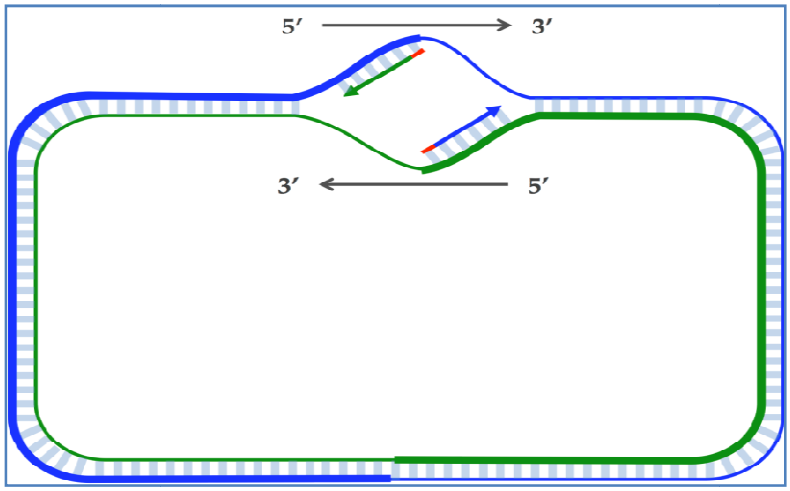
ولكن فى الواقع فأن DNA polymerase يستطيع النسخ فى اتجاة واحد فقط وهو 3′ ⟶5′ كما بالشكل فأن الثابت هو شريط 5′ ⟶ 3′ ويتم تكوين شريط المعاكس 3′ ⟶ 5′



فعندما نحدد الطرق من oriC الى terC فنجد ان لدينا اربع مسارات تنقسم الى اثنين forward half-strands حيث يكون الاتجاة تبعا 5′ ⟶ 3′ واثنين reverse half-strands من 3′ ⟶ 5′ كما موضح بالشكل الخطوط الرفعية والعريضة بالترتيب



شرحنا ما سبق لنبين قاعدة مهمة الا وهى ان عملية النسخ asymmetric حيث ان دائما DNA polymerase يستطيع ان يتحرك دائما فى اتجاة عكسى لل DNA فأنة يستطيع نسخ النيوكليوتدات من oriC الى terC فى reverse half-strands بسهولة اما فى المسارات forward (5′ ⟶ 3′) direction فلا يستطيع فهو يجب ان يعمل فى اتجاة عكسى الى oriCكما موضح بالصورة



ولهذا فأن DNA polymerase يعمل النسخ بسرعة فى المسارات reverse half-strands بينما يتأخر قليلا فى المسارات forward half-strand وسوف نتحدث الان كيف تمكن العلماء من الاستفادة من هذة الخاصية لايجاد منطقة بداية النسخ oriC للمزيد عن عملية **Asymmetry of Replication** فى الرابط

http://education-portal.com/academy/lesson/how-okazaki-fragments-of-the-lagging-strand-dna-are-replicated.html

***كيف تمكن العلماء من الاستفادة من خاصية Asymmetry of Replication لايجاد منطقة بداية النسخ oriC؟**

استفاد العلماء من هذة الظاهرة فى تحديد موقع بداية النسخ oriC حيث ان التأخر والسرعة فى النسخ من مسار الى مسار فى اى شريط يؤدى الى انة اذا كان هناك نيوكلوتيدة يحدث لها mutation كثير فى احد الشريطيين فأنها تقل فى الشريط المقابل للتوضيح نأخذ الجدول التالى وفية تم حساب عدد تكرار الاربع نيوكليدات فى Reverse half-strand و Forward half-strand

	C	G	A	T
Entire strand	427419	413241	491488	491363
Reverse half-strand	219518	201634	243963	246641
Forward half-strand	207901	211607	247525	244722
Difference	**+11617**	**-9973**	-3562	-1919

نلاحظ من الجدول ان نسبة تواجد A ليست مختلفة كثيرا فى المسارين Reverse half-strand و-Forward half strand ايضا هكذا لنيوكليوتيدة T، اما بالنسبة الى C فأنها توجد فى Reverse half-strand بنسبة اكبر من Forward half-strand وعلى الجانب الاخر ال G يحدث العكس يوجد بنسبة مرتفعة فى Forward half-strandعنها فى Reverse half-strand وهذا يجعلنا نبحث فى الجينوم عن المنطقة التى بها اختلاف بين قيم النيوكليدات C-G فتكون هى منطقة بداية النسخ oriC

للتطبيق نحن نعلم ان الجينوم circular لذا للبحث سوف نقطع جزء منة ليكون linear ونبدأ البحث فيها عن المنطقة التى يحدث اختلاف فى نسبة C, G ونطلق على *Skewi*(*Genome*) بأنها دالة تحسب مقدار الاختلاف بين C,G من بداية الجينوم

مثال للتوضيح

Skew(Prefix.i("GTCGA")) for i from 0 to 5 is
0 1 1 0 1 1
Because:
Prefix.0 is the string "" so skew is 0
Prefix.1 is the string "G" so skew is 1 (1 G - 0 C)
Prefix.2 is the string "GT" so skew is 1 (1 G - 0 C)
Prefix.3 is the string "GTC" so skew is 0 (1 G - 1 C)
Prefix.4 is the string "GTCG" so skew is 1 (2 G - 1 C)
Prefix.5 is the string "GTCGA" so skew is 1 (2 G - 1 C)

الان هل يمكن صياغة برنامج لايجاد المنطقة من الجينوم التى بها اقل skew

Minimum Skew Problem: Find a position in a genome minimizing the skew.
Input: A DNA string Genome.
Output: All integer(s) i minimizing Skewi (Genome) among all values of i (from 0 to |Genome|).

Sample Input:
TAAAGACTGCCGAGAGGCCAACACGAGTGCTAGAACGAGGGGCGTAAACGCGGGTCCGAT

Sample Output:
11 24

البرنامج بلغة البايثون

```
def Skew(text):
  """
  Skew('CATGGGCATCGGCCATACGCC')
  Sample Output:
  0 -1 -1 -1 0 1 2 1 1 1 0 1 2 1 0 0 0 0 -1 0 -1 -2
  """
  i=0
  prefix=''
  skewList=[]
  skewMinList=[]
  while i<=len(text):
    prefix=text[:i]
    skewList.append(countMySkew(prefix))
    i=i+1
  skewMin= min(skewList)
  myskewMin=findmyElement(skewList,skewMin)
  print myskewMin

def  countMySkew(p):
  number_G=p.count('G')
```

```
  number_C=p.count('C')
  different= number_G- number_C
  return different

def findmyElement(skewList,skewMin):
  j=0
  myskewMin=''
  while j<len(skewList):
    if skewList[j]==skewMin:
      myskewMin= myskewMin+' '+str(j)
    j=j+1

  return myskewMin
```

الان بعد ان استطعنا تحديد المنطقة التى فى الجينوم التى بها اقل skew فأننا حصلنا على المنطقة التى هى تقريبا منطقة بداية النسخ oriC وهى فى بكتريا E-coil التى درسنها من قبل فى الموقع 3923620 **ولكن اين هى الرسالة المخفية بها ما هو dnaA box** لنعود لبرنامجنا Frequent Words Problem ونبدأ البحث عن اكثر pattern مكرر متكون من 9 نيوكليوتيدات من الموقع 3923620 فى الجينوم لها عند الرجوع الى برنامج Frequent Words Problem فى الجزء الاول نجد الاجابة بعد التطبيق انة لا يوجد تكرار لاى pattern طولة 9 من بداية هذا الموقع مشابه تماما لل **ATGATCAAG** ومكملها **CTTGATCAT**

لكن قبل ان نفقد الامل يمكن ان نلاحظ شيء ان بالاضافة الى حدوث ثلاثة تكرار لل**ATGATCAAG** ومكملها **CTTGATCAT** فنلاحظ ايضا حدوث **ATGATCAAC** ومكملة **CATGATCAT** والذى يختلف مع **ATGATCAAG**و**CTTGATCAT** فقط بنيوكلوتيدة واحدة لذا

atca**ATGATCAAC**gtaagcttctaagc**ATGATCAAG**gtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaag**ATGATCAAG**agaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaag**CATGATCAT**ggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacct**CTTGATCAT**cgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagct**CTTGATCAT**gtt
tccttaaccctctatttttttacggaaga**ATGATCAAG**ctgctgct**CTTGATCAT**cgtttc

فأننا نجد تكرار بمقدار 9 لل pattern ATGATCAAG ومكمله CTTGATCAT ولو كان ال pattern التى اجدها متشابة معهم

افضل من ايجاد ال pattern ATGATCAAG ومكمله CTTGATCAT فقط 6 مرات

*اكتب برنامج كلمة ونص ورقم ويوجد كل المواقع التى بها حدوث لهذة الكلمة فى النص والمواقع للكلمات المختلفة عنها بمقدار dفقط

Approximate Pattern Matching Problem: ***Find all approximate occurrences of a pattern in a string.***

Input: **Two strings *Pattern* and *Text* along with an integer *d*.**

Output: **All positions where *Pattern* appears in *Text* with at most *d* mismatches.**

Sample Input:

ATTCTGGA

CGCCCGAATCCAGAACGCATTCCCATATTTCGGGACCACTGGCCTCCACGGT ACGGACGTCAATCAAAT

3

Sample Output:

6 7 26 27

البرنامج بلغة البايثون

```
def ApproximatePatternMatching(pattern,text,nmatch):
  """
ApproximatePatternMatching('ATTCTGGA','CGCCCGAATCCAGAACGCATTCCCATATTTCGGGACCAC
TGGCCTCCACGGTACGGACGTCAATCAAAT',3)
Sample Output:
  6 7 26 27
  """
  listOfIndex=''
  i=0
  while i<=len(text)-len(pattern):
    n =findwithnmatch(pattern,text,i,nmatch)
    if n<=nmatch:
      listOfIndex=listOfIndex+str(i)+' '
    i=i+1
  return  listOfIndex
def findwithnmatch(pattern,text,i,nmatch):
```

```
n1=0
text1=text[i:i+len(pattern)]
j=0
while j<len(pattern):
  if pattern[j]!=text1[j]:
      n1=n1+1
      if n1>nmatch:
        break
  j=j+1

return n1
```

الان سوف نغير البرنامج ونجعلة يبحث عن الكلمة والكلمات التى تتشابهة معها فى نص ما بمقدار d للتوضيح فمثلا
CGAAT and CGGAC هما نصين الاختلاف بينهم d=2

*اكتب برنامج يأخذ كلمة ونص ورقم يعبر عن عدد الاختلاف d ويعطى المخرج جميع الكلمات التى فى النص مساوية للكلمة اومختلفة معها بأختلاف درجة d

Frequent Words with Mismatches Problem: ***Find the most frequent k-mers with mismatches in a string***
Input: **A string *Text* as well as integers *k* and *d*. (You may assume *k* ≤ 12 and *d* ≤ 3.)**
Output: **All most frequent *k*-mers with up to *d* mismatches in *Text*.**

Sample Input:
ACGTTGCATGTCGCATGATGCATGAGAGCT 4 1
Sample Output:
GATG ATGC ATGT

```
def FrequentWordswithMismatches_1(text,kmer,nmatch):
  """
  FrequentWordswithMismatches_1('ACGTTGCATGTCGCATGATGCATGAGAGCT',4,1)
Sample Output:
  GATG ATGC ATGT
  """
  list_1=[]
```

```
  str_1=''
  i=0
  while i<=len(text)-kmer:
    pattern_1=text[i:i+kmer] # cut to decide my pattern
    list_1.extend(generatepermutationWith_Dmismatch(pattern_1,nmatch))
    i=i+1
  Final_list=[]
  str=''
  i=0
  while i<len(list_1):
    count =countPatternwithnmatch(list_1[i],text,nmatch)#try to find number of mismatches in whole text
    Final_list.append([list_1[i], count])
    i=i+1
  max=0
  for x in xrange(len(Final_list)):
       if Final_list[x][1]>max:
        max=Final_list[x][1]
        most_frqcount=max
  for y in xrange(len(Final_list)):
    if Final_list[y][1]==most_frqcount and str.find(Final_list[y][0])==-1:
         str=str+' '+Final_list[y][0]
  print('the most frequent', kmer,'-mer are ', str)
  print("with frquent count =",most_frqcount)
  return str

def countPatternwithnmatch(pattern,text,nmatch):

  m=0
  count=0 # sum of occurence of mismatch=nmatch
  while m<=len(text)-len(pattern):
    text1=text[m:m+len(pattern)]
   # print 'my tex'+text1
```

```
    j=0
    n1=0
    while j<len(pattern):
      if pattern[j]!=text1[j]:
        n1=n1+1
        if n1>nmatch:
          break
      j=j+1
    if n1<=nmatch:
      count=count+1
    m=m+1
  return count
def permutation(list):
  letter='ACGT'
  mylist=[]
  for pattern in list:
    i=0
    while i<len(pattern):
      j=0
      while j<len(letter):
        if letter[j]!=pattern[i]:
          mylist.append(pattern[:i]+letter[j]+pattern[i+1:])
        j=j+1
      i=i+1
  return mylist

def generatepermutationWith_Dmismatch(pattern,d):
  ilist=[pattern]
  for i in range(d):
    ilist =permutation(ilist)
  return ilist
```

لكن للسائل ان يسأل ولماذا لا نبحث ايضا عن complement فعلا نقطة مهمة نغير فى البرنامج لايجاد الكلمة والكلمات التى تشابها والمكمل لها والكلمات التى تتشابهة مع المكمل

Frequent Words with Mismatches and Reverse Complements Problem: *Find the most frequent k-mers (with mismatches and reverse complements) in a DNA string.*

Input: A DNA string *Text* as well as integers *k* and *d*.

Output: All *k*-mers *Pattern* maximizing the sum $Count_d(Text, Pattern) + Count_d(Text, Pattern)$ over all possible *k*-mers.

Sample Input:

ACGTTGCATGTCGCATGATGCATGAGAGCT
4 1

Sample Output:

ATGT ACAT

البرنامج بلغة البايثون

```
def FrequentWordswithMismatchesReverseComplements(text,kmer,nmatch):
  """
FrequentWordswithMismatchesReverseComplements('ACGTTGCATGTCGCATGATGCATGAGAGCT',4,1)
Sample Output:
   ATGT ACAT
  """
  list_1=[]
  str_1=''
  i=0
  while i<=len(text)-kmer:
    pattern_1=text[i:i+kmer] # cut to decide my pattern
    list_1.extend(generatepermutationWith_Dmismatch(pattern_1,nmatch))
    list_1=list(set(list_1))
    i=i+1
  list_reverse=[]
```

```
    # find comlemment to each pattern in list
    for p in list_1:
        list_reverse.append(complement(p))
    Final_list=[]
    str=''
    i=0
    totalCount=0
    while i<len(list_1):
        count_ =countPatternwithnmatch(list_1[i],text,nmatch)#try to find number of mismatches in whole text
        count_reverse =countPatternwithnmatch(list_reverse[i],text,nmatch)#try to find number of
mismatches in whole text
        totalCount=count_+count_reverse
        Final_list.append([list_1[i],list_reverse[i], totalCount])
        i=i+1
    max=0
    for x in xrange(len(Final_list)):
            if Final_list[x][2]>max:
             max=Final_list[x][2]
             most_frqcount=max
    for y in xrange(len(Final_list)):
        if Final_list[y][2]==most_frqcount and str.find(Final_list[y][0])==-1 and
str.find(Final_list[y][1])==-1:
             str=str+' '+Final_list[y][0]+' '+Final_list[y][1]
    print('the most frequent', kmer,'-mer are ', str)
    print("with frquent count =",most_frqcount)
    return str


def countPatternwithnmatch(pattern,text,nmatch):
    m=0
    count=0 # sum of occurence of mismatch=nmatch
    while m<=len(text)-len(pattern):
        text1=text[m:m+len(pattern)]
       # print 'my tex'+text1
```

```
    j=0
    n1=0
    while j<len(pattern):
      if pattern[j]!=text1[j]:
        n1=n1+1
        if n1>nmatch:
          break
      j=j+1
    if n1<=nmatch:
      count=count+1
    m=m+1
  return count

def permutation(list1):
  list1=list(set(list1))
  letter='ACGT'
  mylist=[]
  for pattern in list1:
    i=0
    while i<len(pattern):
      j=0
      while j<len(letter):
        if letter[j]!=pattern[i]:
          mylist.append(pattern[:i]+letter[j]+pattern[i+1:])
        j=j+1
      i=i+1
  return mylist

def generatepermutationWith_Dmismatch(pattern,d):
  ilist=[pattern]
  for i in range(d):
    ilist =permutation(ilist)
```

```
  return ilist

def reverse(word):
  reverse_word=''
  i=len(word)-1
  while i>=0:
    reverse_word=reverse_word+word[i]
    i=i-1
  return reverse_word

def complement(p):

  rPattern=reverse(p)
  cPattern=''
  for i in range(len(rPattern)):
    if rPattern[i]=='A':
      cPattern=cPattern+'T'
    elif rPattern[i]=='T':
      cPattern=cPattern+'A'
    elif rPattern[i]=='C':
      cPattern=cPattern+'G'
    elif rPattern[i]=='G':
      cPattern=cPattern+'C'
    else:
      cPattern=cPattern+'_'

  return cPattern
```

اخيرا لايجاد الرسالة المخفية فى منطقة بداية النسخ لبكتريا Ecoil فنتبع الاتى:

1- ايجاد minimum skew باستخدام برنامج skew لتحديد منطقة بداية النسخ oriC

2- استخدام برنامج FrequentWordswithMismatchesReverseComplements لايجاد dnaA box الرسالة المخفية التى هى ال patterm الاكثر تكرار ومكملة ومتشابهة

3- هذا رابط فية الجينوم الكامل لبكتريا Ecoil

https://stepic.org/media/attachments/lessons/10/E-coli.txt

نطرح بعض المشاكل المفتوحة لتكملة البحث فمثلا بكتريا *Salmonella typhimurium* *هى متشابهة مع بكتريا Ecoil*

وبعد ما تعلمنا ايجاد dnaA box لبكتريا Ecoil فهل يمكنكم ايجاد ال dnaA box لبكتريا Salmonella typhimurium

Open problem:

- Multiple Replication Origins in a Bacterial Genome
- Finding Replication Origins in Draft Bacterial Genomes
- Finding Replication Origins in Archaea
- Finding Replication Origins in Yeast
- Computing Exact Probabilities of Patterns in a String and the Overlapping Words Paradox

رَبِّ ابْنِ لِي عِنْدَكَ بَيْتًا فِي الْجَنَّةِ

نسألكم بالله الدعاء لنا بالجنـــة وأن يتقبل الله عز وجل العمل خالصـــا لوجة- اميـــــــــن

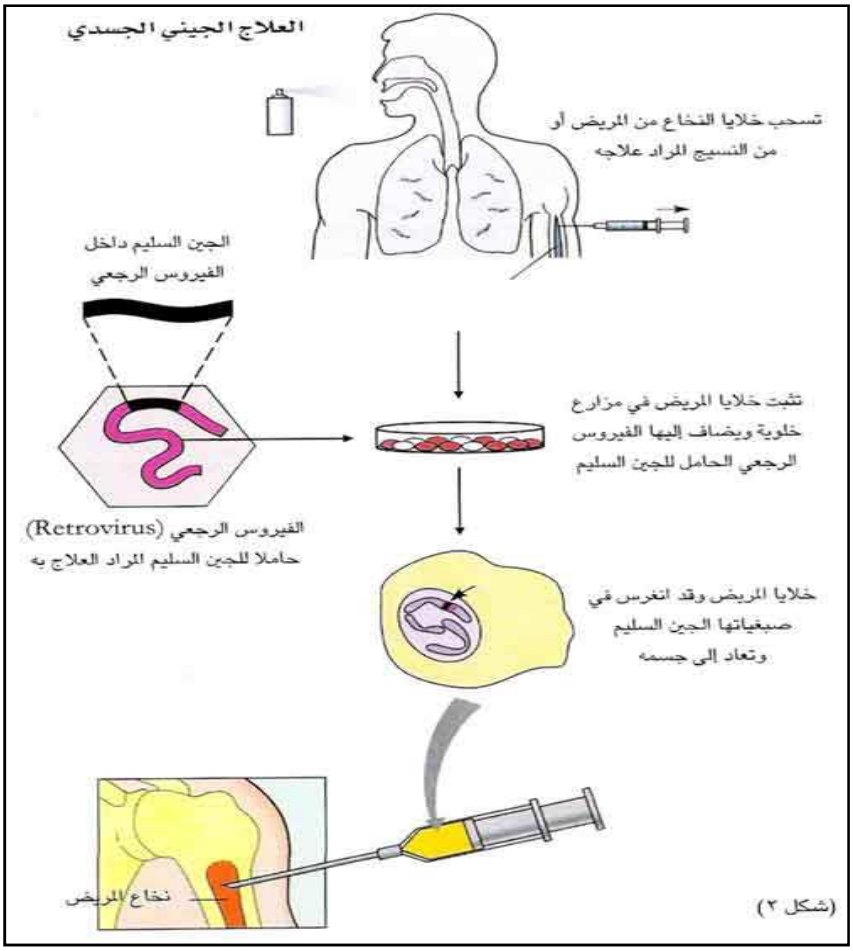
الحمد الله رب العالميين –اللهم دومها نعمة واحفظها من الزوال –امين -اللهم ارزقنا بكل حرف مغفرة ورضوان امين يا رب

إِنَّ اللَّهَ وَمَلَائِكَتَهُ يُصَلُّونَ عَلَى النَّبِيِّ يَاأَيُّهَا الَّذِينَ آمَنُوا صَلُّوا عَلَيْهِ وَسَلِّمُوا تَسْلِيمًا

~~مثـــــــال على أهميـــة معرفـــة منطقـة بدايـة النسـخ~~

العلاج الجينى **gene therapy** يمكن تعريف العلاج الجيني على أنه علاج أمراض عن طريق استبدال الجين المعطوب بآخر سليم (gene replacement) أو إمداد خلايا المريض بعدد كاف من الجينات السليمة (Gene transfer) تقوم هذه الجينات بالعمل اللازم وتعوض المريض عن النقص في عمل جيناته المعطوبة

كيفية العلاج الجينى gene therapy يتم العلاج الجينى عن طريق حقن المريض الذي يفتقر الى الجين السليم بناقل فيروسي viral vector يحتوي على جينات اصطناعية مشفرعليها البروتين العلاجى ثم داخل الخلية هذة الجينات الاصطناعية يتم نسخها لتنتج البروتين العلاجى والذى بدروة يعالج المريض. للتأكد من أن هذة الجينات الاصطناعية تقوم بعملية النسخ replicate داخل الخلية بشكل صحيح ، يجب أن يتم معرفة oriC الذى يبدأ عندة نسخ الجين . الشكل يوضح خطوات العلاج الجينى .



~~~~The Gold Bug puzzle~~

القصة باختصار تحكى عن معرفة احدهم بوجود كنز فى منطقة ما ويريد فك شفرة لرسالة ما - لمعرفة موقع هذا الكنز والرسالة وهـــــــــــــــــــى:

53‡‡†305))6·;4826)4‡.)4‡);806·;48†8^60))85;161;:‡·8

†83(88)5·†;46(;88·96·?;8)·‡(;485);5·†2:·‡(;4956·2(5

·—4)8^8·;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4

)485†528806·81(‡9;48;(88;4(‡?34;48)4‡;1‡(;:188;‡?;

فهل يمكن حل اللغز ؟

فى الحقيقة تم التفكير كثيرا والنظر فى الشفرة كثيرا وقد لاحظ ان هناك ثلاث حروف متتابعيين مكررين فى النص وهم **48;**

53‡‡†305))6·;4826)4‡.)4‡);806·;48†8^60))85;161;:‡·8

†83(88)5·†;46(;88·96·?;8)·‡(;485);5·†2:·‡(;4956·2(5

·—4)8^8·;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4

)485†528806·81(‡9;48;(88;4(‡?34;48)4‡;1‡(;:188;‡?;

وبما ان كاتب اللغز يعلم اللغة الانجليزية فقط فسأل ما هى اكثر الادوات المعرفة فى اللغة الانجليزية هى the لذا هو وضع الحرف ; مقابل حرف t ورقم 4 مقابل حرف h ورقم 8 مقابل حرف e الان فى كل الشفرة قم بعملية احلال نحصل على:

53‡‡†305))6·THE26)H‡.)H‡)TE06·THE†E^60))E5T161T:‡·E

†E3(EE)5·†TH6(TEE·96·?TE)·‡(THE5)T5·†2:·‡(TH956·2(5

·—H)E^E·TH0692E5)T)6†E)H‡‡T1(‡9THE0E1TE:E‡1THE†E5TH

)HE5†52EE06·E1(‡9THET(EETH(‡?3HTHE)H‡T1‡(T:1EET‡?T

هل يمكنك تكملة حل اللغز ؟ اذا لم تعلم اليك الرابط للحل

http://classiclit.about.com/library/bl-etexts/eapoe/bl-eapoe-goldbug.htm