Eksamensopgave i Bioinformatik og Programmering

Denne eksamensopgave består af to dele, som vægtes lige i bedømmelsen:

1. Et sæt programmeringsopgaver.
2. Et sæt bioinformatikopgaver.

Filer til brug ved eksamen

Udover denne PDF-fil som indeholder eksamensopgaverne, har du også downloaded tre andre filer fra Digital Eksamen, som du skal bruge til at løse eksamensopgaven:

- `progexam.py`: Det er i denne fil, du skal skrive de Python funktioner, der bedes om i eksamensopgavens programmeringsdel.
- `test_progexam.py`: Det er denne fil, du kan bruge til at teste de funktioner du skriver i `progexam.py`.
- `bioinfexam.py`: Det er i denne fil, du skriver svarene på eksamensopgavens bioinformatik-del.

Sådan løser du programmeringsopgaverne

Start med at åbne din terminal og naviger ind i den folder som Digtal Eksamen har lavet på din computer. Der er muligt at folderens navn indeholder mellemrum. For at navigere ind i en folder der indeholder et mellemrum vha. terminalen, kan man skrive starten af foldernavnet og så trykke på Tab. Så fuldendes navnet automatisk. F.eks.: Hvis folderen hedder "Digital Eksamen", kan man skrive: "cd Digital" og så trykke Tab. Så fuldendes navnet og man kan trykke Enter.

Som i programmeringsprojekterne fra kurset skriver du din kode i `progexam.py` og kører koden sådan her:

```
python progexam.py
```

Som i programmeringsprojekterne i kurset kan du teste din kode sådan her:

```
python test_progexam.py
```

Test scriptet er tilgængeligt som en hjælp til at teste din kode, men du har selv det fulde ansvar for rigtigheden af din kode.

Det er tilladt at bruge løsninger af opgaver til at løse senere opgaver. Man må altså gerne kalde tidligere definerede funktioner inde i andre funktioner man senere bliver bedt om at skrive.

Følgende er *afgørende* for at din eksamensbesvarelse kan evalueres korrekt:

1. Hver funktion skal navngives *præcis* som angivet i opgaven. Funktioner der ikke er navngivet korrekt regnes som ikke besvarede.
2. Det er ikke tilladt importere kode fra andre filer du har skrevet eller installeret. Det vil sige at du ikke må bruge `import` statements i din fil.
3. Når du afleverer `progexam.py` må den *kun* indeholde definitioner af de funktioner der er beskrevet i eksamensopgaven. Al kode udenfor funktionsdefinitioner skal slettes inden du afleverer, så sørg for at teste i god tid inden aflevering, om dine funktioner stadig virker, når du sletter sådan ekstra kode.

Allervigtigst: Funktioner der ikke fuldstænding opfylder opgavens beskrivelse regnes som ikke besvarede. Så sørg for at lave dine funktioner færdige, så de klarer *alle* tests. Hvis

ingen af dine funktioner er *helt rigtigt besvaret* får du *ingen point*.

Sådan løser du bioinformatikopgaverne

Bioinformatikdelen af eksamensopgaven består af et sæt af opgaver, der hver dækker et emne. Hver opgave indeholder flere delopgaver. Der er to typer delopgaver:

1. Udsagn der enten er sande eller falske og som skal besvares med `True` eller `False`.
2. Spørgsmål der skal besvares med et tal.

Filen `bioinfexam.py` er en Python fil og indeholder en variabel for hver delopgave. For eksempel: den variabel der hører til delopgave tre i emne syv hedder `emne_7_del_3` og har som default værdien `None`:

```
emne_7_udsagn_3 = None
```

Du besvarer sandt/falsk udsagn ved at udskifte `None` med enten `True` eller `False`. Du besvarer spørgsmål ved at udskifte `None` med et tal. Det er anført `bioinfexam.py` om en delopgave skal besvares med `True`/`False` eller et tal.

Følgende er *afgørende* for at din eksamensbesvarelse kan evalueres korrekt: Udsagn der ikke er besvaret med `True`, `False` eller et tal betragtes som forkert besvarede, *så du er bedst tjent med at gætte fremfor ikke at svare*.

I nogle af opgaveemnerne refererer statements til en vist illustration. Alt efter størrelsen på din skærm kan du være nødt til at "zoome ind" på illustrationerne i det program du bruger til at vise denne PDF, ellers kan der være detaljer du ikke kan se.

Sådan afleverer du din eksamensopgave i Digital Eksamen

Inden du afleverer skal du tjekke at `progexam.py` kun indeholder definitioner af de funktioner der er beskrevet i eksamensopgaven. Hvis du har skrevet yderligere kode for at teste dine funktioner, skal du slette den inden du oplader din fil.

Du afleverer din eksamensbesvarelse ved at uploade disse to filer til Digital Examen:

- `progexam.py` skal afleveres som ***hoveddokument***
- `bioinfexam.py` skal afleveres som ***bilag***.

Eksamensopgaverne starter på næste side

Programming assignments

This assignment is in English to make it most like what you are used to from the programming exercises in the course.

In the assignment, Python code will look like this:

```
print(”hello world”)
```

Whenever I refer to Python code inside a sentence it will be styled like `this`.

In this exam assignment, you will work with DNA sequences and we will assume that DNA sequences are always upper-case and that they can only contain the characters A, T, G, and C.

The actual assignment starts below.

Happy coding.

Problem 1

You want to be able to compare two DNA sequences and find out which of them is the longest.

Write a function, `longest_sequence`, which takes two arguments:

1. A string, which represents a DNA sequence.
2. A string, which represents a DNA sequence.

The function must return:

- An string, which represents the longest of the two function arugments. If the two string arguments have the same length, the function must return the *first* argument.

Example usage: If the function is called like this:

```
longest_sequence('ATG', 'TTCG')
```

then it should return:

```
'TTCG'
```

Problem 2

You would like to be able to determine how many times each base occurs in a DNA sequence. You can assume that the only characters in the sequence are A, T, G, and C.

Write a function, `numbers_of_bases`, which takes one argument:

1. A string, which represents a DNA sequence.

The function must return:

- A dictionary, in which keys are strings that represent bases and values are integers that represent the number of occurrences of each base in the sequence. If a base is not found in the sequence, its count must be zero.

Example usage: If the function is called like this:

```
numbers_of_bases("AGTTTT")
```

then it should return (not necessarily with key-value pairs in the same order):

```
{"A": 1, "T": 4, "C": 0, "G": 1}
```

Problem 3

You would like to be able to count overlapping triplets (three consecutive bases) in a sequence. That is, you want to determine how many times each triplet occurs in the sequence. You can assume that the only characters in the sequence are A, T, G, and C.

Write a function, `numbers_of_triplets`, which takes one argument:

1. A string, which represents a DNA sequence.

The function must return:

- A dictionary, in which keys are strings that represent triplets (three consecutive bases) and values are integers that represent the number of occurrences of each triplet in the sequence. The dictionary must only contain triplets that are found in the sequence.

Example usage: If the function is called like this:

```
numbers_of_triplets("AGTTTT")
```

then it should return (not necessarily with key-value pairs in the same order):

```
{"AGT": 1, "GTT": 1, "TTT": 2}
```

Problem 4

Given a dictionary like that returned by `numbers_of_triplets`, you want to find the triplet with the highest count. You are allowed to assume that there is not more than one triplet with this maximum count.

Write a function, `most_common_triplet`, which takes one argument:

1. A dictionary, which represents counts of triplets.

The function must return:

- An string, which represents the *key* in the dictionary that is associated witht the highest *value*. I.e. the triplet with the highest count.

Example usage: If the function is called like this:

```
most_common_triplet({'AGT': 1, 'GTT': 1, 'TTT': 2})
```

then it should return:

```
'TTT'
```

Problem 5

A list of lists represents a matrix. If the number of sub-lists equals the length of each sub-list, then you have square matrix. Such a matrix can have any size. Given such a square matrix, you want to be able to compute the column sums of the matrix. In the small 2 by 2 matrix below, the these column sums are 2 and 5:

```
[[1, 2],
 [1, 3]]
```

The function should work on square matrices of any size. I.e. not just on 2 by 2 matrices

Write a function, `column_sums`, which takes one argument:

1. A list of lists of integers, that represents a sqare matrix.

The function must return:

- An list of integers, which represents the column sums in the matrix represented by the function argument.

Example usage: If the function is called like this:

```
column_sums([[1, 2], [1, 3]])
```

then it should return:

```
[2, 5]
```

Problem 6

Given a DNA sequence, you want to compute the mean distance between all pairs of positions in the sequence that is occupied by some base. E.g if the sequences is `'ATTATTAT'`, then there are three As that can be paired in different ways. The distances between these three pairs are 3, 6 and 3. So the mean distance betweeen pairs is 4.

Write a function, `mean_base_dist`, which takes two arguments:

1. A string, which represents a DNA sequence.
2. A string, which represents a single base.

The function must return:

- An float, which represents the mean distance between all pairs of positions in argument one, where the base in argument two is found. The function should return `None` if the specified base does not exist in the DNA sequence.

Example usage: If the function is called like this:

```
mean_base_dist('ATTATTAT', 'A')
```

then it should return:

```
4
```

Problem 7

Given two sequences of equal length, you want to be able to find all the positions where they differ.

Write a function, `find_differences`, which takes two arguments:

1. A string, which represents a DNA sequence.
2. A string, which represents a DNA sequence of same length as argument one.

The function must return:

- An list of integers, which represents the indexes where the two sequences do not have the same base.

Example usage: If the function is called like this:

```
find_differences('AAAAAA', 'ATAATA')
```

then it should return:

```
[1, 4]
```

Problem 8

Given two sequences of equal length, you want to change the bases from uppercase to lowercase at all the positions where the two sequences differ.

Write a function, `lowercase_differences`, which takes two arguments:

1. A string, which represents a DNA sequence.
2. A string, which represents a DNA sequence of same length of argument one.

The function must return:

- A list of two strings, which each represents a modified copy of the two arguments.

Example usage: If the function is called like this:

```
lowercase_differences('TTT', 'ATG')
```

then it should return:

```
['tTt', 'aTg']
```

Problem 9

You would like to find all unique subsequences in a larger DNA sequence. A subsequence is a sequence that is smaller than the full sequence. You can assume that the only characters in the two sequences are A, T, G, and C.

Write a function, `get_subsequences`, which takes one argument:

1. A string, which represents a DNA sequence.

The function must return:

- A list of strings that each represents a substring of the larger string argument. This list must be sorted alphabetically and contain all unique subsequences. The list must not include empty strings or duplicate subsequences. The list must be empty if the string argument is shorter than two, in which case no subsequences can be found.

Example usage: If the function is called like this:

```
get_subsequences("ATGC")
```

then it should return:

```
["A", "AT", "ATG", "C", "G", "GC", "T", "TG", "TGC"]
```

Problem 10

You would like to find all subsequences that a DNA sequence shares with a number of other DNA sequences of equal length. You can assume that the only characters in the two sequences are A, T, G, and C.

Write a function, `get_shared_subsequences`, which take two arguments:

1. A strings that represents a DNA sequence.
2. A list of strings that each represent a DNA sequence.

The function must return:

- A list of substrings that the first argument shares with all strings in the list argument. The returned list must be sorted alphabetically and contain all unique shared substrings. The list must not include empty strings or duplicate substrings. The list must be empty if no shared substrings are found.

Example usage: If the function is called like this:

```
get_shared_subsequences("AATG", ["ATGT", "CATG"])
```

then it should return:

```
["A", "AT", "ATG", "G", "T", "TG"]
```

Bioinformatikopgaver

Emne 1: Databaser

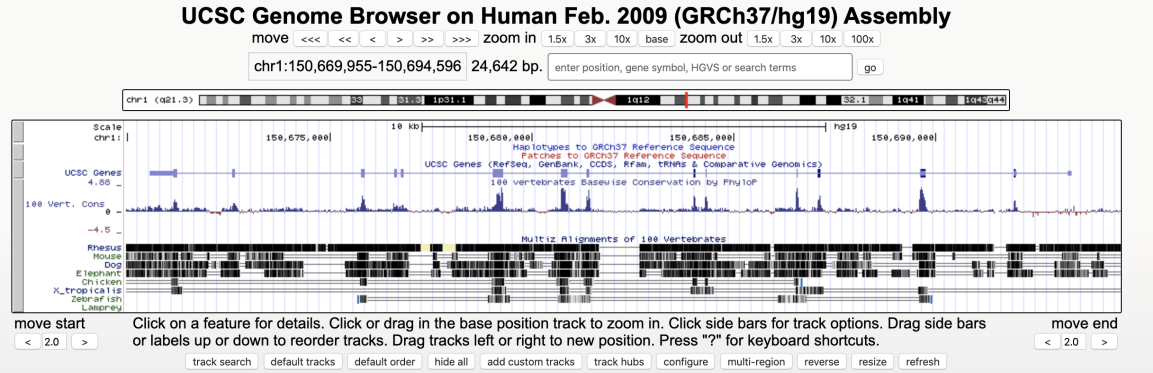
Online databaser giver adgang til et væld af forskellig information. Nogen indeholder rå sekvens information. Andre samler forskellige typer af information relevant for f.eks. gener, proteiner, sygdomme, eller pathways.

Udsagn og/eller spørgsmål:

1. OMIM er en database, der sammenfatter information om arvelige sygdomme. (True/False) True
2. Genbank er en database, der indeholder alle offentligt tilgængelige nukleotid sekvenser. (True/False) True
3. PDB er en database, der indeholder information om RNA sekundær strukturer. (True/False) False
4. Gene er en database, der sammenfatter metainformation om hvert enkelt gen. (True/False) True

Emne 2: UCSC genome browser

Figuren nedenfor er et screenshot fra UCSC genome browseren og viser "UCSC Genes" og "Conservation" tracks for HORMAD1 genet. Du kan være nødt til at zoome ind for at se alle detaljer:



Udsagn og/eller spørgsmål:

1. Hvor mange exons har genet? (talværdi) 15
2. Introns på figuren er generelt mere konserverede end exons. (True/False) False
3. Hvor mange exons er del af den kodende region? (talværdi) 14, vi skal kun tælle de tykkeste
4. Længden af det viste gene er 24642 baser. (True/False) False, det er længde af det afsnit vi kigger på

Emne 3: Needleman-Wunsch og Smith-Waterman

Needleman-Wunsch og Smith-Waterman algoritmerne er begge algoritmer til henholdsvis globalt og lokalt parvist alignment.

Udsagn og/eller spørgsmål:

1. Begge algoritmer identificerer et parvist alignemnt, der er optimalt givet en scoringsmatrix og en gap score. (True/False) True
2. Der er aldrig mere end ét optimalt alignment for en given scoringsmatrix og gap score. (True/False) False
3. Tiden det kræver at beregne et lokalt alignment ved hjælp af Smith-Waterman algoritmen, er proportional med produktet af de to sekvensers længde. (True/False) True
4. Smith-Waterman algoritmen vil i nogen tilfælde producere et globalt alignment. (True/False) True

Emne 4: Needleman-Wunch versus Smith-Waterman

Du har det splicede transcript af et gen med flere exons (dvs. uden intros). Du vil gerne aligne denne sekvens til den genomiske (usplejsede) sekvens, som du tror dit splejsede transkript stammer fra.

Udsagn og/eller spørgsmål:

1. Du kan med fordel benytte Smith-Waterman algoritmen. (True/False) True
2. Du kan med fordel benytte afine cost til at score gaps. (True/False) True
3. Du kan med fordel benytte en lav gap cost for gaps i enderne. (True/False) True
4. Når der benyttes afine cost i Needleman-Wunch algoritmen, er tiden det kræver at beregne et globalt alignment proportional med produktet af de to sekvensers længde. (True/False) True, man skal lave det samme antal beregninger uanset om det er afine cost eller ej

Emne 5: Globalt alignment af to korte sekvenser

Nedenfor ser du to korte DNA sekvenser:

```
sekvens 1: TGG
sekvens 2: AATG
```

Lav et globalt alignment af de to sekvenser vha. Needleman-Wunsch algoritmen. Du skal bruge en match score på 1, en mismatch score på -1 og en gap score på -2 . Du kan gøre dette ved at udfylde en tabel vha. dynamisk programmering.

Udsagn og/eller spørgsmål:

1. Hvad er den optimale alignment score? (talværdi) -3
2. Hvor mange optimale alignments er der? (talværdi) 3
3. Der findes et optimalt alignemnt, der har tre gaps. (True/False) False
4. Der findes et optimalt alignment der har to gaps. (True/False) False
5. Alle optimale alignments aligner de to sidste baser i hver sekvens (G) med hindanden. (True/False) True

Emne 6: Lokalt alignment af to korte sekvenser

Nedenfor ser du en dynamisk programmerings matrix for Smith-Waterman alignment af følgende to sekvenser:

```
sekvens 1: GGTCG
sekvens 2: GGCG
```

S		G_1	G_2	C_3	G_4
	0	0	0	0	0
G_1	0	1	1	0	1
G_2	0	1	2	0	1
T_3	0	0	0	1	0
C_4	0	0	0	1	0
G_5	0	1	1	0	2

Der er brugt en match score på 1, en mismatch score på -1, og en gap score på -2.

Udsagn og/eller spørgsmål:

1. Havd er den optimale alignment score? (talværdi) 2
2. Hvor mange optimale alignments er der? (talværdi) 2
3. Der findes et optimalt lokalt alignment, der spænder tre baser i sekvens 1 og to baser i sekvens 2. (True/False) False
4. Der findes et optimalt lokalt alignment, hvor baserne CG fra sekvens 1 aligner til baserne CG fra sekvens 2. (True/False) True

Emne 7: Parvist alignment ved hjælp af EBIs web tools.

Følgende to sekvenser er blevet alignet med et af EBI's web tools:

>A
GCSHIHSIHVKEFVRVENVKGAVWTVDEVEYQKRRSQKITGSPTLVKNIPTSLGYGAALNASLQAALAESSLP
LLSNPGLINNASSGLLQAVHEDLNGSLDHIDSNGNSSPGCSHIHSIHVKEEPVIAEDEDCTANHSPELED

>B
FVRVENVKGAVWTVEYQKQKITGSPTLVKNIPTSLGYGAALNASLQAALAESSLPLLSNPGLINNASSGLLQA
VHEDLNGSLDHIDSNGNSSPGCSHIHSIHVKEEPVIAEDEDCPMSLVTTANHSPELED

Nedenfor ser du et screen shot med output fra et parvist alignment foretaget ved hjælp af EBIs webservice.

```
#=======================================
#
# Aligned_sequences: 2
# 1: A
# 2: B
# Matrix: EBLOSUM62
# Gap_penalty: 10.0
# Extend_penalty: 0.5
#
# Length: 137
# Identity:     125/137 (91.2%)
# Similarity:   125/137 (91.2%)
# Gaps:          12/137 ( 8.8%)
# Score: 610.5
#
#
#=======================================

A                 13 FVRVENVKGAVWTVDEVEYQKRRSQKITGSPTLVKNIPTSLGYGAALNAS     62
                     |||||||||||||   |||||   ||||||||||||||||||||||||||
B                  1 FVRVENVKGAVWT---VEYQK---QKITGSPTLVKNIPTSLGYGAALNAS     44

A                 63 LQAALAESSLPLLSNPGLINNASSGLLQAVHEDLNGSLDHIDSNGNSSPG    112
                     ||||||||||||||||||||||||||||||||||||||||||||||||||||
B                 45 LQAALAESSLPLLSNPGLINNASSGLLQAVHEDLNGSLDHIDSNGNSSPG     94

A                113 CSHIHSIHVKEEPVIAEDEDC------TANHSPELED    143
                     |||||||||||||||||||||      ||||||||||
B                 95 CSHIHSIHVKEEPVIAEDEDCPMSLVTTANHSPELED    131
```

Udsagn og/eller spørgsmål:

1. Det viste alignment er et globalt alignment. (True/False) False
2. Det viste alignment benytter afine gap scoring. (True/False) True
3. De seks gap karakterer i sekvens A vil bidrage med en score på -60. (True/False) False, -12,5
4. De seks gap karakterer i sekvens B reducerer tilsammen alignmentscoren mere end de seks gap karakterer i sekvens A gør. (True/False) True

Emne 8: Multipelt alignment

Der findes flere forskellige programmer til multipelt alignment. Et af dem er Clustal Omega, men der er mange andre.

Udsagn og/eller spørgsmål:

1. Needleman-Wunch algoritmen kan udvides til at aligne tre sekvenser ved at fylde en tabel ud i tre dimentioner i stedet for to. (True/False) True (men ikke hurtigt)
2. Clustal Omega producerer altid et optimalt alignment for en given scoringsmatrix og gap score. (True/False) False
3. Multiplet alignment kan identificere homologi mellem fjernt beslægtede sekvenser, som ikke kan identificeres ved hjælp af parvist alignment. (True/False) True
4. Forskellige programmer til multipelt alignment vil producere det samme alignment, hvis der benyttes den samme scoringsmatrix og gap cost. (True/False) False

Emne 9: Blast

BLAST programmet implementerer en algoritme til at søge med en sekvens i en stor database af sekvenser. NCBI giver online adgang til BLAST mod sekvenserne i Genbank, men man kan også lave sin egen lille database på sin egen computer og så søge mod den.

Udsagn og/eller spørgsmål:

1. BLAST algoritmen garanterer at finde det bedste alignment mellem en sekvens i databasen og den sekvens man søger med. (True/False) False
2. Den score, der angives for hver identificeret databasesekvens, repræsenterer antallet af baser, der aligner til søgesekvensen. (True/False) False
3. Den E-værdi, der angives for hver for hver identificeret databasesekvens, angiver sandsynligheden for at finde et match i databasen, med samme eller bedre score ved rent tilfælde. (True/False) True
4. Hvis man finder et perfekt match i en stor database, vil E-værdien være større end hvis man finder samme match i en lille database. (True/False) True

Emne 10: Modeller for DNA evolution

Jukes-Cantor og Kimura modellerne er modeller for DNA evolution.

Udsagn og/eller spørgsmål:

1. Begge modeller kan bruges til at udregne den evolutionære afstand mellem to sekvenser ud fra andelen af forskelle mellem dem. (True/False) True
2. Jukes-Cantor modellen lader os korrigere for, at den samme positition i sekvensen har muteret flere gange. (True/False) True
3. Kimura modellen korrigerer for, at transitioner og transversioner er ikke er lige hyppige. (True/False) True
4. Hvis 75% af baserne i to alignede sekvenser er forskellige, så vil den Jukes-Cantor-korrigerede afstand være uendeligt stor. (True/False) True

5. **Models that assume all nucleotides occur at equal frequencies (25%)**
    1. *The Jukes-Cantor (**JC**) model*
        1. All substitutions are equally likely.
        2. All nucleotides occur at the same frequency (25%).
        3. One parameter: the rate of subsitution (alpha).
    2. *Kimura two parameter (**K2P**) model*
        1. Transitions and transversions happen at different rates.
        2. All nucleotides occur at the same frequency.
        3. Two parameters: transition rate (alpha) and transversion rate (beta).

Emne 11: PAM og BLOSUM matricer

Når man konstruerer substitutionmatricer for proteiner (PAM og BLOSUM matricer), så beregnes hver score ud fra information i et sæt alignments.

Udsagn og/eller spørgsmål:

1. Størrelsen på hver indgang i en PAM1 scoringsmatrix repræsenterer, hvor ofte vi forventer at se en aminosyre blive erstattet med en anden, når man tager frekvensen af forskellige aminosyrer i betragtning. (True/False) True
2. PAM80 og BLOSUM80 repræsenterer omtrent lige store evolutionære afstande. (True/False) False
3. PAM160 og BLOSUM60 repræsenterer omtrent lige store evolutionære afstande. (True/False) True
4. PAM120 er konstrueret ved ekstrapolation fra en PAM1 matrix. (True/False) True
5. Jo hyppigere en aminosyre er, jo større score vil substitioner til denne aminosyre have i en PAM matrix. (True/False) False

   Scoren er beregnet ved s= p(a,b)/q(a)*q(b) betyder det ikke netop at en højere hyppighed (krakteriseret ved frekvensen q)

Emne 12: UPGMA

Tabellen nedenfor viser de parvise afstande mellem fire sekvenser: A, B, C og D.

	A	B	C
A			
B	12		
C	30	30	
D	18	18	30

De fire sekvensers indbyrdes afstande kan repræsentres som et træ ved hjælp af clustering algoritmen UPGMA. Det konstruerede træ har ingen rod.

Udsagn og/eller spørgsmål:

1. UPGMA vil begynde med at clustre A og B. (True/False) True
2. UPGMA vil slutte med at sammensmelte et cluster af A og B med et cluster af C og D. (True/False) False
3. Den korteste gren i det konstrurerede træ har længde 3. (True/False) True
4. Den længste gren i det konstrurerede træ har længde 30. (True/False) False
5. Hvis vi byggede et træ kun med sekvenserne A, B og C, så ville vi kunne bruge sekvens D som outgroup på dette træ. (True/False) False
6. Givet de parvise afstande i tabellen vil UGPMA og Neighor-Joining konstruere træer med de samme grenlængder. (True/False) True, da tallene er de samme (ergo gennemsnit og mindsteværdi for clustre er ens og bliver det samme for nye clustre)

Emne 13: Fylogeni

Metoder til konstruktion af fylogenier kan opdeles i karakterbaserede metoder, og metoder der clustrer sekvenser baseret på forudberegnede afstande mellem par af sekvenser.

Udsagn og/eller spørgsmål:

1. Den konstruerede fylogeni afhænger af den benyttede model for DNA evolution. (True/False) True
2. PhyML er et program til konstruktion af en fylogeni. (True/False) True
3. Neighbor-joining algoritmen tager højde for at mutationsraten ikke er konstant. (True/False) True
4. UPGMA algoritmen tager højde for at mutationsraten ikke er konstant. (True/False) False

Emne 14: Felsensteins algoritme

Felsensteins algoritme benyttes i karakterbaserede metoder til konstruktion af fylogeni.

Udsagn og/eller spørgsmål:

1. Felsenteins algoritme finder det bedste træ givet et sæt sekvenser. (True/False) False, (vi tænker at det siger hvor godt det er, men ikke finder det bedste. Holdet siger dog TRUE)
2. Felsensteins algoritme beregner sandsynligheden for et givet træ med en given base på hvert blad. (True/False) True
3. Felsensteins algoritme antager uafhængighed mellem mellem alignment kolonner. (True/False) False
4. Den sandsynlighed, Felsensteins algoritme producerer, afhænger af hvilken model for molekylær evolution der benyttes. (True/False) False
5. Felsensteins algoritme benytter rekursion. (True/False) True

Emne 15: Fil og data formater

Bioinformatisk data findes i mange former og formater. Omformatering af et format til et andet er i nogen tilfælde ikke muligt. I tilfælde hvor det er muligt kan det indebære et tab af information fordi et format indeholder en type information som et andet format ikke indeholder.

Udsagn og/eller spørgsmål:

1. En Fasta fil kan formateres til en Fastq fil. (True/False) False
2. En Fastq fil kan formateres til en Fasta fil. (True/False) True
3. En Fasta fil kan formateres til en Genbank fil uden tab af information. (True/False) True (måske lidt False
4. Data i Newick format repræsenterer en RNA sekundær struktur. (True/False) False

Emne 16: Aflæsning af genforudsigelser

Du har en sekvens fra E.coli og vil forudsige, hvilken gener denne sekvens koder for. Til dette har du brugt Easygene og fået følgende output.

```
##gff-version 2
##source-version easygene-1.2b
##date 2019-12-05
##Type DNA
# model:  ECO02 Escherichia coli O157:H7
# seqname               model   feature start   end        score       +/-     ?       startc      odds
# ------------------------------------------------------------------------------------------------------------------------
NC_017634.1_4000-10000  ECO02   CDS     32      1027    1.42992e-27     +       0       #TTG    132.526
NC_017634.1_4000-10000  ECO02   CDS     4142    5095    6.35347e-62     +       0       #ATG    211.565
NC_017634.1_4000-10000  ECO02   CDS     5210    5797    1.00958e-12     +       0       #ATG    69.1498
NC_017634.1_4000-10000  ECO02   CDS     1587    2363    3.46265e-15     -       0       #ATG    83.5159
NC_017634.1_4000-10000  ECO02   CDS     2433    3863    4.44842e-24     -       0       #ATG    131.079
# ------------------------------------------------------------------------------------------------------------------------
```

Udsagn og/eller spørgsmål:

1. Hvor mange gener er der fundet? (talværdi) 5
2. Hvor mange af de fundne gener er på forward strand? (talværdi) 3
3. På hvilken position starter det første gen? (talværdi) 32
4. To af de forudsagte gener har et alternative startkodons. (True/False) False
5. Hvor mange aminosyrer koder det længste gen for? (talværdi) 477 (den sidste)
   476, fordi stopkodon ikke bliver aa
6. Easygene er baseret på en skjult Markov model. (True/False) True

Emne 17: Generelt om skjulte Markov modeller

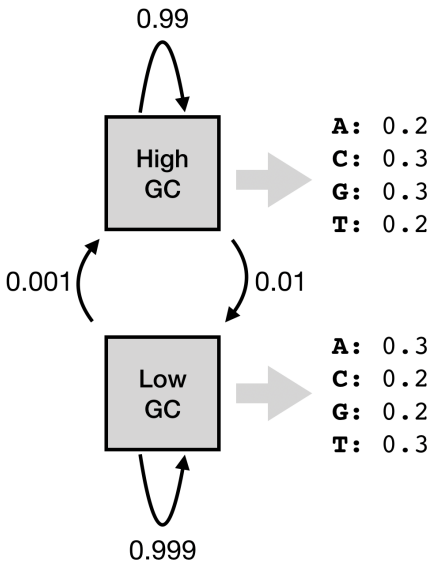
Skjulte Markov modeller (hidden Markov models, eller HMM) er en klasse af matematiske modeller, der ofte benyttes i bioinformatiske sammenhænge.

Udsagn og/eller spørgsmål:

1. En skjult Markov models emissionssandsynligheder angiver, hvor sandsynligt det er at gå fra en skjult tilstand (state) til en anden. (True/False) False
2. Skjulte Markov modeller benyttes blandt andet til at kategorisere delsekvenser i en større sekvens. (True/False) True
3. Hvis man bruger en skjult Markov model til at lave en forudsigelse (prediction) på en sekvens, så vil der være en forudsagt skjult tilstand for hver position i sekvensen. (True/False) True
4. En skjult Markov models parametre skal estimeres, før modellen kan anvendes til forudsigelse. (True/False) True
5. Forward algoritmen beregner de optimale transitions- og emmissions-sandsynligheder. (True/False) False (men mange siger True)
6. Viterbi algoritmen identificerer den mest sandsynlige sti gennem den skjulte Markov model. (True/False) True
7. Posterior decoding identificerer en serie af skjulte tilstande, der repræsenterer en sti gennem den skjulte Markov model. (True/False) False

Emne 18: En simpel skjult Markov model

Herunder ses et eksempel på en simpel skjult Markov model, der kan kategorisere en genomisk sekvens i områder med henholdsvis højt og lavt GC-indhold.

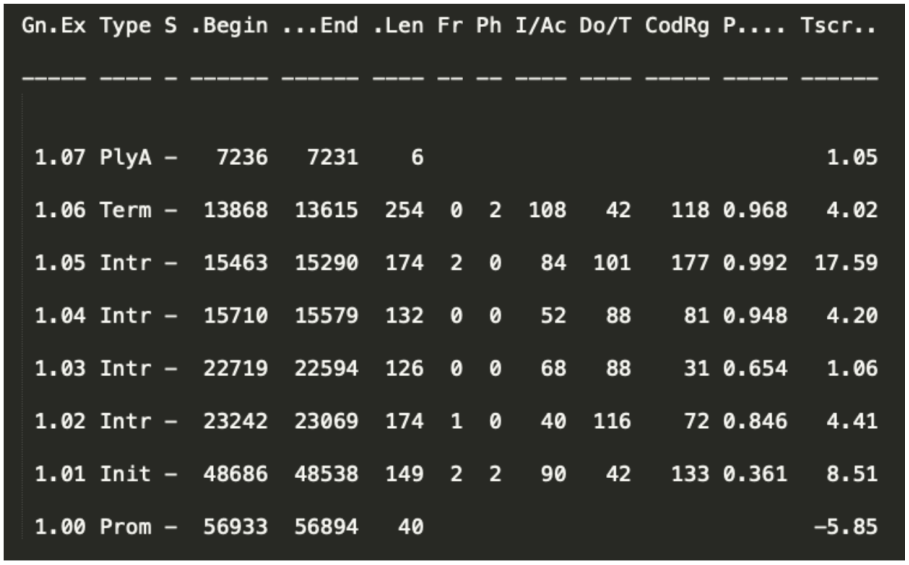


Udsagn og/eller spørgsmål:

1. Hvad er transitionssandsynligheden fra "High GC" til "Low GC"? (talværdi) 0.01
2. Modellens parametre angiver, at regioner med højt GC-indhold generelt er længere end regioner med lavt GC-indhold. (True/False) False
3. Hvad er emmisionssandsynligheden for basen C i en region med lavt GC-indhold? (talværdi) 0.2
4. Hvad er den forventede længde (i antal baser) af regioner med højt GC-indhold givet modellens parametre? (talværdi) 100 (1 for den første, 99 for dens gentagelser)

Emne 19: Genforudsigelse med GenScan

GenScan er et program, der benyttes til at forudsige gener i blandt andet det menneskelige genom. Nedenfor kan du se et screenshot med output fra GenScan.



Gn.Ex	Type	S	.Begin	...End	.Len	Fr	Ph	I/Ac	Do/T	CodRg	P....	Tscr..
1.07	PlyA	-	7236	7231	6							1.05
1.06	Term	-	13868	13615	254	0	2	108	42	118	0.968	4.02
1.05	Intr	-	15463	15290	174	2	0	84	101	177	0.992	17.59
1.04	Intr	-	15710	15579	132	0	0	52	88	81	0.948	4.20
1.03	Intr	-	22719	22594	126	0	0	68	88	31	0.654	1.06
1.02	Intr	-	23242	23069	174	1	0	40	116	72	0.846	4.41
1.01	Init	-	48686	48538	149	2	2	90	42	133	0.361	8.51
1.00	Prom	-	56933	56894	40							-5.85

Udsagn og/eller spørgsmål:

1. Genet som GenScan har fundet ligger på forward strand. (True/False) False
2. Hvor mange exons har det gen, der er fundet? (talværdi) 6
3. Hvad er længden af den første exon i det gen, der er fundet? (talværdi) 149
4. Hvad er positionen for den første base i den første exon? (talværdi) 48686
5. Det forudsagte gen har en poly-A hale. (True/False) True

Emne 20: Chou-Fasman

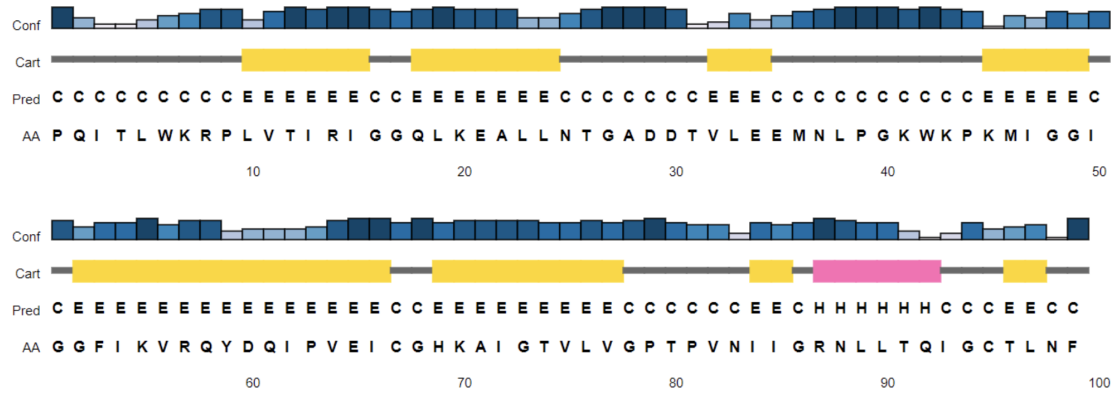
Der er mange forskellige måder at forudsige protein sekundær struktur. En af de mest simple metoder er Chou-Fasman algoritmen.

Udsagn og/eller spørgsmål:

1. Chou-Fasman algoritmen udnytter, at phi of psi vinklerne i peptid kæden er anderledes i alpha-helices end i beta-strands. (True/False) False
2. Chou-Fasman algoritmen udnytter, at forskellige aminosyrer optræder med forskellige hyppigheder i alpha-helices og beta-strands. (True/False) True
3. Chou-Fasman algoritmen udnytter, at fordelingen af afstande mellem C-atomer i peptidkæden er anderledes i alpha-helices end i beta-strands. (True/False) False
4. Chou-Fasman algoritmen udnytter, at forskellige aminosyrer optræder med forskellige hyppigheder i hairpin turns mellem beta-strands. (True/False) True

Emne 21: Protein structure

Nedenfor ses et output fra PSIRED, hvor strukturen af et protein er blevet forudsagt ud fra proteinets sekvens.



Gul: beta
Pink: alpha
Grå: coils

Udsagn og/eller spørgsmål:

1. PSIRED er baseret på en skjult Markov model. (True/False) False, er baseret på neuralt netværk
2. PSIRED forudsiger proteinets tertiære struktur. (True/False) False, den sekundære
3. Hvor mange aminosyrer indgår i den længste af de forudsagte beta-strenge? (talværdi) 15
4. Hvor mange strækninger (på en eller flere aminosyrers længde) forusiges til at være random coils? (talværdi) 10