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 i `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, some of the problems involve DNA sequences. In those problems we will assume that DNA sequences can only contain the characters A, T, G, C, a, t, g, and c.

Hint: In some of the problems below you will be asked to produce sorted lists. If you have a list, you can make a new sorted list using the `sorted` built-in function like this:

```
sorted_list = sorted(unsorted_list)
```

The actual assignment starts below.

Happy coding.

Problem 1

A list of lists represents a matrix. Here is an example:

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

If you write it like this, you can see that the matrix it represents has three rows and two columns:

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

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

1. A list of lists, which represents a matrix.

The function must return:

- An integer, which represents the number of rows in the matrix.

Example usage: If the function is called like this:

```
nr_rows_in_matrix([[1, 2], [1, 2], [1, 2]])
```

then it should return:

```
3
```

Problem 2

Counting stuff is always useful.

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

1. An iterable. I.e. a Python object that can be iterated over in a for loop. E.g. a string, a list or a dictionary.

The function must return:

- A dictionary in which keys are the different values in the iterable. The value associated with each key should be the number of times that the key appears in the iterable.

Example usage: If the function is called like this:

```
count_stuff('banana')
```

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

```
{'b': 1, 'a': 3, 'n': 2}
```

Problem 3

If you have a dictionary with keys and values, you may need to create a new dictionary where the keys and values are swapped so keys become values and values become keys. E.g. if you have a dictionary that lets you look up a persons name using a CPR number, you can convert that to a dictionary that lets you look up a persons CPR number using his/her name. You can assume that all the values in the dictionary are unique.

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

1. A dictionary where all values are unique.

The function must return:

- An dictionary where keys are the values in the dictionary argument and values are the keys in the dictionary argument.

Example usage: If the function is called like this:

```
invert_dictionary({'Preben': 'P', 'Mogens': 'M'})
```

then it should return:

```
{'P': 'Preben', 'M': 'Mogens'}
```

Problem 4

In a DNA string, uppercase and lowercase letters are used to represent different properties of bases.

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

1. A string, which is DNA sequence.

The function must return:

- An integer, which represents the number of uppercase bases.

Example usage: If the function is called like this:

```
count_upper_case_bases('agATgcGCgc')
```

then it should return:

```
4
```

Problem 5

You want to extract every second base in a DNA sequence (starting with the first one).

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

1. A string, which is DNA sequence.

The function must return:

- An a list of strings, that represent every second base from the string argument (starting with the first base)

Example usage: If the function is called like this:

```
every_second_base('ATATATAT')
```

then it should return:

```
['A', 'A', 'A', 'A']
```

Problem 6

Say you have a DNA string of a gene where uppercase bases represent the CDS and lowercase bases represent introns and UTRs (we ignore the existence of startcodons, reading frame etc.).

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

1. A string, which is a DNA sequence.

The function must return:

- A string of only the uppercase characters. The characters must appear in the same order as in the string argument.

Example usage: If the function is called like this:

```
splice_upper_case('agATgcGCgc')
```

then it should return:

```
'ATGC'
```

Problem 7

Say you have a small DNA sequence (what we have also called that a kmer), and you want to know if that kmer is found one or more times in a larger DNA sequence, and if it is, where in the larger sequence it is found.

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

1. A string, which is small DNA sequence (kmer).
2. A string, which is large DNA sequence.

The function must return:

- A *sorted* list of integers, which represent the indexes of the first base of each occurence of the kmer in the large string (argument two).

Example usage: If the function is called like this:

```
kmer_locations('TCT', 'AAATCTAAATCT')
```

then it should return:

```
[3, 9]
```

Problem 8

Say you have *two* small DNA sequences (what we have also called that a kmer). The two kmers have the same length. You want to know if they are found in a larger DNA string in a way so the two kmers overlap.

Write a function, `kmers_overlap`, which takes three arguments:

1. A string, which is small DNA sequence (kmer).
2. A string (with the same length as argument one), which is small DNA sequence (kmer) .
3. A string, which is large DNA sequence.

The function must return:

- An boolean. `True` if argument one and two are found overlapping in argument three and `False` otherwise.

Example usage: If the function is called like this:

```
kmers_overlap('TTAA', 'AATT', 'CCCCTTAATTCCCC')
```

then it should return:

```
True
```

Problem 9

You have a DNA sequence and want to find out how many different kmers (short sub-strings of length k) that there are in the sequence.

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

1. A string, which is DNA sequence.
2. An integer, which represents the length of kmers to search for.

The function must return:

- A *sorted* list of strings. Each string represents a kmer found in argument one. Each kmer must appear only once in the list.

Example usage: If the function is called like this:

```
unique_kmers('ATATATAT', 3)
```

then it should return:

```
['ATA', 'TAT']
```

Problem 10

You have two DNA sequences and you want to find out how many unique kmers (short sub-strings of length k) that are found in *both* sequences.

Write a function, `nr_unique_kmers_shared`, which takes three arguments:

1. A string, which is DNA sequence.
2. A string, which is DNA sequence.
3. An integer, which represent the length of kmers to search for.

The function must return:

- An integer, which represents the number of unique kmers that are shared by the two sequences (argument one and two).

Example usage: If the function is called like this:

```
nr_unique_kmers_shared('AAAAT', 'ATTTT', 2)
```

then it should return:

```
1
```

(because the two sequences only share `'AT'`).

Problem 11

You have a list of DNA sequences and you want to find out how many unique kmers that are shared by each pair of sequences in the list. You must create a matrix (a list of lists) that contains this information. If we call the resulting matrix for `matrix`, then `matrix[1][2]` and `matrix[2][1]` must both be the number of unique kmers shared by the DNA sequence with index `1` and `2` in argument one. `matrix[1][1]` must be the number of unique kmers that the DNA sequence with index `1` shares with itself.

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

1. A list of strings. Each string represents a DNA sequence.
2. An integer, which represent the length of kmers to search for.

The function must return:

- A list of lists of integers. Each integer represents the number of unique kmers shared between to sequences (see above).

Example usage: If the function is called like this:

```
kmer_sharing_matrix(['GAAAAT', 'GATTTT', 'AAAAAA'], 3)
```

then it should return:

```
[[3, 0, 1], [0, 3, 0], [1, 0, 1]]
```

or shown like a matrix below. Note that the matrix is symmetical around the diagonal:

```
[[3, 0, 1],
 [0, 3, 0],
 [1, 0, 1]]
```

Bioinformatikopgaver

Emne 1: Databaser

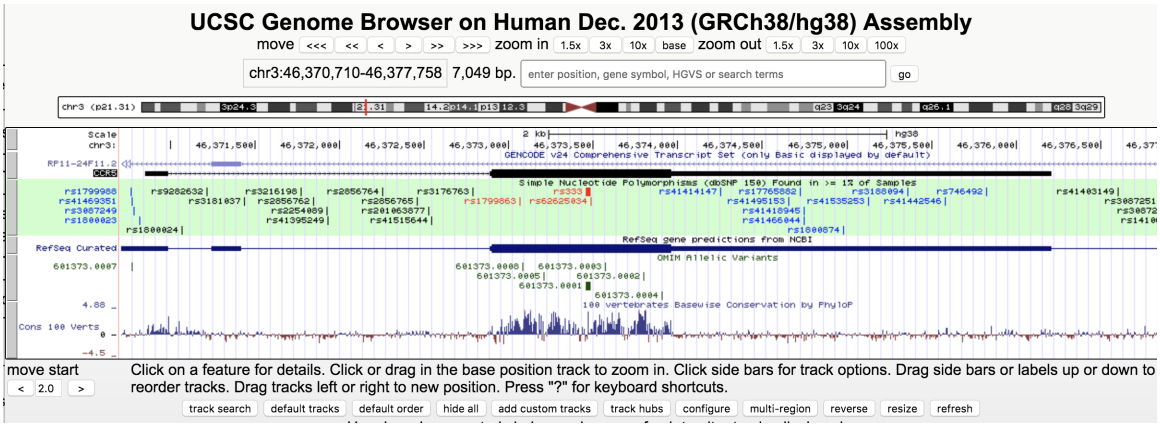
Online databaser giver adgang til et væld af forskellig information. Nogen indeholder rå sekvens information. Andre samler forskellige typer af information relevant for f.eks. gener, proteiner, sygdomme eller pathways.

Udsagn og/eller spørgsmål:

1. Genbank er en database, der indeholder alle offentligt tilgængelige nukleotid sekvenser. (True/False)
2. OMIM er en database, der sammenfatter ontologisk metainformation på tværs af arter. (True/False)
3. dbSNP er en database, der indholder information om enkelte base varianter i mennesker. (True/False)
4. Gene er en database, der sammenfatter metainformation om hvert enkelt gen. (True/False)

Emne 2: UCSC genome browser

Figuren nedenfor viser et screenshot fra UCSC genome browseren (du kan være nødt til at zoome ind for at se alle detaljer):



CCR5-delta32 er en variant af CCR5 genet med en deletion af 32 fortløbende nukleotider. Varianten optræder i databaser om rs333.

Udsagn og/eller spørgsmål:

1. CCR5 genet ligger på kromosom 3. (True/False)
2. SNPen med id rs333 ligger i den protein-kodende region af genet. (True/False)
3. SNPen med id rs333 er associeret med en sygdom beskrevet i OMIM databasen. (True/False)
4. SNPen med id rs333 er en frame-shift mutation. (True/False)

Emne 3: Needleman-Wunsch og Smith-Waterman

Needleman-Wunsch og Smith-Waterman algoritmerne er algoritmer til henholdsvis globalt og lokalt alignment af to sekvenser.

Udsagn og/eller spørgsmål:

1. Needleman-Wunch algoritmen identificerer mindst ét parvist alignemnt, der er optimalt givet en scoringsmatrix og en gap score. (True/False)
2. Der kan højest være ét optimalt lokalt parvist alignment for en given scoringsmatrix og en gap score. (True/False)
3. Tiden, det kræver at beregne et globalt alignment ved hjælp af Needleman-Wunsch algoritmen, er proportional med produktet af de to sekvensers længde. (True/False)
4. Smith-Waterman algoritmen vil aldrig producere et globalt alignment. (True/False)

Emne 4: Globalt alignment af to korte sekvenser

Nedenfor ser du to korte DNA sekvenser:

```
sekvens 1: TGGTA
sekvens 2: TTTGT
```

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

Udsagn og/eller spørgsmål:

1. Der findes et optimalt alignment hvor det sidste A i sekvens 1 aligner til det sidste T i sekvens 2. (True/False)
2. Alle optimale alignments har to gaps. (True/False)
3. Der er tre optimale alignments. (True/False)
4. Alle optimale alignments af de to sekvenser har det samme antal matches (alignmentkolonner hvor de to baser er ens). (True/False)

Emne 5: Parvist alignment ved hjælp af EBIs web tools.

Vi aligner dele af FOXP2 proteinet fra henholdsvis et menneske med en genfejl og fra en rotte.

```
Human FOXP2
FVRVENVKGAVWTVDEVEYQKRRSQKITGSPTLVKNIPTSLGYGAALNASLQAALAESSLPLLSNPGLIN
NASSGLLQAVHEDLNGSLDHIDSNGNSSPGCSHIHSIHVKEEPVIAEDEDCPMSLVTTANHSPELED

Rat FOXP2
SPTLVKNIPTSLGYGAALNASLQAALAESSLPLLSNPGLINNASSGLLQAVHEDLNGSLDHIDSNGNSSP
GCSPQPHIHSIHVKEEPVIAEDEDCPMSLVTTANHSPELEDDREI
```

Nedenfor ser du to screen shots med output fra to parvise alignments foretaget ved hjælp af EBIs webservice.

A

```
#=======================================
#
# Aligned_sequences: 2
# 1: Human
# 2: Rat
# Matrix: EBLOSUM62
# Gap_penalty: 10.0
# Extend_penalty: 0.5
#
# Length: 111
# Identity:     108/111 (97.3%)
# Similarity:   108/111 (97.3%)
# Gaps:           3/111 ( 2.7%)
# Score: 543.0
#
#
#=======================================

Human              30 SPTLVKNIPTSLGYGAALNASLQAALAESSLPLLSNPGLINNASSGLLQA     79
                      |||||||||||||||||||||||||||||||||||||||||||||||||||
Rat                 1 SPTLVKNIPTSLGYGAALNASLQAALAESSLPLLSNPGLINNASSGLLQA     50

Human              80 VHEDLNGSLDHIDSNGNSSPGCS---HIHSIHVKEEPVIAEDEDCPMSLV    126
                      |||||||||||||||||||||||   |||||||||||||||||||||||||
Rat                51 VHEDLNGSLDHIDSNGNSSPGCSPQPHIHSIHVKEEPVIAEDEDCPMSLV    100

Human             127 TTANHSPELED    137
                      ||||||||||
Rat               101 TTANHSPELED    111
```

B

```
#=======================================
#
# Aligned_sequences: 2
# 1: Human
# 2: Rat
# Matrix: EBLOSUM62
# Gap_penalty: 10.0
# Extend_penalty: 0.5
#
# Length: 144
# Identity:     108/144 (75.0%)
# Similarity:   108/144 (75.0%)
# Gaps:          36/144 (25.0%)
# Score: 543.0
#
#
#=======================================

Human              1 FVRVENVKGAVWTVDEVEYQKRRSQKITGSPTLVKNIPTSLGYGAALNAS     50
                                                 ||||||||||||||||||||
Rat                1 ------------------------------SPTLVKNIPTSLGYGAALNAS     21

Human             51 LQAALAESSLPLLSNPGLINNASSGLLQAVHEDLNGSLDHIDSNGNSSPG    100
                     ||||||||||||||||||||||||||||||||||||||||||||||||||||
Rat               22 LQAALAESSLPLLSNPGLINNASSGLLQAVHEDLNGSLDHIDSNGNSSPG     71

Human            101 CS---HIHSIHVKEEPVIAEDEDCPMSLVTTANHSPELED----    137
                      ||   ||||||||||||||||||||||||||||||||||||
Rat               72 CSPQPHIHSIHVKEEPVIAEDEDCPMSLVTTANHSPELEDDREI    115
```

Udsagn og/eller spørgsmål:

1. Alignment B er et lokalt alignment. (True/False)
2. Alignment A er et globalt alignment. (True/False)
3. Alignmnet B benytter liniær gap scoring. (True/False)
4. Det gap på tre aminosyrer, som man ser i alignment B, vil bidrage med en score på -11. (True/False)

Emne 6: 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)
2. Clustal Omega vil altid producere et optimalt alignment givet en scoringsmatrix og gap score. (True/False)
3. Jo flere sekvenser man aligner, jo bedre vil man kunne identificere homologi mellem fjernt beslægtede sekvenser. (True/False)
4. Forskellige heuristikker til multipelt alignment vil producere det samme alignment, hvis der benyttes den samme scoringsmatrix og gap score. (True/False)

Emne 7: 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 er garanteret at finde det bedste alignment mellem en sekvens i databasen og den sekvens man søger med. (True/False)
2. Den score, der angives for hver identificeret databasesekvens, repræsenterer en alignment score for et lokalt alignment af en sekvens i databasen og den sekvens man søger med. (True/False)
3. E-værdien angiver sandsynligheden for at finde et match i databasen med samme eller lavere score ved rent tilfælde. (True/False)
4. E-værdien forventes at være mindre, hvis man spøger mod en lille database, end hvis man søger mod en stor database. (True/False)

Emne 8: Modeller for DNA evolution

Jukes-Cantor modellen og Kimura 2-parameter modellen 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)
2. Kimura modellen korrigerer for, at transitioner og transversioner er ikke er lige hyppige. (True/False)
3. Kimura modellen tillader, at G og C baser har en højere hyppighed end A og T baser. (True/False)
4. Hvis 50% af baserne i to alignede sekvenser er forskellige, så vil den Jukes-Cantor-korrigerede afstand være uendeligt stor. (True/False)

Emne 9: PAM og BLOSUM matricer

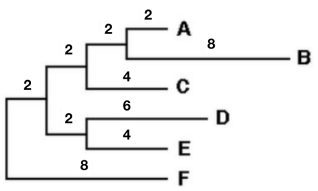
Når man konstruerer substitutionmatricer for proteiner (PAM og BLOSUM matricer), så beregnes hver score ud fra information i et sæt alignments.

Udsagn og/eller spørgsmål:

1. Størrelsen på hver indgang i en PAM1 scoringsmatrix repræsenterer, hvor ofte vi ser en aminosyre i det alignment der benyttes til at beregne matrixen. (True/False)
2. PAM80 og BLOSUM80 repræsenterer omtrent lige store evolutionære afstande. (True/False)
3. PAM120 er konstrueret ved ekstrapolation fra en PAM1 matrix. (True/False)
4. Ved beregning af en BLOSUM62 substitutionmatrix er sekvenserne i de brugte alignments højest 62% identiske. (True/False)
5. Jo hyppigere en aminosyre er, jo større score vil substitioner til denne aminosyre have i en PAM matrix. (True/False)

Emne 10: Afstande på et træ

Nedenfor ser du et træ for seks sekvenser: A, B, C, D, E og F. Længden af hver gren er angivet.



Udsagn og/eller spørgsmål:

1. Sekvensafstanden mellem B og D er 20. (True/False)
2. Sevensafstanden mellem A og B er mindre end mellem A og C. (True/False)
3. Hvis man fjernede F, ville træets totale grenlængde være 30. (True/False)
4. Afstandene på træet indikerer, at substitutionsraten er mindre på den gren der leder til A, end den gren der leder til B. (True/False)

Emne 11: 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 B og C. (True/False)
2. UPGMA vil slutte med at sammensmelte ("joine") et cluster af A og B med et cluster af C og D. (True/False)
3. Hvad er længden på den korteste gren i træet? (talværdi)
4. Hvis vi byggede et træ med kun sekvenserne B, C og D, så ville vi kunne bruge sekvens A som outgroup på dette træ. (True/False)

Emne 12: Fylogeni

Der findes flere typer af metoder til konstruktion af fylogeni.

Udsagn og/eller spørgsmål:

1. PhyML konstruerer en fylogeni ud fra forudberegnede afstande mellem par af sekvenser. (True/False)
2. MrBayes er et program til konstruktion af en fylogeni. (True/False)
3. Neighbor-joining algoritmen tager højde for, at mutationsraten ikke er konstant. (True/False)
4. UPGMA algoritmen tager højde for, at mutationsraten ikke er konstant. (True/False)

Emne 13: Felsensteins algoritme

Felsensteins algoritme benyttes i programmer, der konstuerer en fylogeni.

Udsagn og/eller spørgsmål:

1. Felsensteins algoritme beregner sandsynligheden for et givet træ med en given base på hvert blad. (True/False)
2. Felsensteins algoritme tager højde for afhængighed mellem alignment kolonner. (True/False)
3. Den sandsynlighed Felsensteins algoritme producerer afhænger af, hvilken model for molekylær evolution der benyttes. (True/False)
4. Den sandsynlighed Felsensteins algoritme producerer afhænger af ligevægts-frekvensen af de fire baser. (True/False)

Emne 14: Genome assembly

Genome assembly betegner den bioinformatiske udfordring, det er at samle mange sekvens reads til et genom.

Udsagn og/eller spørgsmål:

1. Repeats i en sekvens kan samles hvis længden af reads er kortere end længden af det enkelte repeat. (True/False)
2. Genome contigs er samt sammen af flere genome scaffolds. (True/False)
3. N50 repræsenterer antallet af baser, der er dækket af mindst 50 reads. (True/False)
4. Hvad er N50 følgende hvis længderne på contigs er 4, 1, 5, 10, 15, 25, 40, 10? (talværdi)

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

Emne 16: Sammenligning af genforudsigelser

Ved hjælp af EasyGene er der blevet lavet en forudsigelse af potentielle "open reading frames" (ORFs) i et stykke DNA fra Bacillus subtilis. EasyGene har givet dette output.

```
##gff-version 2
##source-version easygene-1.2b
##date 2017-12-14
##Type DNA
# model:  BS03 Bacillus subtilis
# seqname   model feature start end   score       +/-  ?  startc  odds
# ------------------------------------------------------------------------
AB010576.1  BS03  CDS     64    324   0.0271875   +    0  #ATG    20.1861
AB010576.1  BS03  CDS     1129  1269  0.0190622   +    0  #ATG    15.7102
AB010576.1  BS03  CDS     1370  2314  2.13273e-12 +    0  #ATG    74.7815
AB010576.1  BS03  CDS     2327  2491  0.0167943   +    0  #ATG    17.2951
# ------------------------------------------------------------------------
```

Udsagn og/eller spørgsmål:

1. Hvor mange ORFs er der fundet? (talværdi)
2. Hvor mange af de fundne ORFs er på forward strand? (talværdi)
3. På hvilken position starter det første ORF? (talværdi)
4. Der er forudsagt alternative startcodons. (True/False)
5. Hvor mange aminosyrer koder det længste ORF for? (talværdi)

Emne 17: Generelt om skjulte Markov modeller

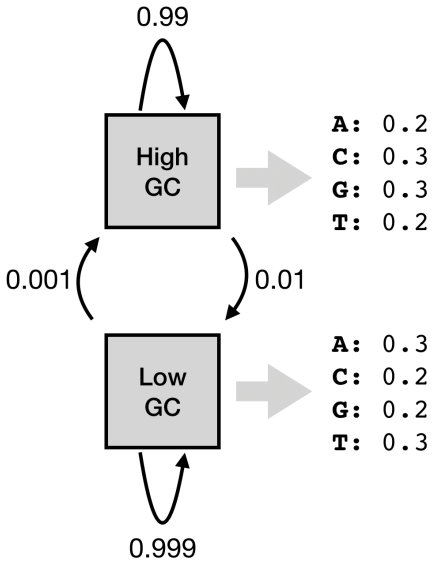
Skjulte Markov modeller (hidden Markov models, eller HMM) er en klasse af matematiske modeller, der ofte benyttes i bioinformatiske sammenhænge.

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)
2. Skjulte Markov modeller benyttes blandt andet til at kategorisere delsekvenser i en større sekvens. (True/False)
3. Hvis man bruger en skjult Markov model til at lave en forudsigelse (prediction) på en sekvens, så vil der være en forudsagt en skjult tilstand for hver position i sekvensen. (True/False)
4. En skjult Markov models parametre skal estimeres, før modellen kan anvendes til forudsigelse. (True/False)
5. Forward algoritmen beregner de optimale transitions- og emmissions-sandsynligheder. (True/False)
6. Viterbi algoritmen identificerer den mest sandsynlige sti gennem den skjulte Markov model. (True/False)
7. Posterior decoding identificerer en serie af skjulte tilstande, der repræsenterer en sti gennem en skjult Markov model. (True/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 "Low GC" til "High GC"? (talværdi)
2. Modellens parametre angiver, at regioner med højt GC-indhold generelt er kortere end regioner med lavt GC-indhold. (True/False)
3. Hvad er emmisionssandsynligheden for basen A i en region med lavt GC-indhold? (talværdi)
4. Hvad er den forventede længde (i antal baser) af regioner med højt GC-indhold givet modellens parametre? (talværdi)

Emne 19: Neurale netværk

Neurale netværk er en klasse af matematiske modeller, der benyttes i bioinformatik.

Udsagn og/eller spørgsmål:

1. Neurale netværk er repræsenteret ved en graf med et sæt forbundne knuder, der hver representer en "neuron". (True/False)
2. Neurale netværk kan anvendes til klassifikation af sekvenser. (True/False)
3. Et neuralt netvært har ingen parametre, som skal estimeres før det anvendes til forudsigelse. (True/False)
4. Neurale netværk kan anvendes til forudsigelse af signalpeptider. (True/False)