Bioinformatics Module

Sample Examination paper with answers

David Gilbert

Answer 3 questions.

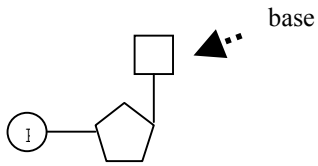
Appendices A,B,C are attached, and are for use in the following questions:

Appendix	Questions
A	1, 2
B	1
C	4

1

1.1 Sketch the composition of a nucleotide, and give the alphabets for sequences of DNA and RNA. [10]

DNA = deoxyribonucleic acid
RNA = ribonucleic acid
Biological macromolecules built as long linear chains of chemical components - ***nucleotides*** = ***sugar+phosphate+base***



Bases: adenine (A), cytosine (C), guanine (G), thymine (T), uracil (U)
4 letter alphabet:
DNA = A C G T (adenine cytosine guanine thymine)
RNA = **A C G U** (adenine cytosine guanine uracil)

1.2 Compute the complementary strand for the following sequence [10]

```
atggtgcacctgactcctgaggagaagtctgcggttactgccctgtggtag
```

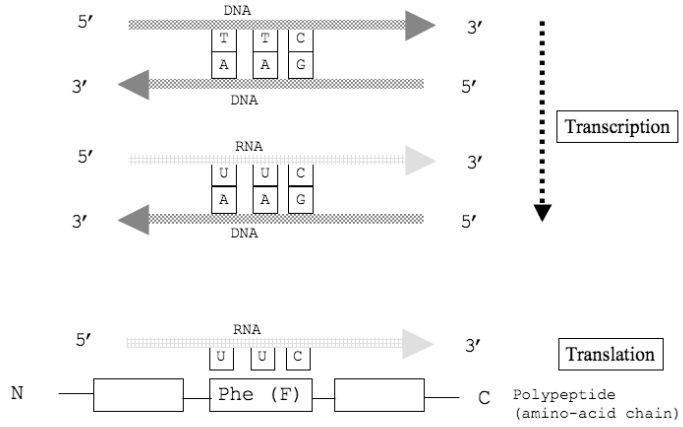
by complements a-t and c-g
add 5' and 3' ends

```
5': atggtgcacctgactcctgaggagaagtctgcggttactgccctgtggtag 3'
3': ta.....................................................tc 5'
```

1.3 What is meant by the terms *transcription* and *translation* and *frame shift errors* in the context of protein synthesis. Illustrate your answer by considering the process of synthesising an amino-acid chain from the DNA sequence above. [20]

Transcription:
The process of copying DNA to RNA
Translation: synthesis of protein from mRNA
Ribosomes are the machines that synthesise proteins from mRNA
reading frame = groupings of codons
translation:
starts with *start codon*
ends with *stop codon*
Triplet code, hence difference between DNA base
Substitution: (hence 1 amino-acid changes)
Insertion / Deletion: "frame shift" (all subsequent amino-acids change)
NB, Indels can be in multiples of 3, and hence...
Also
"Silent mutation" - DNA changes but amino-acid doesn't change
"Nonsense mutation" - a single DNA base substitution resulting in a stop codon.

1.3 Transcription & translation




```
atg gtg cac ctg act cct gag gag aag tct gcg gtt act gcc ctg tgg tag
met val his leu thr pro glu glu lys ser ala val thr ala leu trp stop
```

by complements a-t and c-g
add 5' and 3' ends

```
5': atggtgcacctgactcctgaggagaagtctgcggttactgccctgtggtag 3'
3': ta.....................................................tc 5'
```

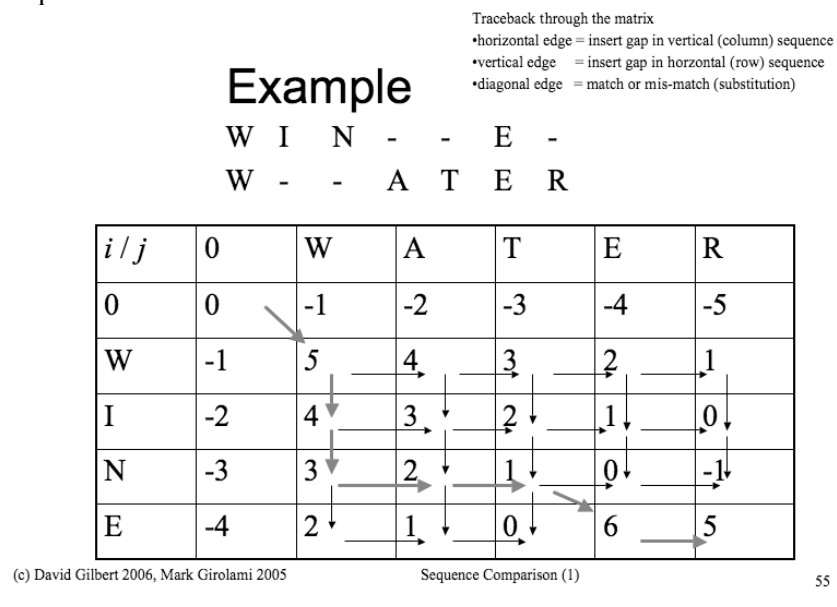
1.4 Define the recurrence relation for the *Edit distance* between two strings and give the order of complexity for a naïve algorithm based on this relation. [10]

$d(i,j) = \min(d(i-1,j)+g,\ d(i,j-1)+g,\ d(i-1,j-1) + t(v_i,w_j))$
$t(v_i,w_j) = m$ if $v_i=w_j$, $t(v_i,w_j) = n$ if $v_i \neq w_j$
g - gap penalty (e.g. 2)
m - match score (e.g. -1)
n - mismatch score (e.g.+1)

Of the order of 2^{2N} *calls to scoring routine*

●1.5 Compute the dynamic programming table, alignments and associated sequence identities for the two strings `WATER` and `WINE`, where Symbol mis-match -5; gap insertion -1; match 5 [50]

As per lectures:



2.1 Define: language, string, symbol, alphabet [10]
A *language* is a set of *strings*
A string is a finite sequence of *symbols*
Symbols are taken from a finite *alphabet*

2.2 Explain the meaning of the symbols in the PROSITE pattern, referring to Appendix A
`[AC]-x-V-x(4)-{ED}`
`and show how it` matches the sequence [30]

DEHSDVLPVLDVCSLKHVAYVFQALIYWIKAMNQQTTLDT

[AC] alternation
x wild card (1 char)
V valine
x(4) 4 wild cards
{ED} not E or D

DEHSDVLPVLDVCSLKHVAYVFQALIYWIKAMNQQTTLDT

match: AYVFQALI where
[AC]=A, x=Y, V=V, x(4)=FQAL, {ED}=I

2.3 Distinguish between *pattern-driven* and *sequence-driven* pattern discovery. [20]
- **Pattern driven:** enumerate all (or some) patterns up to certain complexity (length), for each calculate the score, and report the best
- **Sequence driven:** look for patterns by aligning the given sequences

2.4 Give an algorithm for a *sequence-driven* discovery algorithm for patterns over amino-acid sequences, where a pattern is defined as one or more (sub)sequences which match an example and illustrate the working of your algorithm on the following sequences: [40]
s_1 = MERIAVLALEDKYGYKV
s_2 = MLEDKYGYLEDERIAVLAKV
s_3 = VKYGYKDEMERIAVLAL

Group similar sequences together (e.g., in pairs);
For each group find a common pattern (e.g., by dynamic programming);
Group similar patterns together and repeat the previous step until there is only one group left

s_1 = MERIAVLALEDKYGYKV
s_2 = MLEDKYGYLEDERIAVLAKV
s_3 = VKYGYKDEMERIAVLAL

e.g. Pattern(s1,s2) = {ERIAVLA , EDKYGY}
Pattern(Pattern(s1,s2),s3) = {ERIAVLA}
assuming that the longest patterns are kept, we reject KYGY as a member of Pattern(Pattern(s1,s2),s3)

3.
3.1 Define, and distinguish between: phylum, taxon and species [20]
Phylum (phyla pl): A primary division of a kingdom, as of the animal kingdom, ranking next above a class in size.
Taxon: any named group of organisms
Species: taxonomic group whose members can interbreed (but more or less able to…)

3.2 Give an algorithm for the Neighbour-Joining method, and show how it can be applied to the data in Table 1 to produce a tree. [40]

Table 1

	1	2	3	4
1	0	0.3	0.5	0.6
2		0	0.6	0.5
3			0	0.9
4				0

Algorithm:

Initialisation:

•Define T to be the set of leaf nodes, one for each given sequence and put L=T

Iterate

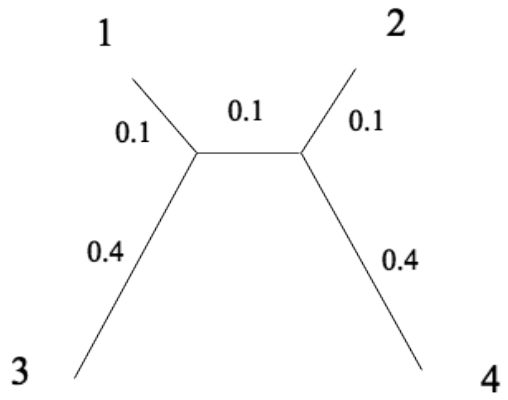
- Pick a pair i,j in L for which D(i,j) is minimal
- Define a new node k and set d(k,m)=1/2*(d(i,m)+d(j,m)+d(i,j) for all m in L
- Add k to T with edges of lengths d(i,k)=1/2*(d(i,j)+ri-rj), d(j,k)=d(i,j)-d(j,k), joining k to i and j respectively

$$D(i,j) = d(i,j) - \left(r_i + r_j\right)$$
$$where$$
$$r_i = \frac{1}{|L|-2}\sum_{k\in L} d(i,k)$$

- Remove i and j from L and add k

Termination

- When L consists of 2 leaves i and j, add the remaining edge between i and j with length d(i,j)



3.3 [40]

(i) What assumption is made by the Neighbour-Joining algorithm in terms of a *molecular clock,* and what test can be applied to tree data to determine whether Neighbour-joining tree reconstruction is likely to be correct?

Additivity must hold – edge lengths are additive if the distance between any pair of leaves is the sum of the lengths of the edges of the paths connecting them.

(ii) Show how a rooted tree might be derived by Neighbour-Joining for the data in Table 1 and indicate what additional information would be required to achieve this

(iii) Name another tree reconstruction algorithm which can be used to reconstruct a tree, and give any assumptions that must be made about the data for correct tree reconstruction.
UPGMA [best to explain this somewhat]
Assumptions:
Molecular clock with constant rate
Additivity of edge lengths

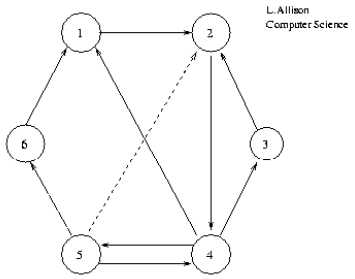
4

4.1. Define *directed graph*, *elementary path*, *elementary circuit* [15]
Graph = (V,A) V = set of vertices {a,b,c…} A = set of arcs {a-b | a,b ∈ V}
A graph is either directed or not
If directed then A – arcs are in form arcs {a→b | a,b ∈ V}

Path = sequence of arcs
($x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$, $x_3 \rightarrow x_4$, … $x_{k-1} \rightarrow x_k$)
Also can write [x_1,x_2 ,x_3,…, x_k]
Elementary if does not use same vertex twice
Circuit = path [x_1,x_2 ,x_3,…, x_k] where initial vertex x_1= terminal vertex x_k
Elementary circuit if all vertices distinct apart from x_1= x_k

4.2 Show how a graph can be represented as an adjacency matrix. [15]
Graphs by Adjacency Matrices.

A graph G= can be represented by a |V|*|V| adjacency matrix A. If G is directed, Aij=true if and only if <vi,vj> is in E. There are at most |V|2 edges in E.
Directed Graph



Directed Graph

	1	2	3	4	5	6
1		T				
2				T		
3		T				
4	T		T		T	
5		T		T		T
6	T					

Adjacency Matrix of Directed Graph.

4.3 Give an algorithm for breadth-first search in a graph and indicate the advantages and disadvantages of this method compared to depth-first search. [30]

Graph (directed or not); using a queue where the vertices found are stored.
bfs (Graph G)
{
all vertices of G are first painted white
the graph root is painted gray and put in a queue
while the queue is not empty

```
        {
          a vertex u is removed from the queue
          for all white successors v of u
          {
             v is painted gray
                 v is added to the queue
          }
          u is painted black
        }
  }
```

Depth first vs Breadth first Search
Depth first and breadth first search both have some advantages. Which is best depends on properties of the problem you are solving. For tree search at least, depth first search tends to require less memory, as you only need to record nodes on the `current' path. If there are lots of solutions, but all at a comparable `depth' in the tree, then you may hit on a solution having only explored a very small part of the tree. On the other hand, that may not be the best solution. Also, depth first search may get stuck exploring long (and potentially infinite) blind alleys, when there is in fact a solution path of only one or two steps. (We could prevent this by setting a depth limit to our search, but then it wouldn't be exhaustive.). So depth first is good when there are many possible solutions, and you only want one (and you don't care which one). It may be less appropriate when there is only one solution, or if you want the shortest one.

Breadth first search may use more memory, but will not get stuck in blind alleys, and will always find the shortest path first (or at least, the path that involves the least number of steps). It may be more appropriate when exploring very large search spaces where there is an expected solution which takes a relatively small number of steps, or when you are interested in all the solutions (perhaps up to some depth limit).

4.4 Apply your algorithm to perform a breadth-first search of the signalling pathway shown in Appendix D, indicating your result by annotating the notes visited in order. *You should interpret a two-way arrow as pointing forwards from the current node that is being processed in the algorithm.* [40]

Answers taken as correct w.r.t. start node & algorithm