Bioinformatics Example Sheet 1

Biology: Background information

Although not required for the examination, it might make some sense to look up the following terms.

Exercise 1 [Chemistry terms] Lookup the following terms relating to Chemistry and Physics:

(a) Atom;

(b) Molecule;

(c) Organic molecules;

(d) Monomers;

(e) Polymers;

(f) Macromolecule;

(g) Acids;

(h) Amino acids;

(i) Nucleic acids;

(j) Proteins (and their relation to amino acids and macromolecules);

(k) Polypeptides.

Exercise 2 [Biology terms] Lookup the following terms relating to Biology:

(a) Nucleotides;

(b) DNA; What are its bases? What does it mean for its strands to be complementary?

(c) RNA; What are its bases?

(d) Genes;

(e) What does the Human Genome project do?

(f) Genetic code;

(g) Basic cell structure. What is stored in ribosomes?

(h) Gene expression.

Further Reading 1 [Isolating DNA] What is the length of a typical DNA strand? (You can also find on YouTube tutorials on how to extract DNA from strawberries or bananas.)

Exercise 3 [DNA functions]

(a) What is DNA transcription?

(b) What is DNA translation?

(c) What are exons/introns in a gene?

Measuring similarity using alignment algorithms

Exercise 4 [Dynamic Programming and the shortest/longest p]

(a) Explain the principle of *dynamic programming.*

(b) Explain how you can use dynamic programming to find the shortest (or longest) path in a *Directed Acyclic Graph (DAG).*

(c) Explain why one can view any (finite space) dynamic programming algorithm as a shortest/longest

path problem in a DAG.

(d) Are there any cases where the shortest/longest path interpretation might lead to (in some sense) worse solutions?

Exercise 5 [Sequence alignment]

(a) What is *sequence alignment*?

(b) How are sequence alignment algorithms used in Bioinformatics?

Exercise 6 [Longest Common Subsequence]

(a) Define the *longest common subsequence (LCS)* problem.

(b) Formulate the recurrence relation and explain how dynamic programming is used to solve it.

(c) Show the DP table for input sequences `BDCABA` and `ABCBDAB`.

(d) Describe how this problem can be interpreted as finding a longest path in a DAG.

(e) (optional) Implement the LCS algorithm (either bottom up or top-down). (You can test your implementation on **[LeetCode 1143]**)

(f) What is the time complexity of your implementation? How does it compare to that of the brute force approach?

(g) What is the space complexity of your implementation? Can you reduce this further?

(h) Explain how you can recover a longest common subsequence. Draw the corresponding table for the example above. What is the time complexity of your approach?

(i) Give a pair of sequences that have more than one LCSs.

(j) (optional ++) How can you count the number of LCSs?

(k) Attempt **[2015P9Q1 (a)]**.

Exercise 7 [Global/local alignment with scoring matrix]

(a) Attempt **[2007P12Q10 (a)]**.

(b) Explain the concept of a *scoring matrix*.

(c) Define the *global alignment* problem. Show how the scoring matrix determines the weights of the edges in the longest path interpretation of the problem.

(d) Define the *local alignment* problem. Show how the scoring matrix determines the weights of the edges in the longest path interpretation of the problem.

(e) Attempt **[2007P12Q10 (b)]**.

(f) Describe the different approaches for handling *gaps* in sequence alignments. How are gaps modelled in the longest path interpretation?

(g) Attempt **[2013P7Q3 (a),(b)]**.

Exercise 8 [Banded Dynamic Programming] Attempt **[2020P8Q2 (a)]**.

Exercise 9 [Multiple sequence alignment with DP]

(a) Describe a DP algorithm for finding an alignment between 3 sequences.

(b) Can you generalise this to k sequences? What is the time complexity of the algorithm?

Further Reading 2 [Multiple sequence alignment is NP-Complete] If you are interested, read the paper "On the complexity of multiple sequence alignment" by Wang and Jiang (e.g. here) for a proof that multiple sequence alignment is NP-Complete.

Exercise 10 [Four-Russians speedup]

(a) Attempt **[2007P12Q10 (d)]**.

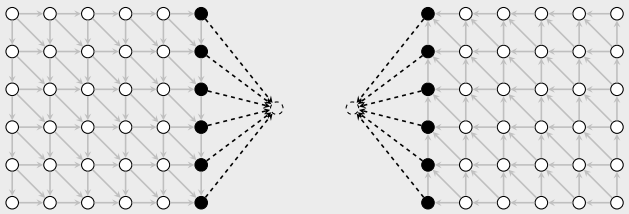
(b) How does the time complexity depend on the size of the alphabet? What does this imply for DNA sequences?

Further Reading 3 [LCS in $\mathcal{O}\left(n^{2-\epsilon}\right)$?] It is an open problem to find an algorithm that computes the LCS between two strings of length n in $\mathcal{O}\left(n^{2-\epsilon}\right)$ time for constant $\epsilon > 0$ (i.e. an asymptotically polynomial change instead of logarithmic). More recently, it has been shown that solving this problem in $\mathcal{O}\left(n^{2-\epsilon}\right)$ time, would imply faster algorithms for a collection of problems (see for example here).

Exercise 11 [Hirschberg's algorithm] Answer the following questions relating to *Hirschberg's algorithm*:

(a) How can the LCS problem be interpreted as a longest path problem in a grid?

(b) Let m and n be the lengths of the two strings. With the aid of the following diagram explain how you can compute the optimal in the column at $n/2$ by solving two instances of LCS for a string of length $n/2$ and a string of length m using linear memory.



(c) If you know the optimal path passes through $(n/2, x)$, how can you find the remaining path, by processing a matrix of total $mn/2$ size (and still using linear memory).

(d) Argue that the time complexity of the algorithm is $\mathcal{O}(mn)$ and the space complexity is $\mathcal{O}(\min(n, m))$. **Hint:** *Consider the sum* $mn + mn/2 + mn/4 + \ldots$.

Exercise 12 [Longest Common Substring] The *longest common substring* between two strings is defined as the longest (continuous) string s that appears in both strings. For example, the longest common substring of `abcarexample` and `simplexactcat` is $s =$ `mple`. Design an algorithm to retrieve the longest common substring. What is the time and space complexity of your algorithm?

Exercise 13 [Nussimov's algorithm]

- What are the three levels of RNA structure?
- What are *RNA foldings*? How are these used in Biology?
- Attempt **[2019P8Q2 (a)]**.
- Attempt **[2015P9Q1 (b)]**.

Exercise 14 [Longest Palindromic Subsequence] (optional) A palindrome is a string that reads the same way forwards and backwards, e.g. `abcba`. The *longest palindromic subsequence* of $a_1, \ldots, a_n$ is the longest subsequence that is a palindrome. For example, $s =$ `anotherexample` then LPS is `aerea`. Design an algorithm that efficiently solves this problem. (If you prefer you can attempt **[2013P1Q6]**)

Algorithms for building trees

If you want to remind yourselves of basic properties of trees, look at Exercise 1 here (and solutions).

Exercise 15 [Phylogeny]

(a) What does the word *phylogeny* mean?

(b) What are *phylogenetic trees*? How are they used in biology? Are there any limitations to these?

(c) Argue that: "Every simple tree with at least two nodes has at least one pair of neighbouring leaves".

Exercise 16 [Distance-based phylogeny]

(a) How can one obtain a distance matrix between species? In what time complexity?

(b) Define the *distance-based phylogeny problem.*

(c) What is an *additive matrix*?

(d) Show an example of a distance matrix that cannot be fit by a tree.

(e) Show an example of a distance matrix that can be fit by multiple trees? How can we make it unique?

(f) (*Four-point condition* +) Argue that there are three possible matchings between any 4 nodes in a complete graph. Argue that if a distance is additive, then for any four nodes two of the three matchings must have the same weight[a].

[a] This is actually an iff statement. You can use a constructive argument (e.g. the algorithm in Exercise 17) to prove it.

Exercise 17 [Additive phylogeny]

(a) Show that the smallest distance in the matrix need not correspond to neighbouring nodes in the tree.

(b) Prove the *limb-length theorem.*

(c) Describe the additive phylogeny algorithm. Argue that the algorithm is correct. What is the time complexity of the algorithm?

(d) What are the disadvantages of the algorithm?

(e) Define the *least-squares distance-based phylogeny problem.*

Exercise 18 [UPGMA algorithm]

(a) What is an *ultrametric tree*?

(b) Describe the UPGMA algorithm. What is the time complexity for each of the steps?

(c) (+) Can you improve its running time to $\mathcal{O}\left(n^2 \log n\right)$ (or better)?

(d) Attempt **[2018P27Q3 (e)]**.

(e) Attempt **[2015P7Q3 (e)]** (or **[2008P13Q1 (b)]**.

(f) (optional) Implement the algorithm.

Exercise 19 [Neighbour joining]

(a) Attempt **[2019P8Q2 (b)]**.

(b) Argue about the time complexity of every step of the algorithm.

Further Reading 4 If you are interested in learning why neighbour joining works look at here and here.

Exercise 20 [Small parsimony problem]

(a) What does the word *parsimony* mean?

(b) Define the *parsimony score* of a string-labelled tree. Give an example.

(c) Define the *small parsimony problem*.

(d) Explain how to reduce the small parsimony problem and how this problem can be solved efficiently.

(e) Derive the time complexity for the algorithm.

Exercise 21 [Large parsimony problem]

(a) Define the *large parsimony problem*.

(b) What is the main disadvantage with this problem?

(c) How does the *nearest neighbour interchange heuristic* work?

(d) (+) Suggest one way to extend the algorithm, when there are no other good local moves remaining.

Further Reading 5 [More approaches to the parsimony problem] You can read further approaches to the parsimony problem: here and here.

Exercise 22 [Revisiting multiple sequence alignment]

(a) Attempt **[2013P9Q1 (a)]**.

(b) Describe *progressive alignment*. What are its advantages and disadvantages?