COURSEWORK ASSIGNMENT **UNIVERSITY OF EAST ANGLIA**
School of Computing Sciences

UNIT :	Data Structures and Algorithms
ASSIGNMENT TITLE :	Coursework 1

DATE SET	**:** 18/10/17	
DATE OF SUBMISSION	**:** 3pm Weds 29/11/17	
RETURN DATE	**:** Week 2, Semester 2	
ASSIGNMENT VALUE	**:** 15%	
SET BY	**:** Jason Lines/Anthony Bagnall	**SIGNED:**
CHECKED BY	**:** Anthony Bagnall/Jason Lines	**SIGNED:**

Aim:

Subject specific

To test your ability to design, analyse and implement algorithms. To test your understanding of the principles underlying an important data structure.

Transferable skills

Describing algorithms using pseudo-code; use of mathematical notation when analysing algorithms; computer programming; use of drawing tools.

Learning outcomes:

Subject specific

Increased experience of programming in Java; increased awareness of the importance of algorithm complexity. An understanding of the principles underlying balanced binary search trees.

Assessment criteria:

Part marks are shown on the exercise sheet.

Description of the Assignment

1 Design and Implement an Algorithm for Subseries Search

Subseries and subsequence matching is a common problem in a range of disciplines. For example, we use it for data mining real valued series in the process of time-series classification with shapelets [1]. Many algorithms in bioinformatics involve matching subsequences of DNA and RNA data. You will design and implement an algorithm to perform this task. The basic question is this: given a series of numerical values, $S =< S_1, S_2, \ldots, S_k >$, of length k and a reference series of numerical values, T, of length m, where $m > k$, what is the subseries of T that is the closest match to S? By subseries, we mean any contiguous subset of values in a series. For example, $Q =< 1,3,4 >$ is a subseries of $T =< 6,3,1,3,4 >$ because Q appears as a contiguous block in T starting at position 3. We will denote the subseries of length k of T starting at T_i by $T[i \ldots i+k-1]$.

To answer the basic question, we need to define what we mean by closest, i.e. we need to define a *distance function* between two series. Given two series S and Q of equal length k, the Euclidean distance between these series is

$$d(S,Q) = \sqrt{\sum_{i=1}^{k} (S_i - Q_i)^2}.$$

The closest subseries between S and T is the subseries of T of length k that has the smallest distance of all possible subseries. This minimum distance between S and a series T is found by evaluating the distance between S and all subseries of T of length k, starting with $T[1 \ldots k]$ then $T[2 \ldots k+1]$, etc. Generally we would return the starting point of the closest subseries.

As an example, let my query series S be

$$S =< 1,0,3 >$$

and my full series T be

$$T =< 1,2,3,0,1,5 > .$$

My algorithm starts by finding the distance between S and $T[1 \ldots 3]$:

$$d(S,T[1 \ldots 3]) = \sqrt{(1-1)^2 + (0-2)^2 + (3-3)^2} = \sqrt{4} = 2.$$

I then need to evaluate

$$d(S,T[2 \ldots 4]) = \sqrt{(1-2)^2 + (0-3)^2 + (3-0)^2} = \sqrt{19},$$

and

$$d(S,T[3 \ldots 5]) = \sqrt{8}$$

and

$$d(S,T[4 \ldots 6]) = \sqrt{6}.$$

The closest match to S is $T[1 \ldots 3]$, so my algorithm should return the index value 1.

We can extend the above ideas to searching a list of series. The closest subseries in a list of series

$$\mathbf{T} =< T_1, T_2, \ldots, T_n >$$

is found by repeating the calculation described above over all n series and recording the position of the closest subseries and the index of the series it came from. For simplicity, let us suppose $m = n$ (i.e. the length of each series is the same as the number of series). We denote the j^{th} element of series i as $T_{i,j}$, so that $T_i =< T_{i,1}, T_{i,2}, \ldots, T_{i,n} >$. $\mathbf{T}$ is essentially an $n \times n$ matrix (a 2-D array), where index i denotes the series and index j denotes the position in the series i.

Your tasks are

1. Write an algorithm in pseudo code that takes as input a subsequence S of length k and a list of series $\mathbf{T} =< T_1, T_2, \ldots, T_n >$, where each series is of length n and returns the index of the series and the starting point of the subseries that has the lowest distance to query series S. [20 marks].
2. Perform a formal analysis of your algorithm. The run time complexity function will be a function of n and k, but assume k is a constant when characterising your run time function. [20 marks]
3. Implement your algorithm (including an implementation of a distance function and a means of handling ties in distance), and run a timing experiment to estimate the average case complexity, using randomly generated data created with the provided code. Assume k is fixed (I don't mind what value you choose) and plot how the time taken changes with n. [15 marks]
4. What is the lower bound for all algorithms to solve this problem? Justify your answer (informally). [5 marks]
5. Can you think of any ways to improve the constant time factor of the algorithm? If so, implement and test this/these improvement(s). HINT: consider when we could abandon the distance calculation. [10 marks]

Assessment

Marks will be awarded as follows

100% Question 1

Submission instructions overleaf.

Submission Procedure

Via Evision: A PDF, formed with the PASS system containing your solution to the exercises, including relevant code listings.

via blackboard: a zipped Netbeans project with your code for question 1.

Written coursework should be submitted by following the standard CMP practice. Students are advised to refer to the Guidelines and Hints on Written Work in CMP (`https://intranet.uea.ac.uk/computing/Links/Reports?_ga=1.7481330.1383599.1413214592`).

Plagiarism: Plagiarism is the copying of close paraphrasing of published or unpublished work, including the work of another student; without due acknowledgement. Plagiarism is regarded a serious offence by the University, and all cases will be investigated. Possible consequences of plagiarism include deduction of marks and disciplinary action, as detailed by UEA's Policy on Plagiarism and Collusion.

References

[1] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advance. *Data Mining and Knowledge Discovery*, pages 1–55, 2016.