Subject Description Form

Subject Code	COMP3022
Subject Title	Algorithms Engineering
Credit Value	3
Level	3
Pre-requisite / Co-requisite/ Exclusion	Pre-requisite: COMP1011/COMP1012/ENG2002 & COMP1411 & COMP2011/COMP2013
Objectives	The objectives of this subject are to: 1. design and implement algorithms for real-life computational problems, with comprehensive consideration of data characteristics, problem instances and implementation options; and 2. design, conduct, and analyse experiments for comparing the performance of algorithms with respect to a problem
Intended Learning Outcomes	Upon completion of the subject, students will be able to: *Professional/academic knowledge and skills* (a) identify the key requirements of real-life computational problems (e.g., response time, memory consumption, solution quality); (b) design comprehensive experiments for comparing the performance of experimental algorithms; (c) develop efficient algorithms for real-life computational problems; and *Attributes for all-roundedness* (d) present algorithms, experimental results and analysis clearly to fellow students.
Subject Synopsis/ Indicative Syllabus	**Topic** 1. **Introduction** Basic concepts (e.g., requirements, computational problems, algorithms, experiments), real-life examples of challenging problems (e.g., vehicle routing, route searching, graph partitioning, near neighbour search) 2. **Problem requirements** Types of requirements (e.g., response time, memory consumption, solution quality), constraints, optimisation objectives

3. **Analysis of data and problem instances**
   Identification of the characteristics of data and problem instances, utilisation of characteristics in designing algorithms

4. **Performance-oriented design of algorithms**
   Performance profiling, analysis on the performance bottleneck, techniques on optimizing the response time, the memory consumption, and the solution quality of algorithms

5. **Implementation options**
   Usage of library functions, usage of compiler options (e.g., optimisation options), multi-threading, vectorised code / single-instruction-multiple-data (SIMD)

6. **Experiments**
   Ways of measuring the performance, preparation of test cases and benchmarks, experimental goals, effects of system and hardware settings, experimental reproducibility, analysis of experimental results

7. **Presentation**
   Guidelines and practices on presenting algorithms and experimental results, in written and oral formats

Teaching/Learning Methodology

Students are expected to be proficient in programming (in C, C++, Java, or Python) and possess basic knowledge in data structures and algorithms.

Lectures will cover the basic concepts and techniques.

Lab sessions offer an opportunity to students for practicing their skills.

Assignments and individual project will be used to assess the abilities of students in developing algorithms, designing experiments, presenting results and analysis.

Assessment Methods in Alignment with Intended Learning Outcomes

Specific assessment methods/tasks	% weighting	Intended subject learning outcomes to be assessed			
		a	b	c	d
1. Individual assignments	30%	✓	✓	✓	
2. Individual project	30%	✓	✓	✓	✓
3. Final exam	40%	✓	✓	✓	
Total	100%				

Explanation of the appropriateness of the assessment methods in assessing the intended learning outcomes:

All assessment methods are used to assess the items a, b, c.

In addition, an individual project is used to assess presentation skills (both written and oral).

Student Study Effort Expected	Class contact:	
	▪ Lectures	26 Hrs.
	▪ Lab exercises	13 Hrs.
	Other student study effort:	
	▪ Individual assignments and individual project	66 Hrs.
	Total student study effort	105 Hrs.
Reading List and References	**Reference books:** 1. Catherine C. McGeoch, A Guide to Experimental Algorithmics. Cambridge University Press 2012, ISBN 978-0-521-17301-8 2. Rudolf Fleischer, Bernard M. E. Moret, and Erik M Schmidt, Experimental Algorithmics: From algorithm design to robust and efficient software, *Lecture Notes in Computer Science, 2547*, Springer, 2002. 3. Paul Cohen, Empirical Methods for Artificial Intelligence, MIT Press 1995. 4. S Halim and F Halim, Competitive Programming 3: The New Lower Bound of Programming Contests, Lulu Press, 2014. 5. Steven S Skiena and Miguel A. Revilla, Programming challenges, Springer, 2003. **Online references:** 1. The DIMACS Implementation Challenges: http://dimacs.rutgers.edu/programs/challenge/ 2. ACM Journal of Experimental Algorithmics (JEA). https://dl.acm.org/journal/jea 3. International Symposium on Experimental Algorithms (SEA). https://dblp.org/db/conf/wea/index.html 4. SIAM Symposium on Algorithm Engineering and Experiments (ALENEX). https://dblp.org/db/conf/alenex/index.html	