

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 & COMP2421 & COMP2011/COMP2013/DSAI2201			
Objectives	The objectives of this subject are to: <ul style="list-style-type: none"> 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: <p><i>Professional/academic knowledge and skills</i></p> <ul style="list-style-type: none"> (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 <p><i>Attributes for all-roundedness</i></p> <ul style="list-style-type: none"> (d) present algorithms, experimental results and analysis clearly to fellow students. 			
Subject Synopsis/ Indicative Syllabus	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Topic</th> </tr> </thead> <tbody> <tr> <td> <p>1. Introduction</p> <p>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)</p> </td> </tr> <tr> <td> <p>2. Problem requirements</p> <p>Types of requirements (e.g., response time, memory consumption, solution quality), constraints, optimisation objectives</p> </td> </tr> </tbody> </table>	Topic	<p>1. Introduction</p> <p>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)</p>	<p>2. Problem requirements</p> <p>Types of requirements (e.g., response time, memory consumption, solution quality), constraints, optimisation objectives</p>
Topic				
<p>1. Introduction</p> <p>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)</p>				
<p>2. Problem requirements</p> <p>Types of requirements (e.g., response time, memory consumption, solution quality), constraints, optimisation objectives</p>				

	<p>3. Analysis of data and problem instances Identification of the characteristics of data and problem instances, utilisation of characteristics in designing algorithms</p> <p>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</p> <p>5. Implementation options Usage of library functions, usage of compiler options (e.g., optimisation options), multi-threading, vectorised code / single-instruction-multiple-data (SIMD)</p> <p>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</p> <p>7. Presentation Guidelines and practices on presenting algorithms and experimental results, in written and oral formats</p>																												
<p>Teaching/Learning Methodology</p>	<p>Students are expected to be proficient in programming (in C, C++, Java, or Python) and possess basic knowledge in data structures and algorithms.</p> <p>Lectures will cover the basic concepts and techniques.</p> <p>Lab sessions offer an opportunity to students for practicing their skills.</p> <p>Assignments and individual project will be used to assess the abilities of students in developing algorithms, designing experiments, presenting results and analysis.</p>																												
<p>Assessment Methods in Alignment with Intended Learning Outcomes</p>	<table border="1" data-bbox="520 1366 1471 1803"> <thead> <tr> <th rowspan="2">Specific assessment methods/tasks</th> <th rowspan="2">% weighting</th> <th colspan="4">Intended subject learning outcomes to be assessed</th> </tr> <tr> <th>a</th> <th>b</th> <th>c</th> <th>d</th> </tr> </thead> <tbody> <tr> <td>1. Continuous Assessment (assignments, project, quizzes)</td> <td>30%</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>2. Final Exam</td> <td>70%</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> </tr> <tr> <td>Total</td> <td>100%</td> <td colspan="4"></td> </tr> </tbody> </table> <p>Explanation of the appropriateness of the assessment methods in assessing the intended learning outcomes:</p> <p>All assessment methods are used to assess the items a, b, c.</p> <p>In addition, an individual project is used to assess presentation skills (both written and oral).</p>	Specific assessment methods/tasks	% weighting	Intended subject learning outcomes to be assessed				a	b	c	d	1. Continuous Assessment (assignments, project, quizzes)	30%	✓	✓	✓	✓	2. Final Exam	70%	✓	✓	✓		Total	100%				
Specific assessment methods/tasks	% weighting			Intended subject learning outcomes to be assessed																									
		a	b	c	d																								
1. Continuous Assessment (assignments, project, quizzes)	30%	✓	✓	✓	✓																								
2. Final Exam	70%	✓	✓	✓																									
Total	100%																												

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	<p>Reference books:</p> <ol style="list-style-type: none"> 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, <i>Lecture Notes in Computer Science, 2547</i>, 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. <p>Online references:</p> <ol style="list-style-type: none"> 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/alnex/index.html 	