Write your answers electronically then submit as a PDF file through the Turnitin link on Moodle before **11:55pm on Thursday 15 December 2016.**

Consider the following two problems

**Subset-Sum Problem**

Given a set $S = \{x_1, x_2, \ldots, x_n\}$ of integers, and an integer $t$ (called *target*) decide if there is a subset of $S$ whose sum is equal to $t$.

**Partition Problem**

Given a set $S = \{x_1, x_2, \ldots, x_n\}$ of numbers, decide if it can be partitioned into two sets such that they both have the same sums.

Choose **one** of these two problems, and develop computer programs to try and solve it, as directed in the related lab guidance notes (Weeks 8–11).

You will consider exact methods (exhaustive search, dynamic programming) and non-exact/approximation methods (greedy strategies, meta-heuristics), and special cases that can be solved in polynomial time. You should test them on instances with varying parameter sizes. Record the time taken and, for the optimization problem, how close these get to a complete solution. You need to test over a large range of instances and produce averaged results for each tested parameters set.

Once you have completed the related in-lab exercises, you should have working heuristics to use. You are encouraged to improve on these wherever possible (GRASP, Simulated Annealing, etc.).

From these experiments you should gain an idea on the capabilities and limitations of the algorithms, which you should analyse and comment on in a short report structured as detailed in the next page.

**Please note the following:**

- You may work on the code development in **groups** of your choosing, provided that the groups are not larger than **6 members**.
  **List the names** of all the group members in your report.
- While the code may be worked on in groups the reports are **individual** (including discussion of the algorithms and analysis of results). Identical reports will be treated as plagiarism.
- The code should be on **GitHub** (or similar) to show evidence of contribution (code commites, issues raised, discussions, wiki contributions, etc.). The repository should be kept **private** until the submission deadline when it should be made public.
- **Referencing**: Support your claims with appropriate citations using the CU-Harvard style. Show the list of references on the last slide.
- **Be concise** – assume that the marker knows everything mentioned in the lectures and the lab sheets, so only concentrate on your achievements and what you have learnt by working on this project. Do not put too much text on the slides – go straight to the point, and use the space on the slides effectively.

Your report should be in the form of a presentation (1-3 slides per item), and structured as follows:

(10 marks) 1) Definition of the problem (Decision, computation, and optimization versions). Fix the notation for the rest of the presentation.

Discuss their complexity class membership (**P**, **NP**, **NP-complete**, **NP-hard**, etc.).

(10 marks) 2) Outline of the testing methodology and random instances sampling strategy.

3) Report on the methodology used to solve the problem. For each method, give its time complexity using O-notation, and discuss its performance in practice (e.g. timing and/or accuracy plot for suitably generated instances).

a) Exact methods:

(10 marks) • Exhaustive search

(10 marks) • Dynamic programming

(10 marks) b) Greedy and GRASP.

(10 marks) c) Any other advanced/alternative methods (Simulated Annealing, . . . ).

(10 marks) d) Special cases that can be solved in polynomial time.

(10 marks) 4) Conclusion with recommendations on when each method is most suitable.

(10 marks) 5) Reflection. (What have you learnt? What could you have done differently? If you collaborated on the coding then specify your contribution clearly.)

(10 marks) 6) Present your work clearly and nicely, using graphs and referencing wherever appropriate using the CU Harvard referencing style.

Code should be on GitHub (or similar). Only provide a link to it.

# Marking scheme

| Mark | Criterion |
|---|---|
| **70-100%** | Excellent work demonstrating strong analytical skills and competent application of classification techniques. Heuristics produce good results and are imaginatively designed. Software well implemented and tested, with well documented results and accurate conclusions. |
| **60-70%** | Problems well analysed and classified. Heuristics work reliably and all implemented software has been well tested and debugged. Results are accurate and conclusions sensible. |
| **50-60%** | A good understanding of classification techniques is present but application may lack precision. Some attempt made in producing heuristics but perhaps confined to simpler methods and approaches. Implementation, results and conclusions are present and satisfactory. |
| **40-50%** | May be lacking in some areas. Demonstrates a basic understanding of the theory but perhaps without succeeding in a complete implementation of all ideas. |
| **Less than 40%** | Little or no demonstration that material is understood. Algorithms may be incorrect in concept and/or implementation. Results may be misreported or absent. |