COMP9414 Assignment 1

1. Put all marks and feedback in one markdown cell at the end of the notebook, and put the total mark in this cell **and** on a spreadsheet with three columns (zID without the "z", the mark you give out of 25, and the late penalty, the number of days/part days late from 1–5, empty if on time). *Take note of the timestamp on the file before marking to determine the late penalty.*
2. If the empirical results seem strange, look at the code to work out if there is something obvious wrong, but there is no need to fix any code.
3. For each question, you can give half marks, but do an overall "sanity check" to make sure the total mark is not too high or too low.
4. The empirical analysis should include plots. The axes on the plots should be labelled, and there should be some interpretation of the results. There may be a problem with images not appearing in some versions of Jupyter, which may require us to contact the student. We do not expect the code to generate plots runs independently of the cells above, so there is no need to run this code.

Marking Rubric

Question 1 (4 marks) Search Spaces for CSP Solvers

Q 1 (4marks)			
1.1 (1 mark)	**1.2 (1 mark)**	**1.3 (1 mark)**	**1.4 (1 mark)**

Informal description of search spaces (1 mark)

Domain Splitting:
Start state is CSP with all variables having domains satisfying unary hard constraints;
Successor function to split domain of one variable into two, with other variables' domains the same;
Goal state is CSP with all variable domains of size 1 whose values satisfy all constraints.

DFS:
Start state is empty assignment, where each variable has no value assigned;
Successor function is to choose a variable without a value and assign it one value from its domain;
Goal state is an assignment of values to all variables that satisfies all constraints.

Use of the terms "meeting" and "day-times" instead of variables/values is OK.
It is OK if the successor function requires that the new value(s) are consistent (if correctly defined).
Assign 0.5 marks for domain splitting, 0.5 for DFS.
0.5 marks if some information is missing but the essential information is included.

Branching factor and maximum depth to find any solution (1 mark)

Branching factor:
2V for domain splitting; D for DFS, where the CSP has V variables with domain size D.
For the branching factor of DFS, 40 (interpreted as the size of the domain) is OK.

Maximum first solution depth:
$V.\log_2(D)$ for domain splitting, V for DFS, where the CSP has V variables with domain size D.

Not having ceiling(log_2) is OK.
Assign 0.5 marks for branching factor, 0.5 for maximum first solution depth.

Worst case time and space complexity of the two search algorithms? (1 mark)

Domain Splitting: $O((2V)^{V*\log_2(D)})$ for time, $O(V^2.\log_2(D))$ for space.
DFS: $O(D^V)$ for time, $O(D.V)$ for space.

0.5 marks for correct but informal answers such as "exponential in the number of variables".
0.5 marks if the answer is consistent with the answer for branching factor and depth (b^d and bd).

Example of a problem that is *easier* for domain splitting than for DFS (1 mark)

The explanation should refer to the number of constraints. A solvable problem with many (hard) constraints should be easier for domain splitting to solve because domains are reduced faster.
0.5 marks for a suitable sample problem with no explanation.

Question 2 (5 marks) Cost Function

Q 2 (5 marks)		
2.1 (3 marks)	**2.2 (1 mark)**	**2.3 (1 mark)**

Implementation of cost function (3 marks)
2 marks for correct implementation (the cost function should actually return the correct cost).
1 mark for readability/efficiency (0 marks only if the code is very hard to read or very complex).
At most 1 mark for code that does not run, even if just for a small technical error.

```
def calculate_cost(self):
    cost = 0
    for var, domain in self.domains.items():
        # sum of costs over all variables of minimum cost value
        costs = []
        functions = self.cost_functions[var]
        for val in domain:
            # cost for val is sum of soft constraint violations
            val_cost = 0
            day = Day_Time().day_number(val)
            hour = Day_Time().day_hour_number(val)
            for function in functions:
                val_cost += function(day, hour)
            costs = costs + [val_cost]
        if len(costs) > 0:
            cost = cost + min(costs)
    return cost
```

Computational complexity in a general form? (1 mark)
$O(D.V)$ where the CSP has V variables with domain size D.

1 mark for the correct answer using Big-O notation.
0.5 for informal answer such as "the function is linear in the number of meetings".

Argument that the cost function *f* never decreases along a path, and why this means the search algorithm is optimal (1 mark)
1 mark for the explanation that if one domain is a subset of another for any variable, with all other variable domains the same, the cost function is always the same or higher, therefore the search algorithm visits paths in order of cost, hence a lower-cost solution will always be visited before any other solution (this is similar to the proof of optimality of A from lectures).*
0.5 marks if the explanation is basically correct but parts are missing.

Question 3 (4 marks) Empirical Evaluation of Domain Splitting Solver

Q 3 (4 marks)		
3.1 (2 marks)	3.2 (1 mark)	3.3 (1 mark)

generate_problem(n) (2 marks)
Try running the code to make sure it works (this code should be independent of the above cells).
Assign 1 mark for correctness, 1 mark for readability/efficiency.
0 marks for readability/efficiency only if the code is very hard to read or very complex (e.g. lots of unnecessary nested loops).
0 marks for code that does not run, even if just for a small technical error.

Plot of the performance of the two constraint solving algorithms against n (1 mark)
The plot should be a line graph with a line for each algorithm, and the x-axis the problem size n.
0 marks if the plot is not present (there may be a problem with images not appearing in some versions of Jupyter, which may require us to contact the student).

Quantification of the performance gain achieved by the use of this cost function (1 mark)
1 mark for some explanation and some quantification of the difference between the two results.
0.5 if the response only reports the qualitative difference without explanation (e.g. "the cost function gives an improvement for n=5", but without explaining a possible reason for this).

Question 4 (5 marks) DFS Solver vs DFS-MRV Solver

Q 4 (5 marks)				
4.1 (1 mark)	4.2 (1 mark)	4.3 (1 mark)	4.4 (1 mark)	4.5 (1 mark)

Worst case time and space complexity of each algorithm (1 mark)
DFS-MRV: $O(D^V)$ for time, $O(D.V)$ for space, with V variables of domain size D.

For DFS, this is the same as Question 1. The answer for DFS-MRV is the same as for DFS.

Properties of the search algorithms (completeness, optimality) (1 mark)
0.5 marks for the answer that both algorithms are complete [since the domain is finite].

0.5 for the answer that both algorithms are not optimal because the search ignores costs, so there is no guarantee that the lowest cost solution is found first.

Example problem *easier* for the DFS-MRV solver than for DFS, and explanation (1 mark)
1 mark for a specific problem and comment on why DFS-MRV outperforms DFS (e.g. a problem with a lot of soft constraints should be easier because values are explored in order of cost).
0.5 marks for a problem without any explanation.
0 marks for an example problem that is unrelated to the assignment (e.g. Sudoku).

Empirical comparison of the quality of the first solution found by DFS and DFS-MRV compared to the optimal solution (1 mark)
1 mark for a plot of DFS/DFS-MRV against problem size, plus explanation of the results
0.5 marks for just one example problem and brief comparison of the solutions for that problem.

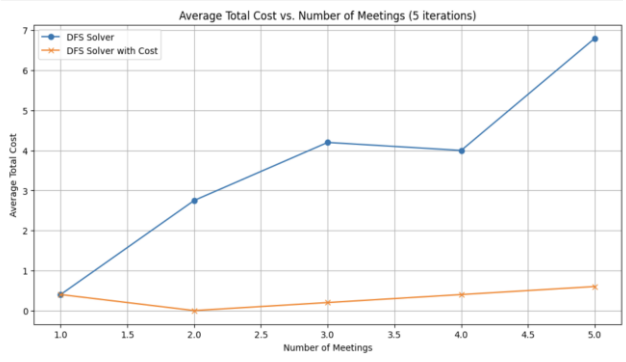
Empirical comparison of DFS-MRV with DFS (1 mark)
1 mark for a plot of number of nodes expanded vs problem size, plus explanation of the results.
0.5 marks for a plot or numbers without explanation.
0 marks if the plots look incorrect.

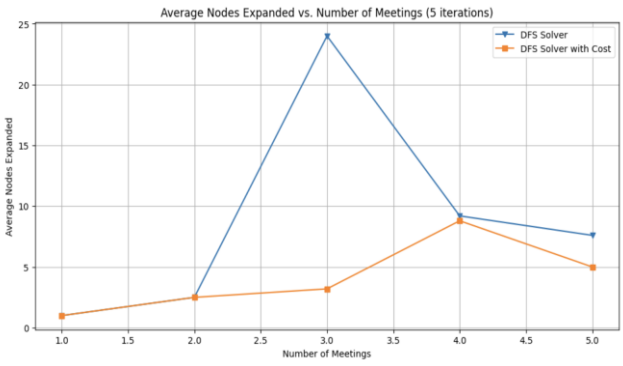
Question 5 (4 marks) DFS Solver with Search Heuristics

Q 5 (4 marks)	
5.1 (2 marks)	5.2 (2 marks)

Implementation of search ordering heuristics in DFS (2 marks)
1.5 marks for correctness (i.e. the algorithm should correctly implement costs in the solver).
0.5 marks for readability/code quality (i.e. reuse of DFS code, and sufficient comments).
At most 0.5 marks for code that does not seem to work correctly.

Empirical comparison of DFS with and without search ordering heuristics (2 marks)
The answer should include plots and conclusions. Both the cost of the first solution found and the number of nodes expanded for DFS <u>with</u> costs should be lower than for DFS <u>without</u> costs.
Assign 1 mark for first solution cost, 1 mark for number of nodes expanded.
In each case, 0.5 marks for plot; 0.5 marks for conclusion.

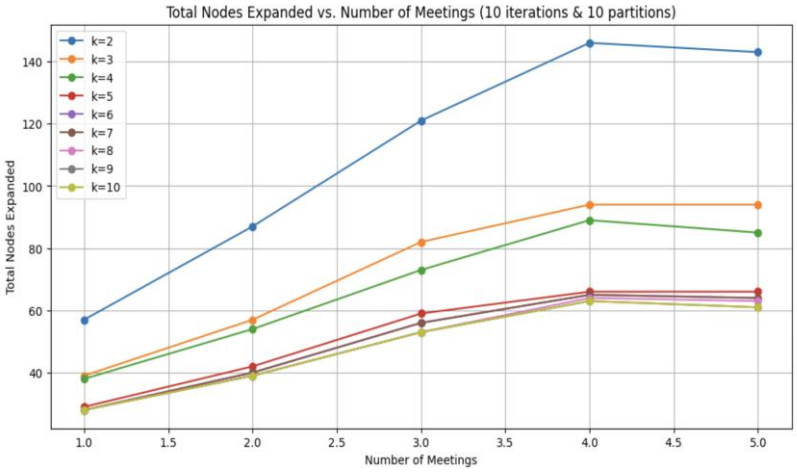




Question 6 (3 marks) Domain Splitting with k Partitions

Q6 (3 marks)	
6.1 (1 mark)	6.2 (2 mark)

Implementation of



partition_domain(k) (1 mark)
1 mark for a correct and well-written function.
0.5 marks if the function does not work for general k.

Performance evaluation of domain splitting solver for a range of values of k (2 marks)
1 mark for a plot of the average number of nodes expanded vs the size of the problem for varying k.
1 mark for the conclusion that using more than 2 partitions improves performance in this domain.