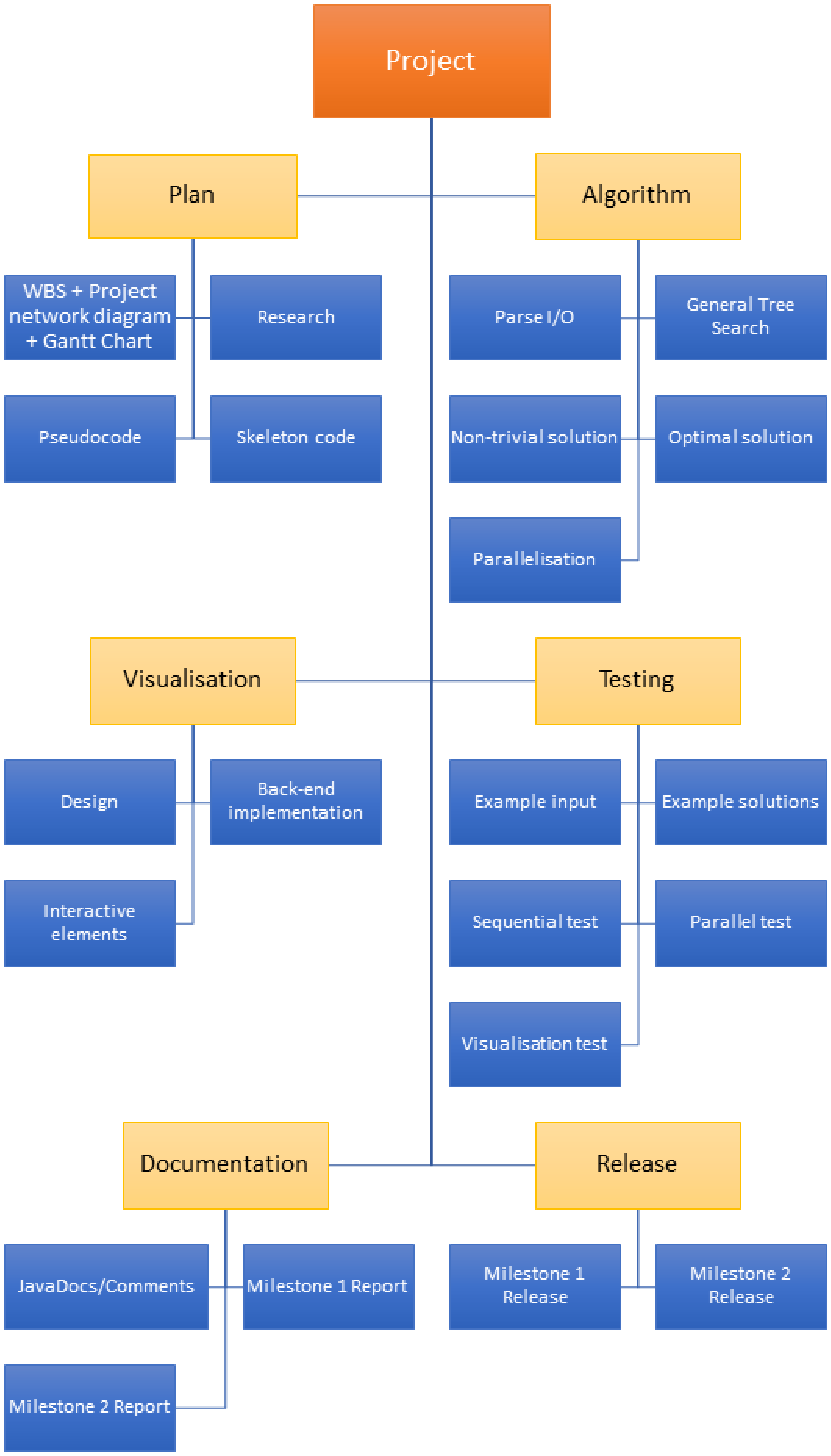
Project 1 Plan

SOFTENG306 Group 14

Whan Jung
Dylan Jung
Keshav Bharanitharan
Jack Lobb
Braden Palmer

Work Breakdown Structure



Work Breakdown Description

Plan

- WBS + Project network diagram + Gantt Chart (The plan and the associated annotations.)
- Research (involves reading through the project document and fully understanding the project. Also may involve seeking inspiration for graphic design and sources for relevant algorithms.)
- Pseudocode (The general plan for the code architecture. This part is also used to determine the types of classes we might need, as well as the relationship between classes.)
- Skeleton code (The important classes and methods will be created beforehand such that different people can work on different classes and methods. If we decide that the creation of a class diagram is necessary, this task can also include the creation of a class diagram.)

Algorithm

- Parse input and output (Reading and Writing .dot files to and from a data structure such as an adjacency list or matrix. We expect to be able to use a library for this.)
- General tree search (The parts of the code needed to search through the state space and also the data structures used to store each state.)
- Non-trivial solution (The algorithm should generate a non-trivial solution for Milestone 1. Should also implement some form of branch and bound to speed up the search to reasonable levels.)
- Optimal solution (The algorithm should generate an optimal solution for Milestone 2. May adjust the code to A* depending on memory and time constraints.)
- Parallelisation (Making our algorithm implementation multithreaded. Each thread should be searching a branch in the state space.)

Visualisation

- Design (The design of the visualization of the search. This task involves generating the view components and styling them.)
- Back-end implementation (The controller of the visualisation. Uses the data from the main algorithm and updates the view components as the search progresses. Should be independent from the algorithm such that it can easily be switched on and off without affecting performance.)
- Interactive elements (Implement features such as ‘magnify’, so that users have more control over the visualisation.)

Testing

- Example input (There should be a range of examples created. Examples can range from 4-26 nodes and there should be various boundary cases and normal cases. It may even be good to generate invalid input see how the program handles it.)
- Solutions to example input (Manually verifying optimal/valid solutions to produce test cases is not trivial work for this problem. In the case of invalid input, the solution should be some sort of error message.)

- Sequential test (Testing the sequential algorithm with our examples.)
- Parallel test (Testing the multithreaded algorithm with our examples. Multithreading can lead to new bugs, so this second test phase is not trivial.)
- Visualisation test (Involves running the visualisation against various examples and seeing whether the intermediate results and the final result is correct, as well as seeing whether there are any other problems such as lag.)

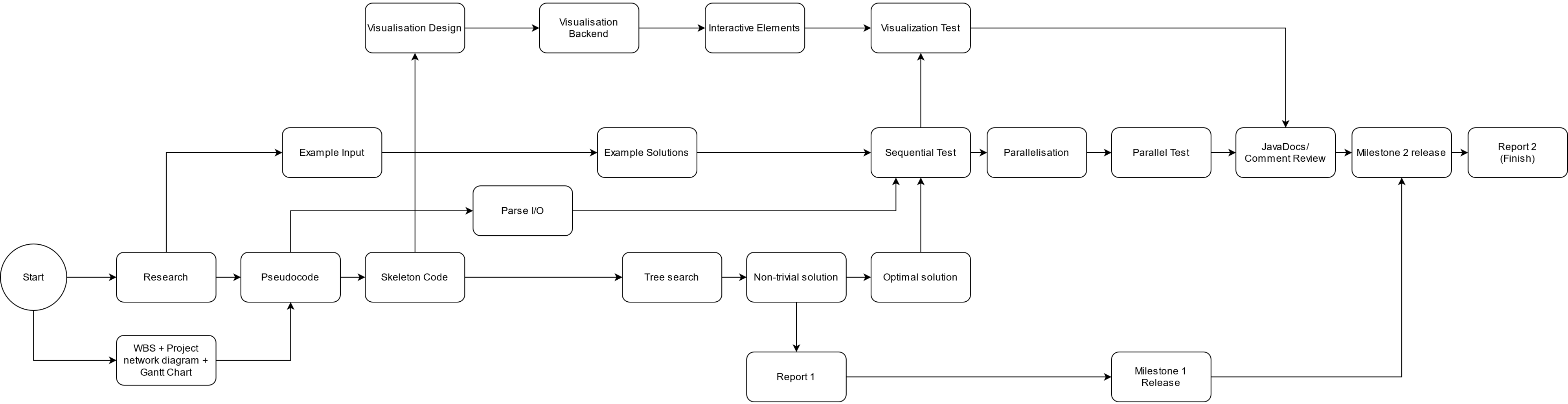
Documentation

- JavaDocs/Comment Review (We intend to comment as we progress. This task involves a comment review and the generation of a JavaDocs-based wiki.)
- Milestone 1 report (2 pages)
- Milestone 2 report (10 pages)

Release

- Milestone 1 release (involves building the release and double checking everything, including the report and making sure that program runs on Linux. Basically, making sure that everything is functional. Can also include refactoring.)
- Milestone 2 release (involves building the release and double checking everything, including the report and making sure that program runs on Linux. Basically, making sure that everything is functional. Can also include refactoring.)

Network Diagram



Network Diagram Description

The general flow of the project is that we are going to create pseudocode to get a general idea of what the code is going to look like. Then we will create skeleton code such that each group member will be able to work on their individual section. The four main sections we have identified are testing, visualisation, algorithm, and documentation.

Testing

The development of the example inputs can be started before the pseudo code. Rigorous testing of each section of the code will proceed after the completion of that section of code. Testing must be performed for both the single-threaded and multi-threaded versions of the algorithm.

Visualisation

The view will be designed and created first before creating the controller layer. It is possible to create visualisation design and backend simultaneously, however we decided that creating the view first would be better as we can test as we go along. Interactive elements would be added last as they are not too important.

Algorithm

We will implement parsing input and output before testing, since testing is impossible without parsing input and output. For the algorithm, we will first implement a search through the state space, and then we will slowly optimise our searching until we can reach the optimal solution in a feasible time. Parallelisation will only be implemented after rigorous testing of the sequential program so that we can easily identify and diagnose any problems related to multithreading.

Documentation

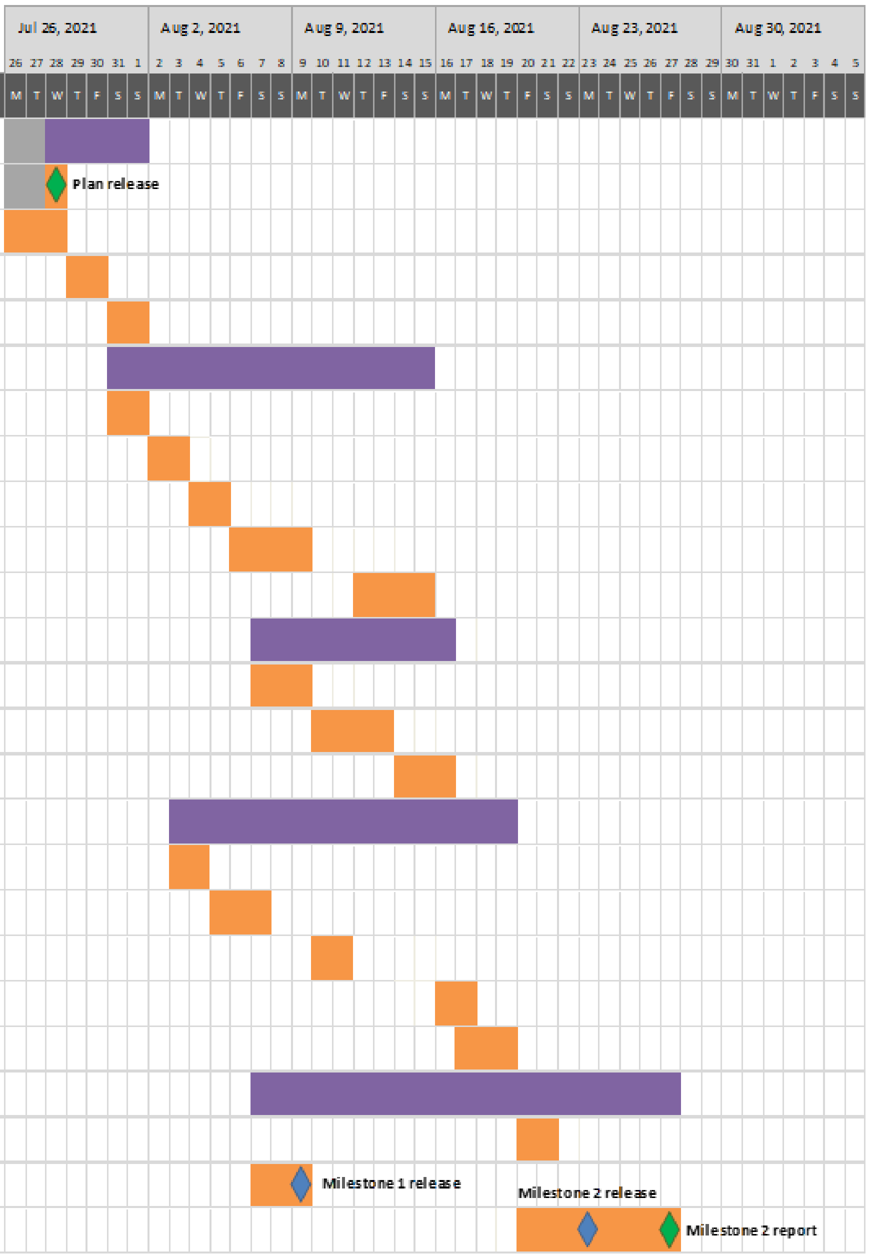
We will do the majority of the commenting during the writing of the code. At the end, once all of the different pieces of code have been tested and finalised, these comments will be refined and the Javadocs generated.

The four sections will converge before the final report because the final report requires the entire project to be finished.

Gantt Chart

Project Start:	Sun, 7/25/2021
Display Week:	1

TASK	PROGRESS	START	END
WBS 1 Plan	40%	7/25/21	8/1/21
1.1 WBS + Project Network Diagram + Gantt Chart	90%	7/25/21	7/28/21
1.2 Research	10%	7/25/21	7/28/21
1.3 Pseudocode	0%	7/29/21	7/30/21
1.4 Skeleton code	0%	7/31/21	8/1/21
WBS 2 Algorithm	0%	7/31/21	8/15/21
2.1 Parse input and output	0%	7/31/21	8/1/21
2.2 General tree search	0%	8/2/21	8/3/21
2.3 Non-trivial solution	0%	8/4/21	8/5/21
2.4 Optimal solution	0%	8/6/21	8/9/21
2.5 Parallelisation	0%	8/12/21	8/15/21
WBS 3 Visualisation	0%	8/7/21	8/16/21
3.1 Design	0%	8/7/21	8/9/21
3.2 Back-end implementation	0%	8/10/21	8/13/21
3.3 Interactive elements	0%	8/14/21	8/16/21
WBS 4 Testing	0%	8/3/21	8/19/21
4.1 Example input	0%	8/3/21	8/4/21
4.2 Solutions to example input	0%	8/5/21	8/7/21
4.3 Sequential test	0%	8/10/21	8/11/21
4.4 Parallel test	0%	8/16/21	8/17/21
4.5 Visualisation test	0%	8/17/21	8/19/21
WBS 5 Documentation	0%	8/7/2021	8/27/2021
5.1 Java doc/comments	0%	8/20/21	8/21/21
5.2 Milestone 1 report	0%	8/7/21	8/9/21
5.3 Milestone 2 report	0%	8/20/21	8/27/21



🔷 Milestone

🟢 Major Deliverable