COMP3121 Assignment 3 – Q3

3.1

It is impossible to perform all n calculations unless the graph G is acyclic as there will be at least one calculation that will indirectly depend on itself if G has a cycle.

To perform all calculations of n, we would have to begin with one vertex to begin the calculation. If G has a cycle, it would mean that the calculation of the current vertex would rely on the vertex before it, and to compute the calculation of the vertex before the current vertex, we would need to compute the calculation of the one before that one and so on, indefinitely looping.

Thus, unless G is an acyclic graph, it is impossible to perform all n calculations.

3.2

Consider an algorithm which runs in $O(n + m)$ time and determines the minimum amount of time required to perform all n calculations on the parallel computer, where m is the number of dependencies (edges).

To determine the overall time needed to perform all the calculations, the algorithm must determine which calculations to do first and the sequence of calculations to follow as some calculations will depend on the result of others.

To determine how to perform all n calculations in graph G, the algorithm will utilise topological sorting and determine the sequence of calculations of most optimal to go through. The topological sort will determine the optimal order for calculations to be performed where the dependencies are done before the calculations that depend on them.

Once the order of calculations is determined, the algorithm will iterate through each calculation at a time, keeping track of the outcome along with any computations which may depend on it.

Since running computations in parallel with the parallel computer can carry out an unlimited number of operations at once, the time needed can be shortened down by computing certain calculations in parallel.

The algorithm will first compute all the calculations which are independent in parallel.

Then, perform all the calculations that are dependent on the calculations done earlier in parallel.

With this method, all the calculations which can be done in parallel are done at the same time, allowing the time needed to be minimised as much as possible.

For each computation, the time required is:

$$\sum(j \in pred(i))r_j + r_i^2,$$

where $\sum(j \in pred(i))r_j$ is the time to compile the results of dependencies, and r_i^2 is computing the result.

As such, the minimum time taken to calculate all n calculations will be equal to the sum of the longest individual independent calculation and the longest individual dependent calculation.

The time complexity of a topological sort is $O(V + E)$ where V is the number of vertices and E is the number of edges in the graph. For this algorithm, the overall run time will be $O(n + m)$ where n is the number of calculations (edges) and m is the number of edges (dependencies).

3.3

Consider an algorithm which runs in $O(s(n+m))$ time and determines the minimum amount of time required to compute all n results using the supercomputer for at most s calculations, and the parallel computer for all other calculations.

This algorithm will adapt the algorithm from question 3.2. As per 3.2, the graph g will be sorted using a topological sort to identify the optimal sequence that the calculations should be performed.

The supercomputer can be used to compile the results of the independent calculations s times. The parallel computer will be used to compute the other calculations not done by the supercomputer.

This algorithm will have a run time of $O\big(s(n+m)\big)$ as a topological sort is done in $O(m+n)$ time, iterating over the sorted sequence of calculations takes $O(m+n)$ time and the use of the super computer to execute s computations results in an overall time complexity of $O(s(m+n))$.