COMP3121 Assignment 2 Q3

3.1

Suppose there is a tome of 5 names and made of 4 glyphs each:

$$\{+, x, =, z\},$$

$$\{+, +, z, =\},$$

$$\{+, -, -, =\},$$

$$\{+, -, +, +\},$$

$$\{x, -, +, =\}.$$

With this tome of names, regardless of how they are arranged, it is impossible to form an alphabet of glyphs.

While comparing the first 1st, 2nd, 3rd, 4th and the 5th names, it is implied in the first column that $+$ comes before x in the alphabet. However, in the 1st and 2nd names, it is implied that in the second column, x comes before $+$ instead, contradicting the first statement and the alphabetical order of these names.

Moreover, comparing the 2nd and 3rd names, it is implied in the second column that $+$ comes before – in the alphabet. However, as observed in the third column of the 3rd and 4th name, $-$ would be before $+$ in the alphabet, contradicting the first statement.

Therefore, the names are not in alphabetical order regardless of how the glyphs are rearranged to form an alphabet.

3.2

Consider an algorithm which will take the number of glyphs g, the number of names n, the number of glyphs in each name k, and a two-dimensional array $S[1..n][1..k]$ containing the numbers 1 through g, where $S[i]$ is an array containing the ith name, and $S[i][j]$ is the index of the jth glyph in that name

This algorithm will involve building a directed graph, where each vertex represents a glyph.

To begin, the algorithm will a initialise directed graph G. Each vertex will be labelled with each unique glyph on the tome. To determine each unique glyph, the algorithm will have to iterate through S and add each unique glyph one as a vertex to the graph.

Next, to add the edges to the graph, the algorithm will loop through each name n and then loop through each glyph in each name.

For each adjacent name, the algorithm will add a directed edge from the first letter to the second character while it is iterating through the name.

To compute the time complexity of this process, we consider the loops used. Since the algorithm first loops through each name n, the time complexity of the first loop is $O(n)$. Moreover, as the algorithm will loop through each glyph k in each name, the time complexity of the nested loop is $O(k)$. The nested loop inside the main loop will result in a time complexity of $O(nk)$.

Once this is done for each name in the array, the directed graph is complete.

The algorithm will then utilise a topological sort on it. If the sort succeeds, then the order of the vertices output can be taken as the alphabet created from the glyphs.

Otherwise, if the sort is unable to product an order of vertices, the algorithm will decide that no alphabet exists.

Running a topological sort will result in an overall time complexity of $O(g + E)$, where g is the total number of unique glyphs (total number of vertices) and E is the total number of edges in the created graph.

In this case, the total number of edges would be at most $n(k-1)$ as each word n will contribute $k-1$ edges to the graph, leaving a time complexity of $O\big(g + n(k-1)\big)$.

To compute the final time complexity, sum up the time complexity of the graph creation and the topological sort.

$$\begin{aligned} O(nk) + O\big(g + n(k-1)\big) &= O\big(nk + g + n(k-1)\big) \\ &= O(nk + g). \end{aligned}$$

The final time complexity is simplified to $O(nk + g)$ as $n(k-1)$ is the lesser dominant term compared to nk. Thus, the time complexity of this algorithm $O(nk + g)$.