COMP4500/7500 Advanced Algorithms & Data Structures
Tutorial Exercises 9 (2019/2)*

School of Information Technology and Electrical Engineering, University of Queensland

October 10, 2019

This material aims to familiarise you with P, NP and NP-completeness. A good treatment of this may be found in CLRS Chapter 34.

1. (CLRS 34.1-1 p1060 [3rd]) Define the problem **LONGEST-PATH-LENGTH** as the relation that associates each instance of an undirected graph G and two vertices u and v with the number of edges k in a longest simple (no duplicates) path between the two vertices. Define the related decision problem **LONGEST-PATH** as

$$\textbf{LONGEST-PATH} = \{\langle G, u, v, k\rangle : \begin{array}{l} G \text{ is an undirected graph, } u, v \in G.\,V, k \geq 0 \text{ is an integer and} \\ \text{there exists a simple path from } u \text{ to } v \text{ in } G \text{ with at least } k \text{ vertices}\} \end{array}$$

   Recall that a decision problem just returns either 0 or 1. Show that the optimisation problem **LONGEST-PATH-LENGTH** can be solved in polynomial time if and only if **LONGEST-PATH** $\in$ P.

2. (CLRS 34.1-3 p1060 [3rd]) Give a formal encoding of directed graphs as binary strings using an adjacency-matrix representation giving the complexity of the encoding's size in terms of the number of vertices in the graph. Do the same using an adjacency-list representation. Argue that the two representations are polynomial related.

3. (CLRS 34.1-4 but for the subset-sum problem) Is the dynamic programming algorithm for the **SUBSET-SUM** problem for tutorial exercises 6 a polynomial time algorithm? You may assume all of the sizes within s are at most the total capacity C. The dynamic programming solution from the early tutorial exercise follows, except here the "sizes" parameter s has been made explicit.

```
SubsetSum(n, C, s)
for ci = 0 to C
    table[0, ci] = 0
for i = 1 to n
    for ci = 0 to C
        if s[i] > ci    // Cannot include item i
            table[i, ci] = table[i − 1, ci]
        else    // s[i] ≤ ci  — can include item i
            with = table[i − 1, ci − s[i]] + s[i]
            without = table[i − 1, ci]
            table[i, ci] = Max(with, without)
return table[n, C]
```

4. (CLRS 34.2-1) Two graphs $G = (V, E)$ and $G' = (V', E')$ are ***isomorphic*** if there exists a bijection (i.e., a total, onto, one-to-one mapping) $f \in V \to V'$ such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in E'$. Consider the decision problem

$$\textbf{GRAPH-ISOMORPHISM} = \{\langle G, G'\rangle : G \text{ and } G' \text{ are isomorphic graphs }\}$$

   Prove that **GRAPH-ISOMORPHISM** $\in$ NP by describing a polynomial-time algorithm to verify a solution to the decision problem.