FIT3155: Week 2 tutorial
Covering concepts from Week 1

Objectives: The tutorials, in general, give practice in problem solving, in analysis of algorithms and data-structures, and in mathematics and logic useful in the above.

Instructions to the class: Prepare your answers to the questions **before** the tutorial. It will probably not be possible to cover all questions unless the class has prepared them all in advance.

Instructions to Tutors:

i. The purpose of the tutorials is not to solve the practical exercises.

ii. The purpose is to check answers, and to discuss particular sticking points, not to simply make answers available.

1. Revise Gusfield's Z-value based linear-time exact matching algorithm

2. In the above Z-algorithm, for any given input string if we found that $Z_2 = q$ (where $q > 0$), then the values of $Z_3, Z_4, \ldots, Z_{q+1}, Z_{q+2}$ can be immediately obtained without additional character comparisons. Reason why?

3. Stare at the lecture slide handling the preprocessing CASE 2b of the Z-algorithm. When $Z_{k-l+1} \geq r-k+1$, the algorithm does explicit comparisons until it finds a mismatch. This is a reasonable way to organize the algorithm, but in fact CASE 2b can be refined so as to eliminate an unneeded character comparison. Argue that when $Z_{k-l+1} > r-k+1$, then $Z_k = r-k+1$, and hence no character comparisons are necessary. Therefore, explicit character comparisons are needed only in the case when $Z_{k-l+1} = r-k+1$.

4. Reason a potential linear-time solution for the following problem: Given two equal-length strings α and β from a fixed alphabet, determine if α is a circular (or cyclic) rotation of β. For example, $\alpha =$ `defabc` is a circular rotation of $\beta =$ `abcdef`.

5. Similar to the above exercise, give a linear-time algorithm to determine whether a linear string α is a SUBstring of a circular string β. Note:

a circular string `str[1..n]` is such that the character `str[n]` precedes character `str[1]`.

6. Give an algorithm that takes in two string α and β of lengths m and n, and finds the longest suffix of α that exactly matches a prefix of β. Reason the run-time of your algorithm.

7. Given a string `str`$[1..n]$, let `len`(**i**) denote the length of the largest suffix of `str`[**i**..n] that is also a prefix of `str`. Give an algorithm that computes `len`(**i**) values. Reason the run-time of your algorithm.

```
-=o0o=-
  END
-=o0o=-
```