FIT2004: Tutorial 4 (held in Week 9)
Covering graph concepts from Weeks 8

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 in advance. There is 1 mark worth for this Tute towards **active** participation. 0.5 marks is towards answering the starred questions (*) indicated below, **before** attending your assigned tutorial. You will have to hand in your work on these starred questions to your tutor at the very start of the tutorial. Remaining 0.5 mark is for participating during the rest of the tutorial.

Instructions to Tutors:

1. The purpose of the tutorials is not to solve the practical exercises!
2. The purpose is to check answers, and to discuss particular sticking points, not to simply make answers available.

1. Work out on paper, using Dijkstra's algorithm, the shortest distance from the source vertex z to every other vertex in the given graph:

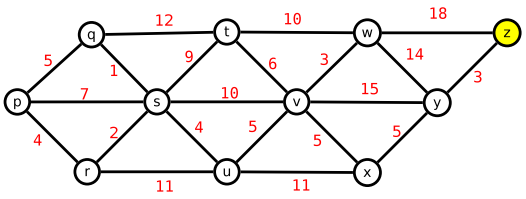

Figure 1: Example graph

2. Reason what is the run time complexity of Dijkstra's algorithm if the set '`Remaining`' is implemented as a linked-list? What is the run time complexity if, instead, the set '`Remaining`' is implemented using a **min-heap**?[1]

[1]For the heap data structure, refer Week 3 preparatory slides/FIT1008 lecture material.

3. Revise Breadth-First Search (BFS) algorithm described in Week 8. Using the graph layout in Figure 1 **and treating the graph to be unweighted (i.e., ignore the speficied edge weights)**, find the shortest paths from vertex z to every other vertex in the graph. Reason the time-complexity of your algorithm.

*** 4.** Graphs are often used as preferred representations to understand the relationships between various objects/entities/concepts. In this exercise we will try to capture the various concepts we have learnt so far into a graph – it would be informative for all of us to understand the relationships between various things we have studied.

We are in Week 9 of this unit, and have covered many concepts over 8 lectures coming into this week: **`L1, L2, L3, L4, ... L8`**. Additionally we have also considered three *assignment* exercises in weeks 4, 6 and 8; denote these as **`A1, A2, A3`**.

Draw an (undirected) graph with $8(lectures) + 3(assignments) = 11$ vertices. Any two vertices are said to be adjacent [2] if there is (some) conceptual relationship/link between the two corresponding lectures/assignments. For example, Lists can be used to represent graphs, hence there is an edge between vertices **`L1`** (where is discussed) and **`L8`** (where graphs were introduced). Another example, the **`mincost`** binary search tree task in assignment 2 uses a Dynamic Programming problem solving strategy, and this problem solving strategy was introduced in the lecture **`L4`**. Therefore, one could draw an edge between **`A2`**−**`L4`**.

Your task in this exercise is to make these conceptual connections between the 11 vertices and represent this (FIT2004 concepts) graph as an **adjacency list**.

As we enter into the last quarter of this unit, this exercise will help you contextualize the *web of interactions* between various things we have covered over the past 8 weeks. Note, there is no one correct answer, as the (subjective) criteria one uses defines how the vertices end up getting connected. However, this will be a **useful** exercise as it will help you explore the '**`Big Picture`**' of what we learnt (or did not) in this unit so far. **Please do this during your self-study**.

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

[2] Two vertices are adjacent when they have an edge between them