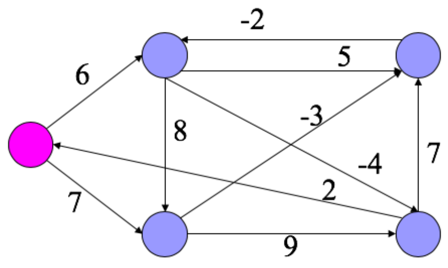
3805ICT Advanced Algorithms – Assignment 1 (100 Marks)

Note:

a) This assignment must be done individually.
b) The programming language to be used is C++ but you may use Python to generate graphs for your reports.
c) For each question requiring a C++ program you must document the algorithm and show any test cases you used. Only submit a single Word document containing the documentation for all questions.
d) The submission time and date are as specified in the Course Profile and the submission method will be communicated during semester.

1. (**20 Marks**) A common geometric problem is, given a set of N lines, how many intersect. The brute force algorithm is O(N^2). Design an O(NlogN) algorithm and implement it in C++ for the case where there are only vertical or horizontal lines. Derive and demonstrate the efficiency of your algorithm.

2. (**10 Marks**) Write a C++ program that uses the Bellman-Ford algorithm to find the shortest paths from the pink node to all other nodes:



3. (**15 Marks**) Most graph computing algorithms assume that the adjacency matrix and adjacency lists can be stored in computer memory so the following 2 operations will be fast:

    - Is vertex *v* connected to vertex *u*?
    - Produce a list of all vertices connected to *v*.

    However, the advent of very large graphs (> 50,000,000 vertices and > 100,000,00) prevents the memory storage of the adjacency matrix and standard adjacency lists for these graphs. Design and implement in C++ a data structure for storing such graphs that is able to effectively perform the 2 operations listed above. Demonstrate the efficiency of your data structure.

4. (**10 Marks**) Design and implement in C++ a data structure for storing unordered lists of integers that:
    a. Can store integers in the range 0 .. n where n is some upper bound.
    b. Duplicate integers are not allowed in the list.
    c. Is O(1) for add, delete, test for being in the list and iterating through the list.

d. Is O(k) (where k is the number of integers in the list) for clearing the list.

5. (**15 Marks**) The goal of a ladder-gram is to transform a source word into the target word on the bottom rung in the least number of steps. During each step, you must replace one letter in the previous word so that a new word is formed, but without changing the positions of the other letters. All words must exist in the supplied dictionary (dictionary.txt). For example, we can achieve the alchemist's dream of changing LEAD to GOLD in 3 steps or HIDE to SEEK in 6 steps. Minimise the number of steps.

6. (**30 Marks**) The file dictionary.txt contains one word per line. Subsets of these words can ordered such that, with the exception of the first word, the second and third letter of each word is identical to the third last and second last of the preceding word. Words may only be used once within a sequence.

   Design algorithms and implement C++ programs that find such sequences separately for words of length 4 through to words of length 15 characters. Target sequence lengths and CPU times in seconds are shown in the following tables. You may be able to find longer sequences.

Results for C++ Implementation

Word Length	Num. Words	Seq. Length	CPU Found	CPU Total	Word Length	Num. Words	Seq. Length	CPU Found	CPU Total
4	3,862	92	0.02	0.20	10	12,308	2,412	95.46	112.55
5	8,548	3,122	65.46	70.75	11	7,850	1,578	41.24	41.55
6	14,383	5,436	61.80	200.20	12	5,194	917	13.28	16.38
7	21,730	8,398	233.15	384.28	13	3,275	503	0.88	5.26
8	26,447	10,304	422.09	548.45	14	1,775	253	1.35	1.40
9	18,844	5,223	204.81	265.65	15	954	98	0.20	0.30

   Analyse in detail the problem and the performance of each of your algorithms. You must produce a detailed Latex/Word paper containing an Introduction, Algorithm Description, Experimental Results and Comparisons and a Conclusion.