Assignment 3 (15% of total marks)

Due date: 31st August 2021, Tuesday by 23:59 (**Absolutely no further extension** ☺)

Scope:

The tasks in this exercise consist of activities in the areas of **Data Structures and Algorithms, in particular, algorithm design strategies – Greedy Algorithm, Branch and Bound, backtracking and Divide and Conquer**. The exercises cover the topics discussed in topics 5, 6, and 7.

Marks:

Total mark: 100
Weightage: 15% of total subject mark

Assessment criteria:

Marks will be awarded for:

- Correct,
- Comprehensive, and
- Appropriate

application of the materials covered in this subject.

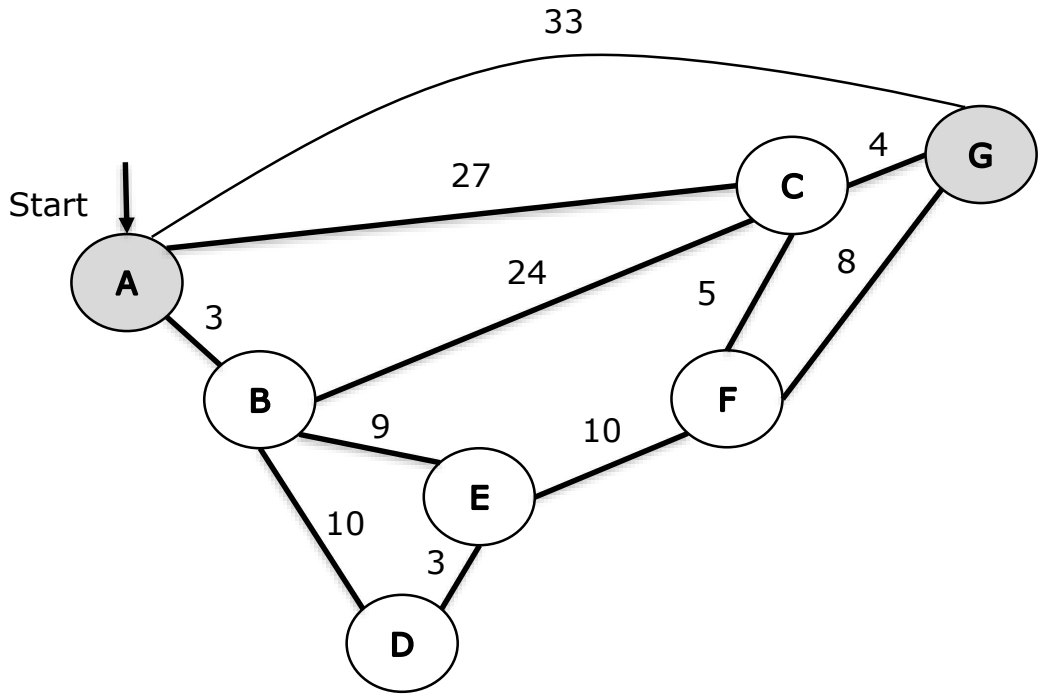
Assignment Specification:

Part A: (40.0 marks)

Question 1 (20.0 marks)

Consider the following network. With the indicated link costs, use Dijkstra's shortest-path algorithm to compute the shortest path from A to all network nodes.

a) Show how the algorithm works by computing a table like the one discussed in class. **(15.0 marks)**
b) Show all the paths from A to all other network nodes. **(5.0 marks)**



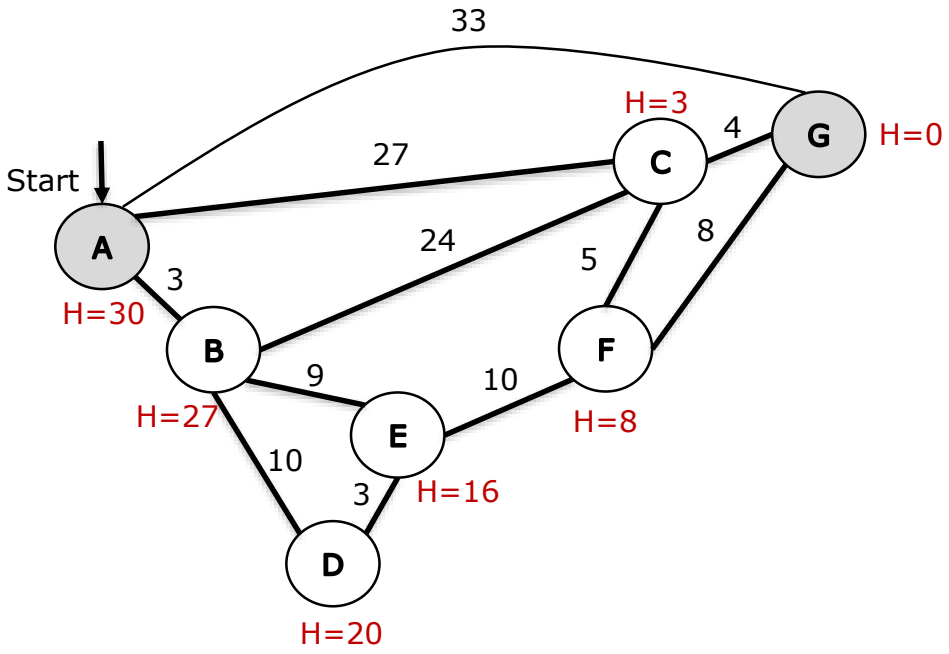
Question 2 (20.0 marks)

Consider the following search problem, represented as a graph. Each node is label by a capital letter and the value of a heuristic function is shown in maroon. Each edge is labelled by the cost to traverse that edge. The start state is 'A' and the only goal state is 'G'. Perform the A* search to find the shortest path from node A to node G.

i. Is the heuristics specified in the problem (shown below) admissible? Justify your answer. If the heuristics is admissible, proceed to answer part (ii). If the heuristics is not admissible, correct it with a sensible value of your choice and proceed to answer part (ii). **(5.0 mark)**

ii. Perform the A* search to find the shortest path from the start state (S) to the goal state (G).

(15.0 marks)



Numbers in maroon are the heuristics for the respective nodes.
Numbers in black are the weight (estimated costs.)

Part B: (60.0 marks)

We have discussed greedy algorithm during lectures. A greedy algorithm is an algorithm that recursively construct a set of objects from the smallest possible constituent parts. At each one of the iterations, the algorithm takes the best that it can get right now, without regards for future consequences. The algorithm hopes that by choosing a local optimum at each one of the iterations, it can end up at a global optimum.

In this assignment, you will **write** a program to schedule final examination for the examination department so that no student has two examinations at the same time. The goal of this assignment is to expose you to the implementation of greedy algorithms that solves a problem with constraints. You will use a greedy algorithm to determine an assignment of classes to examination slots (schedules) such that:

1. No student, enrolled in two subjects, is assigned to the same examination slot (schedule.)
2. Any attempt to combine two slots into one would violate rule 1.

Input to the program will consist of the name of a data file. This file will contain the following data:

- The number of students enrolled in the current semester
- Repeated rows of the following:
    - Name of the student and the total number of subjects enrolled
    - The subject code the student is enrolled in.

A sample of an input file is as follow:

```
3
Melissa, 4
CSCI203
CSCI235
CSCI222
CSCI205
Bernard, 4
CSCI213
CSCI222
CSCI204
CSCI203
Terrence, 4
CSCI212
CSCI203
CSCI235
CSCI213
```

The output of the program should be a list of time slots with the subjects whose final examination will be given at that slot and the total number of students taking the final examination in that slot. One possible output is as follow:

Slot 1: CSCI212, CSCI222 3
Slot 2: CSCI204, CSCI235 3
Slot 3: CSCI205, CSCI213 3
Slot 4: CSCI203 3

The algorithm:

- Read the enrolment information from the input file. As the records are read, build an adjacency matrix representing the relationships among the students and the subject the students enrol in. You should notice that this adjacency matrix is a graph representing the relationships. Each node of the graph will be a subject taken by at least one student in the current semester. An edge between two nodes will mean there is at least one student taking both subjects. The weight of an edge could be the number of students enrols with both subjects.
- Your aim in solving this problem is to construct a *maximal independent set* in the graph. This can be achieved by finding an examination schedule satisfying the two constraints mentioned earlier, as follow:
    (i) Construct a candidate list of subjects.
    (ii) Order the subjects in descending order by total number of inconnectivity.
    (iii) Starting from the subject with the highest number of inconnectivity, create a slot.
    (iv) Search for a subject to which it is not connected. If you find one, add the subject to the same slot and remove it from the candidate list.
    (v) Next, try to find another subject that is not connected to any of those already in the time slot. Similarly, if you find one, add the subject to the same slot and remove it from the candidate list. Continue to do so until there is no more un-connected subject can be found.
    (vi) Accumulate the total number of students enrolled from the adjacency matrix. (How can you do that? Give it a thought.)
    (vii) Repeat steps (iii) through (vi) until all the subjects are removed from the candidate list.

- Note that no pair of time slots can be combined without creating a time conflict with a student. Also note that depending on how you select a subject from the candidate list, there may be different schedule can be formed. Any schedule satisfying the two-mentioned constrained will be acceptable.

You are allowed to implement your algorithm for Part B using C++, Java, or Python.

Submissions

This assignment is due by 31st August 2021 at 12:00 am (23:59) Singapore time.

- For Part A, type your answer for each question in a MS Word or equivalent document format and save it in a pdf formatted file, name your file as YourName-A3-SolPartA.pdf
- For Part B, the name of your program should be greedyAlgo.cpp, greedyAlgo.java, or greedyAlgo.py depending on the programming language that you use to develop your program. Execute your program with the test data provided, that is, A3Data.txt and **screen capture** your output. Next, zip your source code, libraries, readme.txt together with your screen capture and name your file as YourName-A3-SolPartB.zip.
- Zip together YourName-A3-SolPartA.pdf and YourName-A3-SolPartB.zip and name your file as YourName-A3.zip. Do not use your own filename.
- All assignments that do not satisfy the submission requirements listed above will not be evaluated and will be returned to the students with 0 marks.

Submit the files **YourName-A3.zip** through Moodle in the following way:

1) Access Moodle at **http://moodle.uowplatform.edu.au/**
2) To login use a Login link located in the right upper corner the Web page or in the middle of the bottom of the Web page
3) When successfully logged in, select a site **CSCI203 (SP321) Algorithms and Data Structures**
4) Scroll down to a section Submissions of Assignments
5) Click at Submit your Assignment 3 here link.
6) Click at a button Add Submission
7) Move a file, for example, **YourName-A3.zip** into an area. You can drag and drop files here to add them. You can also use a link *Add…*
8) Click at a button Save changes,
9) Click at a button Submit assignment,
10) Click at the checkbox with a text attached: By checking this box, I confirm that this submission is my own work, … in order to confirm authorship of your submission,
11) Click at a button Continue.

A policy regarding late submissions is included in the subject outline.

Only one submission per student is accepted.

Assignment 3 is an individual assignment and it is expected that all its tasks will be solved individually without any cooperation with the other students. Plagiarism is treated seriously. Students involved will likely receive zero. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or over e-mail.

********** End of Assignment Specification **********