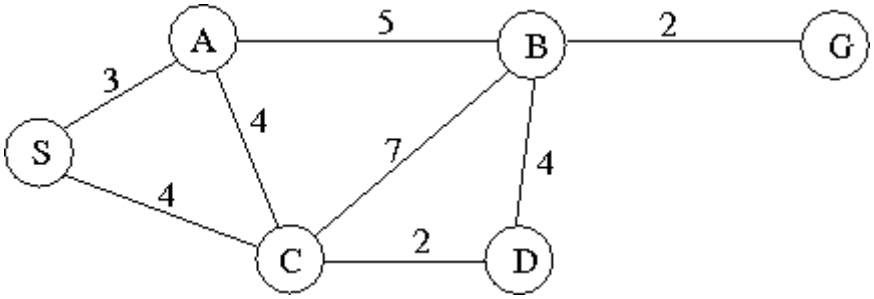
Monash University Faculty of Information Technology
FIT3080 Week 3 Lab 2: Tree and Graph Search

Exercise 1: *Search Algorithms*

The diagram below depicts the cost of travelling between cities.



(a) Draw the search *tree* generated by the basic Tree Search algorithm to reach the goal G starting from S. Use Uniform Cost Search (the cost incurred so far) to choose a node from the frontier. For instance, after expanding S, your frontier has nodes A (cost 3) and C (cost 4), so you should choose node A. **Indicate clearly the frontier after each execution of Step 4 of the algorithm.** What is the generated path? What is its cost?

(b) Draw the search *graph* generated by the Graph Search algorithm to reach the goal G starting from S. As above, use Uniform Cost Search to choose a node from the frontier. **Indicate clearly the frontier and explored set after each execution of Step 5 of the algorithm.** What is the generated path? What is its cost?

Exercise 2: *Problem Formulation*

You have a 9×9 grid of squares, each of which can be colored red or blue. The grid is initially colored all blue, but you can change the color of any square any number of times. Imagining the grid divided into nine 3×3 sub-squares, you want each sub-square to be all one color but neighboring sub-squares to be different colors.

(a) Formulate this problem in the straightforward way. Compute the size of the state space.

(b) You need color a square only once. Reformulate, and compute the size of the state space. Would breadth-first graph search perform faster on this problem than on the one in (a)? How about iterative deepening tree search?

(c) Given the goal, we need consider only colorings where each sub-square is uniformly colored. Reformulate the problem and compute the size of the state space.

(d) How many solutions does this problem have?

(e) Parts (b) and (c) successively abstracted the original problem (a). Can you give a translation from solutions in problem (c) into solutions in problem (b), and from solutions in problem (b) into solutions for problem (a)?

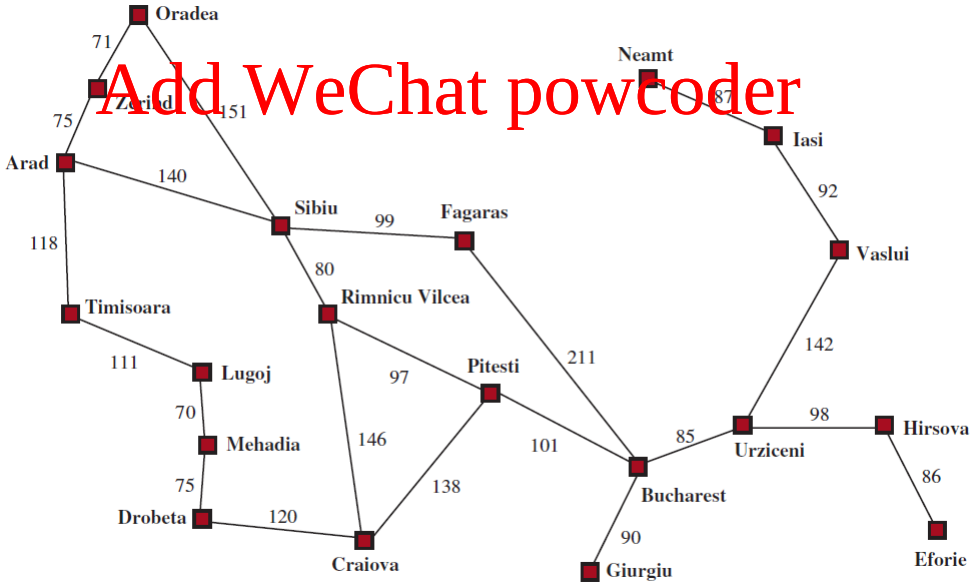
Exercise 3: ***Irrevocable search***

At the end of this week's lecture we heard about **tentative** search strategies and **irrevocable** search strategies. We can describe these approaches in the following way:

- *Tentative algorithms* proceed by growing a tree of all possible solution paths, rooted at the initial state. The search is free to grow any of these candidates, by expanding whichever one appears most attractive according to its queuing function.
- *Irrevocable algorithms* proceed by growing only a small set of possible solution paths, often just one. One way to implement this idea, known as beam search, proceeds as follows:

    1. At each expansion step, **remove each node from the OPEN list**
    2. If any removed node is the goal, return that node as the solution.
    3. Else, **expand each node** and add its successors to a single list
    4. **Rank all the successors** according to some queuing function.
    5. Add **the top k successors** to the OPEN list (the rest are discarded).

(a) What is the resulting path when we solve the Romania example (moving from Arad to Bucharest) using beam search with k=1? Refer to the map below and use the distance between cities (edge labels) as your ranking function.



(b) What happens as we increase k? What are the main advantages of this algorithm?

(c) Give a theoretical analysis of the beam search algorithm.

(d) <OPTIONAL / HOMEWORK> How could we modify this algorithm to achieve completeness and optimality? Would it still be considered irrevocable?