Project 2
Artificial Intelligence

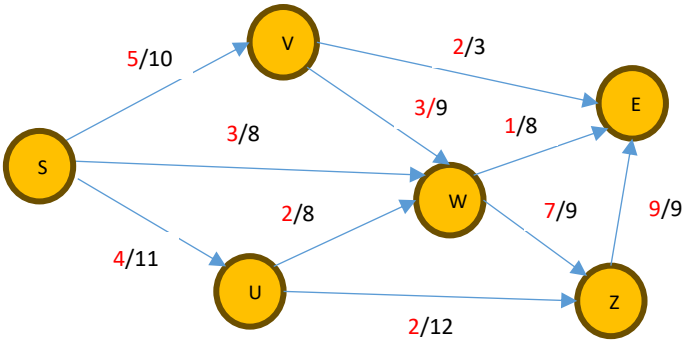
Fall 2024

[Solutions to this assignment must be submitted via CANVAS prior to midnight on the due date. These dates and times vary depending on the milestone to be submitted.

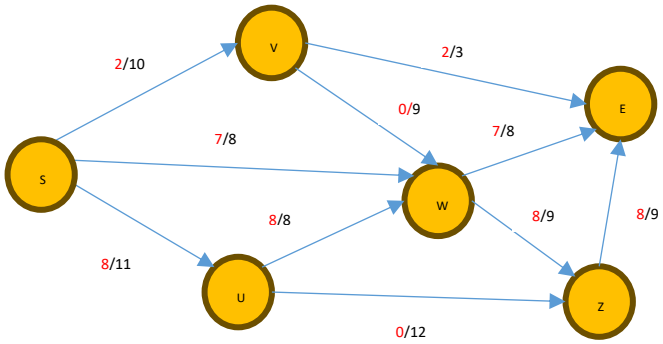
This project may be undertaken in pairs or individually. If working in a pair, state the ***names*** of the two people undertaking the project and the ***contributions*** that each has made. Only ONE submission should be made per group.

You will make use of the Networkx environment that you should be familiar with from Project 1.

Purpose: To gain a thorough understanding of the working of an agent that maximizes flow through a network. Maximizing network flow ensures that the network's resources are fully utilized according to their capacities, and this reduces time to move commodities through the network. Numerous real-world applications benefit from maximizing flow such as maximizing data throughput in a computer network, maximizing the flow of oil through a network of pipelines, etc. Figures 1 A and B below provide a concrete example of a network flow problem. The source node is S, and the sink (destination) node is E.



Flow into E before optimization = 2+1+9=12



Flow into E after optimization with SA = 2+7+8=17

Figures 1 A and B

Each link has a *capacity* (e.g., maximum number of barrels of oil that can flow per hour) and an *actual* flow. The actual flow in the above figure for each edge is indicated in red while the capacity is given in black. Figure 1 A represents a sub optimal flow as we can see that the flow across most of the edges do not utilize their full capacity. The solution in Figure 1 B better utilizes the capacity of the network and is clearly a better solution.

One constraint that must be enforced is the *law of conservation*: the sum of the outflows from a node cannot exceed the sum of inflows into that same node. Thus, for example, the sum of the inflows into node w must be equal to the sum of the outflows from node w.

In practice, on a large network it may take too long to obtain the optimal solution and therefore we will use a Simulated Annealing (SA) algorithm (see Tutorial 4 for details) to construct a solution. We saw in the lectures that SA can return high quality solutions that avoid getting stuck in local minima and return solutions that are as close to the global (optimal) solution as possible. The solution that you generate **must** also satisfy the conservation law across all nodes in the network.

SA requires an initial solution to start off with and this solution can simply set all flows across the network to 0. Edge capacities can be assigned at random to be an integer between 1 and 10.

Once an initial solution is in place, SA needs to generate in each iteration, *successors* of the solution that existed in the previous iteration until the algorithm ends when the *temperature* parameter reduces to 0.

Jot down any questions/doubts that you may have and feel free to ask me questions in class or in person. Together with your partner, ***work out a strategy before you start coding the solution in Python.*** Given the limited timeframe for the project, some simplifications have been applied.

Environment Description: The environment is a network that is specified in the form of a graph structure as represented in Figures 1 A and B (simply a sample, not the entire network). The graph that you will use to represent the network will consist of 30 nodes, with two different average connectivity values (see below).

Your task in this project is to implement the following requirements. The project has two milestones, with Part A representing design of algorithms and Part B consisting of algorithm implementation and experimentation, just as with Project 1. The milestones have different deadlines as shown below.

Part A

Produce pseudo code versions of algorithms needed to construct:

a) A heuristic function suitable for a simulated annealing solution for the network resource problem.
b) A successor function – a successor function specifies how a node is expanded into its child nodes. Your successor function should contain code that generates paths from the *sink* node to the *source* node.
c) The driver algorithm (i.e., the Simulated Annealing algorithm) that takes *source*, *sink* nodes, *initial temperature T* and α (the cooling schedule controller) as parameters and calls the two functions mentioned in a) and b) above.

Due at midnight on Sunday 6 October **(35 marks)**

Part B

Using the pseudo code that you produced in Part A above to implement a Python program to maximize the flow. For all of the requirements below use 0 and 29 for source and sink nodes respectively. Your Python program should meet the following requirements:

R1:

Visualize your solution for the graph given in Figure 1, showing the *capacities* and the *actual flow* along each edge that is returned by executing your SA algorithm. This is an important test case for your code

and will require you to hard code your graph G. Run your SA code for the graph in Figure 1 A and present its answer to the total flow into the sink node E. Verify that the value returned is approximately the same value given in Figure 1 B (due to randomness in the operation of SA your solution may be different numerically but it should be close). **(30 marks)**

R2:

Run your SA code developed in Python on a graph G containing 30 nodes and an average connectivity of 3 (connectivity parameter of 0.1). In your graph G, *randomly generate capacity (C) values across edges in the range 1 to 10*. Work out the total flow value tf into node *T* and compare it with the *total flow* value tf_net returned by the Edmonds Karp algorithm supported by Networkx. The Edmonds Karp method is an optimal algorithm that guarantees the optimal flow value for any given graph. Present both values, tf and tf_net. (**10 marks)**

R3:

This requirement has 3 sub parts:

a. Run your SA code with 30 different random graphs G (having 30 nodes and an average connectivity of 3). Work out the average value of the total flow into sink node T. Present the total flow tf_net_avg(100) with initial temperature set to 100. **(10 marks)**
b. Now re-run your SA code on 30 different random graphs, each of size 30 nodes once again and with the same connectivity, but this time with an initial temperature T of 1000. Compute and present the average flow tf_net_av(1000). **(10 marks)**
c. Do you notice a significant difference between tf_net_avg(100) and tf_net_avg(1000)? If so, why do you think this occurs? On the other hand if no significant difference occurs, why do you think this is the case? **(5 marks)**

Due at midnight on Sunday 20 October

Notes:

1. Produce ONE pdf document that contains pseudo code, actual Python code and another pdf that contains answers to the questions. Do NOT bury answers to questions as comments in your code. If you are unsure how to produce an original pdf from your Google Co-lab notebook, refer to this tutorial on Youtube: Bing Videos. Do NOT simply use the print option as this will result in an image. If you submit an image version of a pdf it will not be graded.
2. **If you finish Part A before its deadline, start working on Part B immediately afterwards.**

End of project specification