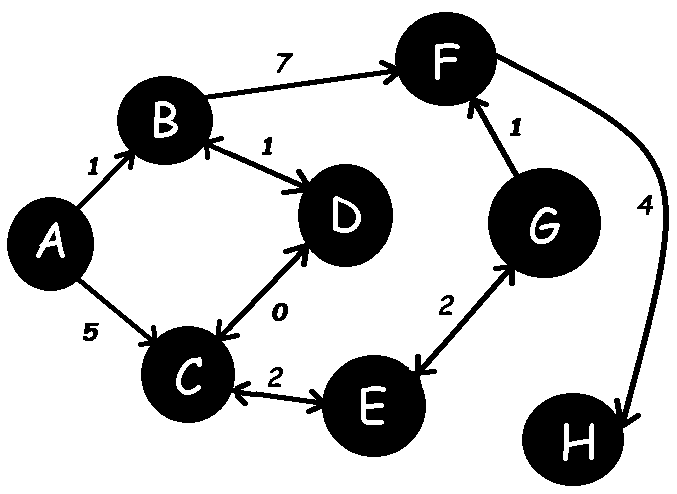
# Final Exam

1. Create a new project based on your answer to Unit 3 Test q1b

* prompts the user for a string
* prints how many of each letter of the alphabet are in the string (case-insensitive, ie. A=a)

Rewrite the code to be better data-oriented and prove via the profiler that the performance has been improved.

1. Represent the following digraph as an adjacency matrix and an adjacency list:





1. Using the digraph of #2, implement the following game along the same lines as Don't Die, but with the added element that each room can be in 1 of 3 states: ice, liquid or gas:

* every second, each node transitions between ice, liquid and gas, such that it changes from ice -> liquid -> gas -> liquid -> ice -> liquid -> etc. Imagine the rooms are constantly heating and freezing.
* the initial state of each node is:

A = ice, B = liquid, C = gas

D = ice, E = liquid, F = gas

G = ice, H = liquid

* the player starts at node A with 5 HP (playerHP) and can only see the rooms with more than (playerHP - 1), since when they move to the next room they lose the edge cost, and if they reach HP = 0, then they die
* the player can wager for more HP while in a room using the trivia api
* the player also has a state of ice, liquid or gas and can change their state for the cost of 1 HP, but they can only change from ice->liquid, liquid->gas, gas->liquid, or liquid->ice
* in order to move to the next room, the player must be in the same state as the room they are entering, otherwise they stay in the same room, but do NOT lose the edge cost
* your program should start by describing that there are 8 rooms in the labyrinth, the initial states of each room and their rules of transition so that the player has a chance of matching the room state upon each move (by keeping track of the time)
* when describing the available exits from each room, you should include the room letters so that the player can possibly determine the state of each neighboring room
* upon reaching room H, the player has escaped the labyrinth and the game should report how many moves and how many seconds it took the player to win

1. Create a console application which defines and uses a singleton class which includes methods to load and save the player's settings using the following JSON format to a hard drive file, serializing and deserializing the structure with the Newtonsoft JSON package:

{"player\_name":"dschuh","level":4,"hp":99,"inventory":["spear","water bottle","hammer","sonic screwdriver","cannonball","wood","Scooby snack","Hydra","poisonous potato","dead bush","repair powder"],"license\_key":"DFGU99-1454"}