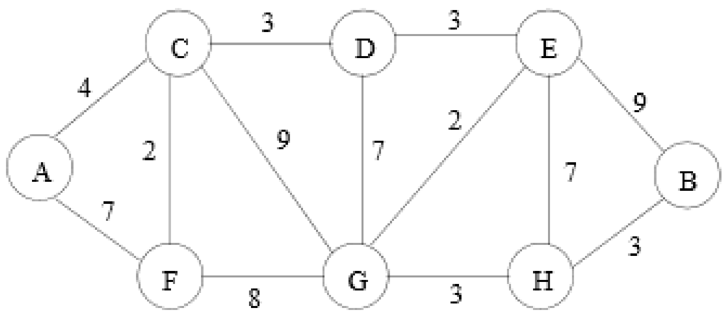
# CS 325 – HW 4

**Problem 1.** *(10 points)* A region contains a number of towns connected by roads. In the graph below, each town is labeled with a circle and each road is labeled with the average number of hours required for a truck to travel between the cities. Suppose that you work for a large retailer that has decided to place a distribution center in town G. (While this problem is small, you want to devise a method to solve much larger problems).



(a) Which algorithm would you recommend be used to find the fastest route from the distribution center to each of the towns? Demonstrate how it would work on the example above if the distribution center is placed at town G. Show the resulting routes.

(b) Suppose one "optimal" location (maybe instead of town G) must be selected for the distribution center such that it minimizes the time travelled to the farthest town. Devise an algorithm to solve this problem given an arbitrary road map. Analyze the time complexity of your algorithm when there are t possible towns (locations) for the distribution center and r possible roads.

(c) In the above graph which is the "optimal" town to locate the distribution center?

(d) Now suppose you can build two distribution centers. Where would you place them to minimize the time travelled to the farthest town? Describe an algorithm to solve this problem given an arbitrary road map.

(e) In the above graph what are the "optimal" towns to place the two distribution centers?

**Problem 2.** *(20 points)* **Euclidean MST Implementation**
Implement an algorithm in the language of your choice (C, C++ or Python) that determines the weight of a MST in a graph $G=(V,E)$ where the vertices are points in the plane and the weight of the edge between each pair of points is the Euclidean distance between those points. To avoid floating point precision problems in computing the square-root, we will always round the distance to the nearest integer. For all $u, v \in V$, $u = (x_1,y_1)$, $v = (x_2,y_2)$ and

$$w(u,v) = d(u,v) = nearestint \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

For example, if $u = (0, 0)$, $v = (1, 3)$,

$$d(u,v) = \text{nearestint}\left(\sqrt{(0-1)^2 + (0-3)^2}\right)$$

$$= \text{nearestint}\left(\sqrt{(-1)^2 + (-3)^2}\right)$$
$$= \text{nearestint}(\sqrt{1+9})$$
$$= \text{nearestint}(\sqrt{10})$$
$$= \text{nearestint}(3.1622 \ldots)$$
$$= 3$$

Assume that the graph $G=(V,E)$ is complete so that there is an edge between ever two vertices. Your program should be named mst.py, mst.c or mst.cpp. Your program should read information about the graph from a file named graph.txt. The first line in graph.txt is the number of tests cases (k ≤ 10) followed by the number of vertices in the test case (n ≤ 100) and the coordinates of the points (vertices) in that test case.

An example of graph.txt is shown below:

```
2
3
0 1
2 1
2 5
5
0 0
0 4
4 4
4 8
1 1
```

The output should be displayed to the terminal and give the test case number followed by the weight of the MST.

```
Test case 1: MST weight 6
Test case 2: MST weight 12
```

Submit to Canvas:

- A verbal describe of your algorithm and data structures
- The pseudo-code
- Theoretical running time

Submit to TEACH in a zip file

- mst.c, mst.cpp or mst.py
- graph.txt
- solution.txt
- the script file HW4_**.sh