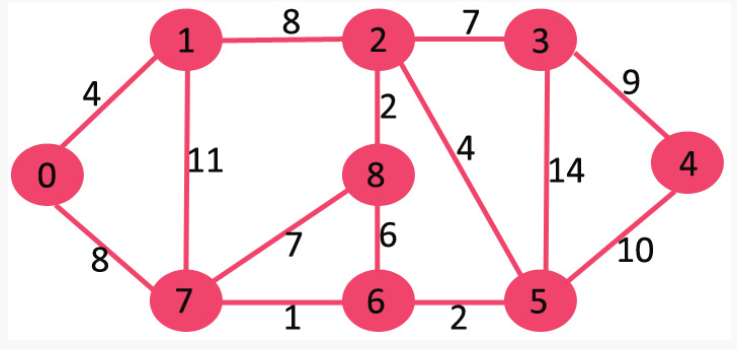
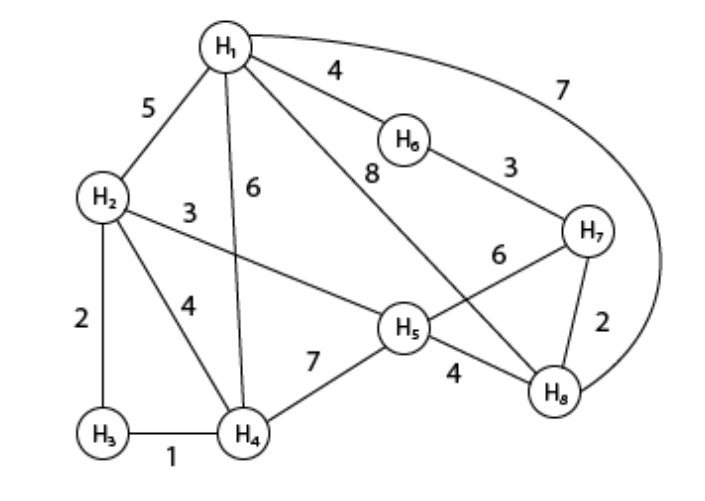
Assignment 2: 10 Marks

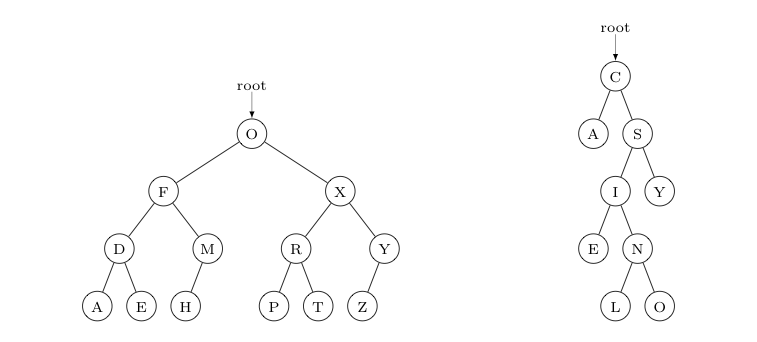
1. Given a graph and a source vertex (node 0) in the graph, find the **shortest paths** from the source to all vertices in the given graph. Which algorithm is best suited?



1. Using Hashing write algorithm
2. To find the duplicate element in a limited range array
3. To Efficiently calculate the frequency of all elements present in a limited range array
4. A newspaper agent daily drops the newspaper to the area assigned in such a manner that he has to cover all the houses in the respective area with minimum travel cost. Compute the minimum travel cost. Which algorithm is best suited?



1. Consider the following two trees:



Is the tree shown on the LEFT in symmetric order?

List the keys seen in a post-order traversal of the tree on the RIGHT.

List the keys seen in a pre-order traversal of the tree on the RIGHT.

List the keys seen in an in-order traversal of the tree on the RIGHT.

In the binary search tree on the RIGHT -- with keys corresponding to the letters shown -- which of these keys are compared to "P" when one attempts to retrieve the value associated with "P"? Write these keys in the order they are compared.

1. Implement a Binary Search Tree (BST) based index for indexing/searching actors or movies in the [IMDb dataset](http://www.imdb.com/interfaces) with the following characteristics:

Your implementation should allow the user to quickly find information associated with an actor, or a movie, based on the actor's or movie's name (e.g., "Arnold Schwarzenegger"), or a prefix of a name (e.g. "Arnold Schwarz\*").

The search should be *case insensitive* (e.g., "arnold schwarz\*" should also work.)

The search key is the short or simplified name of a movie or actor.

The information associated with the key should be of type [MovieInfo](https://mathcenter.oxford.emory.edu/site/cs171/probSetBst/MovieInfo.java).

The BST index should be implemented as a stand-alone class named BSTIndex, having at least the public methods listed below (you may add as many additional private helper methods as needed). You will need to define a Node class within the BSTIndex class that contains 3 fields: val (of MovieInfo type), and left & right references to the children nodes. As the key for comparison with other nodes (i.e., the short name) is already stored in the MovieInfo object, make a key() method in the node class to access this key rather than storing duplicate information.

* 1. public BSTIndex() : a constructor to initialize the BST. The data element should be an object of type MovieInfo as described above.
  2. public MovieInfo findExact(String key) : returns the data MovieInfo element where the shortName matches the key exactly (although possibly with different capitalization).
  3. public MovieInfo findPrefix (String prefix) : returns a data element MovieInfo where the shortName starts with the prefix (and possibly has different capitalization). If there is more than one match, you can return the MovieInfo for any of these -- your choice.
  4. public void insert(MovieInfo data) : insert the given data element into the proper place in the BST structure.