Part 1

Given the following graph below,

3) Use Kruskal’s algorithm to find the minimum spanning tree of G.

4) Use Prim’s algorithm to find the minimum spanning tree of G.

Part 2 - Intro to recurrences

Remember the search problem from our very first assignment. You designed and wrote an iterative algorithm, which searches for an item in an array with $\Theta(n)$ worst case running time complexity. If you have no other information about your array, then linear search is the only thing you can do. However, if you know that your array is sorted, you can search in a different way, which would improve your running time complexity.

The binary search algorithm is one of the simplest divide conquer algorithms. We use it to search in a sorted array. Here is a recursive version of it:

```python
BinarySearch (A, val, minIndex, maxIndex)
    if(minIndex > maxIndex)
        return -1
    Let midIndex = floor((maxIndex-minIndex) / 2 ) + 1
    if( A[midIndex] == val ) then
        return midIndex;
    end if
    if( A[midIndex] > val) then
        return BinarySearch (A, val, minIndex, midIndex-1)
    else
        return BinarySearch (A, val, midIndex+1, maxIndex)
    end if
```
1) For the array below, trace the algorithm above, and report how many comparisons will be made when searching for 7.

[4, 7, 11, 24, 33, 34, 39, 45, 49, 54, 102, 307, 330, 353, 370]

2) What is the worst case running time complexity of Binary Search?

Try after class Consider the following mystery algorithm below,

```python
mystery (n)
    if (n <= 1)
        print(": ")
    else
        print((n % 2) + " ")
        mystery (n/2)
        print(n + " ")
    endIf
endFunction
```

1) Write a recurrence for the running time of the algorithm.

2) Solve the recurrence using recurrence trees.