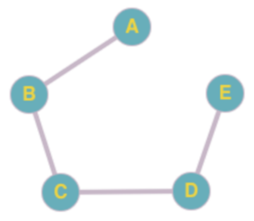
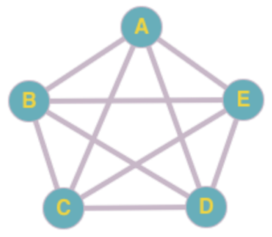
Jake Wiseberg
Writing Assignment 4

1. Given an undirected graph with n nodes.
    a. If graph had smallest number of edges:
        i. The minimum number of edges with respect to n, in a connected graph, will have n-1 edges, this will ensure that there is a direct or indirect path between any 2 nodes.
        ii. With n=5 nodes there will be 4 edges to make sure the graph is connected, there is a path between any 2 nodes, direct or indirect.



        iii. This graph is called a spanning tree, in which the graph is connected, contain n nodes, and n-1 edges.
    b. If graph has the largest number of edges:
        i. The graph will have n(n-1)/2, this is accounting for that each node will have n-1 connections going out, this applies for n/2 nodes due to overlapping connections, so the graph with the largest number of edges will have n(n-1)/2 edges.
        ii.



        iii. Graphs with an edge between every pair of vertices are called complete graph.
2. Quicksort algorithm
    a. First element is used as pivot
        i. O(n^2) for sorted input, worst case runtime for quicksort
        ii. O(n^2) for reverse-sorted input, worst case runtime for quicksort
    b. Middle element is used as pivot
        i. O(nlogn) for sorted input, average case runtime with middle element as pivot
        ii. O(nlogn) for reverse-sorted element, average case runtime with middle element as pivot.
3.

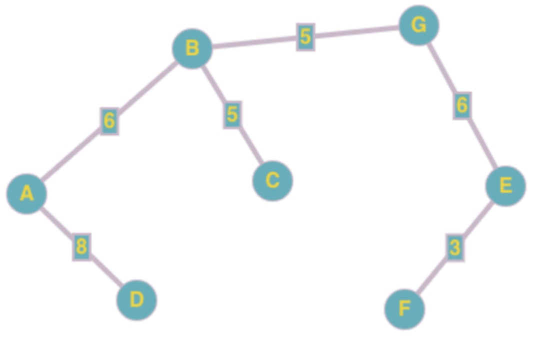
a. Using two iterators, one that starts at the beginning, one that starts at the end of the array, and iterating based on while (i<j), where i is the beginning and j is the end iterator. Nested within the while loop there will be 3 conditions. The first condition will check if the key for arr[i] is "Red" and if so it will iterate with i (i++). The second will check if the key for arr[j] is "Blue" and if so it will iterate in reverse (j--). The final condition will check and swap elements if needed. The final condition will check if both arr[i] is "Blue" and if arr[j] is "Red", and if so it'll swap the two. Rough code that does work for part b can be found on the last page of this document.

b. Below is the output for each iteration using the code on the last page of this doc

```
[ | Red,3 | Blue,5 | Blue,7 | Red,1 | Red,4 | Blue,2 | Red,6 | ]
[ | Red,3 | Red,6 | Blue,7 | Red,1 | Red,4 | Blue,2 | Blue,5 | ]
[ | Red,3 | Red,6 | Red,4 | Red,1 | Blue,7 | Blue,2 | Blue,5 | ]
```

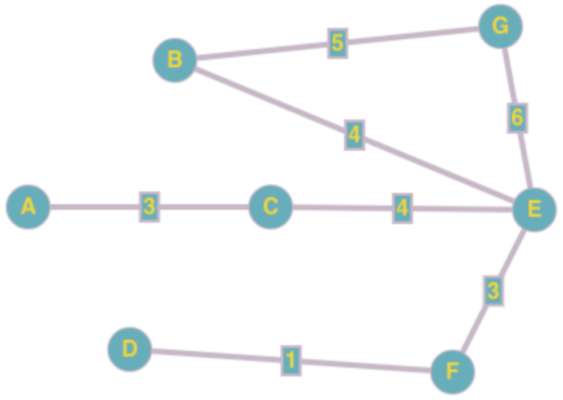
c. Using the algorithm described in part A, the runtime complexity will be O(N). This is because the algorithm will only iterate through the array once. By using two iterators that will only iterate when necessary the loop will break once the iterators meet.

4. Using the given undirected graph
   a. Dijkstra's starting at A
      i. A,B,D,G,C,E,F →



      ii. No, Dijkstra algorithm maintains estimates of the shortest paths from v to every vertex along with their associated costs, this only concerns 2 nodes and would not produce a minimum spanning tree.

   b. Prim's
      i. A,C,E,F,D,B,G →



      ii. Yes, the goal of Prim's algorithm is to return the set of edges that creates a minimum spanning tree for the given graph connecting all the nodes in the graph.

```
public class Colors {
  public static void main(String[] args) {
    RedBlue[] arr = new RedBlue[7];
    arr[0] = new RedBlue("Red", 3);
    arr[1] = new RedBlue("Blue", 5);
    arr[2] = new RedBlue("Blue", 7);
    arr[3] = new RedBlue("Red", 1);
    arr[4] = new RedBlue("Red", 4);
    arr[5] = new RedBlue("Blue", 2);
    arr[6] = new RedBlue("Red", 6);
    print(arr);
    sort(arr);
  }

  public static void sort(RedBlue[] arr) {
    int i=0;
    int j=arr.length-1;
    while (i<j) {
      if (arr[i].color.equals("Red"))
        i++;
      if (arr[j].color.equals("Blue"))
        j--;
      if (arr[i].color.equals("Blue") && arr[j].color.equals("Red")) {
        swap(arr,i,j);
        i++;
        j--;
      }
      print(arr);
    }
  }

  public static void print(RedBlue[] arr) {
    System.out.print("[ | ");
    for (RedBlue a : arr)
      System.out.print(a.color + "," + a.value + " | ");
    System.out.println("]");
  }

  public static void swap(RedBlue[] arr, int i, int j) {
    RedBlue temp = arr[j];
    arr[j] = arr[i];
    arr[i] = temp;
  }
}

class RedBlue {
  String color;
  int value;
  public RedBlue(String c, int v) {
    color = c;
    value = v;
  }
}
```