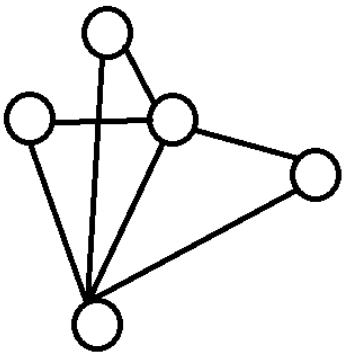
Mahitha Valluru

CS 4800 – Algorithms and Data

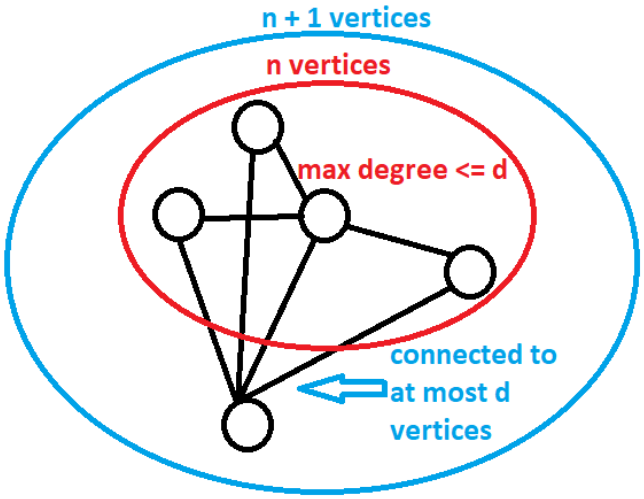
Homework 4

1.

a. Let us say we have a graph that looks like this:



This graph contains some vertices and some edges connecting those vertices to each other.
Now, we must prove that if the degree of every vertex is at most d, then the graph is d+1 colorable.
Let's make our graph a little easier to visualize so we can prove this:



When we take out the vertex at the bottom and only look at the graph in the red circle, we can say that there are some n number of vertices with a maximum degree of less than or equal to d. However, when we add the bottom vertex and

consider the graph in the blue circle, we know that there is $n + 1$ vertices and is connected to at most d vertices, so the degree is less than or equal to d.
This means that this vertex must have one additional color because it cannot have any of the colors of the d vertices it is connected to, which ultimately means that an additional vertex means the graph must require $d + 1$ colors.

b. Say a graph has k vertices. This means that the number of edges that this graph can have to connect all the vertices is $\binom{k}{2}$, which is $\frac{k^2-k}{2}$, which approximately equals k^2. We can write this as:

$$|E| \leq \binom{k}{2} \approx k^2$$

Now based on the problem, we know:

$$c_1 * n \leq |E|$$

Now, we can simplify this:

$$c_1 * n \leq k^2$$
$$\sqrt{c_1 * n} \leq k$$
$$\sqrt{c_1} * \sqrt{n} \leq k$$

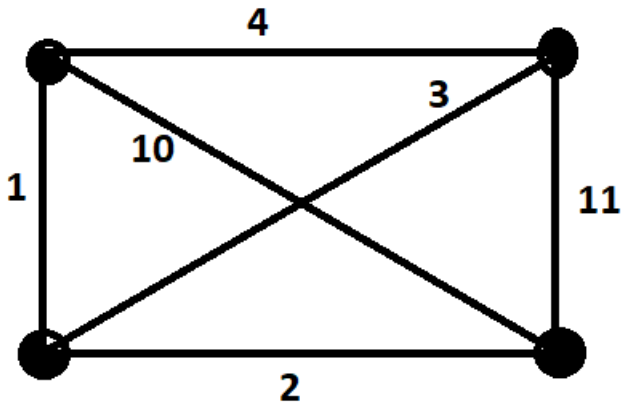
Which means: $c_2 = \sqrt{c_1}$
And finally: $c_2 * \sqrt{n} \leq k$
To answer the question, there exists a constant c_2 which is equal to $\sqrt{c_1}$.

2. Proof by induction on n, the number of vertices in a tree T.
Basis step: If $n = 1$ or $n = 2$, the center is the entire tree which is either a vertex or an edge.

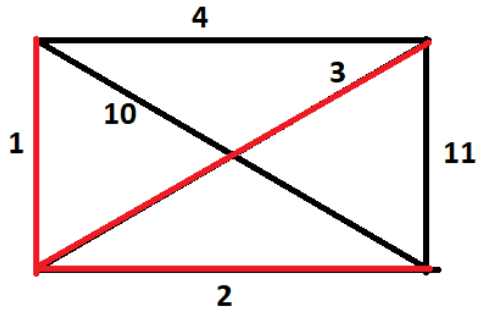
Induction hypothesis:

Let $n > 2$
Let T be a tree with n vertices.
Assume the center of every tree with less than n vertices is a vertex or an edge.
Form T' by deleting the leaves of T. Since the internal vertices on paths between leaves remain, T' has at least one vertex.
The eccentricity of a leaf in T is greater than that of its neighbor. Hence the vertices of minimal eccentricity in T are the same as the vertices of minimal eccentricity in T'. This implies that the center of T' is the same as the center of T.
By induction hypothesis, the center of T' is either a vertex or an edge.

3. This algorithm will not create the minimal spanning tree that we are looking for. Given a graph that looks like this:



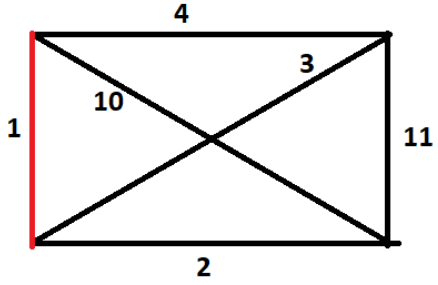
The Minimal Spanning Tree that we want to get will look like this, with the preferred edges in red:



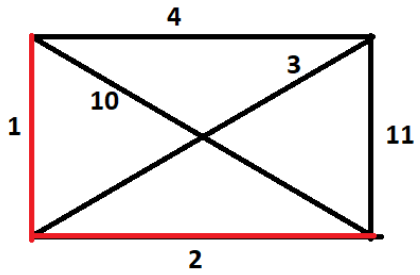
The weight of the MST equals: $1 + 2 + 3 = 6$

However, the following is what happens with the algorithm that is written in the homework.

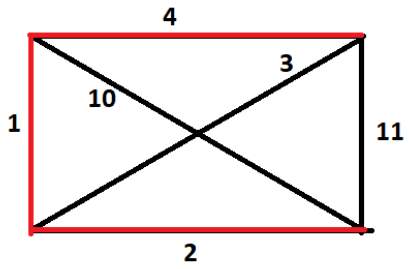
Step 1 – Choose and edge e_1 of minimum weight.



Step 2 – Choose another edge e_2 that has not been previously chosen, is a union of disjoint simple paths with the previously found edges, and has a weight that is as small as possible subject to the rules above.



Step 3 – Choose another edge e_3 that has not been previously chosen, is a union of disjoint simple paths with the previously found edges, and has a weight that is as small as possible subject to the rules above.

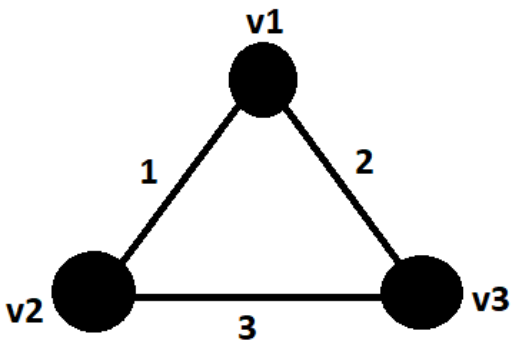


Now we have found our tree, but we need to know if it is of minimal weight. We know from above that the Minimal Spanning Tree has to have a weight of 6, but let's see if this one has that minimal weight.

$$1 + 2 + 4 = 7$$
$$7 < 6$$

Because the path we have found has more of a weight than the minimal spanning tree, we know that this is not the minimal path, and therefore this algorithm will not work.

4. Let's see the subgraph below:



If a Minimal Spanning Tree was created for the whole graph G, that would have to include the vertices of the subgraph above. By the definition of a Minimal Spanning Tree, it will have to compare at least 2 of the 3 edges to find the minimum edge to include in the MST.

All the possible comparisons and their results are shown below:

$\{v_1, v_2\} > or < \{v_1, v_3\}$ ▻ $1 > or < 2$ ▻ $1 < 2$ ▻ $\{v_1, v_2\} < \{v_1, v_3\}$
$\{v_1, v_2\} > or < \{v_2, v_3\}$ ▻ $1 > or < 3$ ▻ $1 < 3$ ▻ $\{v_1, v_2\} < \{v_2, v_3\}$
$\{v_1, v_3\} > or < \{v_2, v_3\}$ ▻ $2 > or < 3$ ▻ $2 < 3$ ▻ $\{v_1, v_3\} < \{v_2, v_3\}$

As you can see, the edge $\{v_2, v_3\}$ has a weight that is greater than the weights of the other two edges. This means that in creation of the Minimal Spanning Tree T, the edge $\{v_2, v_3\}$ will not be chosen since the other two edges will have higher precedence over that edge, thus meaning either or both of those two will be chosen and this edge will not.

5.

a. We can use either Kruskal's or Prim's algorithms will work to calculate he minimal required length of cable if there is only one power plant.
b. This is a multi-step solution:
   Step 1 – Create a vertex with edges that connect to all existing powerplants. Make sure that each of these newly created edges have weights of 0.
   Step 2 – Implement Prim's algorithm. This will ensure that all the factories are connected to powerplants or to factories which are connected to powerplants. We know this will work because, at each stage, it will connect the factories with the closest powerplant without connecting the factory to another powerplant.
   Step 3 – Delete the newly created vertex and its edges.

   And there you have it. As a thank you for reading this, here's a funny quote by Yogi Berra that I found to be pretty funny: "When you come to a fork in the road, take it."