Wyatt Emery

Talon Marquard

COSC 3020

2/23/16

Project 2 Task 2

The problem we face is to create an unweighted, undirected, acyclic graph, which is in essence a tree. We must then find the longest path in this tree in linear time. To keep the graph unweighted, it is simple. We do not have to worry about adding weight to our edges. The undirected is also easy. This just means things can flow two ways along the graph. For finding the longest path, this doesn’t particularly matter because we will probably only follow a path in one direction. The acyclic means in our drawn representation that no vertices can connect to another that are on a higher or equal level. Another issue we may face when building it will be keeping the size within the bounds. We need to make sure that it doesn’t get larger than out vector is allowed and break something. We will need to plan ahead for this.

The building of this tree is our first issue. How can we represent this? Some of first thoughts involved a two dimensional vector or maybe a vector of lists. We decided that maybe the best way to approach this is as a vector of vectors. Our design is based off of the adjacency list. This is one of the easiest to follow and will give us a great deal to work with. We don’t need to worry about storing real values anywhere but just having the construct of our tree. The vector of vectors allows easy traversal through the tree. We may face some issues with this designs as implementation begins. It must be able to handle recursion, and recursion is never easy to plan for by simply thinking about it.

Other issues we will face come down to the real issue of finding the longest branch. So far we have been very consumed with just building the tree, but now how to we find the longest path. We could simply follow each branch until is ends and counting how far it goes, but this will not be fast, and may not even be linear. We need to check through the process and make sure it won’t be slow and will operate as efficiently as possible. If it is found to be slower than we think it will be, we will have to rethink the best way of approaching the issue.

The timing of this issue should not be an issue as far as we can tell. We have lots of experience timing things like this now.