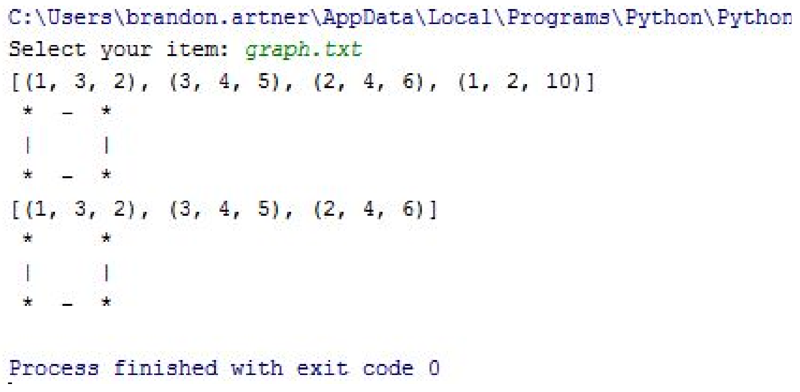
Timothy Indrieri
Brandon Artner
Edward Morkunas

Assignment 3

The program reads in data from a file that we call “graph.txt” where the first integer is the grid size parameter, first two integers in the triple denote two(distinct nodes) , and the last followed by a semicolon gives the weight between the two edges. The first number is the number of nodes per row and is followed by a list of connections. We read in this file “graph.txt” for example the following input would be as the following 2:(1,2;9),(3,4;5),(2,4;6),(1,3;2) with a regular expression. After we create this list we then sort the edges in descending order. This means that we have the lowest cost edge at the beginning of the list and the highest cost at the end of the list. This implementation comes from Kruskal’s algorithm. After we sort the edges, we calculate the total number of nodes by squaring the first number that was read in from the input. In our input, “graph.txt” , the first number is 2. This means that we would have a total of 4 nodes within our graph. Next we perform a check to make sure that we are indeed working with a graph. For this assignment a graph is described as “Every node must be connected to at most the nodes directly above and below, and to the two nodes immediately to the left and right.” Our check involves a for loop that iterates through the edges list. We check to make sure that each node does not exceed the number of nodes or that the node does not move below zero. If any of these checks do not pass then we know that we do not have a graph as indicated by the instructions of the assignment. Next, we implement another component of Kruskal’s algorithm. We check whether an addition of an edge creates a cycle. Following algorithm 2.2 in the text, we merge components if they do not form a cycle. The method used to printed out the graph was heavily influenced by the github repository on problem 2.6. There is also a second test run that we used as shown below with “graph2.txt” that we used for a more complex grid. The displays below first shows the original graph followed by the MCST of the graph with ‘*’ representing the nodes and ‘-’ & ‘|’ representing the connections to those nodes.

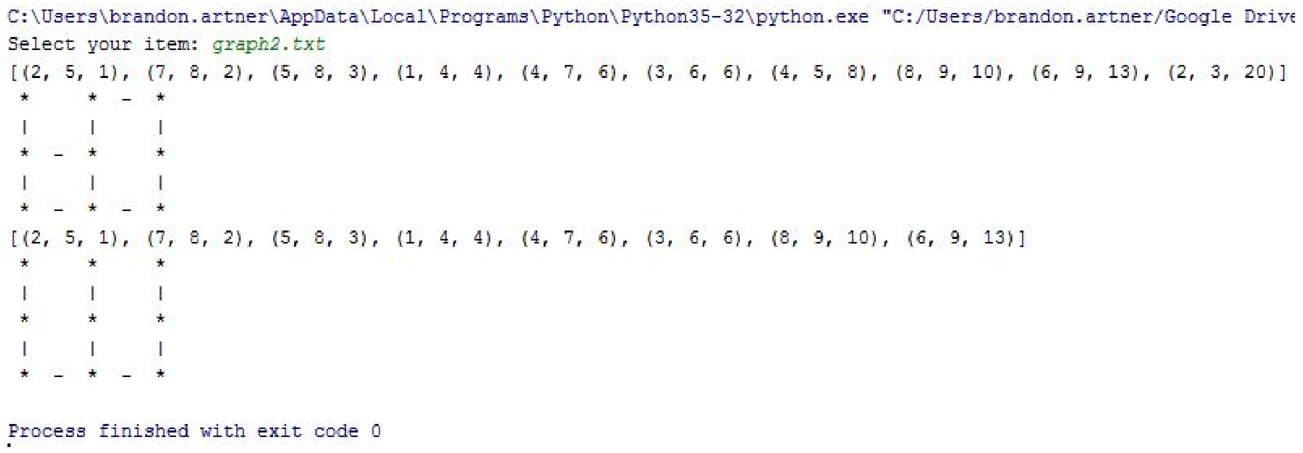
<u>Sample Run with graph.txt:</u>



```
C:\Users\brandon.artner\AppData\Local\Programs\Python\Python
Select your item: graph.txt
[(1, 3, 2), (3, 4, 5), (2, 4, 6), (1, 2, 10)]
 *  _  *
 |     |
 *  _  *
[(1, 3, 2), (3, 4, 5), (2, 4, 6)]
 *     *
 |     |
 *  _  *


Process finished with exit code 0
```

Sample Run with graph2.txt:



```
C:\Users\brandon.artner\AppData\Local\Programs\Python\Python35-32\python.exe "C:/Users/brandon.artner/Google Drive
Select your item: graph2.txt
[(2, 5, 1), (7, 8, 2), (5, 8, 3), (1, 4, 4), (4, 7, 6), (3, 6, 6), (4, 5, 8), (8, 9, 10), (6, 9, 13), (2, 3, 20)]
 *     *  _  *
 |     |     |
 *  _  *     *
 |     |     |
 *  _  *  _  *
[(2, 5, 1), (7, 8, 2), (5, 8, 3), (1, 4, 4), (4, 7, 6), (3, 6, 6), (8, 9, 10), (6, 9, 13)]
 *     *     *
 |     |     |
 *     *     *
 |     |     |
 *  _  *  _  *

Process finished with exit code 0
```