Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 5b (48 points) **Fall 2019, CU-Boulder**

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.
- Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.
- For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.
- You may work with other students. However, **all solutions must be written independently and in your own words.** Referencing solutions of any sort is strictly prohibited. You must explicitly cite any sources, as well as any collaborators.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 5b (48 points) **Fall 2019, CU-Boulder**

1. (25 pts) For this question, you are going to implement Kruskal's algorithm and union-find to build an MST from supplied data. Refer to the python starter code **MST_Q1_starter_code.py** on Canvas that generates a graph of US cities, where the cities are the vertices and the edges are the distances between them. The code requires **miles_dat.txt.gz** file as the graph data source so keep it in the same folder as the code. Before you start writing any code, make sure you can build the code that's been supplied. The code uses the networkx library. You may need to install this library for the code to run.

   Read all instructions for this question carefully.

   (a) (5 pts) Complete the code to find the edges that are part of the MST. You should add these edges in the list *kruskal_selected_edges*. Do not change the existing format of the edges. They are represented as a tuple of vertices and a vertex is represented like v = "Waukegan, IL". Read the comments in the code for more information. You don't need to read/understand the *miles_graph*() and *draw_graph*() functions.

   (b) (10 pts) Implement the *union*() function to implement Kruskal's.

   (c) (10 pts) Modify your code slightly so that you can produce disconnected components. Let's call these components clusters. The spacing of any particular clustering (group of clusters) is defined as the smallest edge between vertices in any pair of different clusters. If we stop Kruskal's k iterations before the algorithm completes, what is the spacing value? Run your code for $k = 2...10$ to generate spacing for all these k values. Your code needs to have this calculation for your answer to receive credit.

   (d) In the pdf that you submit for this assignment, please include the following:

      i. One of the generated graphs **MST.png** that your code produces that shows the MST for that run. Note that on each run, you can get a different number of edges to begin with. Thus, you can expect a different answer each time you run.

      ii. The spacing values for each k value that you use.

      iii. Your .py file for this question needs to be submitted to Canvas.

Name: Jonathan Phouminh

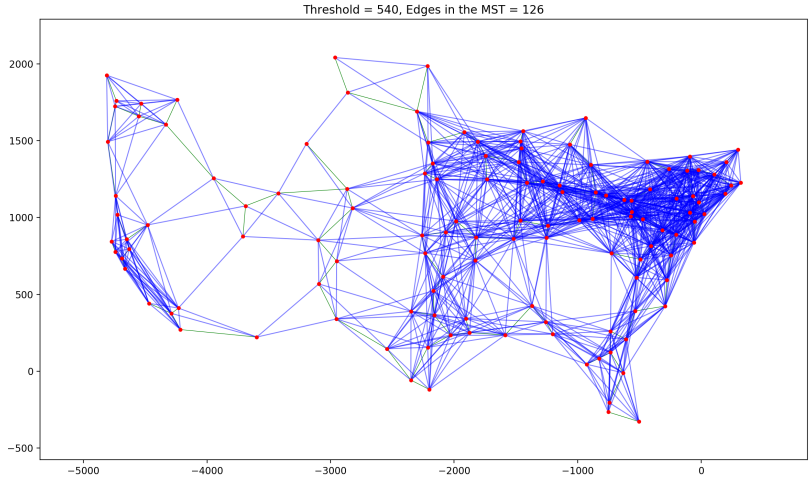
ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 5b (48 points) **Fall 2019, CU-Boulder**

```
Loaded miles_dat.txt containing 128 cities.
digraph has 128 nodes with 8128 edges
Edges considered (in ascending order) for this graph =  1236
for k =: 2 spacing is:
{'weight': 418}
---------
for k =: 3 spacing is:
{'weight': 358}
---------
for k =: 4 spacing is:
{'weight': 357}
---------
for k =: 5 spacing is:
{'weight': 344}
---------
for k =: 6 spacing is:
{'weight': 324}
---------
for k =: 7 spacing is:
{'weight': 320}
---------
for k =: 8 spacing is:
{'weight': 278}
---------
for k =: 9 spacing is:
{'weight': 271}
---------
for k =: 10 spacing is:
{'weight': 236}
---------

Number of edges selected by Kruskal's =  126
```



2. (3 pts) How many disconnected components are there when you stop Kruskal's k round before you complete the MST? Justify your answer.

   Solution.

   There will be $k + 1$ components because every iteration combines two clusters therefore if we stop at round k we will have k + 1 clusters since we did not combine the next set of clusters.

Name: Jonathan Phouminh

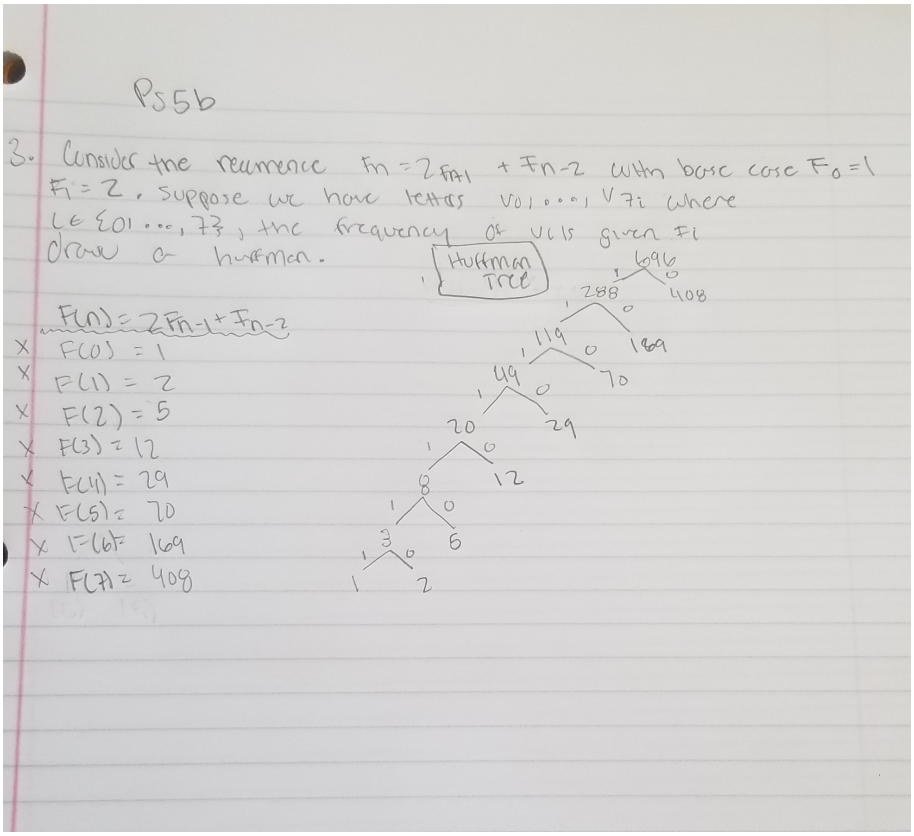
ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 5b (48 points) **Fall 2019, CU-Boulder**

3. (5 pts) Consider the recurrence $F_n = 2F_{n-1} + F_{n-2}$, with the base cases $F_0 = 1$ and $F_1 = 2$. Suppose we have letters $v_0, \ldots, v_7$; where for $i \in \{0, \ldots, 7\}$, the frequency of v_i is given by F_i. Draw a Huffman tree for $v_0, \ldots, v_7$.

Solution.



Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 5b (48 points) **Fall 2019, CU-Boulder**

4. (5 pts) Assume you run your Huffman tree algorithm and you produce the following pre-fix codes. Describe why there must be an error in your algorithm.

```
S = 00
c = 01
i = 001
e = 011
n = 101
```

Solution.
There must be an error in the Huffman tree because when we construct the tree we will see that not every tree has a value associated with it. Also we notice that the coding for 'S' is a prefix for 'i' therefore causing ambiguity the two values. Also we see this same problem with codings for 'c' and 'e'.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 5b (48 points) **Fall 2019, CU-Boulder**

5. (10 pts) Assume you're given an integer matrix that represents a plot of land, where the value at that location in the matrix represents the height above sea level. A value of zero indicates water. A pond is a region of water connected vertically, horizontally, or diagonally. The size of the pond is the total number of connected water cells. Write an algorithm to compute the sizes of all ponds in the matrix.

   Example:

   ```
   0 2 1 0
   0 1 0 1
   1 1 0 1
   0 1 0 1
   ```

   would output 1, 2, 4.

   (a) (3 pts) Describe the graph data structure that your algorithm will use for this problem.

   Solution.
   We will use an adjacency list graph representation. We will have all the matrixe's vertex point to a linked list that contain that vertices neighbors.

Name: Jonathan Phouminh

ID: 106054641

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 5b (48 points) **Fall 2019, CU-Boulder**

(b) (2 pts) Provide a 3-4 sentence description of how your algorithm works, including how the matrix is converted to the graph, how adjacent vertices are identified, and how the algorithm traverses the graph to identify connected vertices.

Solution.
We make all vertex's in the matrix point to an array that holds the neighbors of its vertexs. We identify the vertexs neighbors by examining what its connected to in the linked list structure. We will traverse the graph by following the neighbors that have a value of - to count the size of the region of the pond.

(c) (5 pts) Write an algorithm to solve this problem.

Solution.
def foo():

- Loop through- entire matrix

-have all elements of the matrix point to a linked list structure

- Keep variable, $count - of - pond - size$, and a vector that puts in all the ponds we encounter.

- Traverse through the first vertex with value 0 and follow its neighbors that have a value 0, increment $count - of - pond$. When you hit a vertex that has no value 0, push a value into the declared vector that denotes size of pond. Should also ensure that we have kept track of vertices that have been visited. repeat until all ponds are found.

- loop through again to update all its linked list to all of its neighbors

Collaborated with:
Sam Williamson
Zach Chommala
Bao Nguyen
Jennifer Palese
Marvin Nguyen
Michael Ren