Summary

Our goal was to find locations for warehouses so that the entire continental US would have one-day shipping. The model we devised is capable of taking in a list of images corresponding with the transit maps of selected zip codes, which represent potential locations for warehouses and over laying those images to find the different combinations of locations that satisfy our goal. Our model has the capability of checking every possible combination of locations given the inputted maps. Despite the fact that our model takes a very long time to calculate its results, it can calculate every possible result, and thus theoretically determine the least amount of warehouses needed, as well as where they should be located. Because any number of maps can be inputted into our model, it can be used to calculate the area covered with one-day shipping depending on what locations a company might be considering, not just the parameters set by the problem.

To account for tax, the program that found the potential locations for warehouses then took those locations and checked which states they belonged to. The program then added up the tax rates from each of those states, which results in a "tax number." A similar process was done to calculate a "tax number" for Part 3 as well, this time counting how many of the warehouses have a clothing tax. With this knowledge, the results from Part 1 could be compared to determine which was the better solution when taxes were taken into consideration.

The Letter

Dear President,

Recently, a demand for our storefront to move to the forefront of the world has arisen. To achieve this goal, we need to increase the amount of warehouses our company has across the continental United States. Since we are a consumer-friendly company, we have promised one-day shipping to all consumers in the continental US. To decrease the cost associated with doing this, we have devised a model to find the possible combinations of locations for warehouses given a desired number of warehouses. From this model, we can find solutions that require the least amount of warehouses. Also, to increase both our profit and customer satisfaction, we conceived a system to comparatively measure the effect of tax on our model, and deemed those scores, “tax numbers.” We then tried to station our warehouses in locations which resulted in situations with the lowest tax numbers.

Our model has given us the result that 64 warehouses should be used, and that they should be located in: NE, IN, ID, MI, WA, NV, MO, NY, OK, AR, KS, AZ, CA, SD, AL, FL, IL, D.C, CO, TX, TN, NM, UT, OR, ME, NH, LA, WV, MT, GA, VT, WY, MS, KY, NJ, RI, OH, and MN.

We chose these locations because they minimized the number of warehouses used. They supply one-day shipping across the entire continental US and favor states that have a smaller sales and clothing tax. We hope you implement this solution as quickly as possible to maximize our profits.

Yours truly, Team #6283

Introduction

An online store is trying to make a model to strategically place warehouses across the contingent United States so that everywhere in the continental US gets one-day shipping. Our team tackled this problem by creating a model that would simulate the area that had one-day shipping dependent on each warehouse location. The primary task was to figure out the least amount of warehouses it would take to be able to provide one-day shipping throughout the entire continental US. Secondly, we were assigned to figure out the liability of tax on customers based on our placement of warehouses and how our chosen warehouse locations could be adjusted to reduce the tax the customer has to pay. Our third responsibility was to analyze and improve the previous models based on the addition of clothing and apparel to inventory and the associated taxes. Lastly, we were tasked to compose a letter to the president of the company about our solutions to these problems. In order to attempt this problem, we needed to create a set of assumptions to specify some constraints that would make the predicted scenarios more reasonable.

Assumption and Justification

A. The mandatory warehouse is located in Portsmouth, New Hampshire. This assumption was made because the first location had to be in New Hampshire, and the transit map from the UPS site for Portsmouth aligned with the given map.

B. Each warehouse has unlimited inventory because One Day Shipping would not be possible for all of the continental United States if all the warehouses in one large region ran out of a certain product.
C. Regions without zip codes in America would not necessarily have to have one-day shipping if that was not possible.
D. A warehouse can be built in every zip code region in the contingent United States.
E. Grocery tax is not taken account for because the company does not sell food.
F. The maps/delivery date provided by the UPS website are constant because it was given in the problem statement.

Model Description

Part I - Obtaining the Maps

The job of our model is to take a number of transit maps from different locations and check if the locations can cover America with one-day shipping. The first step was to obtain transit maps from the UPS website (5), the same website as what was used to obtain the example map present in the problem. To obtain these transit maps, a zip code was necessary. Zip codes from all 48 continental US states were hand selected based on which gave the most unique coverage (see Figure 1). First zip codes which likely had good coverage were found by looking at zip codes from large cities or near highways. Next, additional zip codes were found from different locations in the state which were far apart. All of these zip codes were checked using the UPS website, and if they had enough coverage, the zip code was kept to be used in the model.

Figure 1. Example of how the zip codes were determined based on zip codes from Iowa. Both the image to the left and to the middle were kept because they covered a large area, and did not cover the same area. The image to the right was not kept, because all of that area was covered by the image to the left.

Creating a Model for Part I

We decided to model the situation using Java. The maps from different possible locations would be laid over each other and the result would be checked to see if one-day shipping (represented by yellow) would cover all of America. Once the zip codes were finalized, we used the UPS website to find the URLs to the corresponding transit maps. From there, we downloaded the images in order to create a faster simulation. Figure 2 below shows the process that the model used to determine a solution.

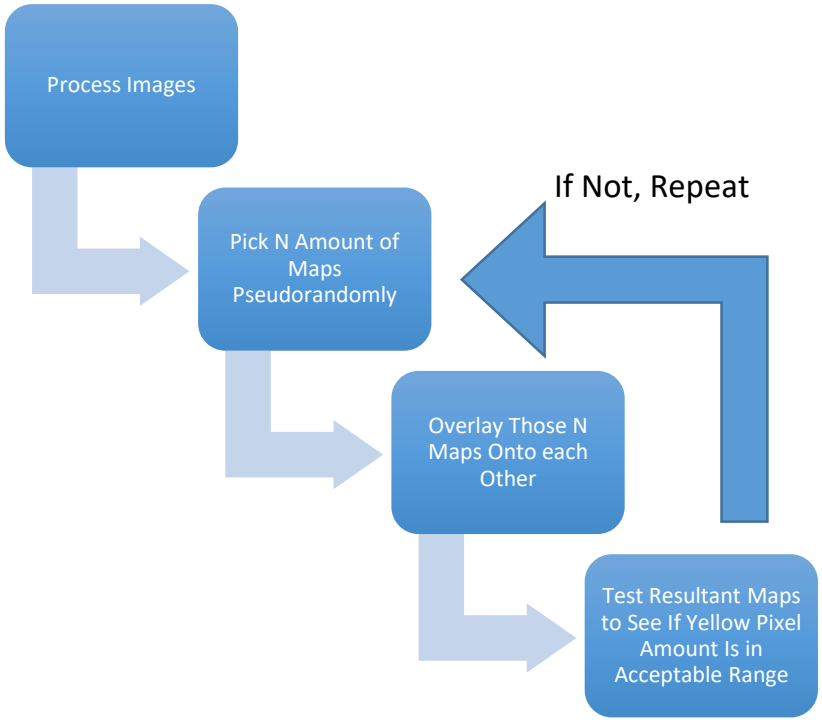

Figure 2. This process was used to determine a solution with N amount of maps. Images were processed to remove unnecessary elements and to add transparency, and then N maps were pseudo randomly picked and compared to a known value of yellow pixels in a complete solution.

The code behind the model iterated over the process described by Figure 2 until a solution was found for that specific N, the amount of maps, and therefore warehouses, used. A pseudo-random selection was used because the number of possible combinations for any N is 2^n - 1, which is an intractably big number to calculate with a standard computer. The known value of pixels in a complete solution is 91,421 pixels, which we found by running part of our code with a manually arranged map with all yellow. If the amount of yellow pixels generated by the aforementioned process was greater than 91,400 pixels - within a negligible range of the accepted value - then that N, number of maps/warehouses, is a possible solution. A small part of the code was adapted from (3) and some coding references were used (2)(7).

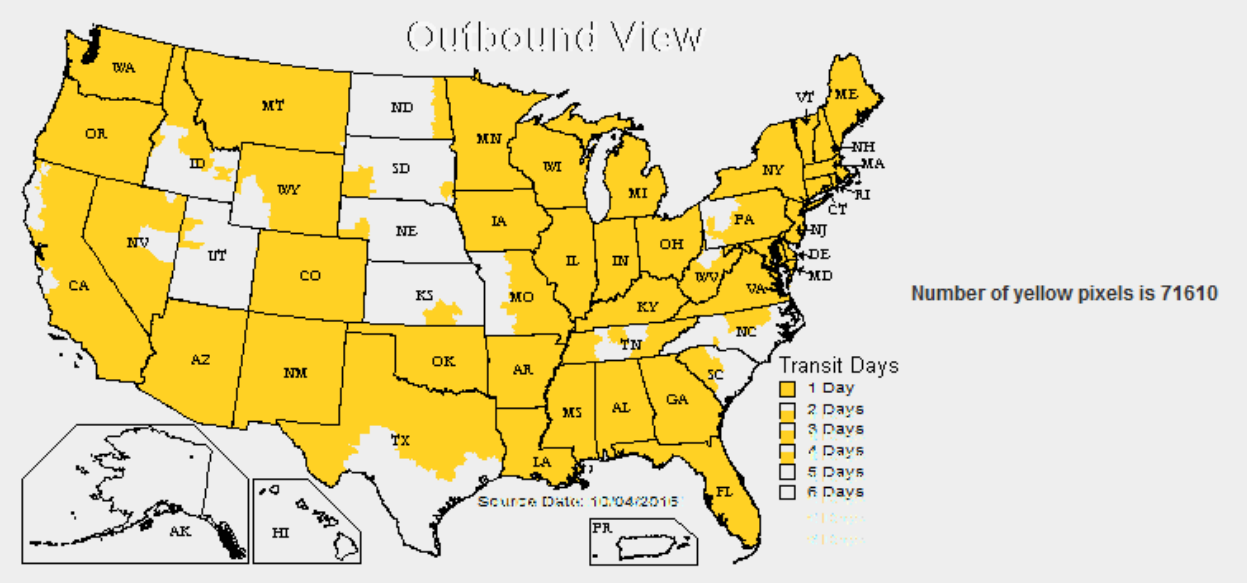

Figure 3. Example of how the results were compared to the total number of yellow pixels required to fill up the diagram of America. This image does not pass because it does not match the total number of yellow pixels.

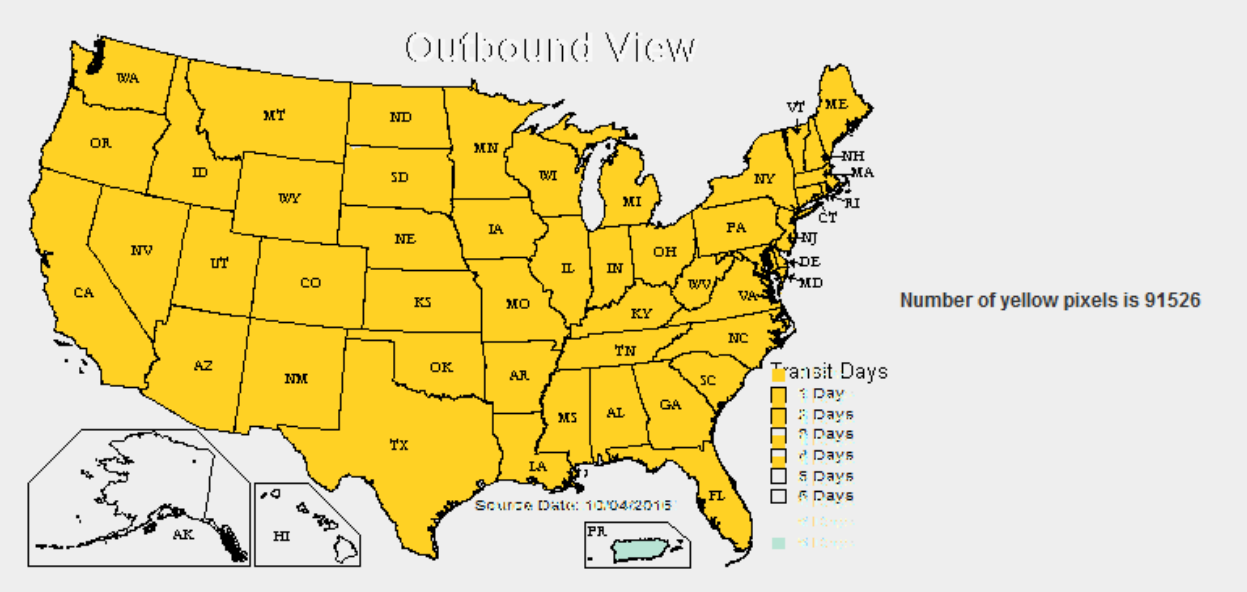

Figure 4. Example of a result that has an equal number of yellow pixels compared to the total number of pixels required to fill a picture of America. There is a slight discrepancy with the number of pixels being higher than the accepted value due to the legend affecting part of the pixel count, but this was deemed negligible.

Creating a Model for Parts 2 & 3

For parts 2 & 3, we compared the situations gathered from the model and compared them by taking into consideration the tax in each location. For part 2, the tax in each state was accounted for, and each situation was given a "tax number" that was the sum of all the taxes from each of the states there was a warehouse in. This was done by taking the names of the images of the maps, figuring out which state each corresponded to, and then adding up the taxes from each state. For part 3, a similar process was deployed, but this time the number of locations in states with a clothing tax were counted. By looking at the two numbers generated, the person using the model can decide which situations are better tax wise, since they will have lower "tax numbers."

Analysis of the Model

Strengths

Our model has the capability of checking every possible solution, and determining without a doubt if the resulting situation can cover all of the continental US with one-day shipping.

Our model is able to use a large number of possible maps representing possible warehouse locations, as well as how the number of possible locations used can be changed. This means that the company could easily use the same model but specify only a specific set of locations they were considering. This also means that our program has the capability of analyzing every possible transit map, although that is not what it is currently set up to do.

Weaknesses with Part I

Our model is based on random chance. Although it has the capability of finding every possible combination for a set number of locations, it will take longer than if it just ran through every possible combination.

The probability of finding of finding a situation that works is x/ (n! /((n-r!) r! - t), where

x = predicted number of actual solutions given a set number of locations allowed

n = number of possible different locations

r = number of locations allowed in each situation

t = amount of failed tries that have already been ran through

The program takes a very long time to code, and there is no assurance as to how soon one possible situation that satisfies the problem will be found. After one solution is found, there is no guarantee as to how long it will take to find the next one, or even if there are any more solutions with that given number of locations.

Extensions

Part 1

For part 1, instead of randomly selecting what locations would go together into a situation, we could have made a program that ran through every possible combination. Although this would reduce the trouble with being uncertain whether or not there are any solutions for a given number of locations, it would also take a long amount of time, more so than our current method.

Part 2 & 3

For parts 2 & 3, taxes were taken into account in order to specify which situation had the locations located in the states with the lowest taxes. While our current model only determines the "tax number" based on solely counting up the tax in of each state that there is a location in, a more accurate model would take into account the different population density of areas in order to judge how many possible customers would be affected by the tax. A more accurate model would also take into account the value of locations in states that reach a large area outside of their state, since any area reached outside of the state that the warehouse is located in would not be taxable.

Results & Conclusion

Part 1

For Part 1, we found that the lowest number of warehouses required was 64. This is probably incorrect for multiple reasons. There are likely situations that require less locations to cover all of the continental US, but because of the time it takes for our program to find just one solution, we could only find the three different situations for 64 locations. With more time, we could use our model to figure out a more optimal solution based on the previous solutions. We can conclude that 64 warehouses are more than enough to be able to cover the continental US. From this we could work down the number line, trying 64, then 64, then 63, etc., until the lowest number of locations required was found. Our solution to the problem reveals multiple things, including how there is a very large number of possible combinations of locations to satisfy the

problem. The ability of our program to pick up on this large range of possibilities allows any company to use our model in trying to gauge where warehouses should be put so that one day shipping could be achieved all across the continental US. The fact that what zip codes or transit maps are used can be changed also means that this model can easily adapt to situations that have only a specific range of locations that the company wants tested.

Part 2 & 3

The results for Part 2 conclude that out of the three solutions created using 64 warehouses, Solution 3 was the most beneficial to consumers. The summation of the state taxes, which is the "tax number" for Solution 1, 2, and 3 are 367, 353, and 355, respectively as shown in Table 1. Solution 3 has the best "tax number" because the third scenario created by the model used in Part 1 depicted the lowest number of warehouses that were in the most strategically located places. When the sum of all of the taxes in the areas where warehouses were being placed was calculated, the result was less than the other two solutions.

The results for Part 3 show how many warehouses out of the 64 total warehouses in the continental US have a clothing tax. These numbers are calculated based on whether the states have a clothing tax based on the information given. The warehouses in each state that had no clothing tax were added altogether. But if the state did have a tax, then the count would increase by the number of warehouses in that state. Based on the solutions created in Parts 1 and 2, the number of warehouses in each solution were added together. Solutions 1, 2, and 3 had 58, 57, and 59 warehouses that had a clothing tax respectively. Solution 2 proved to be the best scenario

because it had the lowest number of warehouses with clothing tax, which is advantageous to consumers.

Table 1: The table lists the sum of the state taxes and warehouses with clothing tax for each of the three solutions that used 64 warehouses as the minimum number needed to cover the US. See extended table in Appendix.

			Solutions for N=64				
Solution 1			Solution 2			Solution 3	
Sum of State Taxes	367		Sum of State Taxes	353		Sum of State Taxes	355
Number of States	58		Number of States	57		Number of States	59
With Clothing Tax			With Clothing Tax			With Clothing Tax	

References

(1) AggData. (2012, February 16). Retrieved November 11, 2016, from https://www.aggdata.com/node/86

(2) Creating a jar File in Eclipse. (n.d.). Retrieved November 11, 2016, from https://www.cs.utexas.edu/~scottm/cs307/handouts/Eclipse Help/jarInEclipse.htm

(3) Java Graphics How to - Overlay 2 images. (n.d.). Retrieved November 11, 2016, from http://www.java2s.com/Tutorials/Java/Graphics_How_to/Image/Overlay_2_images.htm

(4) Thompson, A. (2013, October 22). How to put integer value in JLabel? Retrieved November 11, 2016, from https://stackoverflow.com/questions/19510198/how-to-put-integer-value-in-jlabel

(5) United States. (n.d.). Retrieved November 11, 2016, from https://www.ups.com/maps?loc=en_US

(6) U.S. ZIP Codes: Free ZIP code map and zip code lookup. (2014). Retrieved November 11, 2016, from http://www.unitedstateszipcodes.org/

(7) Writing/Saving an Image. (n.d.). Retrieved November 11, 2016, from https://docs.oracle.com/javase/tutorial/2d/images/saveimage.html

Appendix

Java Code

```java
package HiMCM;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.awt.image.FilteredImageSource;
import java.awt.image.ImageFilter;
import java.awt.image.ImageProducer;
import java.awt.image.RGBImageFilter;
import java.io.File;
import java.io.IOException;
import java.util.Random;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;

public class Tester2 {

  public static void main(String[] args) throws Exception {

    /* http://www.java2s.com/Tutorials/Java/Graphics_How_to/Image/Overlay_2_images.htm *****/

    File[] file = new File[94];

    file[0] = new File("C:/Users/Niall/Desktop/HiMCM/Maps/backgroundMap.gif");
    file[1] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0506.gif");
    file[2] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0422.gif");
    file[3] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0522.gif");
    file[4] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0475.gif");
    file[5] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0022.gif");
    file[6] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0079.gif");
    file[7] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0154.gif");
    file[8] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0131.gif");
    file[9] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0498.gif");
    file[10] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0346.gif");
    file[11] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0250.gif");
    file[12] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0283.gif");
    file[13] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0396.gif");
    file[14] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0203.gif");
    file[15] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0415.gif");
    file[16] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0021.gif");
    file[17] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0005.gif");
    file[18] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0260.gif");
    file[19] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0308.gif");
    file[20] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0197.gif");
    file[21] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0384.gif");
    file[22] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0334.gif");
    file[23] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0288.gif");
    file[24] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0519.gif");
    file[25] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0007.gif");
    file[26] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0028.gif");
    file[27] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0045.gif");
    file[28] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0116.gif");
    file[29] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0313.gif");
    file[30] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0216.gif");
    file[31] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0428.gif");
    file[32] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0550.gif");
    file[33] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0073.gif");
    file[34] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0006.gif");
    file[35] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0128.gif");
    file[36] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0322.gif");
    file[37] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0178.gif");
    file[38] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0437.gif");
    file[39] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0503.gif");
    file[40] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0018.gif");
    file[41] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0090.gif");
    file[42] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0563.gif");
    file[43] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0099.gif");
    file[44] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0295.gif");
    file[45] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0484.gif");
    file[46] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0170.gif");
    file[47] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0504.gif");
    file[48] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0192.gif");
    file[49] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0536.gif");
    file[50] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0160.gif");
    file[51] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0138.gif");
    file[52] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0497.gif");
    file[53] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0358.gif");
    file[54] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0241.gif");
    file[55] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0285.gif");
    file[56] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0393.gif");
    file[57] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0205.gif");
    file[58] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0413.gif"); // take out?
```

```
    file[59] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0277.gif");
    file[60] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0268.gif");
    file[61] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0286.gif");
    file[62] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0195.gif");
    file[63] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0370.gif");
    file[64] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0401.gif");
    file[65] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0489.gif");
    file[66] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0513.gif");
    file[67] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0113.gif");
    file[68] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0436.gif");
    file[69] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0326.gif");
    file[70] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0184.gif");
    file[71] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0461.gif");
    file[72] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0092.gif"); // take out?
    file[73] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0168.gif");
    file[74] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0085.gif");
    file[75] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0306.gif");
    file[76] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0488.gif");
    file[77] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0424.gif");
    file[78] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0543.gif");
    file[79] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0132.gif");
    file[80] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0491.gif");
    file[81] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0377.gif");
    file[82] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0257.gif");
    file[83] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0088.gif");
    file[84] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0407.gif"); // take out?
    file[85] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0198.gif");
    file[86] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0427.gif");
    file[87] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0543.gif");
    file[88] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0490.gif");
    file[89] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0372.gif");
    file[90] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0470.gif");
    file[91] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0510.gif");
    file[92] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0493.gif"); // take out?
    file[93] = new File("C:/Users/Niall/Desktop/HiMCM/zips/map_0472.gif");

    int numCheck = 65; /********************* NUMBER OF MAPS USED **********************************/

    // List to hold zips used
    int[] list = new int[numCheck];

    Runnable r = new Runnable() {
      @Override
      public void run() {

        // Tells if solution has been found or not
        boolean solutionFound = false;
        // For yellow pixels
        int counter = 0;

        int numFiles = file.length;

        BufferedImage[] Images = new BufferedImage[numFiles];

        // Repeat while no solution has been found
        while (solutionFound == false) {

            // Read in files to image array
            for (int i = 0; i < numFiles; ++i) {
                try {
                    Images[i] = ImageIO.read(file[i]);
                    System.out.println("Passed: " + i);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            // Temp for transformed images
            Image[] temp = new Image[numFiles];

            for (int i = 1; i < numFiles; ++i) {
                temp[i] = TransformColorToTransparency(Images[i], new Color(0,0,0), new Color(255, 200, 255));
            }

            // Transfer back to buffered images
            for(int i = 1; i < numFiles; ++i) {
                Images[i] = ImageToBufferedImage(temp[i], Images[i].getWidth(), Images[i].getHeight());
            }

            // Initialize graphics
            Graphics2D g;

            // Create pseudo random rng
            Random rng = new Random(System.currentTimeMillis());

            // Get base picture
            int temp1 = rng.nextInt(94); /************* WAS 98 ******************************************/
            g = Images[temp1].createGraphics();

            //gui.add(new JLabel(new ImageIcon(Images[0])));
```

```
            // Add base picture
            g.drawImage(Images[0], 0, 0, null);

            // list was here

            // Add unique identifier to unassigned list elements
            for (int i = 1; i < numCheck; ++i) {
                list[i] = -1;
                System.out.println(list[i]);
            }

            // Add the background map to used zips
            list[0] = 0;

            // Flag for repeats in list, should have only unique zips
            boolean flag = false;

            // Add in unique zips to list
            for (int i = 1; i < numCheck; ++i) {
                do {
                    int temp2 = rng.nextInt(94); // *********** WAS 98 *******************************************/

                    flag = false;

                    for (int j = 0; j < numCheck; ++j) {
                        if (list[j] == temp2) {
                            flag = true;
                        }
                    }

                    if (flag == false) {
                        g.drawImage(Images[temp2], 0, 0, null);
                        list[i] = temp2;
                    }
                } while (flag == true);
            }


            // Debug
            System.out.println("Flag is " + flag);

            // Reset yellow pixel counter
            counter = 0;
```

```
            // Debug
            System.out.println("Counter is: " + counter);

            // Sum up the number of yellow pixels
            for (int x = 0; x < Images[temp1].getWidth(); x++) {
                for (int y = 0; y < Images[temp1].getHeight(); y++) {
                    Color color = new Color(Images[temp1].getRGB(x,y));
                    int red = color.getRed();
                    int green = color.getGreen();
                    int blue = color.getBlue();
                    if (red == 255 && green == 209 && blue == 36) {
                        counter++;
                    }
                }
            }


            // Print out counter to system and set up label for gui
            System.out.println("Counter is: " + counter);
            JLabel lblTemp1 = new JLabel("");
            lblTemp1.setText("Number of yellow pixels is " + String.valueOf(counter));
            //gui.add(lblTemp1);

            // Adjust for discrepancy in California and Wyoming
            boolean californiaFlag = false;
            boolean wyomingFlag = false;

            for (int i = 0; i < list.length; ++i) {
                if (list[i] == 49) {
                    californiaFlag = true;
                } else if (list[i] == 65) {
                    wyomingFlag = true;
                }
            }

            // 91,421 is the number of yellow pixels in a full America grid, enter IF close to value and CA + WY has been accounted for
            if (counter >= 91400 && californiaFlag == true && wyomingFlag == true) {

                // Output
                JPanel gui = new JPanel(new GridLayout(1, 0, 5, 5));
                System.out.println("Fills America");

                // Set solution to found
                solutionFound = true;
                gui.add(new JLabel(new ImageIcon(Images[temp1])));
```

```
271
272                 // More output
273                 JLabel lblTemp = new JLabel("");
274                 lblTemp.setText("Number of yellow pixels is " + String.valueOf(counter));
275                 gui.add(lblTemp);
276
277                 // Counter to store number of unused
278                 int count = 0;
279
280                 // List to hold the number of unused zips
281                 int[] list2 = new int[94-numCheck]; // WAS 98
282
283                 // Fill up list2 with unused zips
284                 for (int i = 0; i < 94; ++i) { // WAS 98
285                     flag = false;
286
287                     for (int j = 0; j < list.length; ++j) {
288                         if (list[j] == i)
289                             flag = true;
290                     }
291
292                     if (flag == false) {
293                         list2[count] = i;
294                         count++;
295                     }
296                 }
297
298                 // More  output
299                 for (int i = 0; i < list2.length; ++i) {
300                     System.out.println("Unused Number: " + list2[i]);
301                     JLabel lblTemp2 = new JLabel("");
302                     lblTemp2.setText("Unused zip " + String.valueOf(file[list2[i]]));
303                     gui.add(lblTemp2);
304                 }
305
306                 JOptionPane.showMessageDialog(null, gui);
307             } else {
308                 // Output if failed
309                 System.out.println("Does not fill America");
310
311                 /*gui.add(new JLabel(new ImageIcon(Images[temp1])));
312                 JLabel lblTemp = new JLabel("");
313                 lblTemp.setText("Number of yellow pixels is " + String.valueOf(counter));
314                 gui.add(lblTemp);*/
315             }
```

```
316
317             // Print out used zips
318             for (int i = 0; i < list.length; ++i) {
319                 System.out.println("Zip Number: " + file[list[i]]);
320             }
321
322             String[] links = new String[numCheck];
323
324             for (int i = 0; i < list.length; ++i) {
325                 links[i] = "https://www.ups.com/using/services/servicemaps/maps25/" + file[list[i]].toString().substring(34);
326                 System.out.println("Link is: " + links[i]);
327             }
328
329             // Check taxes
330             PartII(links);
331             PartIII(links);
332
333         }
334     }
335   };
336   SwingUtilities.invokeLater(r);
337
338 }
339
340 private static Image TransformColorToTransparency(BufferedImage image, Color c1, Color c2)
341 {
342   final int r1 = c1.getRed();
343   final int g1 = c1.getGreen();
344   final int b1 = c1.getBlue();
345   final int r2 = c2.getRed();
346   final int g2 = c2.getGreen();
347   final int b2 = c2.getBlue();
348   ImageFilter filter = new RGBImageFilter()
349   {
350     public final int filterRGB(int x, int y, int rgb)
351     {
352       int r = (rgb & 0xFF0000) >> 16;
353       int g = (rgb & 0xFF00) >> 8;
354       int b = rgb & 0xFF;
355       if (r >= r1 && r <= r2 &&
356           g >= g1 && g <= g2 &&
357           b >= b1 && b <= b2)
358       {
359         // Set fully transparent but keep color
360         return rgb & 0xFFFFFF;
```

```
        }
        return rgb;
      }
    };

    ImageProducer ip = new FilteredImageSource(image.getSource(), filter);

    return Toolkit.getDefaultToolkit().createImage(ip);
  }

  private static BufferedImage ImageToBufferedImage(Image image, int width, int height)
  {
    BufferedImage dest = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2 = dest.createGraphics();
    g2.drawImage(image, 0, 0, null);
    g2.dispose();
    return dest;
  }

  public static void PartII(String[] urls){
        int n = urls.length;
        String[] imageNumbers = new String[n];
        double taxNumber = 0;


        for (int i = 0; i < urls.length; i++){
            String imageNumber = urls[i].substring(urls[i].length() - 8, urls[i].length() - 4);
            imageNumbers[i] = imageNumber;
        }

        for (int i = 0; i < imageNumbers.length; i++){
            if (imageNumbers[i].equals("0079") || imageNumbers[i].equals("0334") ||
                    imageNumbers[i].equals("0007") || imageNumbers[i].equals("0550")) {
                taxNumber = taxNumber + 0.00;
            } else if (imageNumbers[i].equals("0475")) {
                taxNumber = taxNumber + 2.90;
            } else if (imageNumbers[i].equals("0168") || imageNumbers[i].equals("0131") ||
                    imageNumbers[i].equals("0415") || imageNumbers[i].equals("0045") ||
                    imageNumbers[i].equals("0322") || imageNumbers[i].equals("0484") ||
                    imageNumbers[i].equals("0170") || imageNumbers[i].equals("0413") ||
                    imageNumbers[i].equals("0138") || imageNumbers[i].equals("0326") ||
                    imageNumbers[i].equals("0488") || imageNumbers[i].equals("0436") ||
                    imageNumbers[i].equals("0370") || imageNumbers[i].equals("0132") ||
                    imageNumbers[i].equals("0407") || imageNumbers[i].equals("0489")) {
                taxNumber = taxNumber + 4.00;
            } else if (imageNumbers[i].equals("0384") || imageNumbers[i].equals("0370")){
                taxNumber = taxNumber + 4.23;
            } else if (imageNumbers[i].equals("0428") || imageNumbers[i].equals("0436")){
                taxNumber = taxNumber + 4.50;
            } else if (imageNumbers[i].equals("0116") || imageNumbers[i].equals("0113")){
                taxNumber = taxNumber + 4.75;
            } else if (imageNumbers[i].equals("0131") || imageNumbers[i].equals("0295") ||
                    imageNumbers[i].equals("0306")){
                taxNumber = taxNumber + 5.00;
            } else if (imageNumbers[i].equals("0510") || imageNumbers[i].equals("0513")){
                taxNumber = taxNumber + 5.13;
            } else if (imageNumbers[i].equals("0090") || imageNumbers[i].equals("0092") ||
                    imageNumbers[i].equals("0088")){
                taxNumber = taxNumber + 5.30;
            } else if (imageNumbers[i].equals("0021") || imageNumbers[i].equals("0288") ||
                    imageNumbers[i].equals("0401")){
                taxNumber = taxNumber + 5.50;
            } else if (imageNumbers[i].equals("0506") || imageNumbers[i].equals("0504")){
                taxNumber = taxNumber + 5.60;
            } else if (imageNumbers[i].equals("0216") || imageNumbers[i].equals("0079")){
                taxNumber = taxNumber + 5.75;
            } else if (imageNumbers[i].equals("0503")){
                taxNumber = taxNumber + 5.95;
            } else if (imageNumbers[i].equals("0498") || imageNumbers[i].equals("0131") ||
                    imageNumbers[i].equals("0283") || imageNumbers[i].equals("0160") ||
                    imageNumbers[i].equals("0154") || imageNumbers[i].equals("0285") ||
                    imageNumbers[i].equals("0203") || imageNumbers[i].equals("0205") ||
                    imageNumbers[i].equals("0079") || imageNumbers[i].equals("0268") ||
                    imageNumbers[i].equals("0260") || imageNumbers[i].equals("0491") ||
                    imageNumbers[i].equals("0073") || imageNumbers[i].equals("0277") ||
                    imageNumbers[i].equals("0128") || imageNumbers[i].equals("0085") ||
                    imageNumbers[i].equals("0018") || imageNumbers[i].equals("0099")) {
                taxNumber = taxNumber + 6.00;
            } else if (imageNumbers[i].equals("0346") || imageNumbers[i].equals("0461") ||
                    imageNumbers[i].equals("0005") || imageNumbers[i].equals("0377") ||
                    imageNumbers[i].equals("0437") || imageNumbers[i].equals("0472") ||
                    imageNumbers[i].equals("0358") || imageNumbers[i].equals("0372") ||
                    imageNumbers[i].equals("0470")) {
                taxNumber = taxNumber + 6.25;
            } else if (imageNumbers[i].equals("0022")){
                taxNumber = taxNumber + 6.35;
            } else if (imageNumbers[i].equals("0422") || imageNumbers[i].equals("0396") ||
                    imageNumbers[i].equals("0192") || imageNumbers[i].equals("0393") ||
                    imageNumbers[i].equals("0424") || imageNumbers[i].equals("0563") ||
                    imageNumbers[i].equals("0427") || imageNumbers[i].equals("0498")) {
```

```
                taxNumber = taxNumber + 6.5;
            } else if (imageNumbers[i].equals("0519") || imageNumbers[i].equals("0286") ||
                    imageNumbers[i].equals("0308")){
                taxNumber = taxNumber + 6.85;
            } else if (imageNumbers[i].equals("0308") || imageNumbers[i].equals("0286")){
                taxNumber = taxNumber + 6.88;
            } else if (imageNumbers[i].equals("0250") || imageNumbers[i].equals("0241") ||
                    imageNumbers[i].equals("0197") || imageNumbers[i].equals("0257") ||
                    imageNumbers[i].equals("0195") || imageNumbers[i].equals("0198") ||
                    imageNumbers[i].equals("0006") || imageNumbers[i].equals("0028") ||
                    imageNumbers[i].equals("0178")) {
                taxNumber = taxNumber + 7.00;
            } else {
                taxNumber = taxNumber + 7.5;
            }
        }
        System.out.println(Math.round(taxNumber));
    }

    public static void PartIII(String[] urls){

        int n = urls.length;
        String[] imageNumbers = new String[n];
        double taxNumber2 = 0;


        for (int i = 0; i < urls.length; i++){
            String imageNumber = urls[i].substring(urls[i].length() - 8, urls[i].length() - 4);
            imageNumbers[i] = imageNumber;
        }

        for (int i = 0; i < imageNumbers.length; i++){
            if (imageNumbers[i].equals("0045") || imageNumbers[i].equals("0006") ||
                imageNumbers[i].equals("0005") || imageNumbers[i].equals("0550") ||
                imageNumbers[i].equals("0334") || imageNumbers[i].equals("0334") ||
                imageNumbers[i].equals("0028") || imageNumbers[i].equals("0073") ||
                imageNumbers[i].equals("0018") || imageNumbers[i].equals("0079")){

            } else {
                taxNumber2++;
            }
        }

        System.out.println(taxNumber2);
    }
```

Extend Table of Solutions When N = 64

			Solutions for N=64				
Solution 1			Solution 2			Solution 3	
Map	State		Map	State		Map	State
map_0168	Alabama		map_0288.gif	Nebraska		map_0021	Maine
map_0543	California		map_0250.gif	Indiana		map_0131	Georgia
map_0377	Illinois		map_0493.gif	Idaho		map_0018	Vermont
map_0503	Utah		map_0260.gif	Michigan		map_0197	Mississippi
map_0334	Montana		map_0563.gif	Washington		map_0407	Louisiana
map_0198	Mississippi		map_0497.gif	Idaho		map_0283	Iowa
map_0018	Vermont		map_0519.gif	Nevada		map_0073	Pennsylvania
map_0216	Ohio		map_0401.gif	Nebraska		map_0205	Kentucky
map_0195	Mississippi		map_0370.gif	Missouri		map_0006	Rhode Island
map_0322	South Dakota		map_0491.gif	Idaho		map_0184	Tennessee
map_0396	Kansas		map_0045.gif	New York		map_0484	Wyoming
map_0536	California		map_0436.gif	Oklahoma		map_0491	Idaho
map_0489	Wyoming		map_0422.gif	Arkansas		map_0326	South Dakota
map_0384	Missouri		map_0396.gif	Kansas		map_0488	Wyoming
map_0241	Indiana		map_0506.gif	Arizona		map_0506	Arizona
map_0498	Washington		map_0543.gif	California		map_0099	West Virginia
map_0491	Idaho		map_0322.gif	South Dakota		map_0138	Georgia
map_0313	North Dakota		map_0277.gif	Michigan		map_0422	Arkansas
map_0472	Texas		map_0170.gif	Alabama		map_0288	Nebraska
map_0286	Minnesota		map_0427.gif	Arkansas		map_0116	North Carolina
map_0006	Rhode Island		map_0154.gif	Florida		map_0510	New Mexico
map_0197	Mississippi		map_0377.gif	Illinois		map_0543	California
map_0550	Oregon		map_0079.gif	D.C.		map_0260	Michigan
map_0170	Alabama		map_0475.gif	Colorado		map_0424	Arkansas
map_0116	North Carolina		map_0192.gif	Arkansas		map_0268	Michigan
map_0522	California		map_0461.gif	Texas		map_0370	Missouri
map_0306	Wisconsin		map_0178.gif	Tennessee		map_0128	South Carolina

map_0407	Louisiana		map_0510.gif	New Mexico		map_0550	Oregon
map_0085	West Virginia		map_0498.gif	Idaho		map_0092	Virginia
map_0154	Florida		map_0503.gif	Utah		map_0493	Idaho
map_0088	Virginia		map_0550.gif	Oregon		map_0503	Utah
map_0326	South Dakota		map_0021.gif	Maine		map_0079	Delaware
map_0461	Texas		map_0007.gif	New Hampshire		map_0377	Illinois
map_0250	Indiana		map_0415.gif	Louisiana		map_0461	Texas
map_0427	Arkansas		map_0504.gif	Arizona		map_0513	New Mexico
map_0277	Michigan		map_0099.gif	West Virginia		map_0519	Nevada
map_0113	North Carolina		map_0536.gif	California		map_0088	Virginia
map_0160	Florida		map_0428.gif	Oklahoma		map_0203	Kentucky
map_0138	Georgia		map_0346.gif	Illinois		map_0498	Idaho
map_0437	Texas		map_0085.gif	West Virginia		map_0475	Colorado
map_0073	Penn Sylvia		map_0334.gif	Montana		map_0543	California
map_0401	Nebraska		map_0138.gif	Georgia		map_0536	California
map_0021	Vermont		map_0160.gif	Florida		map_0250	Indiana
map_0506	Arizona		map_0018.gif	Vermont		map_0160	Florida
map_0490	Idaho		map_0489.gif	Wyoming		map_0085	West Virginia
map_0370	Missouri		map_0372.gif	Illinois		map_0415	Louisiana
map_0519	Nevada		map_0413.gif	Louisiana		map_0384	Missouri
map_0413	Louisiana		map_0326.gif	South Dakota		map_0306	Wisconsin
map_0393	Kansas		map_0198.gif	Mississippi		map_0007	New Hampshire
map_0268	Michigan		map_0424.gif	Arkansas		map_0427	Arkansas
map_0563	Washington		map_0131.gif	Georgia		map_0489	Wyoming
map_0475	Colorado		map_0522.gif	California		map_0334	Montana
map_0510	New Mexico		map_0384.gif	Missouri		map_0178	Tennessee
map_0424	Arkansas		map_0203.gif	Kentucky		map_0277	Michigan
map_0257	Indiana		map_0090.gif	Virginia		map_0286	Minnesota
map_0022	Connecticut		map_0028.gif	New Jersey		map_0346	Illinois
map_0543	California		map_0006.gif	Rhode Island		map_0497	Idaho

map_0028	New Jersey		map_0241.gif	Indiana		map_0322	South Dakota
map_0504	Arizona		map_0216.gif	Ohio		map_0522	California
map_0079	Delaware		map_0472.gif	Texas		map_0470	Texas
map_0493	Idaho		map_0184.gif	Tennessee		map_0393	Kansas
map_0090	Virginia		map_0286.gif	Minnesota		map_0504	Arizona
map_0346	Illinois		map_0088.gif	Virginia		map_0154	Florida
map_0288	Nebraska		map_0132.gif	Georgia		map_0472	Texas
Sum of State Taxes	367		Sum of State Taxes	353		Sum of State Taxes	355
Number of States	58		Number of States	57		Number of States	59
With Clothing Tax			With Clothing Tax			With Clothing Tax	