PURPLE AMERICA

COMP446 – Algorithms Design and Analysis

Deniz Toprak - 36235

20 ARALIK 2017

TABLE OF CONTENTS

ASSIGNMENT DESCRIPTION

HISTORICAL INFORMATION

During coverage of the 2000 presidential election, Tim Russert coined the political terms red states and blue states to refer to states that predominantly vote for the Republican presidential candidate (red) or the Democratic presidential candidate (blue). The news media use red-state blue-state maps, such as the one below, to display election results. (Wayne)

PROBLEM DESCRIPTION

We are provided with geographic data that describes the boundary of each state and country in United States of America and election return data that describes the results for each presidential election, by state and county in .txt format. And the assignment consists of 3 parts.

WHITE AMERICA

Writing a program White.java that takes the name of a region as the command-line argument and produces an outline map.

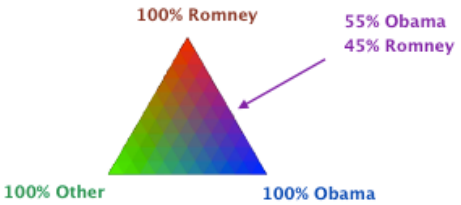
REDBLUE AMERICA

Write a program RedBlue.java that takes two command-line arguments (the name of the region and the year of the election) and produces a red-state blue-state map.

PURPLE AMERICA

Write a program Purple.java that takes two command-line arguments (the name of the map and the year of the election) and produces a Purple-America map. Each region is colored in a shade of red, green, and blue, according to the proportion of votes received by each candidate. If the Republican, Independent, and Democratic candidates receive a1, a2, and a3 votes, respectively. Formula to decide coloring of each region:

$$(R,\ G,\ B) = \left(\frac{a_1}{a_1 + a_2 + a_3},\ \frac{a_2}{a_1 + a_2 + a_3},\ \frac{a_3}{a_1 + a_2 + a_3} \right)$$



RUNNING CONDITIONS

Running conditions are important because later we will discuss about performance and optimization solutions.

MACHINE

- Processor: Intel Core i7-6700HQ @ 2.60 GHZ
- RAM: 16,0 GB
- Graphics Card: Nvidia GeForce GTX 965M 4GB

RUNNING ENVIRONMENT

- Operating System: Windows 10
- IDE: Eclipse Oxygen
- Runtime Environment: jre-8u151
- External Libraries: ACM

SOLUTION DESCRIPTION AND ANALYSIS

There are 3 main programs: DataReader.java, ElectionData.java and Test.java. Test.java is the main controller, runs scanner and waits for an input from user. It waits 3 types of different inputs. First type starts with “white” string, and name of region in USA. It can be whole USA, USA-county or any state. This input runs white.java program. The second type consists “RedBlue” string, name of a region in USA and a valid election year. This input runs RedBlue.java program. The third type consists “Purple” string, name of a region in USA and a valid election year. This input runs RedBlue.java program. If the input is given incorrectly it will produce errors accordingly. DataReader takes an argument of file name and reads the appropriate file, returns the file data in ArrayList<String>.

WHITE AMERICA

Creates an instance of DataReader class and gathers latitude and longitude values from the.txt file of the region. Draws the region with ACM GraphicsProgram according to their latitude and longitude values.

REDBLUE AMERICA

Creates an instance of DataReader class and gathers latitude and longitude values from the.txt file of the region. Draws every sub region in the region with ACM GraphicsProgram according to their latitude and longitude values. For every sub region it reads the election results from the according txt file. Stores each sub region’s information, election results and drawn shape in a Hash Map. This Hasp Map’s key is the shape (GPolygon object) and value is ElectionData object. This

Hash Map is for GUI interactivity. Then the sub region is colored with the assigned color of the winner party.

Whenever user clicks on a sub region, the program shows information about the clicked sub region.

PURPLE AMERICA

Creates an instance of DataReader class and gathers latitude and longitude values from the.txt file of the region. Draws every sub region in the region with ACM GraphicsProgram according to their latitude and longitude values. For every sub region it reads the election results from the according txt file. Stores each sub region's information, election results and drawn shape in a Hash Map. This Hasp Map's key is the shape (GPolygon object) and value is ElectionData object. This Hash Map is for GUI interactivity. Then the sub region is colored according to votes each party got.

Whenever user clicks on a sub region, the program shows information about the clicked sub region.

ANALYSIS

My program uses names of all regions and their coordinates, election result and user's mouse clicks. And outputs Polygons filled with according color and information and election result display.

There were some problems I encountered. I put this part in analysis part because these problems led me to use different solutions to the assignment. Firstly, some names of cities were not the same in geographic data txt and election result txt. Also, some were not in both. I ignored missing data problems and left them filled white. Some inputs were uppercase in geographic data but lowercase in election result. So, I always ignored case. But the main problem was, some cities were written differently in both input txt files. Especially in state Louisiana (LA). Every city was ending with word "Parish" in geographic data but without in election result data. I couldn't just check first string separated with white space, because some were 3-word names, some were 2. I thought about getting rid of the last word, but some other cities had the same problem and this solution was not working for them. So, I came up with a different solution. Because 90% of the data did not have this problem, while searching for a city in election result list, I used linear search and if found return the value. But if it is not found I separate every word and try searching for them. To give more information: I create a string with only first word of the search string and search for it. If found return. If not add the next string and continue. Until it finds the searched election result. It was working with worst case O(4*n) = O(n). Expected running time was O(n). This algorithm runs for every sub region so, real expected running time is O(n) * (# of sub regions). Space complexity is O(1).

I discovered that all the inputs in both geographic and election result data were sorted in alphabetical order. I tried to use this information, but as I said before some data were missing in the files. Also, I found that some data were extra. Some cities had two election results in a year. And some were out of order (my discovery was wrong). So, my new solution did not work. Input data mistakes were forcing me to use a search algorithm.

I had to use a search algorithm, but linear search was inefficient. I wanted my search algorithm to run as fast as it can, so I sorted my input data properly. I used merge sort for this cause. I chose merge sort because it is an in-place sort algorithm (does increase space complexity) and running time is O(nlogn). While searching for election data of every city or region I used binary search algorithm. Running time of it is worst case O(logn) and space complexity is O(1). For my algorithm it has running time of O(nlogn) + O(4*logn). And it is multiplied by number of regions.

This solution did effect running time. It decreased from ~1600ms to ~1000ms. But I knew the election result data was nearly sorted. I was spending too much time while sorting it. I again did binary search, but this time the chance of not finding increased. So, the algorithm continues as: If not found search linearly in the array. My new time complexity in worst case is O(4*logn) + O(n). And it is multiplied by number of regions. And running time decreased to ~800ms.

I wanted to show both my algorithms. So, the last solution is coded in Purple.java and the other is in RedBlue.java.

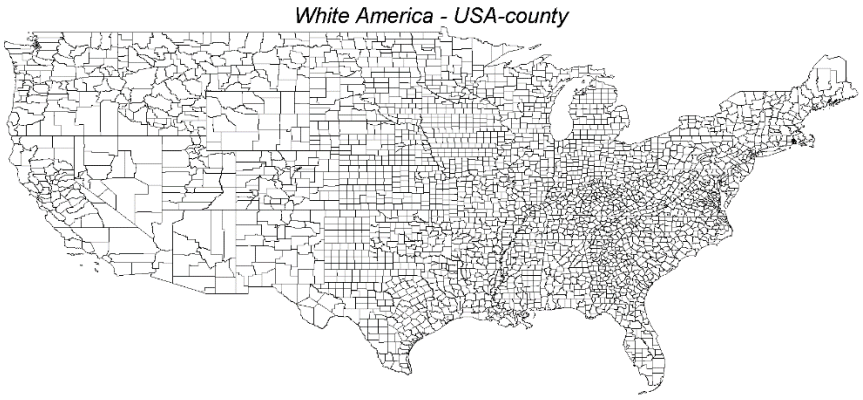
For some extra credit I implemented a GUI interaction to show information of clicked sub region. I briefly explained it before, now I will analyze it. Whenever user clicks on a sub region, the program shows information of the specific sub region. Information consists name, state which it is in and election results. Implementation: Whenever the program receives the election results. It saves it in a Hash Map with its shape on the map. (Key = GPolygon, Value=Information). Information is stored as ElectionData(String name, String republicanVotes, String democratVotes, String otherVotes, Sting state) object. I did not focus on the optimization of this part, because it does effect running time of the whole program. It only creates a minimal delay between click and display and it is not even visible. Because of these reasons I chose Hash Map to store these values. According to my research, hashMap.get(key) function runs with worst case O(n). And in my opinion, it is good. Even binary search I implemented is working with worst case O(logn). Also, this search is per click of user in maximum number of elements 3206 according to our data. Possibly I could use a AVL tree instead of Hash Map to increase performance. But as I said it is not needed.

CONCLUSION

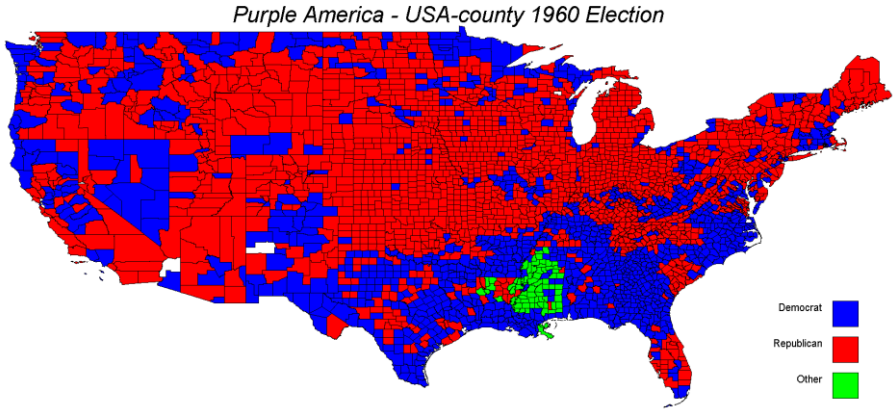
This assignment was considerable easy for this course. But this easiness left me more time to think about how to improve it. The hardest part was optimizing process of finding appropriate election result data in the input list. I three different algorithm designs to have a better solution. The most important thing I learned while doing this assignment is in some cases expected running time can be more important than the worst case of an algorithm. The running time does not only depend on algorithm design but also on the type and structure of the input your algorithm is working on. Knowledge I learned in learned in class COMP446 helped me a lot in this assignment especially while implementing search and sort algorithms. While designing my algorithm I already knew how will it result with what time and space complexity.

RUNNING RESULT IMAGES

Input: White USA-county 1960



Input: RedBlue USA-county 1960



Input: Purple USA-county 1960