Project 3: Mass Extinction Calculator

Administrative:
Team 21
Dimitri Distefano and George Rauta
GitHub Link: https://github.com/DimitriDistefano/COP3530Project3
Video Link: https://www.youtube.com/watch?v=k45e1sXuHyA

Proposal:

The problem we are trying to solve is the issue of disastrous climate change caused by increasing greenhouse gas emissions. In our simulation, we are trying to represent this issue of global warming. Essentially, as world populations continue increasing and as those populations continue producing more and more emissions, there is a greater and greater build-up of those emissions in the atmosphere. The increase of these emissions in the atmosphere leads to increases in global temperatures until, at a certain threshold, temperatures increase by such an amount that global warming will accelerate no matter how much we reduce emissions. Since these emissions are fairly difficult to remove from the atmosphere and will only probably become harder to remove as global warming progresses, it is crucial that we take steps to reduce how many emissions we release into the atmosphere. This is the problem our simulation is trying to represent.

This is a major problem because most people don't realize just how close this impending disaster is. But another thing they also don't realize is how small but constant changes can drastically shape how this disaster will play out. Through this simulation, users can learn to realize these aspects of the crisis and figure out different scenarios that may not only postpone this apocalypse but stop it altogether.

Features:

- Simple to use UI
- Can run as many simulations as is desired
- Can change global emission and population change percents for each simulation
- Can change the above percents for each country for each simulation
- Receives a time estimation of when a critical temperature is reached
- Displays "Little Timmy", a quick visual reflection of how dire the simulation is
- Can search for data relating to a specific country on a specific date for each simulation
- Can search for the first data point where a specific temperature is reached for each simulation
- Displays operation time for each data structure for each button

<u>The data we used</u>

- The top 20 countries in CO2 emissions (we used the top 12)
- Equation for global temperature increase (this is a very rudimentary equation for a complex, multi-layered problem)
- Little Timmy (was edited slightly)

<u>Tools/Languages/APIs/Libraries used</u>

- Visual Studio C++ Windows Form Application was used for the UI of the calculator
- Languages: All code written by group members were in C++
- No APIs were used for this project
- The standard C++ library and the date/time library is used

We implemented two separate data structures. It is important to note that both of them used an ID as a way to sort the data. This ID consists of two digits denoting a unique country or global data and four digits denoting how many months have passed since the start of the simulation. One of the data structures implemented and used was a hash table using vectors as the foundation and using a special open hashing collision resolution algorithm that would work great with our data set. Using the first two digits of the ID to hash the data would place it in that specific spot in the hash table. It would be re-sizable to make it easier on us if we wanted to use a different amount of countries. After that, we would use the last four digits to place it in a specific location in the bucket of a specific country. The data is added in the bucket in a similar way to the country data but since the data is generated and stored sequentially, by date, it's equivalent to simply pushing back data on a bucket. The careful creation and knowledge of our data set means that we can always extract country and date information instantly and that all country information is stored in the same bucket.

The other data structure used and implemented was a Red-Black tree. It was implemented using a Node class that would store an ID, data value, left/right child, and its parent. It being a BST tree means that by inserting IDs into the tree, we could order Nodes by country and sequentially by dates and then use an inorder traversal to go through data in chronological order. Searching for a specific country and date for information would be the same as a regular BST search algorithm. The reason for using a Red-Black tree would be that since it is well balanced, there is no worry about skewing which actually would have been a problem considering our IDs are added sequentially by dates. This heavy right skewing is solved by the balancing properties of the red-black tree and makes insertions and accessing significantly faster.

George Rauta was in charge of implementing and testing the data structures to be used by the program. He created both the hash table as well as the Red-Black tree and various methods that would be needed by the simulation, such as accessing a specific country and date ID or searching for the first instance of a temperature. A deletion method was also needed for the Red-Black tree whenever the simulation would refresh.

Dimitri Distefano was in charge of creating the GUI elements of the simulation as well as generating the data based on user inputs. He created various fields where users can input percentages for both emission and population change, one for a global percentage and one for each country. There are also fields that allow the user to search the data and display the received information and a display of “Little Timmy”, showing how dire the simulation is. For the generation of the data, he created various equations that would take in the user percentages and calculate monthly changes in emission which would be converted to a temperature change.

<u>Analysis:</u>

One major change that had to be made to the project was the type of tree that would be implemented. Our original idea was to implement an AVL tree because of its balancing properties allowing for quick insertions and searching because our previous experiences in implementing the tree would make it easy to do so. However, we were told we couldn't use an AVL tree because it was used in a previous project, so we ending up deciding to use a Red-Black tree instead. Its properties would be similar, it would be balanced and have quick insertions and searches, but the way it balances means different algorithms have to be implemented to allow it to function. At the very least a Red-Black tree is balanced faster than an AVL tree so it probably makes the insertions slightly faster than AVL and thus saves time.

<u>Worst Case Time Complexities:</u>

Void HashTable::insert(int ID, double placeholder)
Worst case is O(1) since the data is added in sequentially, meaning that the worst resize that must occur is one space and data is inserted directly into the correct spot.

dataNode HashTable::search(int ID)
Worst case is O(1) since the data can always be instantly accessed. If the ID doesn't exist then we can quickly tell if it goes out of bounds and returns empty data.

int HashTable::searchVal(double val)
Worst case is O(n), where n is the number of data points recorded. Since temperature data is isn't part of the ID, it is impossible to use a hash to instantly find the needed value and so we must search sequentially through all of the data. What makes improves things is that temperature is only in global data, so only a small fraction of all the data points must be searched through.

RBtree::Node* RBTree::insert(int ID, int placeholder)
Worst case would be O(log(n)), where n is the number of data points recorded. Since the tree is always balanced, it would take at most "log(n)" nodes before a proper place is found for a value. After that the node must be found again using a BST search which is O(log(n)) time and the tree needs to be balanced. Each check and rotation only takes O(1) time and much be performed on at most "log(n)" nodes as it recurses back up so it is over all O(log(n)).

dataNode RBTree::search(int ID)
Worst case is O(log(n)) since it uses a BST search on a balanced tree which only needs to search through at most "log(n)" nodes before finding the required node.

int RBTree::searchVal(double val)
Worst case is O(n), where n is the number of data points recorded. Since temperature data is isn't part of the ID, we will need to traverse the tree in inorder to access the data points sequentially. Again, temperature data is only on global points so we only need to search through a fraction of the nodes before returning. However, we still do need to traverse all global nodes.

All other functions just implemented these functions and their time complexity would have come from them too.

<u>Reflection:</u>

As a group, the experience for the project was quite good. We met up several times online to work on the project and had very few problems in scheduling these meetings. We both worked diligently on our assigned parts, both of which were approximately equal work-wise and well made. There were no major delays on anybody's part but if the other was having some kind of issue, maybe concerning some kind of programming error or figuring out the logic for an algorithm to be implemented, the other would be there to assist. We both feel that the result of our work is something to be proud of. The GUI is user-friendly, the simulation runs quite fast, and there is enough complexity to allow the user to experiment on various scenarios they have come up with.

One of the challenges that we encountered was that if the percentage settings were set too high then we would have very few data points to work with and that if it were set too low, then the simulation would never end as the final temperature would never be reached. At first, we thought about making the simulation run daily but if that were the case, the user would only be able to change the new daily percentages by a minuscule amount or they blow up the simulation. Besides, having it daily makes it harder to comprehend what the simulation is trying to say. For example, we believe it is easier to understand a 1% increase over a month rather than a 0.1% increase every day. Instead, to solve this, we have the simulation always run for a fixed amount of times to generate enough data points but also stop if the simulation enters an infinite loop.

If we were to start the project over, we don't think there would be many things that we would change. If we did have a full group of three instead of the two we ended up with we might have been able to fit more features into the final simulation. At the same time, however, we believe that we have hit upon the more important features that we could possibly add and so aren't too torn about the situation.

I, George Rauta, learned a few things through this project. One relates to the results of the project, specifically how the simulation shows just how quickly small changes in our behaviors can compound in the many months that pass. Just a few tenths of a percent change can either prolong the disaster to many hundreds of years down the road or quicken it to within a few decades. It highlights just how important the issue is but also how change doesn't have to be drastic, just constant. The other relates to the actual project. The creation of a specialized ID helped us make the simulation more useful by having the data structures be able to quickly search for specific values. On data structures, I also learned a lot about them specifically the Red-Black tree I had to implement. Through implementing it, I was able to appreciate just how the tree functions and how useful it is for storing and accessing many different nodes.

I, Dimitri Distefano, had a good experience on this project. I got to employ a lot of the programming related skills I have learned in my education, and it did not feel tedious doing so. This project felt a lot easier to work on than other ones, because of the control of the topic and execution of that topic. Some of the challenges for me in this project were finding and implementing an equation that could roughly predict temperature based on CO2 emissions. The equation used definitely cannot account for what seems like infinite amount of variable that effect climate change, like other more severe forms of pollution and the effects of evaporation into the atmosphere. I learned how to use a windows form application in C++, which seems valuable to know. Through this project, I deepened by understanding of climate change and had a pleasant experience.

References:

- https://www.ucsusa.org/resources/each-countrys-share-co2-emissions
- https://www.worldometers.info/world-population/population-by-country/
- https://en.wikipedia.org/wiki/List_of_countries_by_carbon_dioxide_emissions