Joey Rockwood

Dr. Khadka

CS 2103.01

10.21.2016

Address Sorter

This program allows the user to choose from four specified files that contain either 100, 200, 500, or 1000 address. The user will then be able to select a specific piece of the address to sort, or the user can choose to sort the address by State then city. This way of sorting was done because each state would be alphabetically sorted, then the cities within repeating states will be sorted alphabetically as well. It uses a timer to calculate the number of clock cycles from when the sort is called to when it finishes the sorting. I’ve calculated the seconds for both, the time taken as a whole and the time taken for each input.

The algorithm used was merge sort. This sort is special in the way it works, so it allows to set priority of sorting. This is how the program is able to perform the nested sorting without resorting over previously sorted information. Merge sort is a conservative sort that prevents the sorted information from being overwritten when sorted again if done in order. This allows for the program to sort the information in a nested fashion while being simple also.

**Program written out**

1. struct address{

string name;

string street;

string city;

string state;

string zip;

};

1. User chooses file to sort
2. User chooses method of sorting. i.e. Sort by name, street, city, state, zip, or nested (state then city)
3. Set a sortChoice equal to the value of the sort they chose
4. Open the file
5. Read each line for the length of the entire document
6. Create an array of struct type, address, address arr[bigsize];
7. Within a while loop(file.good())
   1. Getline(file, line, ‘ | ‘) searches for each bar and pulls the separate parts of the document and assigns them to the piece of the struct they belong. i.e. arr[i].name = line;
8. Close file
9. Begin timer, timer = clock()
10. Call mergeSor(arr, 1, arrSize, sortChoiceNum);
11. Determine if nested sorting is required,
12. If sortChoiceNum == 6, merge sort with sortChoiceNum as 3 first to sort by City, then sort by 4 which is state. Merge sort allows the previously sorted information to be saved.
13. End Timer, timer = clock() – timer; //subtract timer from clock to get total time taken.
14. Display the clock cycles, which is the default fomat.
15. Display the seconds, (float)timer /clocks\_per\_sec
16. Display timer per input, timer/arrSize for clock cycles and (float)timer /(clocks\_per\_sec \* arrSize)
17. mergeSort(address arr[], int low, int high, int sortChoice)
    1. address outputArr[bigNumber]
    2. originalLow = low;
    3. outputLow = low;
    4. midpoint = mid +1;
       1. while((originalLow <= mid) AND (midpoint <= high))
       2. switch case to determine what sort choice was selected. (1-6)
       3. ex. Case1: //sort by name
          1. if(arr[originalLow].name <= arr[midpoint].name)

{

outputArr[outputLow] = arr[originalLow];

originalLow++;

}

Repeat for each case (street, zip, nested, etc.)

* After midpoint is reached, set the outputArr[outputLow] = arr[comparison], where comparison is the midpoint.

if (originalLow>mid)

{

for (comparison = midpoint; comparison <= high; comparison++)

{

outputArr[outputLow] = arr[comparison];

outputLow++;

}

}

else

{

for (comparison = originalLow; comparison <= mid; comparison++)

{

outputArr[outputLow] = arr[comparison];

outputLow++;

}

}

for (comparison = low; comparison <= high; comparison++)

arr[comparison] = outputArr[comparison];

* table that shows the execution time for each input for the algorithm
* performance graph (running time vs. input).
* The printArray uses ofstream out to save the sorted information to a text file called sorted.txt
  + Use for loop, I = 1, I < arrSize
  + Cout << arr[i].name << “ | “ << arr[i].street << “ | “ …. Etc.
* The printArray uses a stringstream buffer to read from the ouput textfile(sorted.txt) and read it back into the terminal. Convert the buffer.str() in order for it to be viewed.
* lineCount(string file)
  + ifstream myfile(file.c\_str());
  + string line.
  + Int lineNum =0;
  + While(getline(myfile, line))
    - ++lineNum;
  + Cout << lineNum //Total Num of Lines
  + Return lineNum;

**Time for nested sort of each file**