

CS1110 – Spring 2017 – “Distance Between World Capitals”

Asgn 6 - 100 points – due in two weeks

Objectives – learn/practice the following new concepts:

- Input & output data files
- Calculating stats on data
- use a pre-existing method and call it
- Using single dimensional array as two-dimensional array
- printf's for creating nicely formatted report

BASIC REQUIREMENTS

- A top comment including your name, lab section day/time, asgn number & name of application
- Appropriate descriptive variable naming, including use of camel case (and variables start with a small letter)
- incremental development
- follow these written specs (and other things specified by your lab instructor)
- output goes to a DATA FILE (though you MAY write to the Console window for your own debugging work)
- follow written specs in terms of which algorithms/approaches to use and the output formatting, etc.
- indent/align program code to show program logic (use NetBeans' dropDown menu Source, then choose Format)
- do not type beyond the red line in NetBeans

PROJECT OVERVIEW

This project uses some of the data from the WorldCapitals.csv file to create the Nearest Neighbors Report (written to the data file).

INPUT FILE: WorldCapitals.csv (a text file which has “Comma Separated Values”). This file contains basic data for every country's capital city in the world¹ (see footnote). The 1st record (line) contain “meta-data”. The rest of the records contain the actual data. Each data record contains these fields, in this order:

Country Name
Capital Name
Capital Latitude
Capital Longitude
~~Country Code~~
Continent Name

Ignore the records that have been removed (stroked) above.

PROCESSING :

1. Calculate intercity distances for all possible combination of world capital cities. Use Haversine distance to calculate pair wise distance. For example, consider two cities, A (lat: 3, long: 10) and B (lat: 12, long: -5). Then distance between cities A & B using Haversine Formula distance is given as

$$dlon = lon2 - lon1$$

$$dlat = lat2 - lat1$$

$$a = (\sin(dlat/2))^2 + \cos(lat1) * \cos(lat2) * (\sin(dlon/2))^2$$

$$c = 2 * \text{atan2}(\text{sqrt}(a), \text{sqrt}(1 - a))$$

¹ Data taken from <http://techslides.com/list-of-countries-and-capitals> list is per 2013.

$Distance(A, B) = R * c$ (where R is the radius of the Earth)

Create a method called `calcDistance()` to do the above calculation. Method takes latitude of city A, longitude of city A, latitude of city B and longitude of city B. Calculated distance should be returned from the method.

2. Store distance values for only unique pairs of cities. As $Distance(A, B)$ is same as $Distance(B, A)$, we only need to calculate and store $Distance(A, B)$. To do this follow below steps
 - a. Create three parallel arrays, **two of type integer and one of type double** (which will be used later to generate report) in the main method scope. Length of these parallel array should be $N(N + 1)/2 - N$. Where N is representing the total number of capitols. Let's call the double parallel array used to store intercity distances as `distArr`. Other two integer parallel arrays used to store indices as `idxArr1` and `idxArr2`. Details is described in Appendix.
 - b. Record unique combination of source – destination city indices in the two integer parallel arrays created in step (a).
 - c. Use the integer parallel arrays to compute distance between two cities. Store the distance result in corresponding index of double parallel array.
3. Copy `distArr` to a new double array of same size called `sortedDistArr`.
4. Sort the `sortedDistArr` using either *insertion* or *selection* sort. As you sort the `sortedDistArr` make same changes in parallel arrays `idxArr1` and `idxArr2` to reflect the sorting.
5. Create a method called `findNN()` which takes above 3 parallel arrays (`idxArr1`, `idxArr2` `distArr` and `sortedDistArr`) as argument and returns a double array of size N . In the NN array value at i^{th} index presents the closest neighboring capitol city's distance.
 - a. To find nearest neighbor for city with index i , first create a temporary array called '*neighborsDist*' for size N to which you will copy distance from i to all other cities.
 - b. Find minimum distance values in the '*neighborsDist*' array call it `minDist`.
6. Search the `sortedDistArr` for the index at which `minDist` was found (*binary search*). Corresponding source and destination cities index are found in arrays `idxArr1` and `idxArr2`.
7. Translate source – destination index to a string for in the below format and generate output file called `neighbors_list.txt`
Country X's capital City A is closest to country Y's capital City B.

Note: Bonus points if you can tell if both cities are in same continent or different continent. For example, output file, should contain

Country X's capital City A is closest to country Y's capital City B. Cities are in same continent.
Or

Country X's capital City A is closest to country Y's capital City B. Cities are in different continent.

OUTPUT REPORT FILE: neighbors_list.txt – looks like the following:

[NOTE: The data below **isn't necessarily accurate**, I'm just showing you the required formatting].

[NOTE: Use this exact formatting].

WORLD CAPITOLS NEAREST NEIGHBORS LIST

Country X's capital City A is closest to country Y's capital City B.

Country U's capital City C is closest to country V's capital City D.

.

.

.

Country E's capital City H is closest to country F's capital City G.

APPENDIX:

Consider following instance example, following cities with given latitude and longitude

City Name	City Index	Latitude	Longitude
My City A	1	2	6
My City B	2	5	3
My City C	3	8	9
My City D	4	7	9

**City names, latitude and longitude is hypothetical. In the given data file city index is not given and your program must give indices to world capital cities.

Three parallel arrays created for above example should look like below

Integer parallel array 1 (idxArr1)

Integer parallel array 2 (idxArr2)

Double parallel array for distance
(distArr)

2	3	3	4	4	4
1	1	2	1	2	3
4.242641	6.708204	6.708204	5.830952	6.324555	1

As you can see max size of parallel array is $4(4 + 1)/2 - 4 = 6$.

sortedDistArr in this example will look like below

1	4.242641	5.830952	6.324555	6.708204	6.708204
---	----------	----------	----------	----------	----------

NN list for each city will look like below. Remember in the example below row with index is explicitly printed, however in your implementation this will be array index.

1	2	3	4
4.242641	4.242641	1	1

Assignment Submission

- Your project folder should be named la6cs1110_yourlastnameFirstInitial_mmddyy, Replace “yourlastnameFirstInitial” and “mmddyy” appropriately.
- Generate a .zip file that contains all of your files in the above java project folder.
- Submit the .zip file via E-learning