Feb 21, 2022

ShortestSplitlineAlgorithm

Jamal Thomas[1]

[1]Ivan Ryan, s96abrar,

1

protocol .

Jamal Thomas

Takes the US Census results and geographic data and creates electoral districts that are iso-populous (same population in each district). This eliminates gerrymandering since it only uses openly available census data, with an algorithm that can be checked by anybody with a common computer.

https://rangevoting.org/SplitLR.html

Jamal Thomas 2022. ShortestSplitlineAlgorithm. **protocols.io** https://protocols.io/view/shortestsplitlinealgorithm-b5cwq2xe

protocol ,

Feb 20, 2022

Feb 21, 2022

58486

- Ubuntu 20.04.3 LTS or similar distro.

- Libglew 2.1.0 or newer.
- Freeglut 3.2.2 or newer.
- Gcc 11.2 or newer.
- Make 0.75 or newer.

*The above libraries can be installed on a Ubuntu system using the following commands:
sudo apt install libglew-dev freeglut3-dev -y
sudo apt install gcc make -y

- state geo_uf1, population uf1, and state district data (provided here: **State_Data.zip**)

:

Make sure to install the required libraries and Linux distros. Program requires sudo (root) access so check with your system administrator if elevations of permissions is needed.

1 Place **block_display.c** , **geoproc.cpp** , "MakeFile":

```
all: block_display geoproc

block_display:
 gcc -g -Wall block_display.c -o block_display -lGL -lglut -lm

geoproc:
 g++ -g -Wall geoproc.cpp -o geoproc

clean:
 rm -rf geoproc block_display *.o
```

into an new folder.
and "run":

```
#!/bin/sh

path=$(realpath ./)
if [ $# -gt 0 ]
then
 path=$(realpath $1)
fi
```

```
#********************** Check all the files **********************#
regions=(al ak az ar ca co ct de dc fl ga hi id il in ia ks ky la
me md ma mi mn ms mo mt ne nv nh nj nm ny nc nd oh ok or pa pr ri
sc sd tn tx ut vt va wa wv wi wy)
blockss=(01 02 04 05 06 08 09 10 11 12 13 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 72 44
45 46 47 48 49 50 51 53 54 55 56)
reg_prefix=00001_uf1.zip
geo_prefix=geo_uf1.zip
blk_suffix=st
blk_prefix=_d00_ascii.zip

display_param_1=(01 02 04 05 06 08 09 10 11 12 13 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
72 44 45 46 47 48 49 50 51 53 54 55 56)
display_param_2=(7 1 8 4 53 7 5 1 0 25 13 2 2 19 9 5 4 6 7 2 8 10
15 8 4 9 1 3 3 2 13 3 29 13 1 18 5 5 19 0 2 6 1 9 32 3 1 11 9 3 8
1)

function check_file_status() {
 if [[ ! -f $(echo $1/$2) ]]
 then
  echo ""
  echo "File $1 don't exists..."
  exit 1
 fi
}


echo "Checking binary files..."
check_file_status . block_display
echo "  block_display...OK"
check_file_status . geoproc
echo "  geoproc.........OK"
echo ""


i=0
echo "...Checking ${#regions[@]} region data file status...";
for region in ${regions[@]}
do
 echo "Checking region $region..."
 file1=$(echo $region$reg_prefix)
 file2=$(echo $region$geo_prefix)
 file3=$(echo $blk_suffix${blockss[$i]}$blk_prefix)

 check_file_status $path $file1
```

```
 echo "  $file1......OK"

 check_file_status $path $file2
 echo "  $file2........OK"

 check_file_status $path $file3
 echo "  $file3...OK"

 i=`expr $i + 1`
done

echo ""
echo "Check done!"
echo ""

#************************************************************#

#************************ Extract, generate block data, generate
image ***************************#

function unzip_file() {
 unzip -q -o -d $1 $2
 if [ $? -ne 0 ]
 then
  echo "Unziping $2 failed"
  exit 1
 fi
}

img_fol=images

mkdir -p out
mkdir -p $img_fol

i=0

echo "...Extracting, generating block data, generating image...";
for region in ${regions[@]}
do
 echo "Unziping $region..."
 file1=$(echo $region$reg_prefix)
 file2=$(echo $region$geo_prefix)
 file3=$(echo $blk_suffix${blockss[$i]}$blk_prefix)

 unzip_file $region $(echo $path/$file1)
 unzip_file $region $(echo $path/$file2)
 unzip_file $region $(echo $path/$file3)
```

```
 echo ""
 echo "Executing geoproc program..."
 cd $region
 ../geoproc $region 101 > pop_data.dat.blocks

 echo ""
 echo "Executing block_display program..."
 cd ..
 ./block_display $region ${display_param_1[$i]} blocks
${display_param_2[$i]} noimage &> log.txt

 echo ""
 echo "Storing generated image files..."
 mkdir -p $img_fol/$region
 mv $region/*.bmp $img_fol/$region
 mv out/*.bmp $img_fol/$region
 rm -rf $region

 echo ""
 echo "Converting bmp to png..."
 cd $img_fol/$region
 for j in *.bmp; do convert "$j" "`basename "$j" .bmp`.png"; done

 cd ../..

 echo ""
 i=`expr $i + 1`
done

rm -rf out


echo ""
echo "Images saved to `realpath $img_fol`"
echo ""
```

into the folder labeled: "State_Data"

2 Unzip the "State_Data" folder. (Warning: Be sure to only unzip the State_Data folder and nothing else as anything else will interfere with program execution)

3

Compile the "geoproc.cpp" and "block_display.c" program

3.1 Change permission to execute **geoproc.cpp** , **block_display.c** and "MakeFile" by using the following command:

```
chmod a+x geoproc.c
chmod a+x blockdisplay.c
chmod a+x Makefile
```

3.2 Compile both programs using the following command:

```
make MakeFile
```

4

Compile the "run" program in the State_Data folder

4.1 Change permission using the following command:

```
chmod a+x run
```

4.2 Run the "run" script using the following command:

```
bash run
```

OR

```
./run
```

5 Check for newly-created "images" folder for completed results. (Warning: If images are not present, check log file located in the "State_Data" folder for trouble-shooting help.)