CIV 201

COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 7

Searching

(C) Robert J. Vanderbei, Princeton University

Hints on Java Applets

Edit **`/u/yourname/.login`** and set **`umask`** to **`022`**.

Copy **`/u/civ201a/lab1/.rhosts`** to **`/u/yourname`**.

Create a directory **`/u/yourname/public_html/JAVA/myutil`**.

Copy all files in **`/u/ci201a/public_html/JAVA/myutil`** to your **`myutil`** directory.

Use **`chmod`** to set permissions on **`/u/yourname/public_html`** and *every directory thereunder* to **`drwxr-xr-x`**.

Use **`chmod`** to set permissions on every file in **`/u/yourname/public_html`** and *in every directory thereunder* to **`-rw-r--r--`**.

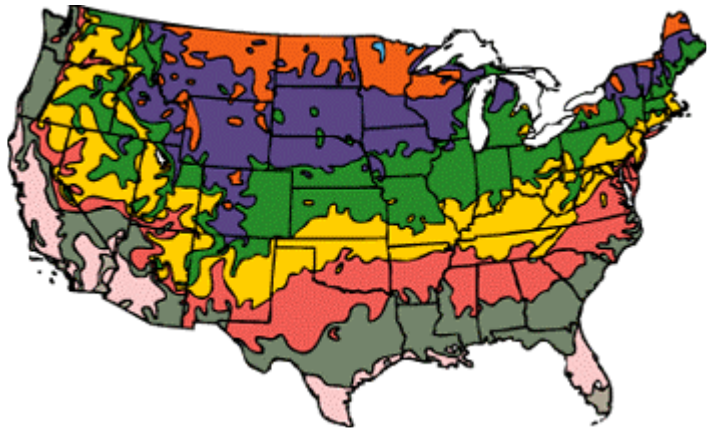
Check **`CLASSPATH`** by typing **`echo $CLASSPATH`**. If it doesn't start out with **`.:/u/yourname/public_html/JAVA`**, edit **`.cshrc`** to correct the problem.

Zip Code Finder

Given a file **`zlla0.dat`**:

```
41371
01001   42.06   72.624   42
01002   42.36   72.451   24
01003   42.38   72.516    0
01004   42.36   72.451    0
01005   42.42   72.106    0
01007   42.27   72.402   27
01008   42.18   72.953    0
01009   42.19   72.308    0
01010   42.11   72.205    3
   .
   .
   .
99357   46.91   119.61    0
99359   46.43   118.10    0
99360   46.11   118.71    0
99361   46.43   118.10    0
99362   46.15   118.31  117
99363   46.11   118.71    0
99371   46.90   118.28    0
99401   46.13   117.06    0
99402   46.16   116.94    0
99403   46.22   117.23   24
```

Write a program that prompts for a zip code and then shows it on a map of the US.



Start with Two Classes:

Class **`Zip`** *-- to Hold Zipcode Data*

```
public class Zip
{
    int zip;
    double lat;
    double lon;
    double x;   // x-coordinate in azimuthal projection
    double y;   // y-coordinate in azimuthal projection
}
```

Class **`ZipFinder`** *-- to hold the Program*

```
public class ZipFinder extends Frame
{
    /* Class-global variables */
    static int n;
    static Zip[] zips;
    static Zip myzip = null;

    public static void main(String[] args)
    { /* details later */ }

    public synchronized void paint (Graphics g)
    { /* details later */ }

    /* some other methods:  details later */
}
```

Class `ZipFinder` needs a `main()`

```
public static void main(String[] args)
{
    int i, zip, j, junk;
    /*************************************
    * Read in data.  (TEDIOUS)             */
    BufferedReader in;
    try {
      in = new BufferedReader(new FileReader("zlla0.dat"));
    }
    catch (FileNotFoundException fe) {
      System.err.println("File not found");
      return;
    }
    n = FileIO.readInt(in);
    zips = new Zip[n];
    for (i=0; i<n; i++) {
      zips[i] = new Zip();
      zips[i].zip = FileIO.readInt(in);
      zips[i].lat = FileIO.readDouble(in);
      zips[i].lon = -FileIO.readDouble(in);
      junk        = fileIO.readInt(in);
    }
    azimuthal();  // compute (x,y) coords in azimuthal proj
    /*****************************************
    * Open graphics window                    */
    ZipFinder bs = new ZipFinder();
    bs.resize(800,800);
    bs.show();
```

main() continued

```
    while (true) {
        zip = Console.in.readInt("Enter a zip: ");
        if (zip < 0) break;

        /* pick one */
        j = binsearch( zip );
        // j = brutesearch( zip );

        if ( j >= 0 ) {
            myzip = zips[j];
            Console.out.println("Found it "+myzip.lat+" "+
                        myzip.lon);
        } else {
            Console.out.println("Not found");
            myzip = zips[-j];
            Console.out.println("How about "+myzip.zip+": "
                        +myzip.lat+" "+ myzip.lon+"?");
        }
        bs.paint(bs.getGraphics());
    }
}
```

Graphics Work is in `paint()`

```
public synchronized void paint (Graphics g)
{
       double minx=1000, maxx=-1000, miny=1000, maxy=-1000, mid;
       int i, zip, j;

       /* compute minx, maxx, miny,maxy:  details later */

      GL.ginit(g, size().width, size().height, this, false);
      GL.ortho2(minx, maxx, miny, maxy);

       /*********************************************
        * Show all the points in red                */

       GL.color(Color.black);
       GL.clear();

       for (i=0; i<n; i++) {
           GL.color(Color.red);
           GL.pnt2(zips[i].x, zips[i].y);
       }
```

Paint() continued

```
      try { Thread.sleep(1000); }
      catch(InterruptedException ie){};

      if (myzip != null) {
          GL.color(Color.yellow);
          GL.pnt2(myzip.x, myzip.y);
          GL.drawString("HERE", myzip.x, myzip.y,
                              "CENTER", "BOTTOM");
      }
   }
```

What is `brutesearch()`?

```
static int brutesearch(
    int   w    // item sought
)
{
    int i;

    for (i=0; i<n; i++) {
        if (w == zips[i].zip) {
            return i;    // found it
        }
    }
    return -1            // not found
}
```

Requires on average *n*/2 iterations if item is present.

Requires *n* iterations if item is absent.

What is `binsearch()`?

Requires data (i.e., zipcodes) to be sorted.

```
static int binsearch(
    int   w    // item sought
)
{
    int low, high, mid;

    low = 0; high = n-1;
    while (low <= high) {
        mid = (low+high)/2;
        if ( w < zips[mid].zip) {
            high = mid - 1;          // in lower half
        } else if (w > zips[mid].zip) {
            low = mid + 1;           // in upper half
        } else {
            return mid;              // found it
        }
    }
    return -1                        // not found
}
```

Requires only $\log_2(n)$ iterations.
$\log_2(41000) \approx 16$.

Odds and Ends

Computing minx, maxx, etc.

```
for (i=0; i<n; i++) {
    if (zips[i].y > maxy) maxy = zips[i].y;
    if (zips[i].y < miny) miny = zips[i].y;
}
for (i=0; i<n; i++) {
    if (zips[i].x > maxx) maxx = zips[i].x;
    if (zips[i].x < minx) minx = zips[i].x;
}
maxx = maxx + 0.05 * (maxx - minx);
minx = minx - 0.05 * (maxx - minx);
maxy = maxy + 0.05 * (maxy - miny);
miny = miny - 0.05 * (maxy - miny);

if (maxx - minx > maxy - miny) {
    mid = (miny + maxy)/2;
    maxy = mid + (maxx - minx)/2;
    miny = mid - (maxx - minx)/2;
} else {
    mid = (minx + maxx)/2;
    maxx = mid + (maxy - miny)/2;
    minx = mid - (maxy - miny)/2;
}
```

Imports:

```
import java.awt.*;
import java.text.*;
import java.io.*;
import myutil.*;
import ccj.*;
```

`Azimuthal()`

Lots of trigonometry - see code online.