ORF 201

COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 7

Searching

Hints on Java Applets

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

Copy **`/u/orf201/lab1/.rhosts`** to **`/u/yourname`**.

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

Copy all files in **`/u/orf201/public_html/JAVA/myutil`** to your **`myutil`** directory.

Do with **`ccj`** as you did above with **`myutil`**.

Use **`chmod`** to set permissions on **`/u/yourname`**, **`/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.

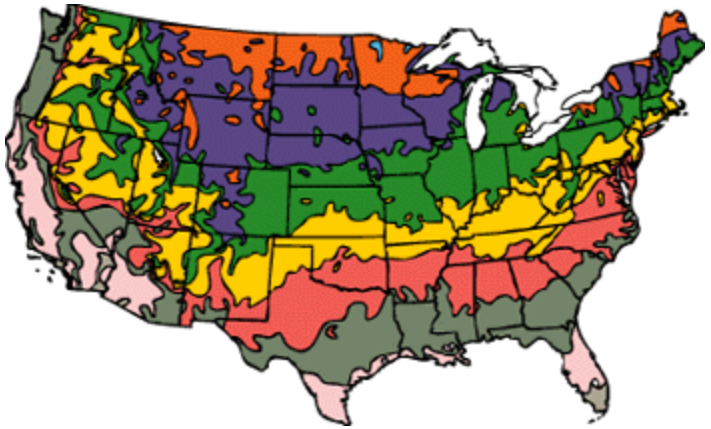
Edit **`clock.html`**, **`gravity.html`**, etc., and set **`codebase`** to "**`../..`**".

Zip Code Finder

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

```
27814
01000     42.377    72.522
01013     42.143    72.612
01100     42.100    72.588
01201     42.447    73.253
01220     42.625    73.117
01222     42.105    73.331
01223     42.355    73.129
01224     42.521    73.240
01224     42.556    73.153
    .
    .
    .
99362     46.070   118.330
99363     46.076   118.355
99371     46.755   118.313
99401     46.392   117.061
99402     46.338   117.045
99403     46.392   117.061
99995     43.710    79.220
99996     43.230    79.950
99997     43.090    79.260
99998     43.260    83.000
```

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*

```
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;

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

     public 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, record_num;

    // Read in data. (TEDIOUS)
    BufferedReader in;
    try {
      in = new BufferedReader(new FileReader("ziplatlon.dat"));
    }
    catch (FileNotFoundException fe) {
      System.out.println("File not found");
      return;
    }
    n = FileIO.readInt(in);
    zips = new Zip[n];
    for (i=0; i<n; i++) {
      if (i%500 == 0) {System.out.println(i)};
      zips[i] = new Zip();
      zips[i].zip = FileIO.readInt(in);
      zips[i].lat = FileIO.readDouble(in);
      zips[i].lon = -FileIO.readDouble(in);
    }
    azimuthal();  // compute (x,y) coords in azimuthal proj

    // Open graphics window
    ZipFinder bs = new ZipFinder();
    bs.resize(800,800);
    bs.show();
}
```

Most of the Work is in `paint()`

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

    /* 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 on black background*/

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

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

We're not done. The best is yet to come. See next page...

Here's the Real Work

```
    /*********************************************
     * Prompt user for a zip. Highlight it on the screen */


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

        /* use only one of the following search methods */
        // j = brutesearch( zip, zips );
        j = binsearch( zip, zips );

        if (j >= 0 ) {
            GL.pnt2(zips[j].x, zips[j].y);
            GL.drawString("HERE", zips[j].x, zips[j].y,
                          "CENTER", "BOTTOM");
        } else {
            Console.out.println("Not found");
        }
    }
}
```

What is `brutesearch()`?

```
static int brutesearch(
    int   w,     // item sought
    Zip[] v      // array of zips
)
{
    int i;

    for (i=0; i<n; i++) {
        if (w == v[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
    Zip[] v      // array of zips
)
{
    int low, high, mid;

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

Requires only $\log_2(n)$ iterations.
$\log_2(27000) \approx 15$.

Odds and Ends

Computing minx, maxx, etc.

```
maxy = -1000; miny = 1000;
maxx = -1000; minx = 1000;
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;
    min= mid - (maxy - miny)/2;
}
```

Imports:

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

`Azimuthal()`

Lots of trigonometry - see online source listing.