**Algorithms CS325: Project 1**

## Run-time analysis

* Assume all lines follow the form Yi=miX+bi
* Let m[n] be a set of slopes of the given lines
* Let b[n] be a set of y-intercepts of the given lines
* Let n be the number of lines given

Algorithm 1:

GetVisibleLines (lines[n], m[n], b[n])

For i=1 to n-2

      For j=0 to i-1

*Where Xj,k is the intersection point of line j and line k*

For k=i+1 to n-1

if(yj(Xj,k)>yi(Xj,k))

line i is not visible

Run-time is θ(n3) as it will go through 3 for loops for each line given.

Algorithm 2:

GetVisibleLines(lines[n],m[n],b[n])

for i=1 to n-2

      for j=0 to i-1

if line j is visible{

*Where Xj,k is the intersection point of line j and line k*

  for k=i+1 to n-1

if(yj(Xj,k)>yi(Xj,k))

line i is not visible

go back to first loop

}

Run-time is Ο(n3) when all lines are visible, as it will have to go through all 3 loops acting line algorithm 1, and Ω(n) when all but the first and last lines are invisible as it only needs to find one example of invisibility before moving on to the next line.

Algorithm 3:

* Let VisibleLines[n] only store indices that represent visible lines.
* Let countVisible represents the number of visible lines.
* Initially, VisibleLines [0]=0; VisibleLines[1]=1;CountVisible=2;

GetVisibleLines(VisibleLines[n], m[n],b[n])

for i=2 to n-1

  for j=0 to countVisible-2

if the intersection point of visible line j and line j+1 is below line i

countVisible=j+1;

break;

  VisibleLines[CountVisible]=i;

  countVisible++;

Run-time is Ο(n2) when all lines are visible as it will have to go through each line with two for loops and Ω(n) if all but the first and last line are invisible.

# Correctness

Yjk-1

Yjk

Yjk+1

Yjk+2

Yi

Yjk-2

**Prove:** if Yjk+1 is the first invisible line, then the following lines also become invisible.

Since Yj1, Yj2…Yjt is the visible subset of Y1, Y2…Yi-1, when add line Yi

Let Yjk be the last visible line

Let Yjk+1 be the first invisible line

Let Yjk+2 be the line we want to prove to become invisible

We know that for all intersection points, we have Xj1,j2 <Xj2,j3 <…<Xjk-1,jk <Xjk,jk+1<Xjk+1,jk+2<….<Xjt-1,jt

According to Claim 1, for line Yjk+1, we have two visible lines Yjk and Yi , for jk<jk+1<Yi,

Such that Yi(Xjk,i) > Yjk+1(Xjk,i)

Also because for all X < Xjk+1,jk+2

We have Yjk+1(X)>Yjk+2(X) (#2 from Visible Line Notes handout)

Then we get Yi(Xjk,i) > Yjk+1(Xjk,i)> Yjk+2(Xjk,i)

So we know that the intersection point of lines Yjk and Yi is also above line Yjk+2

Therefore, line Yjk+2 is invisible, same reason for line Yjk+2 to Yjt.

**Prove:** if Yjk is the last visible line, then Yj1 to Yjk-1 are visible as well.

Since Yj1, Yj2…Yjt is the visible subset of Y1, Y2…Yi-1, when add line Yi

Let Yjk be the last visible line

Let Yjk-1 be the line we want to prove to be visible

We know that for all intersection points, we have Xj1,j2 <Xj2,j3 <…<Xjk-1,jk <Xjk,jk+1<Xjk+1,jk+2<….<Xjt-1,jt

For all X < Xjk,i

We have Yi(X)< Yjk(X) (#1 from Visible Line Notes handout)

Also because for all X < Xjk,jk-1

We have Yjk(X)<Yjk-1(X) (#1 from Visible Line Notes handout)

Then we get Yi(X) < Yjk (X)> Yjk-1(X) for all X < Xjk,jk-1

So we know that the line Yi doesn’t affect the visibility of line Yjk-1

Therefore, line Yjk-1 is visible, same reason for line Yjk-2 to Yj0.

# Experimental Analysis

## Extrapolation and Interpretation

Using the equations derived from the log-log chart using excel we have a rough trend line of the performance of the algorithms. After solving for X in Wolfram Alpha we found that in one hour algorithm 1 can handle 16825 lines, algorithm 2 can handle 4.57x1012, and algorithm 3 can handle 2.9x1011.

Looking at the comparison of algorithm 2 and 3 it seems that 2 is faster. This is because of an issue with the random data generated. We could not generate a random data set that would provide any visible lines other than the first and last lines. This means that all of the algorithm data we recorded represents the best case running time. Because of this the running time is extremely small leading to high variability in testing times and skewing the data. This also explains why the equations calculated from excel match the best case running times we came up with in the Run-Time Analysis section.