CS325 Project 1 Report

Erika Crowe and Dean Johnson

October 16, 2014

Run-time Analysis

ALGORITHM 1: ENUMERATION

$n =$number of lines
$j = 0$
$i = j + 1$
$k = i + 1$
lines(i, j, k) visible = true
for $k < n$
  y_{jk} = y-intersect(lines(j, k))
  y_i = [slope(line(i)) $*$ (y-0-intersect(line(j)) - y-0-intersect(line(k)))]
    + [y-intersect(line$(i), 0$) $*$ slope(line$(k)-$ slope(line(j)))]
  if $y_{jk} > y_i$
    line(i) visible = false
  $j = j + 1$
  $i = i + 1$
  $k = k + 1$
return

ALGORITHM 2: BETTER ENUMERATION

$n =$number of lines
$j = 0$
$i = j + 1$
$k = i + 1$
lines(i, j, k) visible = true
for $k < n$
  if line(i) visible = false
    $j = j + 1$
    $i = i + 1$
    $k = k + 1$
    continue
  y_{jk} = y-intersect(lines(j, k))
  y_i = [slope(line(i)) $*$ (y-0-intersect(line(j)) - y-0-intersect(line(k)))]
    + [y-intersect(line$(i), 0$) $*$ slope(line$(k)-$ slope(line(j)))]
  if $y_{jk} > y_i$

line(i) visible = false
$j = j + 1$
$i = i + 1$
$k = k + 1$
return

ALGORITHM 3: EVEN BETTER ENUMERATION
A = set of all lines
V = set of visible lines
$V = A$
$j = 0$
$i = j + 1$
$k = i + 1$
if $V \leq 2$
return V
for length of A
line1 = $A[j]$
line2 = $A[i]$
comparison line = $A[k]$
y-int = y-intersect(line1,line2)
comp-int = [slope(comparison line) $*$ (y-0-intersect(line1) - y-0-intersect(line2)]
+ [y-intersect($comparisonline, 0)*$ (slope(line2) - slope(line1))]
if y-int < comp-int
line2 visible = false
remove line2 from V
$j = j + 1$
$i = i + 1$
$k = k + 1$
return V

Asymptotic Analysis

All three algorithms are $\theta(n)$ to each other. While each algorithm is more efficient than the last, they all iterate through the set of lines at a linear rate.

Correctness of Claim 2/Algorithm 3

Theorem 1. *If* $\{y_{j1}, y_{j2}, ..., y_{jt}\}$ *is the visible subset of* $\{y_1, y_2, ..., y_{i-1}\}$ *where* $(t \leq i-1)$ *then* $\{y_{j1}, y_{j2}, ..., y_{jk}, y_i\}$ *is the visible subset of* $\{y_1, y_2, ..., y_i\}$ *where* y_{jk} *is the last line such that* $y_{jk}(x^*) \geq y_i(x^*)$ *where* $(x^*, y_{jk}(x^*))$ *is the point of intersection of the lines* y_{jk} *and* y_{jk-1}.

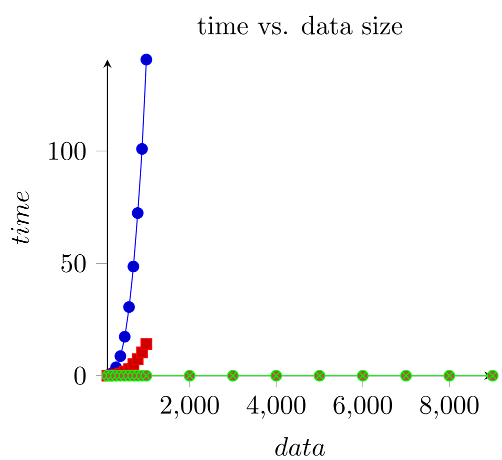
Proof. Suppose there exists a set of intersecting lines A and that the slope of each line is unique. Further suppose that a subset of those lines V are the only lines in A that are visible when viewed from above. Now, if a new line is appended to V, a subset V_1 is created within V that consists of the original

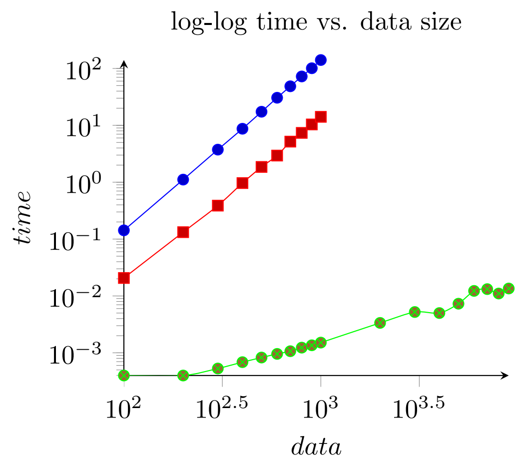
lines within V.
V contains lines $\{y_{j1}, \ldots, y_{jt}\}$
A contains lines $\{y_1, \ldots, y_{i-1}\}$
Let y_i be the line added to V and let y_{jk} be the last line in the set that has an intersection point with the line before it that is greater than the intersection point between itself and the new line (symbolically explained previously).
In order for a line to be visible, it must have at least one point on it whose x value is greater than the point of intersection between it and the next closest line (Equation 1 on the Visible Lines Handout). Therefore, any new line added to V that is visible must have at least one visible point on it that is greater than the point of intersection between itself and the next closest visible line (y_{jk}). If there exists a visible line already in V whose point of intersection between itself and y_{jk} has a greater x value than the x value of the point of intersection with the new line and y_{jk} then this line is no longer visible. However, all visible lines with x-intersect values less than that of the new line will remain visible and thus form a subset V_1 within V. $\square$

Experimental Correctness

Solutions to the instances in the file provided have been submitted to TEACH.

Experimental Analysis





Extrapolation and Interpolation

1. *The biggest instances that could be solved within one hour were not able to be determined.*

2. (a) Algorithm 1 (blue) slope is $\frac{\log(48.599125/8.711271)}{\log(700/400)} = 3.072$

   (b) Algorithm 2 (red) slope is $\frac{\log(5.17444/.965603}{700/400} = 2.99$

   (c) Algorithm 3 (green) slope is $\frac{\log(.00106699999999/.000685000000001}{700/400} = .792$

   It appears that Algorithms 1 and 2 are similar enough to each other to be θ to each other, but Algorithm 3 increases at such a lower rate that it is closer to O for both 1 and 2.