CS325 - Project 1

Group #6
William Jernigan, Alexander Merrill, Sean Rettig

October 16, 2014

Correctness

Proof of Claim 2: Direct Proof

> Claim 2: If $\{y_{j_1}, y_{j_2}, ..., y_{j_t}\}$ is the visible subset of $\{y_1, y_2, ..., y_{i-1}\}(t \leq i{-}1)$ then $\{y_{j_1}, y_{j_2}, ..., y_{j_k}, y_i\}$ is the visible subset of $\{y_1, y_2, ..., y_i\}$ where y_{j_k} is the last line such that $y_{j_k}(x^*) \geq y_i(x^*)$ where $(x^*, y_{j_k}(x^*))$ is the point of intersection of lines y_{j_k} and $y_{j_{k-1}}$.

Let $V = \{y_{j_1}, y_{j_2}, ..., y_{j_t}\}$ be the visible subset of $A = \{y_1, y_2, ..., y_{i-1}\}$.
Let $V^+ = \{y_{j_1}, y_{j_2}, ..., y_{j_k}, y_i\}$ be the visible subset of $A^+ = \{y_1, y_2, ..., y_i\}$.

To prove Claim 2, we must prove that each new line y_i is visible at the time it is added to V^+, that each culled line y_{j_p} (where $k < p < i$) was no longer visible due to being covered by y_i, and that each remaining line y_{j_1} ... y_{j_k} remains visible in V^+.

Prove that $y_i \in V^+$

Let $A^+ = A \cup \{y_i\}$.
Because $m_i > m_n$ for all $0 < n < i$, y_i is visible by the Claim 1 proof in the "Visible Line Notes" handout.
Since y_i is visible and $y_i \in A^+, y_i$ must also be in V^+, the visible subset of A^+.

Prove that $y_{j_k} \in V^+$

Let $(x^*, y_{j_k}(x^*))$ be the point of intersection of the lines y_{j_k} and $y_{j_{k-1}}$.
Since $y_{j_k}(x^*) \geq y_i(x^*)$ by definition, y_{j_k} is visible with respect to y_i.
Since y_{j_k} was already in V, it is defined to be visible with respect to all other elements in V.
$\therefore y_{j_k} \in V^+$.

Prove that $y_{j_n} \in V^+, 0 < n < k$

Because $y_{j_n} \in V$, it is defined to be visible with respect to all other elements in V.
So we must show that y_{j_n} is visible with respect to y_i as well.
Let $(x_n^*, y_{j_n}(x_n^*))$ be the point of intersection of the lines y_{j_n} and $y_{j_{n+1}}$.
By definition, $m_{j_n} < m_{j_{n+1}}$, so $\forall x_n < x_n^*, y_{j_n}(x_n) \geq y_{j_{n+1}}(x_n)$.
$\therefore y_{j_1}(x_1^*) = y_{j_2}(x_1^*) \geq y_{j_3}(x_1^*), y_{j_2}(x_2^*) = y_{j_3}(x_2^*) \geq y_{j_4}(x_2^*), ..., y_{j_{n-1}}(x_{n-1}^*) = y_{j_n}(x_{n-1}^*) \geq y_{j_{n+1}}(x_{n-1}^*), ..., y_{j_{k-1}}(x_{k-1}^*) = y_{j_k}(x_{k-1}^*) \geq y_{j_i}(x_{k-1}^*)$
$\therefore y_{j_n}$ is visible with respect to y_i.
$\therefore y_{j_n} \in V^+$.

Prove that $y_{j_p} \notin V^+, k < p < i$

Let $(x_p^*, y_{j_p}(x_p^*))$ be the point of intersection of the lines $y_{j_{p-1}}$ and y_{j_p}.
By definition, $m_{j_{p-1}} < m_{j_p}$, so $\forall x_{p-1} < x_{p-1}^*, y_{j_{p-1}}(x_{p-1}) \geq y_{j_p}(x_{p-1})$.

$\therefore y_{j_1}(x_2^*) = y_{j_2}(x_2^*) \geq y_{j_3}(x_2^*), y_{j_2}(x_3^*) = y_{j_3}(x_3^*) \geq y_{j_4}(x_3^*), ..., y_{j_{k-1}}(x_k^*) = y_{j_k}(x_k^*) \geq y_{j_{k+1}}(x_k^*), ..., y_{j_{p-1}}(x_p^*) = y_{j_p}(x_p^*) \geq y_{j_{p+1}}(x_p^*), ..., y_{j_{t-2}}(x_{t-1}^*) = y_{j_{t-1}}(x_{t-1}^*) \geq y_{j_t}(x_{t-1}^*)$
In order for y_{j_p} to not be visible in V^+, $y_i(x_p^*) > y_p(x_p^*)$ by definition of visible.
$\therefore y_{j_1}(x_2^*) = y_{j_2}(x_2^*) \geq y_{j_3}(x_2^*), y_{j_2}(x_3^*) = y_{j_3}(x_3^*) \geq y_{j_4}(x_3^*), ..., y_{j_{k-1}}(x_k^*) = y_{j_k}(x_k^*) \geq y_{j_i}(x_k^*), ..., y_{j_{p-1}}(x_{p-1,p}^*) = y_{j_p}(x_{p-1,p}^*) < y_{j_i}(x_{p-1,p}^*), ..., y_{j_{t-1}}(x_t^*) = y_{j_t}(x_t^*) < y_{j_i}(x_t^*)$
$\therefore y_{j_p} \notin V^+$.

Proof of Algorithm 3: Direct Proof

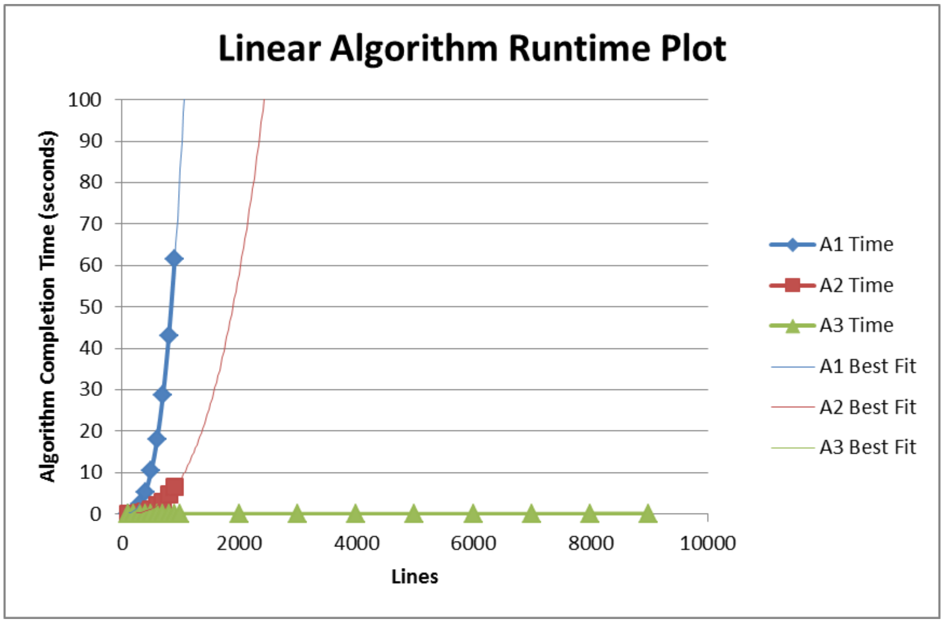
As Algorithm 3 is simply an implementation of Claim 2, and Claim 2 was proven above, Algorithm 3 is also proven to be correct. Our Claim 2 proof shows that each new line y_i is visible at the time it is added to V^+, that each culled line y_{j_p} was no longer visible due to being covered by y_i, and that each remaining line y_{j_1} ... y_{j_k} remains visible in V^+.

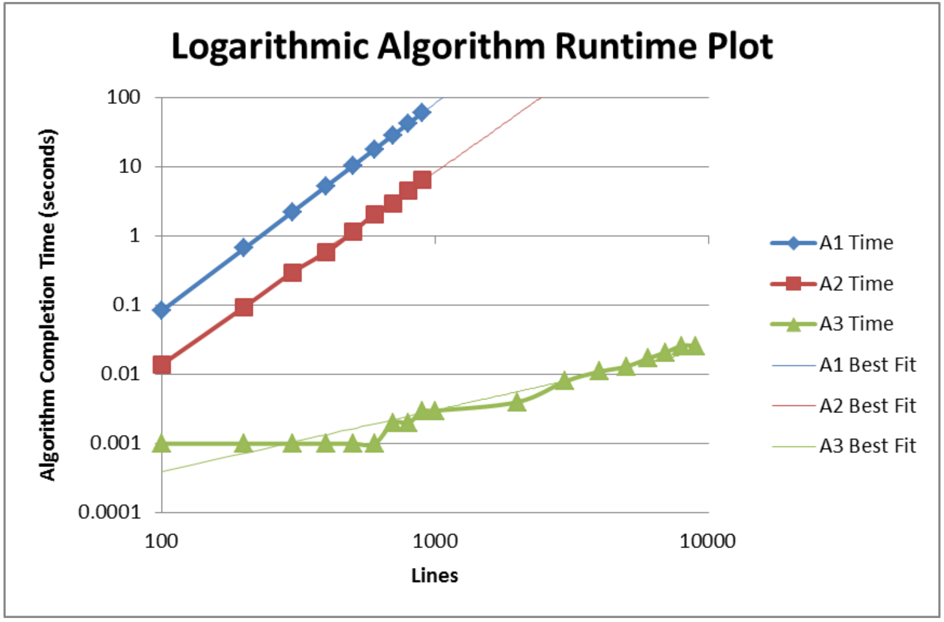
Experimental and Asymptotic Run Time Analysis

Experimental Run Time Data

Lines	A1 Time	A2 Time	A3 Time
100	0.084	0.014	0.001
200	0.665	0.095	0.001
300	2.246	0.300	0.001
400	5.324	0.588	0.001
500	10.414	1.163	0.001
600	18.104	2.100	0.001
700	28.801	2.989	0.002
800	42.930	4.628	0.002
900	61.510	6.617	0.003
1000			0.003
2000			0.004
3000			0.008
4000			0.011
5000			0.013
6000			0.017
7000			0.021
8000			0.026
9000			0.026

Experimental Run Time Plots





Experimental Run Time Analysis

Algorithm 1: $y = 8 \times 10^{-8} x^{3.0026}$
Algorithm 2: $y = 4 \times 10^{-8} x^{2.7892}$
Algorithm 3: $y = 7 \times 10^{-6} x^{0.8868}$

Given these equations, we can use both the slopes and our code to analyze the run times of the algorithms.

Asymptotic Run Time Analysis

Algorithm 1: $\Theta(n^3)$
Algorithm 1 iterates over the set of lines in a triple-nested loop.

Algorithm 2: $\Theta(n^3)$
Similar to Algorithm 1, Algorithm 2 still has the triple-nested loop—it simply bypasses some unnecessary calculations within the triple-nested loop to speed it up.

Algorithm 3: $\Theta(n)$
Algorithm 3 only iterates over the set of lines once. Though the removeCovered function can recurse several times, it does not scale with the size of the input list.

Discrepancies

We note that the slopes from our experimentally-derived equations are close to $\Theta(n^3)$, $\Theta(n^3)$, and $\Theta(n)$, but not exact. These discrepancies have many possible contributing factors, including a small sample size, a low timing resolution (particularly for Algorithm 3), and randomness in run time present in Algorithms 2 and 3 (which unlike Algorithm 1, do not simply perform a fixed number of operations, but rather change what operations they run depending on the input lines given. This randomness is likely to be the primary cause, as the experimental running time for Algorithm 1 is by far the closest to our asymptotic estimation.

Estimated Number of Lines Per Hour

Algorithm 1: $\sim 3,532$
Algorithm 2: $\sim 8,460$
Algorithm 3: $\sim 7,919,210,000$

Pseudocode

```
algorithm1(lines):
    for j in lines[0 ... n]:
        for i in lines[j+1 ... n]:
            for k in lines[i+1 ... n]:
                Xjk, Yjk = intersection(j, k)
                Yi = i.slope * Xjk + i.intercept
                if Yjk > Yi:
                    i.visible = False
    return lines

algorithm2(lines):
    for j in lines[0 ... n]:
        for i in lines[j+1 ... n]:
            for k in lines[i+1 ... n]:
                if i.visible:
                    Xjk, Yjk = intersection(j, k)
                    Yi = i.slope * Xjk + i.intercept
                    if Yjk > Yi:
                        i.visible = False
    return lines

algorithm3(lines):
    vlines = []
    for i in lines:
        vlines.append(i)
        removeCovered(vlines)
    return lines

removeCovered(vlines):
    if len(vlines) < 3: # All lines are visible if there are only 1 or 2.
        return vlines
    else:
        a, b, c = vlines[n-2 ... n]
        Xab, Yab = intersection(a, b)
        Yc = c.slope * Xab + c.intercept
        if Yc > Yab: # If line b is covered, remove it and recurse.
            b.visible = False
            vlines.remove(b)
            return removeCovered(vlines)
        else: # If line b is still visible, do nothing.
            return vlines
```