CS325 Project 2 Merge Visible Lines
October 27, 2014

Zhong, Yaonan
Deng, Daoxiang
Chen, Weipeng

Part 1 Proof of Claim 3

1. First, let's consider the visible line segments in two visible sets Z and Z'. In both sets, the visible range of x is [-∞, +∞]. Since each line segment has a unique slope, there must be a cross point between the first set of visible segments and the second set. Figure 1 shows that Zi in the first set intersects with Z'j in the second set at point P1. Now let's demonstrate that {Z'1,Z'2,..,Z'j-1} is invisible after merged.

2. Let P1 = (P1x,P1y). Since Z'j is a visible segment in the second set and P1 is on Z'j, based on the definition of visible lines, we have P1y>=Z't(P1x), for 1<=t<=j-1. After merging Z and Z', P1 is the cross point of Zi and Z'j, using claim 1 in project 1, we know for 1<=t<=j-1, Z't is invisible. Using the same analysis, we can show that Zt is invisible for 1<=t<=i-1 in the left set.

3. Now let's demonstrate {Z1,Z2,..,Zi} and {Z'j,Z'j+1,..,Z's} are visible. In step 2, we remove all the segments that are invisible after merged. In the first set, the visible range of x becomes [-∞, P1x]. In the second set, the visible range is [P1x, +∞]. Since both ranges do not overlap with each other except at P1. They are visible as they are in the original sets. So we show that {Z1,Z2,..,Zi} and {Z'j,Z'j+1,..,Z's} are visible.

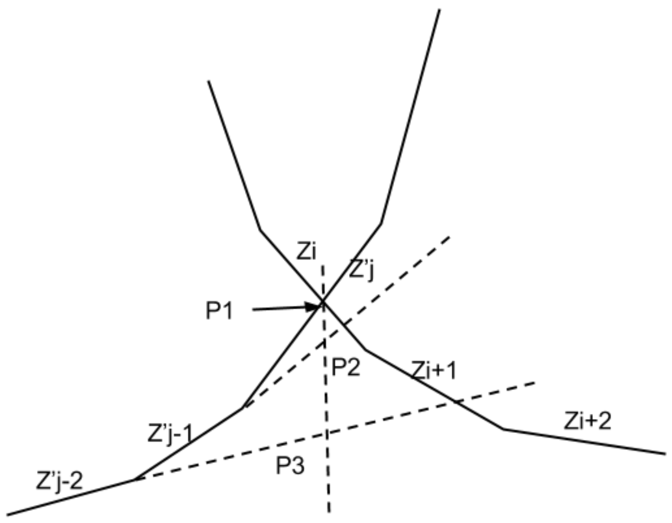


Figure 1

Part 2 Pseudocode and Run-time Analysis

MergeVisible

```
 MergeVisible(L[1..a], R[1..b])
        Li = Ri = 1
        while Li < len(L) and Ri < len(R):
                (P1x, P1y) <- the cross point of L[Li] and L[Li+1]
                (P2x, P2y) <- the cross point of R[Ri] and R[Ri+1]
                if P1x <= P2x:
                        if (P1x, P1y) is above or in the line R[Ri]: Li = Li + 1
                        else: break
                else:
                        Ri = Ri + 1
        targetL = Li

        Li = Ri = 1
        while Li < len(L) and Ri < len(R):
                (P1x, P1y) <- the cross point of L[Li] and L[Li+1]
                (P2x, P2y) <- the cross point of R[Ri] and R[Ri+1]
                if P2x <= P1x:
                        if (P2x, P2y) is below or in the line L[Li]: Ri = Ri + 1
                        else: break
                else:
                        Li = Li + 1
        targetR = Ri

        return L[1..targetL] + R[targetR..b]
```

This MergeVisible algorithm runs in linear time O(n). There are two independent while loops. The first one computes the target index in left visible set with run-time O(min[len(L), len(R)]). The second one computes the target index in right visible set with run-time O(min[len(L), len(R)]). len(L) and len(R) are the sizes of two visible sets. Let n = len(L) + len(R), then the total running time is O(2*min[len(L), len(R)]) <= O(n).

Algorithm 4

```
 FindVisible_4(Lines[1..n])
        if n <= 2:
                return Lines[1..n]
        if n == 3:
```

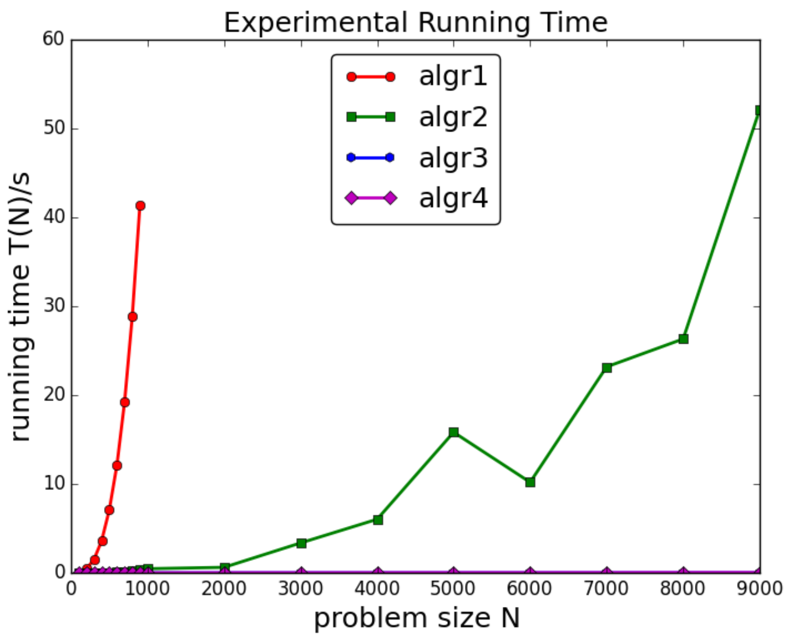
```
        if the cross point of Lines[1] and Lines[3] is above Line[2]:
            return Lines[1]+Lines[3]
        else:
            return Lines[1..n]
    m = n/2
    Left = FindVisible_4(Lines[1..m])
    Right = FindVisible_4(Lines[m+1..n])
    result = MergeVisible(Left, Right)
    return result
```

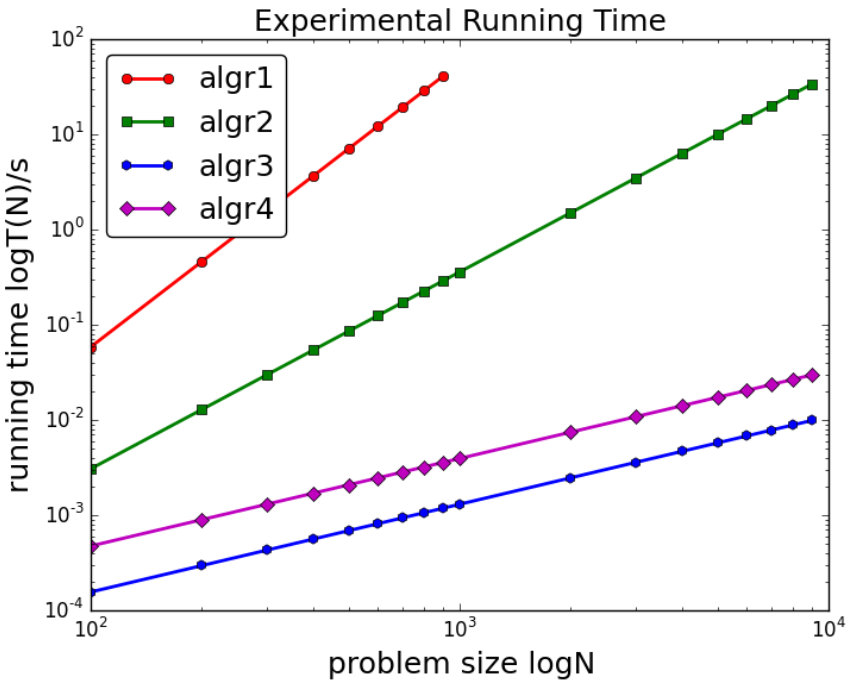
The running time of algorithm 4 is O(nlogn). If the size of lines is n, using the recurrence, the running time is $T(n) = 2T(n/2) + O(n) + C$, and C is a constant. Solve this recurrence and we get $T(n) = O((n+C)\log n) = O(n\log n)$.

Part 3 Proof of correctness by induction:

(1) Base case: if n<=2, all lines are visible. if n=3, we verify the second line. So algorithm4 correctly return visible lines.
(2) Induction hypothesis: suppose that for any set of lines of size < n, algorithm4 correctly return the set of visible lines.
(3) Prove: now consider a set L of n lines. Algorithm4 divide L into two halves Lines[1..m] and Lines[m+1..n] of size < n. Therefore, Left and Right are visible lines by our induction hypothesis. Using claim3, we know that our MergeVisible algorithm is correct. So algorithm4 correctly return the visible lines of L.

Part 4 Experimental Analysis





Part 5 Extrapolation and Interpretation

1. From the experimental running time data, it's a bit difficult to tell the size of the biggest instance that could be solved by each algorithm in one hour, because longest running time in our experiment is about 50 seconds.
   But we can estimate the problem size solved in one minute:
   Algr 1: about 1000 lines
   Algr 2: about 10000 lines
   Algr 3: very large, larger than algr1 and algr2
   Algr 4: Also very large, larger than algr1 and algr2

2. Use polynomial fitting, we can get the coefficients of the fitting lines in log-log plot:
   logy = k*logx + b, for [k,b] we have
   Algr 1: [2.97810842, -16.55340591]
   Algr 2: [2.06735771, -15.30717024]
   Algr 3: [0.92187809, -13.012448]
   Algr 4: [0.91925452, -11.88769513]

   The slope of the lines:
   Algr 1: 2.97810842
   Algr 2: 2.06735771
   Algr 3: 0.92187809
   Algr 4: 0.91925452

From the log-log plot, we can see that the running time of algr4 is smaller than those of alg1 and alg2, but larger than algr3. In project 1, the theoretical running times is Theta(n^3), O(n^3) and Theta(n). And the theoretical time of algr4 is O(nlogn), so the experimental running time of algr4 is larger than that of algr3. But the slope of algr4 is a little bit smaller than algr3 in the log-log plot. This does not match with the theoretical analysis.