CSCE 586 - Design and Analysis of Algorithms
Date: Thursday 6th December, 2018
Name: Micah Hayden
Assignment: Final Exam - Take Home Portion, Version A
Documentation:

---

1 Hidden Surface Removal Problem:

1.1 Which algorithmic paradigm will you use to solve this problem?

I will utilize Divide-and-Conquer to solve this problem.

1.2 Why did you chose the algorithmic paradigm selected above to solve this problem?

I believe this paradigm is suited to divide and conquer. Each line has similar characteristics, following the same equation $y_i = a_i \cdot x + b_i$. The problem can be divided into small, independent sub-problems using the slope of each line. The base case occurs when $n \leq 3$: because no 3 lines intersect at a single point, the resulting "uppermost" lines can be found in constant time.

1.3 Give an algorithm that takes n lines as input, and in $O(n \log n)$ time returns all of the lines that are visible. Provide a clear description of the algorithm.

Let L be a set of lines, $|L| = n$, where $L_i = m_i \cdot x + b_i$.

Algorithm 1 Hidden-Surface-Removal: HSR(L)

Sort L by ascending slope, such that L_i has slope m_i and $m_i < m_{i+1}$ for all i.
$|L| = n$
if $n \leq 2$ **then**
  return The set of lines L, and their intersection a.
else if $n = 3$ **then**
  Let $a =$ the intersection of L_1 and L_3
  Let $b =$ the intersection of L_1 and L_2
  if $x_b < x_a$ **then**
    Let $c =$ the intersection of L_2 and L_3.
    return L and $\{b, c\}$
  else
    return $L - \{L_2\}$ and $\{a\}$
  end if
else
  $L, A = HSR(\{L_1, \ldots, L_{\frac{n}{2}}\})$
  $L', B = HSR(\{L_{\frac{n}{2}+1}, \ldots, L_n\})$
end if
Merge A and B into C by increasing x coordinate
Find the first element in C for which $L' > L$, denote this element as c_k.
Let $L_i \in L$ be the uppermost line in L immediately before c_k.
Let $L_j \in L'$ be the uppermost line in L' immediately after c_k.
Let point p_{int} be the intersection of lines L_i and L_j.
$L = \{L_1, L_2, \ldots, L_i\} \cup \{L'_j, L'_{j+1}, \ldots L_n\}$
$C = \{A_1, \ldots A_{i-1}\} \cup p_{int} \cup \{B_j, \ldots B_{n-1}\}$
return L and C

1.4 Perform asymptotic analysis of your algorithm's running time. Also, consider the run time performance of a best case, worst case, and average case input model scenario.

Worst Case:
Best Case:
Average Case:

1.5 Provide a proof that your algorithm works correctly:

2 Bipartite Matching Problem

2.1 Which algorithmic paradigm best describes this algorithm?

2.2 Why did you choose the algorithmic paradigm selected above?

2.3 Give an example of a bipartite graph G for which this algorithm does not return the maximum matching.

2.4 Let M and M' be matchings in a bipartite graph G. Suppose that $|M'| > 2 \cdot |M|$. Show that there is an edge $e' \in M'$ such that $M \cup e'$ is a matching in G.

2.5 Using the previous claim (and your supporting proof) to further prove that the algorithm is optimal or that the algorithm is ρ-optimal approximate (in this case be sure to derive the value of ρ as part of your proof).

3 Number Partitioning Problem

3.1 Problem Statement:

Show that the *Number Partitioning* is NP-complete using the *Subset Sum* problem.

Show *Number Partitioning* $\in$ NP

Let $Y = Number\ Partitioning$

Choose an NP-Complete problem X:

Let *Subset Sum* be X.

Prove that $X \leq_P Y$