Assignment 6 – Divide and Conquer 2

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: Kayley Seow　　Wisc id: kseow

Divide and Conquer

1. *Kleinberg, Jon. Algorithm Design (p. 248, q. 5)* Hidden surface removal is a problem in computer graphics where you identify objects that are completely hidden behind other objects, so that your renderer can skip over them. This is a common graphical optimization.

   In a clean geometric version of the problem, you are given n non-vertical, infinitely long lines in a plane labeled $L_1 \dots L_n$. You may assume that no three lines ever meet at the same point. (See the figure for an example.) We call L_i "uppermost" at a given x coordinate x_0 if its y coordinate at x_0 is greater than that of all other lines. We call L_i "visible" if it is uppermost for at least one x coordinate.

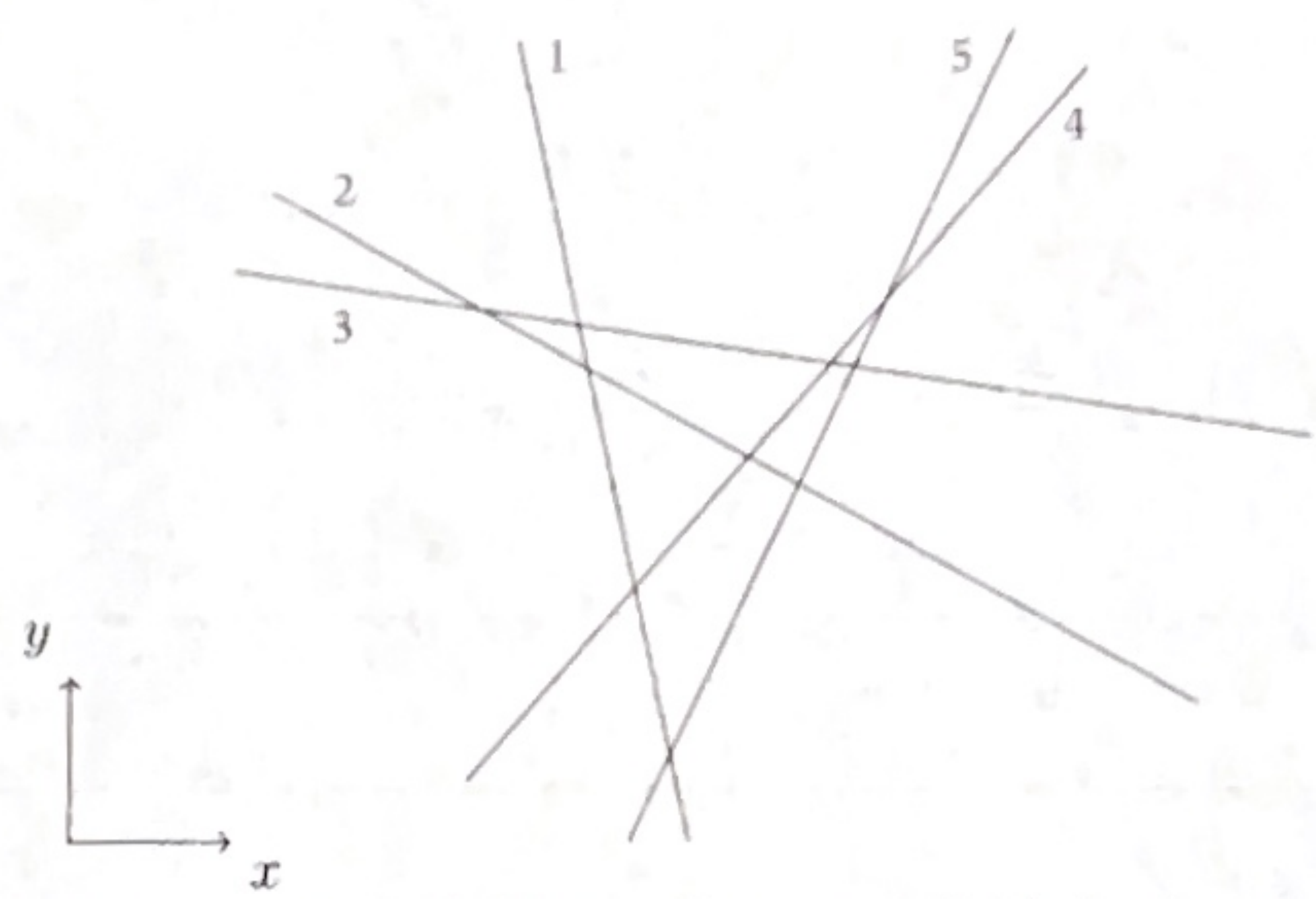


Figure 5.10 An instance of hidden surface removal with five lines (labeled *1-5* in the figure). All the lines except for *2* are visible.

   (a) Give an algorithm that takes n lines as input and in $O(n \log n)$ time returns all the ones that are visible.

Solution:
Because we need to sort the lines, I would sort them using mergesort, and have them in descending slope. First would be the line with the steepest slope, and then last one would be steepest negative. We know that lines on each end are visible. To find intersections, we need to compare the lines by where they intersect. Using a modern sliding window approach, we would then we would compare the lines to see which is dominant at a point, and which is visible.

(b) Write the recurrence relation for your algorithm.

Solution:

$T(n) = 2T(n/2) + cn$

sorting in increasing order, linear time operations

$(T(n))$

$\frac{cn}{2}$ $\frac{cn}{2}$

$\frac{cn}{4}$ $\frac{cn}{4}$ $\frac{cn}{4}$ $\frac{cn}{4}$

$\frac{cn}{2^k}$

$\sum_{i=0}^{H} \frac{2^i \cdot cn}{2^i} =$

$cn \sum_{i=0}^{H} 1$

$= cn(\text{height}+1)$

$= O(n \log n)$

(c) Prove the correctness of your algorithm.

Solution:

Soundness: uses merge sort, so our lines are sorted to find intersection. Base case of $n=1$, no visible segment, will be visible

Inductive: for visible segment list size j, $1 \le j \le k$, algo returns intersection. On the $j+1$ case if the line intersects, it returns correct and input less than k will have inductive hypothesis applied! half other of list input will prove inductive hypothesis.

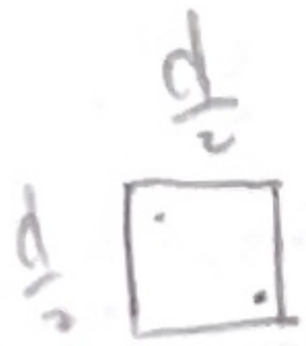
Termination, will intersect on the right and left on the plane, as none are parallel. Therefore, the algo will terminate and for any number it will terminate

2. In class, we considered a divide and conquer algorithm for finding the closest pair of points in a plane. Recall that this algorithm runs in $O(n \log n)$ time. Let's consider two variations on this problem:

$\frac{\sqrt{2}}{2} d < d$

If two pts fall inside one grid, dist

$\frac{d}{2} \times \frac{d}{2}$



(a) First consider the problem of searching for the closest pair of points in 3-dimensional space. Show how you could extend the single plane closest pairs algorithm to find closest pairs in 3D space. Your solution should still achieve $O(n \log n)$ run time.

Solution:

merge

1 go through all points, find out those with distance to middle $\leq d$ $O(n)$

2 For these point, compute pairwise distance $O(1) \cdot O(\log n) \cdot O(n)$

#: ≤ 15

$T(n) = 2T(n/2) + O(n) \to O(n \log n)$

use plane to divide 3D in half, search closest pair using half $O(\log n)$ runtime, find δ. and if $\exists P_1$ and P_2 smaller than δ, will be δ. find minimum pair, check next 63, 64-1, takes $O(n)$

(b) Now consider the problem of searching for the closest pair of points on the surface of a sphere (distances measured by the shortest path across the surface). Explain how your algorithm from part a can be used to find the closest pair of points on the sphere as well.

Solution:

Use xy plane to divide in half. Can use part a algorithm to check closest pair for each point $P \in S$, you can check cube within error sphere that surrounds it with sphere δ to look for any two point with smaller distance between them than delta, checking 63 points.

(c) Finally, consider the problem of searching for the closest pair of points on the surface of a torus (the shape of a donut). A torus can be thought of taking a plane and "wrap" at the edges, so a point with y coordinate MAX is the same as the point with the same x coordinate and y coordinate MIN. Similarly, the left and right edges of the plane wrap around. Show how you could extend the single plane closest pairs algorithm to find closest pairs in this space.

Solution:

Solve same algorithm. We would apply the distance formula to get difference between points. Then we need to take 8 more distances between points to check the distance wrapping around as planes. The distances we need to check the minimum of the distances.

3. *Erickson, Jeff. Algorithms (p. 58, q. 25 d and e)* Prove that the following algorithm computes $\gcd(x, y)$ the greatest common divisor of x and y, and show its worst-case running time.

```
BINARYGCD(x,y):
if x = y:
    return x          Base
else if x and y are both even:
    return 2*BINARYGCD(x/2,y/2)
else if x is even:
    return BINARYGCD(x/2,y)
else if y is even:
    return BINARYGCD(x,y/2)
else if x > y:
    return BINARYGCD( (x-y)/2,y )
else
    return BINARYGCD( x, (y-x)/2 )
```

recurse

$T(n) = T(n/2) + 1$

$O(\log n)$

Strong induct on x+y

Solution: Base case: x=y. GCD returns x which is correct GCD, holds.

IS: show x+y = k+y returns correct

IH: for all 2 < x+y ≤ k, Binary GCD returns correct value, all calls decrease problem size and x ≠ y. x+y is in the range of the IH.

Case 1: x and y are both even - 2 divides x and y. $x/2$, $y/2$ both integers and factor of them, therefore of GCD so you can take out 2, x at the end.

Case 2: x or y is odd divide even one by 2 because 2 divides by definition and not change GCD

Case 3: both odd. $x = 2k+1$, $y = 2m+1$ → subtracting will cancel out 2 and give even number, correct input passed in, euclidian, correct.

4. Here we explore the structure of some different recursion trees than the previous homework.

   (a) Asymptotically solve the following recurrence for $A(n)$ for $n \geq 1$.

$$A(n) = A(n/6) + 1 \qquad \text{with base case} \qquad A(1) = 1$$

Solution: $A(n)$

$1 + 1/6$ $\quad A(n/6)$

$1 + 1/36$ $\quad 1 \quad A(n/36)$

$1 + 1/6^3$ $\quad 1 \quad A(n/6^3)$

⋮

$1 + 6^i$ $\quad 1 \quad A(n/6^H)$

$\sum_{i=0}^{H} 1 + \sum_{i=0}^{H} \frac{1}{6^i}$

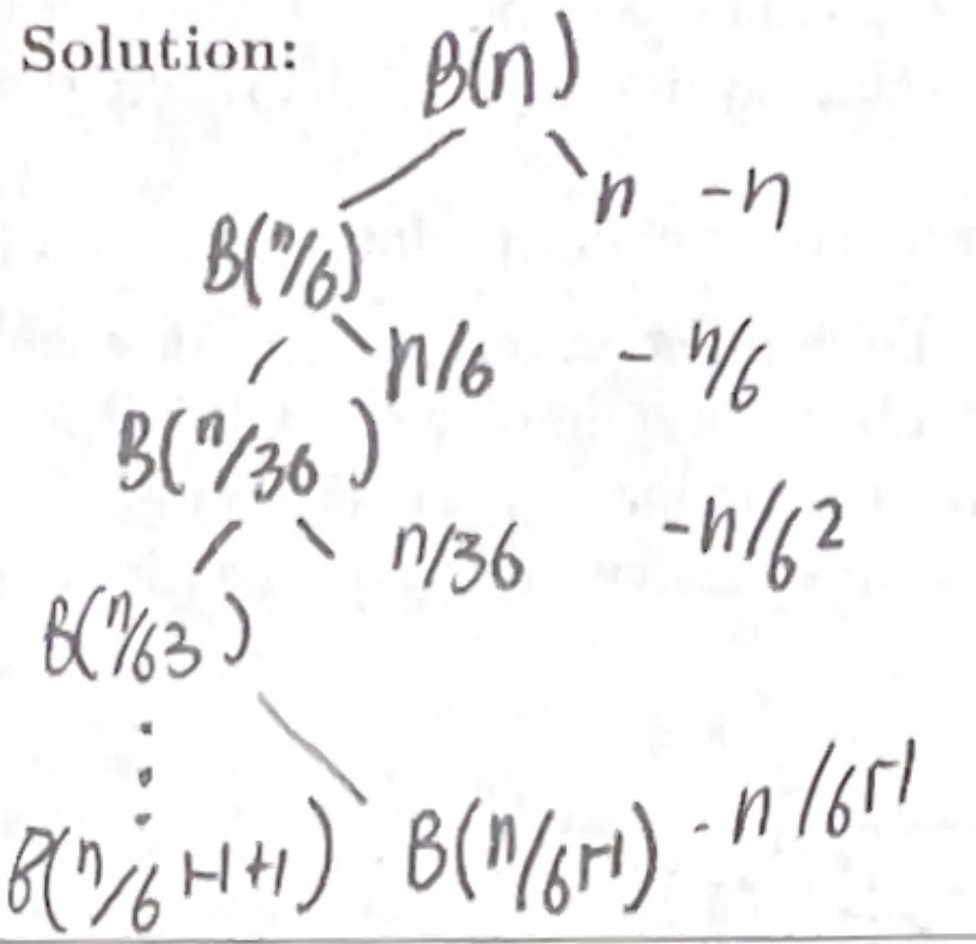
= (height + 1) constant

$\log n$

(b) Asymptotically solve the following recurrence for $B(n)$ for $n \geq 1$.

$$B(n) = B(n/6) + n \qquad \text{with base case} \qquad B(1) = 1$$

Solution:



$$\sum_{i=0}^{H} \frac{n}{6^i} \to n \sum_{i=0}^{k} \frac{1}{6^i}$$

$$= n \cdot \frac{\frac{1}{6}}{1-\frac{1}{6}}$$

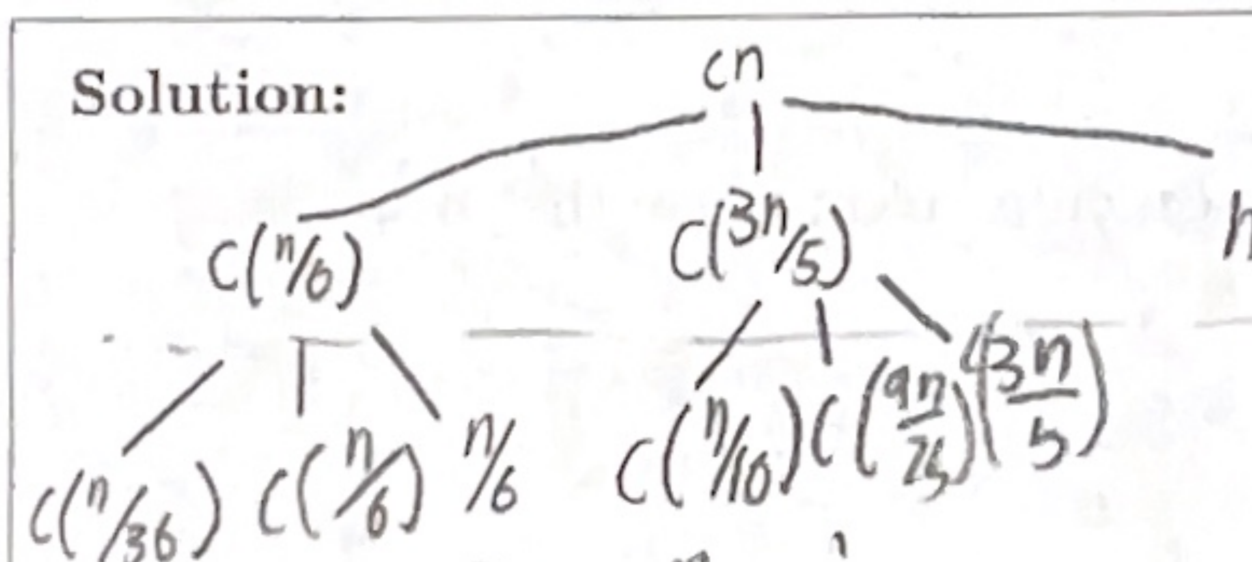
$$= n \cdot \frac{\frac{1}{6}}{5/6}$$

$$= \frac{1}{5} n = O(n)$$

(c) Asymptotically solve the following recurrence for $C(n)$ for $n \geq 0$.

$$C(n) = C(n/6) + C(3n/5) + n \qquad \text{with base case} \qquad C(0) = 0$$

Solution:



$-n$

$3n/5 + n/6$

$-\ 3^k n/5^k + n/6^k$

$$cn = \sum_{i=0}^{H} \frac{3^i n}{5^i} + \frac{n}{6^i}$$

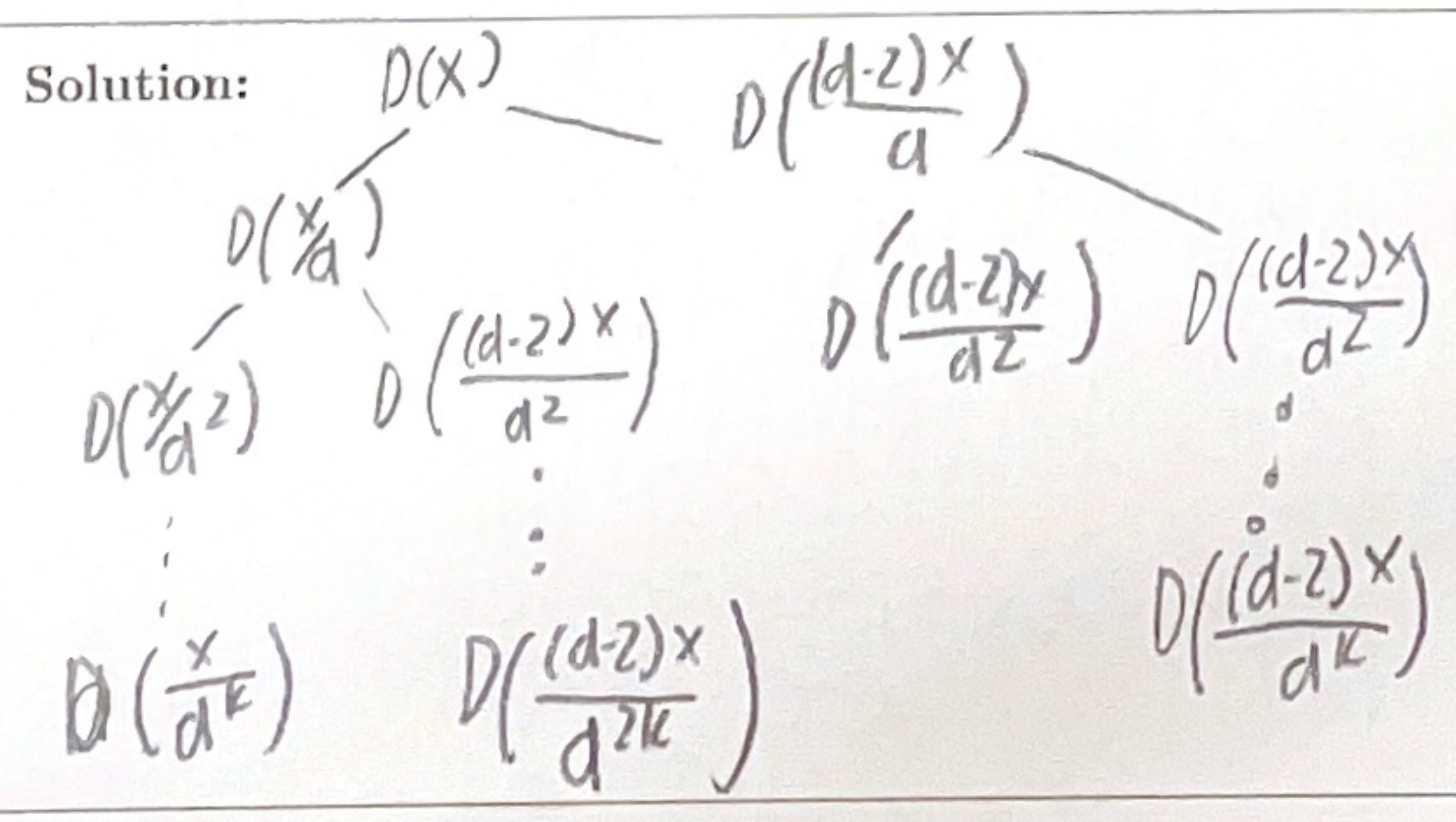
$$= n \sum_{i=0}^{H} \left(3/5 + 1/6\right)^i$$

$= O(n)$ constant

(d) Let $d > 3$ be some arbitrary constant. Then solve the following recurrence for $D(x)$ where $x \geq 0$.

$$D(x) = D\left(\frac{x}{d}\right) + D\left(\frac{(d-2)x}{d}\right) + x \qquad \text{with base case} \qquad D(0) = 0$$

Solution:



$$= \sum_{i=0}^{k} \left(\frac{x}{d^i} + \frac{(d-2)x}{d^i}\right)$$

$$= x \sum_{i=0}^{k} \frac{1}{d^i} + (d-2)x \sum_{i=0}^{k} \frac{1}{d}$$

$$= x + (d-2)x$$

$$= (d-2)x$$

$$= O(dx)$$