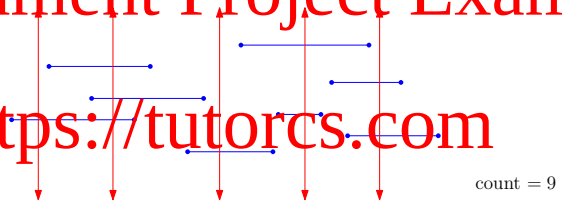
CS/ECE 374 A (Spring 2022)
Homework 5 (due March 3 Thursday at 10am)

Instructions: As in previous homeworks.

In algorithm design questions, usually an algorithm is best described by pseudocode (not actual code!), along with explanation or justification of correctness (especially if it is not obvious), and analysis of the running time. See `https://courses.engr.illinois.edu/cs374/sp2022/A/hw_policies.html#content`.

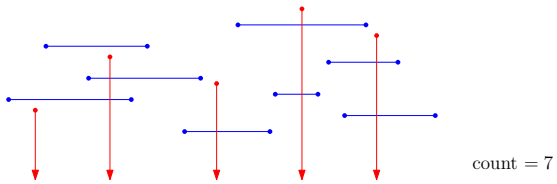
Problem 5.1:

(a) (20 pts) Suppose that we are given a set H of horizontal line segments and a set V of vertical lines with $|H| + |V| = n$. (A horizontal line segment has two endpoints and can be specified by two x-coordinates and one y-coordinate; a vertical line is unbounded from above and from below, and can be specified by one x-coordinate.) Describe an $O(n \log n)$-time algorithm to count the total number of intersections between H and V. You may use sorting as a subroutine.



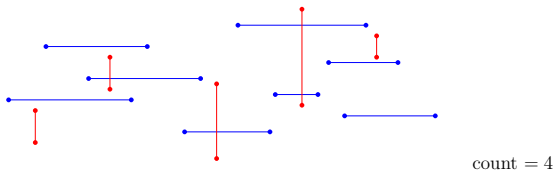
count = 9

(b) (70 pts) Next, suppose that we are given a set H of horizontal line segments and a set V of vertical downward rays with $|H| + |V| = n$. (A vertical downward ray is unbounded from below, and can be specified by the x- and y-coordinates of its top endpoint.) Describe an algorithm to count the total number of intersections between H and V. Your algorithm should use divide-and-conquer and have running time $O(n \log^2 n)$ or better. You may (and should) use part (a) as a subroutine. [Hint: divide using a median horizontal line. . .]



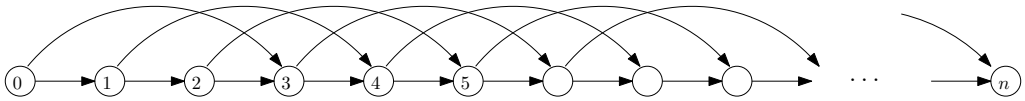
count = 7

(c) (10 pts) Finally, suppose that we are given a set H of horizontal line segments and a set V of vertical line segments with $|H| + |V| = n$. Describe an algorithm to count the total number of intersections between H and V. Your algorithm should have running time $O(n \log^3 n)$ or better. You should use part (b) as a subroutine. [Hint: one way is to use divide-and-conquer again, but there is also a slicker way, using (b)...]



[Note: You may assume that all x-coordinates and all y-coordinates are distinct.]

Problem 5.2: Consider the following directed graph G_n:



We would like to count the number of the paths that go from vertex 0 to vertex n in G_n. Let X_n denote this number.

It is not difficult to see that X_n satisfies the recurrence

$$X_n = X_{n-1} + X_{n-3} \tag{1}$$

for all $n \geq 3$ (since we can enter vertex n either from vertex $n-1$ or from vertex $n-3$). For the base cases, $X_0 = X_1 = X_2 = 1$.

From this recurrence, it is straightforward to obtain an algorithm that computes X_n using $O(n)$ arithmetic operations. But we can do better...

(a) (10 pts) Prove that for all $m, n \geq 2$,

$$X_{m+n} \;=\; X_m X_n + X_{m-2} X_{n-1} + X_{m-1} X_{n-2}.$$

[Hint: in G_{m+n}, a path from vertex 0 to vertex $m+n$ may either go through vertex m, or skip over m in two possible ways...]

(b) (10 pts) Use part (a) (and Eq. (1))[1] to express $X_{2n}, X_{2n-1}, X_{2n-2}$ in terms of X_n, X_{n-1}, X_{n-2}.

(c) (45 pts) Using part (b), design and analyze an algorithm that computes X_n for a given n, using only $O(\log n)$ arithmetic operations.

(d) (35 pts) For large n, the number X_n is exponentially large (how many bits?), and so we can't assume that arithmetic operations take constant time. Show that your algorithm in part (c) can be implemented in $O(n^{\log_2 3})$ time, if multiplications are done using Karatsuba's algorithm. (In contrast, the naive approach using Eq. (1) would require $O(n^2)$ time when bit complexity is taken into account.)

[Note: If you are unable to do (a), you can still do (b)–(d), assuming the formula from (a).]

[1] It may be helpful to know, from Eq. (1), that $X_{n-3} = X_n - X_{n-1}$, for example.