CS120: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Problem Set 2

Harvard SEAS - Fall 2021 *Due: Wed Sep. 22, 2021 (12 noon)*

1. (reductions) Consider the following computational problem:

Input	: Points $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ in the $\mathbb{R}^2$ plane that are the vertices of a convex polygon (in an arbitrary order) whose interior contains the origin
Output	: The area of the polygon formed by the points

Computational Problem AreaOfConvexPolygon

(a) Give an $O(n)$-time *reduction* of AreaOfConvexPolygon to Sorting arrays of length n. That is, you should give an algorithm that solves AreaOfConvexPolygon using one call to a hypothetical "oracle" that can instantly solve Sorting on arrays consisting of n item-key pairs . Treating the oracle call as a single time step, your algorithm should run in in $O(n)$ time. In this part, you may assume that a point $(x, y) \in \mathbb{R}^2$ can be converted into polar coordinates (r, θ) in constant time.

You may find the following useful:

- The polar coordinates (r, θ) of a point (x, y) are the unique real numbers $r \geq 0$ and $\theta \in [0, 2\pi)$ such that $x = r\cos\theta$ and $y = r\sin\theta$. Or, more geometrically, $r = \sqrt{x^2 + y^2}$ is the distance of the point from the origin, and θ is the angle between the positive x-axis and the ray from the origin to the point.
- The area of a triangle is $A = \sqrt{s(s-a)(s-b)(s-c)}$ where a, b, c are the side lengths of the triangle and $s = \frac{a+b+c}{2}$ (Heron's Formula).

(b) Deduce that AreaOfConvexPolygon can be solved without any Sorting oracle in time $O(n \log n)$.

(c) Come up with a way to avoid conversion to polar coordinates and any other trigonometric functions in solving AreaOfConvexPolygon in time $O(n \log n)$. Specifically, design an $O(n)$-time reduction that makes $O(1)$ calls to a Sorting oracle on arrays of length at most n, using only arithmetic operations $+$, $-$, $\times$, $\div$, and $\sqrt{\ }$, along with comparators like $<$ and $==$. (Hint: first partition the input points according to which quadrant they belong in, and consider $\tan\theta$ for a point with polar coordinates (r, θ).)

Similar techniques to what you are using in this problem are used in algorithms for other important geometric problems, like finding the Convex Hull of a set of points, which has applications in graphics and machine learning.

2. (modelling algorithms as decision trees, proving lower bounds) In our 9/7 class and the Zoom video recorded by Salil to complete the lecture, we proved a lower bound of $\Omega(n \log n)$ on runtime of comparison-based sorting algorithms. (Watch Salil's video under the Zoom tab in Canvas if you haven't already done so.) In this problem, you'll use a similar technique to show that searching a sorted array requires time at least $\Omega(\log n)$, and thus binary search is asymptotically optimal. Specifically, we are interested in the following computational problem:

Input	: A *sorted* array A of item-key pairs $((I_0, K_0), (I_1, K_1), \ldots, (I_{n-1}, K_{n-1}))$, where each key $K_i \in \mathbb{R}$, and a search key $K \in \mathbb{R}$
Output	: A pair (I_i, K_i) such that $K_i = K$ if one exists, else $\perp$

Computational Problem SearchSortedArray

The measure of efficiency we will consider for this problem is the *number of memory accesses*. That is, to read any of the $2n + 1$ keys or items in the input costs the algorithm one time step. You will prove that worst-case running time of any algorithm is $\Omega(\log n)$ even when restricted to the set $\mathcal{I}$ of inputs where the keys K_i come from the set $\{0, 1, 2\}$, the items are $I_i = i$ for $i = 0, \ldots, n - 1$, and search key is $K = 1$. Thus the worst-case running time over all inputs must also be $\Omega(\log n)$.

(a) Describe how the behavior of every algorithm for SearchSortedArray on inputs in $\mathcal{I}$ can be modelled as *ternary* tree whose depth is the worst-case number of memory accesses. (A ternary tree is a tree where every node has at most 3 children.)

(b) Prove that the aforementioned tree must have at least n distinct leaves, and hence has depth $\Omega(\log n)$.

(c) (*challenge) Show that the depth of the tree is at least $\log_2 n - O(1)$.