15-451/651 Algorithm Design & Analysis, Spring 2025

Recitation #12

Objectives

- Problem-Solving with Computational Geometry
- Using tools such as Convex Hull Algorithms, Randomized Incremental Algorithms, etc.

Recitation Problems

1. **(Width of a Set of Points)** You're given a set $S = \{p_1, \ldots, p_n\}$ of n points in the plane. A strip of width w is the region between two parallel lines, where the distance between the two lines is w. The goals is to find the strip of minimum width that contains all the points.

   (a) Argue that the smallest strip containing S is the same as the smallest strip containing the convex hull of S.

   (b) Now, argue that there exists a strip of minimum width whose sides are parallel to one of the sides of the convex hull.

   (c) Using Part (a), give an $O(n^2)$ algorithm for this problem.

   (d) Now we make an important observation that allows us to speed up the algorithm. Suppose you have a side of the convex hull (a_i, a_{i+1}) and you know that the farthest point from (a_i, a_{i+1}) is the point q_j. Argue that if I consider the next counterclockwise side of the polygon (a_{i+1}, a_{i+2}), that the farthest point from this side is some $q_{j'}$ where $j' \in [j, i]$. In words rather than math, the new farthest point $q_{j'}$ is counterclockwise of q_j and before a_i.

   (e) Use the observation in Part (d) to give an $O(n \log n)$ algorithm for this problem.

2. **(Randomized Dart Throwing)** Consider the following randomized dart game and its associated cost function:

   Procedure `DartGame`:

   i Initially. there is a dart board of n consecutive, empty squares, arranged in a row.

   ii For n iterations, throw a dart at a uniformly random empty square, and pay cost equal to the number of consecutive empty squares to the left and right of the dart.

   The goal is to analyze the expected cost of this procedure. In this problem we will explore two alternative ways to analyze the expected cost of the game, using the ideas from ***randomized incremental algorithms*** and their analysis.

   In both cases, our goal is to bound the *expected cost of the i^{th}* iteration of the game, call it c_i. The total cost, by linearity of expectation, will therefore be

$$\mathbb{E}[\text{total cost}] = \mathbb{E}\left[\sum_{i=1}^{n} c_i\right] = \sum_{i=1}^{n} \mathbb{E}[c_i].$$

   (a) We can evaluate this expectation by conditioning on the locations of all the previous darts at time i. Derive an expression for the expected cost for the i^{th} iteration of the game:

$$\mathbb{E}[c_i] = \sum_{\substack{\text{All possible} \\ \text{dart locations} \\ |D|=i-1 \\ D \subseteq [n]}}$$

   Instead, to bound $\mathbb{E}[c_i]$, we consider the state of the game after i iterations - when there are i darts on the board. We now consider all possible locations at which the i^{th} dart could have landed[1]. We want to compute the expected cost over these possibilities.

   (b) (**Probability method**) Consider a particular *empty square* on the dart board. What is the probability that this square contributes to c_i? i.e., what is the probability that this empty square is one of the consecutive empty squares to the left or right of the i^{th} dart?

   (c) Use Part (b) to compute $\mathbb{E}[c_i]$.

   (d) (**Average-cost method**) We can simplify the problem with one very important observation. We don't actually need to compute the cost of the i dart locations individually, we only need to know the *expected cost* over all the possible locations. Another way to put it is that we only need to know the *average cost* of all the possible locations that the dart could land in. The average cost can be computed as the **total sum cost** of the possible dart locations divided by the number of possible locations, which will give us the expected cost without needing to compute the cost of any particular location individually! Use this idea to compute $\mathbb{E}[c_i]$.

[1] This technique is sometimes called *backward analysis* because instead of looking at the current state of the game and casing over all of the possible *next* moves, we look at the current state of the game and case over all of the possible *previous* moves, i.e., we look at the current state and ask "how did I get here"? We look *backwards* one step instead of looking forward one step.

(e) Compute the expected cost of the game using Part (c) or (d).