Coding Assignment 5

ECS 122A Algorithm Design and Analysis

Radio Towers Yet Again – Practice

Problem Description

Note. The following problem is slightly different from CA5, the only difference being that we don't insist that every city is covered by a single tower, rather that it be covered by at least one.

In ByteLand, there are n cities located along the x axis with the i^{th} city located at $(A_i, 0)$ for $1 \le i \le n$.

In ByteLand, there is a telecommunications company that manufactures radio towers. Each tower has radius of coverage d, i.e., a tower at $(y, 0)$ covers a city at $(x, 0)$ if and only if $|x - y| \le d$.

The company have identified m potential locations for placing radio towers. The j^{th} of these locations is at $(B_j, 0)$. The company is allowed to place radio towers only at these locations.

For every integer k with $1 \le k \le m$, compute the number of ways to place radio towers at **exactly** k out of the m given locations so that every city is covered by **at least one** tower. Since the answer may be large, output it modulo $10^9 + 7$, i.e., if the answer is x, print the remainder when x is divided by $10^9 + 7$.

Solution

We use dynamic programming technique to solve this problem. The first step is to identify suitable smaller sub-problems from which we can compute the answer for our desired problem. Here is one way to do so. Let $\mathsf{dp}(i, j)$ denote the number of ways to choose tower positions so that (i) the rightmost tower is at B_i and (ii) exactly j positions are chosen and (iii) every city to the left of B_i is covered.
We can derive a recurrence relation for $\mathsf{dp}(i, j)$ by fixing the immendiate next tower to the left of B_i. Suppose the tower to the left of B_i was $B_{i'}$. This means that every city between $B_{i'}$ and B_i must be covered by one of these towers. Equivalently, there must not be any city in the interval $[B_{i'} + d + 1, B_i - d - 1]$. This immediately gives us the following recurrence. Let $\mathsf{count}(i', i)$ denote the number of cities in the interval $[B_{i'} + d + 1, B_i - d - 1]$. [1]

$$\mathsf{dp}(i, j) = \begin{cases} \sum\limits_{i' < i} \mathbb{1}[\mathsf{count}(i', i) = 0] \cdot \mathsf{dp}(i', j), & \text{if } j > 1 \\ \mathbb{1}[A_1 \ge B_i - d], & \text{otherwise} \end{cases}$$

Given i', i we can compute $\mathsf{count}(i', i)$ using binary search over the array A in $O(\log n)$ time. Hence the above recurrence can be implemented as an $O(m^3 \log n)$ time solution to the problem. To optimize this we can observe that the values i' that satisfy $\mathsf{count}(i', i) = 0$ form a contiguous subarray $[i^\star, i - 1]$ for some $i^\star$.
Computing $i^\star$: More specifically $i^\star$ can be computed as follows: Suppose last_i is first city to the left of B_i that is uncovered (i.e., $A_{\mathsf{last}_i} < B_i - d$), then $i^\star$ is the first tower position that is at least $A_{\mathsf{last}_i} - d$. If every city to the left of B_i is already covered by B_i then, we set $i^\star = 1$. It is easy to see that $i^\star$ may be computed in $O(\log n)$ time using binary search (or even amortized $O(1)$ time).
Computing the summation: Knowing $i^\star$, we can use prefix sums to compute sub-array sums efficiently. More precisely we define $\mathsf{sdp}(i, j) = \sum\limits_{i' \le i} \mathsf{dp}(i', j)$ and then we can compute $\mathsf{sdp}(i, j) = \mathsf{sdp}(i - 1, j -$

[1] We use the notation $\mathbb{1}[\text{condition}]$ to return 1 whenever condition is true and 0 otherwise.

$1) - \mathsf{sdp}(i^\star - 1, j-1)$. We can compute $\mathsf{sdp}(i,j)$ along with $\mathsf{dp}(i,j)$ in $O(1)$ time using the recurrence $\mathsf{sdp}(i,j) = \mathsf{sdp}(i-1,j) + \mathsf{dp}(i,j)$. With this additional observation we can reduce the time complexity to $O(m^2 \log n)$ or $O(m^2)$.

Computing the final answer using `dp` *table*:

Method 1. Once we have computed the `dp` table, we can compute the final answer by aggregating over all possible locations of the rightmost tower, i.e, the j^{th} answer is given by $\sum_i \mathbb{1}[A_n \leq B_i + d] \cdot \mathsf{dp}(i,j)$. Note that the condition $A_n \leq B_i + d$ ensures that all cities are covered.

Method 2. Alternatively we can modify the input by adding an imaginary city and tower location at $\max(A_n, B_m) + d + 1$, i.e., define $A_{n+1} = \max(A_n, B_m) + d + 1$ and $B_{m+1} = A_{n+1}$. We then compute the `dp` table for this new instance. Observe that every valid choice of tower locations must contain the imaginary location B_{m+1} to cover the imaginary city. Hence the answer to our problem is simply $\mathsf{dp}(m+1, j-1)$.