Coding Assignment 2

Due: 31 October, 2025 6pm PT

About This Assignment

- Please download the file `CA2.zip` from Canvas (see the corresponding Canvas assignment). It contains all the visible test cases for the problem, as well as template code that handles input/output for you.
- Please submit your solution on Gradescope. Your solution will be evaluated on the visible test cases and a set of hidden test cases.

Radio Towers

Problem Description

In LineLand, there are n ($\leq 10^5$) cities located along the x axis with the i^{th} city located at $(A_i, 0)$ ($0 \leq A_i \leq 10^9$ for $1 \leq i \leq n$).

In LineLand, a telecommunications company manufactures radio towers. Each tower has a radius of coverage d (input to the algorithm), i.e. a tower at $(y, 0)$ covers a city at $(x, 0)$ if and only if $|x - y| \leq d$.

The company has identified m (≤ 1200) potential locations for placing radio towers. The j^{th} of these locations is at $(B_j, 0)$ ($B_j \leq 10^9$ for $1 \leq j \leq m$). The company is allowed to place radio towers only at these locations. The company also faces difficulties if more than one tower services a city. For example, suppose there are two towers T_1, T_2, and some city A_i within the coverage regions of both towers. In such a scenario, devices in A_i cannot decide which tower among T_1, T_2 to communicate with. To avoid this problem, they would like to build towers so that such a scenario never occurs. Hence, the company wants to place some towers so that (i) every city is covered by at least one tower and (ii) no city is covered by more than one tower.

To help them find such an assignment, solve the following problem. For every integer k with $1 \leq k \leq m$, compute the number of ways to place radio towers at exactly k out of the m given locations so that each city is covered by a single 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 $10^9 + 7$.

Hint: You may want to solve the following (seemingly harder) problem: For every integer $1 \leq i \leq m$, and $1 \leq j \leq m$, how many ways can we choose tower locations so that (i) the rightmost tower is at $B[i]$, (ii) there are exactly j towers selected (including $B[i]$) among $B[1], B[2], \ldots B[i]$ and (iii) every city to the left of $B[i]$ is uniquely covered by a tower? See if you can use these questions to reduce the original problem into smaller sub-problems. The sample solution has complexity $O(m^2 + n)$ or $O(m^2 \log n + n)$. If your solution takes $\Theta(m^3)$ time, you should expect (at most) **60%** of the points.

Input

Each input file contains several test instances. The first line contains a single integer C, representing the number of test instances. The description of the C instances follows.

Each test instance consists of three lines:

- The first line contains three space-separated integers n, m, d.
- The second line contains n space-separated integers $A_1, A_2, \ldots A_n$.
- The third line contains m space-separated integers $B_1, B_2, \ldots B_m$.
- It is guaranteed that the city and potential tower locations are given in increasing order, i.e., $A_i > A_{i-1}$ and $B_j > B_{j-1}$ holds for every $1 < i \leq n$ and $1 < j \leq m$.

Output

Output m space separated integers, where the i^{th} integer represents the number of ways to select exactly i towers to uniquely cover all the cities.

In this assignment, you are provided with sample submissions that contain code for input and output. You only need to complete a function that, given d, and the locations of the cities and towers, returns an array of m integers.

Warning: The sample case is not part of the visible test cases.

Constraints

Each input file contains at most 5 test instances. The time limit (total for all test instances in a single file) is 1 seconds for C/C++ and 3 seconds for Python. Each test instance satisfies the following additional constraints:

- $1 \leq n \leq 10^5$.
- $1 \leq m \leq 1200$.
- $1 \leq d \leq 10^9$.
- $1 \leq A_i, B_j \leq 10^9$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.
- $A_i > A_{i-1}$ for $1 < i \leq n$ and $B_j > B_{j-1}$ for $1 < j \leq m$.

The following additional constraints are satisfied for 60% of the test cases.

- $1 \leq m \leq 120$.
- Sum of m across all instances is at most 240.

(**Warning**: For Python users, you are advised against using the standard dictionary data structure, which is rather slow.)

Test Cases

Your program will be evaluated on 5 visible test cases and 5 hidden test cases. Each test case is worth 0.6 points.

Sample Input 1:

```
2
5 5 1
1 2 3 4 5
1 2 3 4 5
60 60 2
1 2 3 8 9 10 15 16 17 22 23 24 29 30 31 36 37 38 43 44 45 50 51 52 57 58 59 64 65
66 71 72 73 78 79 80 85 86 87 92 93 94 99 100 101 106 107 108 113 114 115 120 121
122 127 128 129 134 135 136
1 2 3 8 9 10 15 16 17 22 23 24 29 30 31 36 37 38 43 44 45 50 51 52 57 58 59 64 65
66 71 72 73 78 79 80 85 86 87 92 93 94 99 100 101 106 107 108 113 114 115 120 121
122 127 128 129 134 135 136
```

Sample Output 1:

```
0 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 486784380 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Sample Explanation

Sample Input 1 contains $C = 2$ test instances. In both instances, the location of the cities and potential tower locations are identical.

In the first test instance, the locations are {1, 2, 3, 4, 5} with $d = 1$. There are two ways to cover the cities. The first is by placing towers at {1, 4} and the second way is at {2, 5}. In the first way cities at $A[1], A[2]$ are covered only by the tower at $B[1]$ and cities at $A[3], A[4], A[5]$ are covered only by the tower at $B[5]$. Both ways require 2 towers. Hence, the answer is {0, 2, 0, 0, 0}.

In the second test instance, the towers are placed such that for every $i \le i \le 20$, the towers $B[3 \cdot i - 2], B[3 \cdot i - 1], B[3 \cdot i]$ only cover $A[3 \cdot i - 2], A[3 \cdot i - 1], A[3 \cdot i]$. Hence, in any valid tower placement, for every $1 \le i \le 20$, we must choose exactly one tower among $B[3 \cdot i - 2], B[3 \cdot i - 1], B[3 \cdot i]$. It is also not hard to verify that any such choice of towers covers all cities according to the given constraint. Hence, all valid towers placement requires exactly 20 towers, and the number of such ways is exactly 3^{20}. Note that $3^{20} = 3 \cdot (10^9 + 7) + 486784380$. Hence, you should output 486784380 for $k = 20$ and 0 for the rest.

Submission Guideline

Write your program in either C, C++, or Python **in a single file**. Submit the file on Gradescope. The time limit on Gradescope is 1 seconds for C/C++ and 3 seconds for Python. You can make at most 10 submission attempts. **You may refer to or directly use the sample code files sample.cpp or sample.py that take input and write output.**