|  |  |  |  |
| --- | --- | --- | --- |
| public int[] getIntersections() { |  | |  |
| int x =0; | O(1) | | O(1) |
| int y=0; | O(1) | | O(1) |
| int counter=0; | O(1) | | O(1) |
| int [] result = new int [array.length]; | O(1) | | O(1) |
| MyHashTable <Position> table = new MyHashTable<Position>(); | O(1) | | O(1) |
|  |  | |  |
| for (int i = 0; i<array.length; i++) { | O(N) | O(N) | O(N) |
| if (array[i] == A2Direction.LEFT) | O(1) | O(1) |
| x--; | O(1) |
| else if (array[i] == A2Direction.UP) | O(1) |
| y--; | O(1) |
| else if (array[i] == A2Direction.RIGHT) | O(1) |
| x++; | O(1) |
| else if (array[i] == A2Direction.DOWN) | O(1) |
| y++; | O(1) |
|  |  |  |
| Position point = new Position(x, y); | O(1) | O(1) |
| if (table.contains(point)) | O(1) | O(1) |
| result[counter++] = i; | O(1) |
| else | O(1) |
| table.insert(point); | O(1) |
| } |  |  |
| int [] finalResult = new int [counter]; | O(1) | | O(1) |
| for (int i=0; i< counter ; i++) { | O(1) | | O(1) |
| finalResult[i] = result[i]; | O(1) | |
| } |  | |
| return finalResult; | O(1) | | O(1) |
| } |  |  |  |

**Complexity (amortized)** = O(1)+ O(1)+ O(1)+ O(1)+ O(1)+ O(N)+ O(1)+ O(1)+ O(1) = O(N)

**Complexity (worst-case)** = O(1)+ O(1)+ O(1)+ O(1)+ O(1)+O(N2)+ O(1)+ O(N) = O(N2)