

Design and Analysis of Algorithms

Problem 1 — Counting Inversions

You are given an array $A[1..n]$ of integers. A pair of indices (i, j) with $1 \leq i < j \leq n$ is called an *inversion* if $A[i] > A[j]$ — a pair that appears out of order. The number of inversions measures how far the array is from being sorted: a sorted array has 0 inversions, while a strictly decreasing array attains the maximum of $n(n-1)/2$.

Task. Design an efficient algorithm that counts the total number of inversions in A , and state its running time. Values may repeat, and equal values do *not* form an inversion.

Constraints: $1 \leq n \leq 10^5$.

Test cases (input -> expected output)

Array A	Inversions	Note
[2, 4, 1, 3, 5]	3	generic
[1, 2, 3, 4, 5]	0	already sorted (minimum)
[5, 4, 3, 2, 1]	10	reversed (maximum = $n(n-1)/2$)
[7]	0	single element
[]	0	empty array
[2, 1, 2, 1]	3	duplicates (equal values are not inversions)

Problem 2 — Even-Sum Sub-arrays

For an array A of n positive integers, a *sub-array* is a contiguous block $A[i..j]$ with $1 \leq i \leq j \leq n$, and its *sum* is the sum of its elements. There are $n(n+1)/2$ sub-arrays in all.

Task. Design a algorithm that counts how many sub-arrays of A have an *even* sum, and state its running time. **Constraints:** $1 \leq n \leq 10^5$.

Test cases (input -> expected output)

Array A	Even-sum count	Note
[1, 2, 3, 4]	4	generic
[4]	1	single even element
[7]	0	single odd element

Array A	Even-sum count	Note
[2, 4, 6]	6	all even -> every sub-array even = $n(n+1)/2$
[1, 1, 1]	2	all odd -> only even-length sub-arrays
[1, 2, 3, 4, 5]	6	larger mixed case

Problem 3 — Lights On

Over a single day, n people use a room. Person i enters at time a_i and leaves at time b_i (with $b_i > a_i$), and all $2n$ times are distinct. The lights are initially off. Whenever a person enters an *empty* room they switch the lights on; whenever a person leaves and the room becomes empty they switch them off; the next person to enter switches them on again.

Task. Given the intervals (a_i, b_i) , determine how many times the lights are switched *on* during the day. (Equivalently: into how many disjoint stretches of time is the room occupied?) State the running time. **Constraints:** $1 \leq n \leq 10^5$.

Test cases (each row is a set of visits -> expected output)

Visits (enter, exit)	Switch-ons	Note
[1,4] [2,3] [6,9] [7,8]	2	two occupied blocks
[1,5]	1	single visit
[1,2] [3,4] [5,6]	3	all disjoint
[1,10] [2,4] [5,9]	1	all inside one block
[1,3] [2,5] [4,7]	1	overlaps chain into one block
[1,2] [3,6] [4,5] [7,10] [8,9]	3	three separate blocks

Problem 4 — Two-Payload Drone Fleet

A logistics hub must airlift n parcels with masses $w_1, \dots, w_n$. Each drone *sortie* can carry at most *two* parcels, and the combined mass on a sortie must not exceed the drone capacity C . Every parcel individually satisfies $w_i \leq C$, so each is liftable on its own.

Task. Determine the minimum number of sorties needed to deliver all parcels, and state the running time. **Constraints:** $1 \leq n \leq 10^5$, $1 \leq w_i \leq C$.

Test cases (masses, C -> expected output)

Masses	C	Sorties	Note
[3, 2, 2, 1]	3	3	generic
[5]	5	1	single parcel
[2, 2, 2, 2]	4	2	everything pairs up
[3, 3, 3, 3]	5	4	nothing fits together (3+3 > 5)
[1, 2, 3]	4	2	{1,3} exact fit, {2} alone
[5, 1, 1, 1]	5	3	heaviest = C flies alone

Problem 5 — Archive Grid Lookup

A cold-storage archive holds $r \cdot c$ crates with distinct serial numbers on an $r \times c$ grid, filled in increasing order *row by row*: reading the grid left-to-right and top-to-bottom lists the serials in strictly increasing order. (Equivalently, each row increases, and the first serial of a row exceeds the last serial of the row above.)

Task. Given a query serial q , decide whether a crate with serial q exists. The algorithm must run in $O(\log(r \cdot c))$ time; you may *not* read all $r \cdot c$ cells. **Constraints:** $1 \leq r \cdot c \leq 10^6$.

For the test cases below, Grid = $[[1, 3, 5, 7], [10, 11, 16, 20], [23, 30, 34, 60]]$.

Query q	Result	Note
16	present	
13	absent	falls in the gap $11 < 13 < 16$
1	present	smallest serial
60	present	largest serial
0	absent	below every serial
100	absent	above every serial

Edge cases to handle: a 1×1 grid, and an empty grid (always absent).

Problem 6 — Recirculating Magazine Dispenser

A dispenser holds a vertical magazine of n chips with distinct serial numbers, loaded from top to bottom. The firmware repeats the following until the magazine is empty: *eject* the top chip (it leaves the machine); then, if any chips remain, move the new top chip to the *bottom* of the magazine.

Task. Given the set of serials, output one top-to-bottom initial loading so that the chips are ejected in *strictly increasing* order of serial. Any valid loading is accepted. State the running time. **Constraints:** $1 \leq n \leq 10^5$; serials distinct.

Test cases (serials -> a valid loading, top -> bottom)

Serials	A valid loading (top -> bottom)
{2, 3, 5, 7, 11, 13, 17}	[2, 13, 3, 11, 5, 17, 7]
{42}	[42]
{1, 1000}	[1, 1000]
{1, 2, 3}	[1, 3, 2]
{1, 2, 3, 4}	[1, 3, 2, 4]
{1, 2, 3, 4, 5}	[1, 5, 2, 4, 3]
