Question 1

Use Case: Shipping Company Optimization

Background

A shipping company needs to efficiently allocate various shipments into available containers. Each shipment consists of a specific number of units, and each container has a maximum capacity of units it can hold. The goal is to determine the minimum number of containers required to accommodate all shipments, even if units from the same shipment are split across multiple containers.

Scenario

Company: SwiftLogistics

Shipments: The company has n shipments to be delivered. Each shipment consists of a different number of units. This is represented by the array shipments where shipments[i] indicates the number of units in the ith shipment.

Containers: The company also has m containers available for use, each with a specific capacity. This is represented by the array container\_limits where container\_limits[j] indicates the maximum number of units that the j-th container can hold.

Problem

The operations team at SwiftLogistics needs to figure out the minimum number of containers required to redistribute the units from all shipments into the available containers.

Example

Inputs:

• shipments = [10, 20, 30] • container\_limits = [15, 15, 20, 10]

Output:

• Minimum number of containers needed: 4 • If there is an insufficient container\_limit the function should return -1.

Explanation

1. The first container (15 units) can take 10 units from the first shipment and 5 units from the second shipment.

2. The second container (15 units) can take the remaining 15 units from the second shipment.

3. The third container (20 units) can take the entire third shipment of 30 units, but since it exceeds its capacity, the fourth container (10 units) can be used to accommodate the remaining 10 units.

Thus, the minimum number of containers needed is 4

SOLUTION

import java.util.Arrays;

import java.util.PriorityQueue;

import java.util.Scanner;

public class Main {

public static int minContainersRequired(int[] shipments, int[] container\_limits) {

if (container\_limits.length == 0) return -1;

Arrays.sort(shipments);

for (int i = 0; i < shipments.length / 2; i++) {

int temp = shipments[i];

shipments[i] = shipments[shipments.length - 1 - i];

shipments[shipments.length - 1 - i] = temp;

}

PriorityQueue<Integer> containerQueue = new PriorityQueue<>();

for (int capacity : container\_limits) {

containerQueue.add(capacity);

}

for (int shipment : shipments) {

while (shipment > 0) {

if (containerQueue.isEmpty()) {

return -1;

}

int containerCapacity = containerQueue.poll();

if (containerCapacity >= shipment) {

containerCapacity -= shipment;

shipment = 0;

} else {

shipment -= containerCapacity;

containerCapacity = 0;

}

if (containerCapacity > 0) {

containerQueue.add(containerCapacity);

}

}

}

return container\_limits.length - containerQueue.size();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of shipments:");

int n = scanner.nextInt();

int[] shipments = new int[n];

System.out.println("Enter the units for each shipment:");

for (int i = 0; i < n; i++) {

shipments[i] = scanner.nextInt();

}

System.out.println("Enter the number of containers:");

int m = scanner.nextInt();

int[] container\_limits = new int[m];

System.out.println("Enter the capacity for each container:");

for (int i = 0; i < m; i++) {

container\_limits[i] = scanner.nextInt();

}

int result = minContainersRequired(shipments, container\_limits);

if (result == -1) {

System.out.println("-1");

} else {

System.out.println("Minimum number of containers needed: " + result);

}

scanner.close();

}

}

INPUTE :-

Enter the number of shipments: 3

Enter the units for each shipment: 10 20 30

Enter the number of containers: 4

Enter the capacity for each container: 15 15 20 10

Link

/2ODgS9aIuKibJ https://www.programiz.com/online-compiler