Lab 6

1. **Problem 1.** You are given a list of files, each with a specific size. The goal is to merge these files into a single file with the least amount of computation required. Merging two files of sizes a and b requires $a + b$ computations. Design an efficient algorithm to compute the minimum total number of computations required to merge all files into one.

   Input:

   $$files = [4, 3, 2, 6]$$

   Output: The minimum total cost to merge all files is **29**.

   Input:

   $$files = [1, 2, 3, 4, 5]$$

   Output: The minimum total cost to merge all files is **33**.

   Input:

   $$files = [10, 20, 30]$$

   Output: The minimum total cost to merge all files is **90**.

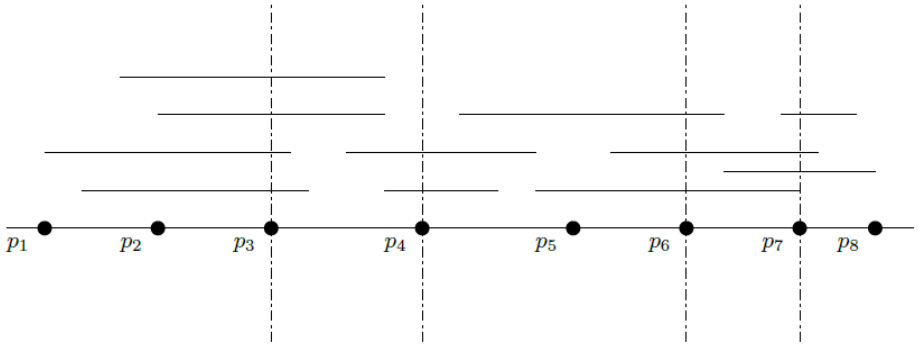
   Input:

   $$files = [5, 7, 9, 13]$$

   Output: The minimum total cost to merge all files is **62**.

2. **Problem 2.** You are given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points on a line. Note that p_i is left of p_j, where $1 \leq i < j \leq n$. You are also given a set of m intervals $I_1, I_2, \ldots, I_m$, where each interval I_j is of the form $[start_j, end_j]$ and $p_1 \leq start_j \leq end_j \leq p_n$. Design a $O(n \log n)$ time algorithm to find a subset $X \subseteq P$ of the smallest cardinality such that each interval contains at least one point from X. Justify the correctness.

   For example, in the figure mentioned below, $X = \{p_3, p_4, p_6, p_7\}$ is a subset of 4 points such that each interval contains at least one point from the subset X.



Test Case 1:

Input:

- Points: $P = \{1, 3, 4, 6, 7\}$
- Intervals:
  - $I_1 = [1, 4]$
  - $I_2 = [2, 5]$
  - $I_3 = [3, 6]$

Output:

- Selected points: $\{4\}$

Explanation:

- $I_1 = [1, 4]$, $I_2 = [2, 5]$, and $I_3 = [3, 6]$ are all covered by the single point 4.

Test Case 2

Input:

- Points: $P = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
- Intervals:
  - $I_1 = [1, 2]$
  - $I_2 = [4, 5]$
  - $I_3 = [7, 8]$

Output:

- Selected points: $\{2, 5, 8\}$

Explanation:

- Each interval is disjoint, so separate points are needed to cover them.

Test Case 3

Input:

- Points: $P = \{1, 2, 3, 4, 5, 6\}$
- Intervals:
  - $I_1 = [1, 6]$
  - $I_2 = [2, 5]$
  - $I_3 = [3, 4]$

Output:

- Selected points: $\{4\}$

Explanation:

- All the intervals overlap, so a single point, 4, can cover all intervals.

Test Case 4

Input:

- Points: $P = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- Intervals:
  - $I_1 = [1, 3]$
  - $I_2 = [4, 6]$
  - $I_3 = [2, 5]$
  - $I_4 = [6, 8]$

Output:

- Selected points: $\{3, 6\}$

Explanation:

- I_1 and I_3 are covered by point 3.
- I_2 and I_4 are covered by point 6.

Test Case 5

Input:

- Points: $P = \{1, 2, 3, 4, 5, 6, 7\}$
- Intervals:
  - $I_1 = [1, 4]$
  - $I_2 = [2, 6]$

- $I_3 = [4, 5]$
- $I_4 = [5, 7]$

Output:

- Selected points: $\{5\}$

Explanation:

- The point 5 is contained in all intervals, so it is sufficient to cover all of them.

3. You are given a set of n intervals $I_1, I_2, \ldots, I_n$ where each interval $I_j = [start_j, end_j]$. You need to partition the intervals into the minimum number of groups such that no two intervals in the same group overlap.

   Equivalently, each group should represent a schedule where all intervals are compatible (non-overlapping).

Objective

Design an efficient algorithm in $O(n \log n)$ to compute:

- The minimum number of groups required.
- The actual grouping of intervals.

Output

- Print the minimum number of groups.
- Print the intervals assigned to each group.

Test Cases

Test Case 1

Input: Intervals: $[0, 30], [5, 10], [15, 20]$

Output: Minimum groups: 2 Grouping: - Group 1: $[0, 30]$ - Group 2: $[5, 10], [15, 20]$

Test Case 2

Input: Intervals: $[1, 3], [2, 4], [3, 5], [7, 8]$

Output: Minimum groups: 2 Grouping: - Group 1: $[1, 3], [3, 5], [7, 8]$ - Group 2: $[2, 4]$

Test Case 3

Input: Intervals: $[1, 2], [3, 4], [5, 6]$

Output: Minimum groups: 1 Grouping: - Group 1: $[1, 2], [3, 4], [5, 6]$

Test Case 4

Input: Intervals: $[1, 10], [2, 7], [3, 6], [8, 9]$

Output: Minimum groups: 3 Grouping: - Group 1: $[1, 10]$ - Group 2: $[2, 7]$ - Group 3: $[3, 6], [8, 9]$