

## HOMEWORK BEFORE FINAL EXAM

- (1) Suppose that algorithm  $A$  has time complexity  $T_A(n) = 7T_A(n/2) + n^2$  and algorithm  $B$  has time complexity  $T_B(n) = kT_B(n/4) + n^2$ . Find the maximum value of integer  $k$  such that  $T_B(n) = O(T_A(n))$
- (2) You are given a list of  $n$  independent tasks,  $A = [a_1, a_2, \dots, a_n]$ , where  $a_i \in \mathbb{N}$  represents the workload of  $i^{\text{th}}$  task in units. A single core processor can process one task per time unit at a uniform rate  $k$ . When it works on task  $i$  for one time unit, it reduces the remaining workload by  $\min(k, \text{remaining workload of task } i)$ . A task with remaining workload zero is considered complete, and the machine may switch to any other incomplete task in the next time unit. The goal is to complete all tasks within at most  $h$  time units.

Write an  $O(n \log(\max_i a_i))$ -time algorithm to find the minimum processing rate  $k$  such that all tasks can be fully completed within the total  $h$  time units.

- (3) Given a non-empty string  $s$  composed only of digits, count the number of valid ways to interpret it as a sequence of letters using the mapping:

$$1 \rightarrow A, 2 \rightarrow B, \dots, 26 \rightarrow Z.$$

Rules:

- A single digit 1 – 9 can be map to a letter.
- Any two-digit number from 10 to 26 (inclusive) can map to letter.
- String with leading zeros (e.g., 06) are invalid.

Write an  $O(n)$ -time algorithm to count the total number of valid decodings. If there are none, return 0. For example, 226 can be decoded into three ways, BZ, VF and BBF.

- (4) Given a list of non-negative integers. Write an  $O(nk \log n)$ -time algorithm, where  $n$  is the number of elements in the list and  $k$  is the maximum length of the string representations of numbers, to arrange all these numbers in some order to create the numerically greatest possible concatenation when the numbers are written side by side. For example,  $[1, 2, 3]$  must be rearranged to 321 to maximize its numeric value.
- (5) Consider there are  $n$  courses labeled from 0 to  $n - 1$ . You are given a list of prerequisites pairs:  $Prerequisites = [[c_1, p_1], [c_2, p_2], \dots, [c_n, p_n]]$ , where  $[c_i, p_i]$  means you must complete course  $p_i$  before taking course  $c_i$ . Write an  $O(n + m)$ -time algorithm, where  $n$  is the number of courses and  $m$  is the number of prerequisite pairs, to find and return a valid order in which to complete all  $n$  courses so that every course is taken after all its prerequisites. If multiple valid orders exist, you may return any one. If no such order exists (for example, due to cyclic dependencies), return an empty list.
- (6) Let  $P = \{p_1, p_2, \dots, p_n\} \subset \mathbb{Z}^2$  be a finite set of  $n$  distinct points in the two-dimensional integer lattice, where each point is represented as  $p_i = (x_i, y_i)$ .

For any pair of points  $p_i$  and  $p_j$ , define the cost of connecting them as their Manhattan distance:

$$\text{cost}(p_i, p_j) = |x_i - x_j| + |y_i - y_j|.$$

You are permitted to construct a set of connections (i.e., undirected edges) between selected pairs of points in  $P$ , such that the following conditions are satisfied:

- The resulting structure is connected; that is, for any pair of points  $p_i$  and  $p_j$ , there exists a sequence of connected edges linking them.
- The total cost, defined as the sum of the costs of all selected connections, is minimized.

Write an  $O(n^2 \log n)$ -time algorithm to determine the minimum possible total cost required to connect all points in  $P$  under these conditions.