Algorithms. Assignment 4

Problem 1

Given an integer p, we wish to compute its n-th power p^n. Consider multiplication as the elementary operation, regardless of the size of the factors. (This makes sense especially when we deal with multiplication modulo a fixed number.)
The obvious algorithm multiplies p with itself $n-1$ times and therefore needs $O(n)$ multiplications. How can you compute p^n using only $O(\log n)$ multiplications?
(There are several possible strategies and ways to analyze them, but probably they are all based on the same idea.)

Problem 2

The following small examples are related to sorting, however they do not address the sorting itself. Rather, they shall demonstrate that various problems become relatively easy to solve when the items in the instances are sorted first. (Well, this is already a general solution hint ...)

(a) We are given two sets, each consisting of n numbers. These numbers are given in no particular order. Any two numbers x, y can be compared in $O(1)$ time. (Comparing means to figure out whether $x < y$, $x > y$ or $x = y$.) How much time is needed to decide whether the two given sets are equal? Give an algorithm that is as fast as possible.

(b) Given n intervals $[s_i, f_i]$ on the real line, we wish to compute the total length covered by them. Of course, we cannot simply add their lengths $f_i - s_i$ in $O(n)$ time, as the intervals may overlap. How can you compute the covered length correctly and still efficiently?

See the reverse page.

(c) We are given a set W of n words, each consisting of at most m characters. (Shorter words may be filled up to length m by a blank symbol.) We would like to abbreviate each word by its shortest unique prefix. That is, no two words in W must get the same abbreviation, and no abbreviation must be prefix of another one.

Example: The set of words
Hydrogen, Boron, Oxygen, Fluorine, Neon, Aluminium, Phosphorus, Sulfur, Argon, Calcium, Cobalt, Nickel, Gallium, Germanium
would be abbreviated
H, B, O, F, Ne, Al, P, S, Ar, Ca, Co, Ni, Ga, Ge.
(Right, this example is a bit tweaked such that the output consists of the usual chemical symbols.)

An obvious $O(n^2m)$ time algorithm to produce these abbreviations would check any two words against each other, and extend their abbreviations until a pair of distinct characters is found.

However, one can do better. Describe and analyze a condiserably faster algorithm. Do not forget to argue why your algorithm is correct.