September 10, 2020

Algorithms and Data Structures

The main objective here is to find efficient solution for computational problems arising in different application areas

Computational problems

① We are given a sequence $(a_1, a_2, \ldots, a_n)$ of positive integers and another positive integer B
Are there two different numbers in the sequence whose sum is B?
Do there exist $1 \leq i < j \leq n$ such that $a_i + a_j = B$?

② The input is the same as before.

Does there exist $I \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{i \in I} a_i = B \ ?$$

③ We are given a finite number of pairs of strings

$(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)$

Can we form a sequence (repeatation is allowed) of these pairs in such a way that the concatenation of the first components equals to the concatenation of the second components?

More formally:

Are there $1 \le i_1, i_2, \dots, i_k \le n$ such that

$$s_{i_1} s_{i_2} \cdots s_{i_k} = t_{i_1} t_{i_2} \cdots t_{i_k} \; ?$$

① We can answer the question quickly:

take all possible pairs a_i, a_j and check if $a_i \cdot a_j = \beta$

How many pairs are there?

$$\frac{n(n-1)}{2} = \binom{n}{2}$$

$$\frac{n(n-1)}{2} = \frac{n^2-n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

quadratic algorithm

- efficient when $n \approx 1000, 2000, \ldots$
- less efficient when $n \approx 1000000, \ldots$

Nevertheless, theoritically it is an efficient algorithm

This is not the most efficient one

(2) The same idea works here as well: take all subsets and check them.

How many subsets are there? 2^n

This is exponentially many cases

In the world of algorithms such an exponential algorithm counts an inefficient algorithm

Non exponential algorithms for this problem are not known; it's very likely that efficient algorithms for this problem don't exist at all.

③ No algorithm can solve this problem!

This can be proved!!!

In this course we will focus on efficient algorithms and problems which can be solved by effiient algorithms.

Sorting

We are given a sequence of numbers (or letters, or strings, objects wich can be compared)

Rearrange the sequence in such a way that the first element is the smallest one, the second one is the second smallest and so on

Input: $(a_1, a_2, \ldots, a_n)$

Output: $(a'_1, a'_2, \ldots, a'_n)$ where $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

We know a lots of efficient (sorting) algorithms to solve this problem.