Actually, most of the algorithms are alike, eventhough they are hsed forbdifferent purposes.

One of the most popular algorithms is 'Brute-force search'. This algorithm is used to find one certain element in an array by going the the whole array. It is not that effective, but a correct result is guaranteed with this method. The main disadvantage of this algorithm is its execution time when the array length is massive. For example, to find a right 4-digit password you will have to go through 9! combinations.

It is better not to use brute-force algorithm, eventhough it is much easier to design such an algorithm.

Here is a code explanation of why you should not use it:

```
BruteForce(needed_num):
 for elem in range(0, 9999):
  if elem == needed_num:
   return elem
```

If there are any information given about element of array that is needed to be found, it is preferred to use other algorithms like 'branch and bound method' (also known as 'BB', BnB' or 'B&B').

If we return to example with 4-digit password:

> Let's imagine that we have to figure out a 4-digit password and we also know that it starts with '6'. If we use brute-force algorithm, it will go though every sing combination all the way from '0000' to '9999' (or needed combination if we put up `break` command). On the other hand, if we use B&B method, we would not iterate through combinations that does not start with '6'. So, we have just cut out 9/10 of combinations and saved a huge amount of time.
>
> B&B method also works with other conditions. For example, we have to find a 4-digit number which value > 6000. That means that the first digit has to be 6 or greater. An algorithm will iterate through every combination from '6000' until a needed one.

Code example for B&B method (4-digit password problem):

```
BnB(clause_num, needed_num):
 for elem in range(clause_num, 9999):
  if elem == needed_num:
   return elem
```