# Sublist search algorithm

***Source:* https://github.com/rusevrosen/codehub.github.io/tree/main/Algorithms**

## 1. What is the sublist search algorithm?

**Sublist search** is an algorithm that is used to find whether a **sublist** (a smaller list) is contained within a larger list. The **sublist search** algorithm takes two lists as input and returns a Boolean value indicating whether the smaller list is contained within the larger list.

## 2. Code example.

```
#include <iostream>
#include <vector>


int sublistSearch(const std::vector<int>& list, int start, int end,
int target) {
  for (int i = start; i < end; i++) {
    if (list[i] == target) {
      return i;
    }
  }
  return -1;
}


int main() {
  std::vector<int> list = {1, 2, 3, 4, 5};
  int index = sublistSearch(list, 1, 3, 2);
  std::cout << "Element found at index: " << index << std::endl;
  return 0;
}
```

```
Output: Element found at index: 1
```

This is because the function searches the **`sublist {2, 3}`** for the element **`2`** and finds it at index **`1`** in the original list.

### *3. Code explanation.*

The example is a function that searches a given sublist of a list for a target element and returns the index at which it is found. The function takes in the following arguments:

- **`list:`** A reference to a vector of integers representing the list to be searched.
- **`start:`** An integer representing the start index of the sublist to be searched.
- **`end:`** An integer representing the end index of the sublist to be searched.
- **`target:`** An integer representing the element to be searched for in the list.

The function begins by iterating through the sublist from the start index to the end index, using a for loop. It then checks if the current element is equal to the target element. If it is, the function returns the index at which the element was found. If the element is not found, the function returns **`-1`**.

In the main function, the **`list {1, 2, 3, 4, 5}`** is defined and passed as an argument to the **`sublistSearch`** function. The start and end indices of the sublist to be searched are 1 and 3, respectively, which corresponds to the **`sublist {2, 3}`**. The target element is **`2`**. When the **`sublistSearch`** function is called, it searches the **`sublist {2, 3}`** for the element **`2`** and returns **`1`**, since **`2`** is at index **`1`** in the original list. The result is then printed to the console using the **`cout`** stream.

### *4. Time complexity.*

The time complexity of the example function is **`O(n)`**, where n is the number of elements in the sublist. This is because the function performs a linear search through the sublist, meaning that it will take longer to run as the size of the sublist increases.

In the worst case, the function will have to iterate through the entire sublist before finding the target element or determining that it is not present. This means that the time taken by the function is directly proportional to the size of the sublist.

For example, if the sublist has 10 elements, the function will take 10 steps to complete in the worst case. If the sublist has 100 elements, the function will take 100 steps to complete in the worst case, and so on. This means that the time complexity of the function is **`O(n).`**

There are more efficient algorithms for searching lists, such as binary search, which have a time complexity of **`O(log n).`** These algorithms require the list to be sorted, which may not always be possible or practical.

# I wish you happy learning,

# Your Rosen Rusev