Theory: Merge sort in Python

🕓 31 minutes 0 / 5 problems solved

Skip this topic Start practicing

233 users solved this topic. Latest completion was 2 days ago.

Merge sort is an efficient sorting algorithm: for a list of size n, it works in $O(n \log n)$ in the worst case. The main idea of the algorithm is to divide an input list into two equal parts, sort them recursively, and then, merge these parts into one sorted list. In this topic, we will learn how this algorithm can be implemented in Python.

§1. Implementing the merge procedure

An implementation of the **merge sort** algorithm can be divided into two procedures. The first one is a procedure that takes two sorted lists and merges their elements into one sorted list:

```python
def merge(left, right):
    merged = [0 for _ in range(len(left) + len(right))]
    i, j, k = 0, 0, 0

    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            merged[k] = left[i]
            i += 1
        else:

            merged[k] = right[j]

            j += 1

        k += 1



    while i < len(left):

        merged[k] = left[i]

        i += 1

        k += 1



    while j < len(right):

        merged[k] = right[j]

        j += 1

        k += 1



    return merged
```

First, we create a list named `merged` to store the sorted elements. Next, we start iterating through the input lists using the `while` loop. At each step, we compare the current elements of the lists, add a smaller one to the `merged` list and change the positions of the current elements. The loop is finished when all elements of one of the input lists are processed. If the other list contains some elements that were not considered in the first loop, they will be added to the end of the `merged` list in one of the next `while` loops. At the end of the function, we return the `merged` list with all the elements sorted in ascending order.

Let's see several examples of how the function works:

Current topic:

Merge sort in Python

Topic depends on:

✓ Merge sort

✓ Algorithms in Python

Table of contents:

```
left = [1, 3, 5, 7]
right = [2, 4, 6, 8]
merged = merge(left, right)
print(merged)  # [1, 2, 3, 4, 5, 6, 7, 8]

left = [1, 5, 7]
right = [5, 8, 10, 15, 20]
merged = merge(left, right)
print(merged)  # [1, 5, 5, 7, 8, 10, 15, 20]
```

As you can see from the last example, for lists with different sizes the function works correctly as well.

§2. Implementing merge sort

Using the **merge** procedure, the **merge sort** algorithm can be implemented as follows:

```
def merge_sort(lst):
    if len(lst) < 2:
        return lst

    mid = len(lst) // 2
    left = merge_sort(lst[:mid])
    right = merge_sort(lst[mid:])

    return merge(left, right)
```

A function named `merge_sort` takes a list named `lst` as input and returns the elements of the list sorted in ascending order. Initially, we check if the size of the input list is less than 2. If the condition holds, we return the list since in this case it is already sorted. Otherwise, we split the list into two parts, sort them recursively, and then, get the whole sorted list using the `merge` procedure and return it as a final result.

Below are several examples of how the function works:

```
lst = [4, 3, 1, 5, 2, 7, 8]
print(merge_sort(lst))  # [1, 2, 3, 4, 5, 7, 8]

lst = [3, -7, 9, 2, 1, 4, -8, 15]
print(merge_sort(lst))  # [-8, -7, 1, 2, 3, 4, 9, 15]
```

Report a typo

27 users liked this theory. **2** didn't like it. **What about you?**

Start practicing

Comments (1)   Hints (0)   Useful links (0)   Show discussion