Merge Sort

Vivek Singh

Information Systems Decision Sciences (ISDS)
MUMA College of Business
University of South Florida
Tampa, Florida

2018

Introduction

The merge sort is based on the divide and conquer methodology.

- It is a recursive algorithm that continually splits a list in halves.
- The list is split repeatedly till we end up with lists of one element each which are inherently sorted.
- Once this process is completed, we merge the sorted lists, combining them together into a bigger sorted list.
- The base case in this recursion is of course a list of size 1.

Python Implementation

By performing the sublist segmentation, we have reduced the total number of shifting operations to sort the list.

```
def merge_sort(input_list):
    if len(input_list)>1:
        mid = len(input_list)//2
        left_half = input_list[:mid]
        right_half = input_list[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i=0
        j=0
        k=0
```

Continued

```
while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                input_list[k]=left_half[i]
                i=i+1
            else:
                input_list[k]=right_half[j]
                j=j+1
            k=k+1

while i < len(left_half):
            print('First')
            input_list[k]=left_half[i]
            print(input_list[k])
            i=i+1
            k=k+1
```

Continued

```
while j < len(right_half):
                print('Second')
                input_list[k]=right_half[j]
                print(input_list[k])
                j=j+1
                k=k+1
a =  [5,1,66,35,72,26,6,7,267,25,24,46,25,54]
merge_sort(a)
[1, 5, 6, 7, 24, 25, 25, 26, 35, 46, 54, 66, 72, 267]
```

Explanation

- The list is repeatedly split into smaller lists that eventually results in lists of one element each.
- Lets call them parent and child lists just for the purpose of understanding.
- The variables i and j point to the 0th position of the child lists and k = 0 points to position 0 of the parent.
- Consider a list of 8 elements that is initially split into lists of 4 elements each and later into list of 2 elements and then into 1 at the end.
- In this case, the left child of size 1 is compared against the right child of size 1 and based on the result, the smaller of the two overwrites the 0th position in its parent list of size 2.
- Following that 'k' is incremented by 1, so that it is pointing to position 1 of the parent list.
- This logic is taken care of by the first while loop

Explanation

- The other two while loops are in place to handle the situation where there are elements leftover in either of the child lists that cannot be compared against any element of its sibling.
- This happens when the left child has its element(s) pushed to the parent list and only the right child alone has elements left to be sorted.
- In this case, the third while loop is called. If the left child has leftover element(s) in the left child, the second while will be called.
- The lists are then merged together progressively and the final sorted list is arrived at. The merge operation places the items back into the original list.

Analysis of Merge Sort

- To analyze merge sort's performance, we need to consider both halves of its implementation, the splitting and merging.
- From Binary search, we know that we can split a list in half 'log n' times where n is the size of the list.
- The next operation is the merge. Each item in the list is processed and eventually placed on the sorted list.
- Hence, the merge operation requires n operations.
- Hence, combining log n splits, each of which costs n results in a total of n log n operations.
- A merge sort is therefore O(n log n) algorithm.
- A critical drawback of merge sort is that it needs extra space to hold all the split lists and this may become too resource intensive for very large lists.

Summary

- Merge sort is based on the Divide and Conquer method to sort elements.
- It offers a great performance improvement over the traditional sorting methods.
- It offers O(n log n) performance but there is a significant tradeoff in space requirements to carry out this algorithm.
- Merge sort is based on recursion where each iteration splits the list and the sublists merge back into the main list after getting sorted.