Merge Sort in R

Rahul Goswami

2022-01-18

This function implements the merge sort algorithm. Let us consider a vector of size n. Intentionally we will take an asort of the vector. The merge sort algorithm is a simple sorting algorithm that works by repeatedly splitting the vector into two halves and then merging the two sorted halves. This algorithm is a stable sorting algorithm. It is average case and O(n log n) time complexity. It was invented by John von Neumann.

Algorithm:

1. Split the vector into two halves

2. Sort the two halves

3. Merge the two sorted halves

Split function - This function splits the vector into two halves

```
split <- function(vec){
    # Split the vector into two halves
    mid <- length(vec)/2
    left <- vec[1:mid]
    right <- vec[(mid+1):length(vec)]
    return(list(left, right))
}
```

Define the merge function - This function merges the two sorted halves
@param left_vec Left vector
@param right_vec Right vector
@return Merged vector

```
merge <- function(left_vec, right_vec)
  {
    # Initialize the merged vector
    merged_vec <- c()
    # Initialize the index of the left vector
    left_index <- 1
    # Initialize the index of the right vector
    right_index <- 1
    # While both the left and right vectors have elements
    while(left_index <= length(left_vec) && right_index <= length(right_vec))
      {
        # If the element at the left index is less than the element at the right index
        if(left_vec[left_index] < right_vec[right_index])
          {
            # Add the element at the left index to the merged vector
            merged_vec <- c(merged_vec, left_vec[left_index])
            # Increment the left index
            left_index <- left_index + 1
          }
        # Else if the element at the left index is greater than the element at the right index
        else
          {
            # Add the element at the right index to the merged vector
            merged_vec <- c(merged_vec, right_vec[right_index])
            # Increment the right index
            right_index <- right_index + 1
          }
      }
    # If the left vector has elements
    if(left_index <= length(left_vec))
      {
        # Add the remaining elements of the left vector to the merged vector
        merged_vec <- c(merged_vec, left_vec[left_index:length(left_vec)])
      }
    # Else if the right vector has elements
    else if(right_index <= length(right_vec))
      {
        # Add the remaining elements of the right vector to the merged vector
        merged_vec <- c(merged_vec, right_vec[right_index:length(right_vec)])
      }
    merged_vec
  }
```

merge.sort function - This function implements the merge sort algorithm.

@param vec Vector to be sorted
@return Sorted vector

```
merge.sort <- function(vec){
    # If the vector has more than one element
    if(length(vec) > 1)
      {
        # Split the vector into two halves
        splitted_list <- split(vec)
        left_vec <- splitted_list[[1]]
        right_vec <- splitted_list[[2]]
        # Sort the two halves
        left_vec <- merge.sort(left_vec)
        right_vec <- merge.sort(right_vec)
        # Merge the two sorted halves
        merged_vec <- merge(left_vec, right_vec)
        return(merged_vec)
      }
    else {return(vec)}
}
```

Example

Take A vector

```
vector = c(7,6,5,1,0,9,5,5)
merge.sort(vector)
```

```
## [1] 0 1 5 5 5 6 7 9
```