**Pseudocode:**

This pseudocode defines two procedures: "merge\_sort" and "merge". The "merge\_sort" procedure sorts an array recursively by dividing it into two halves, sorting each half, and then merging the sorted halves back together. The "merge" procedure merges two sorted arrays into a single sorted array.

PROCEDURE merge\_sort(arr: List[T])

IF arr.size() > 1

mid = arr.size() / 2

left = arr[0..mid]

right = arr[mid..arr.size()]

merge\_sort(left)

merge\_sort(right)

merge(arr, left, right)

PROCEDURE merge(arr: List[T], left: List[T], right: List[T])

i = 0

j = 0

k = 0

WHILE i < left.size() AND j < right.size()

IF left[i] <= right[j]

arr[k] = left[i]

i = i + 1

ELSE

arr[k] = right[j]

j = j + 1

k = k + 1

WHILE i < left.size()

arr[k] = left[i]

i = i + 1

k = k + 1

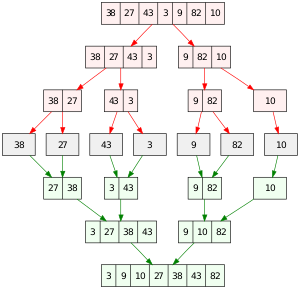
WHILE j < right.size()

arr[k] = right[j]

j = j + 1

k = k + 1

**Merge Sort:**



**Code:**

// Merge Sort

#include <iostream>

using namespace std;

void merge(int array[], int const left, int const mid,

int const right)

{

auto const subArrayOne = mid - left + 1;

auto const subArrayTwo = right - mid;

auto \*leftArray = new int[subArrayOne],

\*rightArray = new int[subArrayTwo];

for (auto i = 0; i < subArrayOne; i++)

leftArray[i] = array[left + i];

for (auto j = 0; j < subArrayTwo; j++)

rightArray[j] = array[mid + 1 + j];

auto indexOfSubArrayOne

= 0,

indexOfSubArrayTwo

= 0;

int indexOfMergedArray

= left;

while (indexOfSubArrayOne < subArrayOne

&& indexOfSubArrayTwo < subArrayTwo) {

if (leftArray[indexOfSubArrayOne]

<= rightArray[indexOfSubArrayTwo]) {

array[indexOfMergedArray]

= leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

}

else {

array[indexOfMergedArray]

= rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

}

indexOfMergedArray++;

}

while (indexOfSubArrayOne < subArrayOne) {

array[indexOfMergedArray]

= leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

indexOfMergedArray++;

}

while (indexOfSubArrayTwo < subArrayTwo) {

array[indexOfMergedArray]

= rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

indexOfMergedArray++;

}

delete[] leftArray;

delete[] rightArray;

}

void mergeSort(int array[], int const begin, int const end)

{

if (begin >= end)

return;

auto mid = begin + (end - begin) / 2;

mergeSort(array, begin, mid);

mergeSort(array, mid + 1, end);

merge(array, begin, mid, end);

}

void printArray(int A[], int size)

{

for (auto i = 0; i < size; i++)

cout << A[i] << " ";

}

void Top5(int A[], int size)

{

int count=1;

for (auto i = size - 1; count<=5 ; i--)

{

++count;

cout << A[i] << " ";

}

}

int main()

{

int arr[] = { 10000, 11000, 13000, 5000, 6000, 7000 }; //employee salary

auto arr\_size = sizeof(arr) / sizeof(arr[0]);

cout << "Given array is \n";

printArray(arr, arr\_size);

mergeSort(arr, 0, arr\_size - 1);

cout << "\nSorted array is \n";

Top5(arr, arr\_size);

return 0;

}

**Output Screenshot:**

