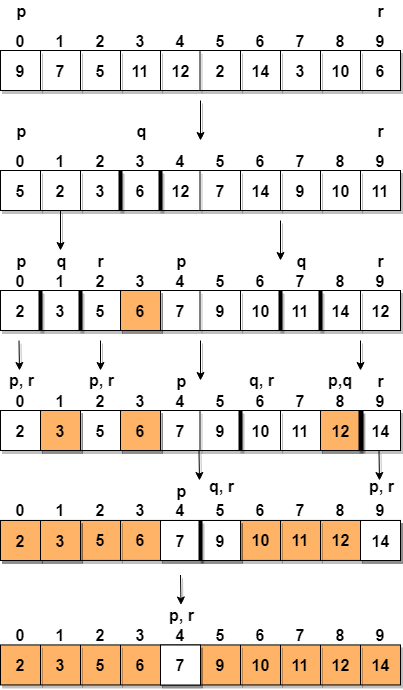
**A pictorial representation of how quick sort will sort the given array**

Let's consider an array with values **{9, 7, 5, 11, 12, 2, 14, 3, 10, 6}**



## Implementing Quick Sort Algorithm

// simple C program for Quick Sort

# include <stdio.h>

// to swap two numbers

void swap(int\* a, int\* b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

/\*

a[] is the array, p is starting index, that is 0,

and r is the last index of array.

\*/

void quicksort(int a[], int p, int r)

{

if(p < r)

{

int q;

q = partition(a, p, r);

quicksort(a, p, q);

quicksort(a, q+1, r);

}

}

int partition (int a[], int low, int high)

{

int pivot = arr[high]; // selecting last element as pivot

int i = (low - 1); // index of smaller element

for (int j = low; j <= high- 1; j++)

{

// If current element is smaller than or equal to pivot

if (arr[j] <= pivot)

{

i++; // increment index of smaller element

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

// function to print the array

void printArray(int a[], int size)

{

int i;

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

{

printf("%d ", a[i]);

}

printf("\n");

}

int main()

{

int arr[] = {9, 7, 5, 11, 12, 2, 14, 3, 10, 6};

int n = sizeof(arr)/sizeof(arr[0]);

// call quickSort function

quickSort(arr, 0, n-1);

printf("Sorted array: n");

printArray(arr, n);

return 0;

}

A pictorial representation of how merge sort will sort the given array.

