Convert array into Zig-Zag fashion

Given an array of distinct elements, rearrange the elements of array in zig-zag fashion in O(n) time. The converted array should be in form a < b > c < d > e < f.

```
Example:
Input:  arr[] = {4, 3, 7, 8, 6, 2, 1}
Output: arr[] = {3, 7, 4, 8, 2, 6, 1}

Input:  arr[] =  {1, 4, 3, 2}
Output: arr[] =  {1, 4, 2, 3}
```

We strongly recommend you to minimize your browser and try this yourself first.

A **Simple Solution** is to first sort the array. After sorting, exclude the first element, swap the remaining elements in pairs. (i.e. keep arr[0] as it is, swap arr[1] and arr[2], swap arr[3] and arr[4], and so on). Time complexity is O(nlogn) since we need to sort the array first.

We can convert in O(n) time using an **Efficient Approach**. The idea is to use modified one pass of bubble sort. Maintain a flag for representing which order(i.e. < or >) currently we need. If the current two elements are not in that order then swap those elements otherwise not.
Let us see the main logic using three consecutive elements A, B, C. Suppose we are processing B and C currently and the current relation is '<'. But we have B > C. Since current relation is '<' previous relation must be '>' i.e., A must be greater than B. So, the relation is A > B and B > C. We can deduce A > C. So if we swap B and C then the relation is A > C and C < B. Finally we get the desired order **A C B**

Refer this for more explanation.

Below is C++ implementation of above algorithm

```
// C++ program to sort an array in Zig-Zag form
#include <iostream>
using namespace std;

// Program for zig-zag conversion of array
void zigZag(int arr[], int n)
{
    // Flag true indicates relation "<" is expected,
    // else ">" is expected.  The first expected relation
    // is "<"
    bool flag = true;

    for (int i=0; i<=n-2; i++)
    {
        if (flag)  /* "<" relation expected */
        {
            /* If we have a situation like A > B > C,
               we get A > B < C by swapping B and C */
            if (arr[i] > arr[i+1])
                swap(arr[i], arr[i+1]);
        }
        else /* ">" relation expected */
        {
            /* If we have a situation like A < B < C,
               we get A < C > B by swapping B and C */
            if (arr[i] < arr[i+1])
                swap(arr[i], arr[i+1]);
        }
        flag = !flag; /* flip flag */
    }
}

// Driver program
int main()
{
    int  arr[] = {4, 3, 7, 8, 6, 2, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    zigZag(arr, n);
    for (int i=0; i<n; i++)
        cout << arr[i] << "  ";
    return 0;
}
```

Output:

```
3  7  4  8  2  6  1
```

Time complexity: O(n)
Auxiliary Space: O(1)