



Sorting arrays into ascending or descending order, and printing the odd and even elements separately.

Presented by:

- IEC2021063 Javesh Lodha
- IEC2021064 Yash Gupta
- IEC2021065 Shashi Kumar Chaubey
- IEC2021066 Pavitra Pandey

Under the supervision of:
Dr. Mohammad Javed

23rd January 2022



Abstract

In this paper we have devised an algorithm to sort an array and print the odd and even elements separately.



Algorithm Design

- Suppose we iterate over the array (length n) and apply swap operation over two adjacent elements if the element with higher index is smaller than the element with lower index.
- We will perform such swap $n-1$ times.
- This will bubble up at least one element to its designated position.
- So we will do these set of operations, n times to ensure every element is at its designated position.
- But we don't need to perform $n-1$ swaps every loop as with every loop at least one element bubbles to its correct position

- So, if we already have ran such sets of swaps i times, i elements at least would be at their designated position.
- So we will only run this loop $n-i-1$ times after, i th set of swaps.
- So we will iterate over $\sum_{i=1}^n (n-1-i) = \frac{n^2}{2} - 3\frac{n}{2}$ times.
- Also if the array contains already sorted elements, we will check if any swaps are taking place, if not then we will terminate the loop.
- We will print odd and even elements by traversing the sorted loop and checking if the element is odd or even.
- Also we will print the arrays in descending fashion by reverting the loop of indices from $n-1$ to 0 , where n is the size of the array.



This is the code for the problem.

Made in c

```
1  #include <stdio.h>
2  int main(){
3      printf("Enter the number of elements to sort\n");
4      int n;
5      scanf("%d", &n);
6      int a[n];
7      printf("\nEnter the elements to sort\n");
8      for(int i=0; i<n; i++){
9          scanf("%d", &a[i]);
10     }
11     int count = 0, c;
12     for(int i=1; i<n; i++){
13         for(int j=0; j<n-i; j++){
14             if(a[j]>a[j+1]){
15                 c = a[j];
16                 a[j] = a[j+1];
17                 a[j+1] = c;
18                 count++;
19             }
20         }
21         if(count == 0) break;
22         count = 0;
23     }
24     printf("\nSorted array is: ");
25     for(int i=0; i<n; i++){
26         printf("%d ", a[i]);
27     }
28     printf("\nSorted array with even numbers is: ");
29     for(int i=0; i<n; i++){
30         if(a[i]%2 == 0){
31             printf("%d ", a[i]);
32         }
33     }
34     printf("\nSorted array with odd numbers is: ");
35     for(int i=0; i<n; i++){
36         if(a[i]%2 == 1){
37             printf("%d ", a[i]);
38         }
39     }
40     printf("\nSorted array in decending order is: ");
41     for(int i=n-1; i>=0; i--){
42         printf("%d ", a[i]);
43     }
44     return 0;
45 }
```



Input

```
Enter the number of elements to sort  
4
```

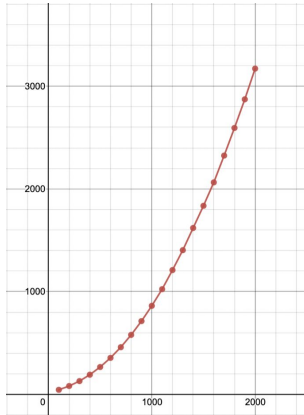
```
Enter the elements to sort  
3 2 4 1
```

Output

```
Sorted array is: 1 2 3 4  
Sorted array with even numbers is: 2 4  
Sorted array with odd numbers is: 1 3  
Sorted array in decending order is: 4 3 2 1
```

Time complexity of the Algorithm

As we know we iterate, $\sum_{i=1}^n (n-1-i) = \frac{n^2}{2} - 3\frac{n}{2}$ times. Would expect the runtime graph of the loop to look something like x^2 graph.



This is the graph of $t \times 10^{-7}$ vs n .

This looks pretty similar to the graph of x^2 . As the highest degree element in the polynomial equation of n vs t is n^2



Conclusion

Bubble sort is an effective way to sort an array, it has time dependency of n^2 .

References

- <https://www.geeksforgeeks.org>
- <https://stackoverflow.com>

Thank You

The end