1.

If , then

is true if for some integer k > 1.

When , we have

, then

2.

For , the sub-array is an empty array, step 1 doesn't require any

time, and step 2 requires constant time. The algorithm runs in time.

For , step 1 need the recursion for again and step 2 runs

in .

So we have a conclusion:

3.

i = 0, j = n - 1

while(1)

{ while (a[i] < 0 && i < j)

i++;

while (a[j] > 0 && i < j)

j--;

if (i < j)

{ swap(a[i],a[j]);

i++;

j--;}

else

break;

}

4.

We have an array of size n, then there exist two scenarios:

- Need an element is present from index 0 and (n-1)

- Need an element is not present from index 0 and (n-1)

So the number of cases = n + 1

We see that:

- The element at middle is returned in the first comparison.

- 2 elements are returned in the second comparison.

Outcomes are based on the result of the first comparison. And so on...

In short, elements that need ‘i’ comparisons =