Find maximum and minimum of an array using minimum number of comparisions

```
#include <stdio.h>
#include <stdlib.h>

struct pair
{
	int min;
	int max;
};

struct pair getMinMax(int *arr, int size)
{
	struct pair minmax;
	int index;

	if(size % 2 == 0)
	{
		if(arr[0] > arr[1])
		{
			minmax.max = arr[0];
			minmax.min = arr[1];
		}
		else
		{
			minmax.min = arr[0];
			minmax.max = arr[1];
		}
		index = 2;
	}
	else
	{
		minmax.min = arr[0];
		minmax.max = arr[0];
		index = 1;
	}

	while( index < size-1 )
	{
		if (arr[index] > arr[index + 1])
		{
			if(arr[index] > minmax.max)
				minmax.max = arr[index];
			if(arr[index+1] < minmax.min)
				minmax.min = arr[index + 1];
		}
		else
		{
			if (arr[index + 1] > minmax.max)
				minmax.max = arr[index + 1];
			if (arr[index] < minmax.min)
				 minmax.min = arr[index];
```

```
                        }
                        index += 2;
            }
  return minmax;
}

int main()
{
            int *arr, size, X;
            printf("Enter size of an array\n");
            scanf("%d", &size);
            //allocate memory for array
            arr = (int *)malloc(size * sizeof(int));

            printf("Enter Array elements ");
            for(int index = 0; index < size; index++)
                        scanf("%d", &arr[index]);

            struct pair minmax = getMinMax(arr, size);

            printf("Maximum element is %d and Minimum element is %d",
                        minmax.max, minmax.min);
            return 0;
}
```

Time Complexity: O(n)
Space Complexity: O(1)

Total Number of comparisons:

If n is odd: 3*(n – 1) / 2;

If n is even: 1 Initial comparison for initializing min and max
and 3*(n – 1) / 2 comparisons for rest of the elements
= 1 + 3 * (n – 1) / 2 = 3n/2 – 2