Dev Jatinbhai Patel

CE091

20CEUOS018

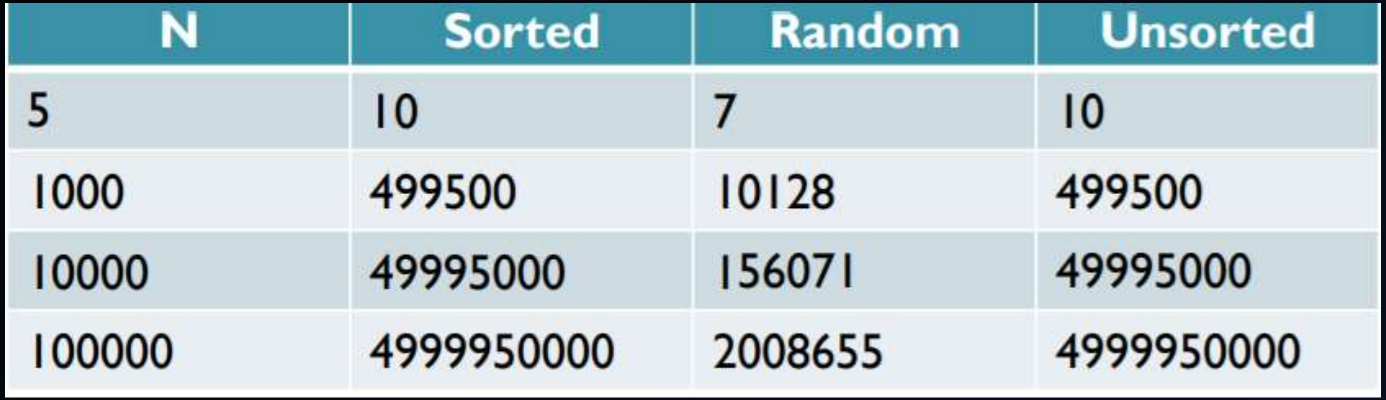
LAB2

Quicksort algorithm:

```cpp
#include <iostream>

using namespace std;
int comp=0;
int swaps=1;
int partition(int arr[],int p,int r)
{
    int x=arr[r];
    int i=p-1;
    for(int j=p;j<=r-1;j++)
    {
        comp++;
        if(arr[j] < x)
        {
            i++;
            swap(arr[i],arr[j]);
            swaps++;
        }

    }
    swap(arr[i+1],arr[r]);
    return (i+1);
}

void Quicksort(int arr[],int p,int r)
{
    if(p<r)
    {
        int q=partition(arr,p,r);
        Quicksort(arr,p,q-1);
        Quicksort(arr,q+1,r);
    }
}
```

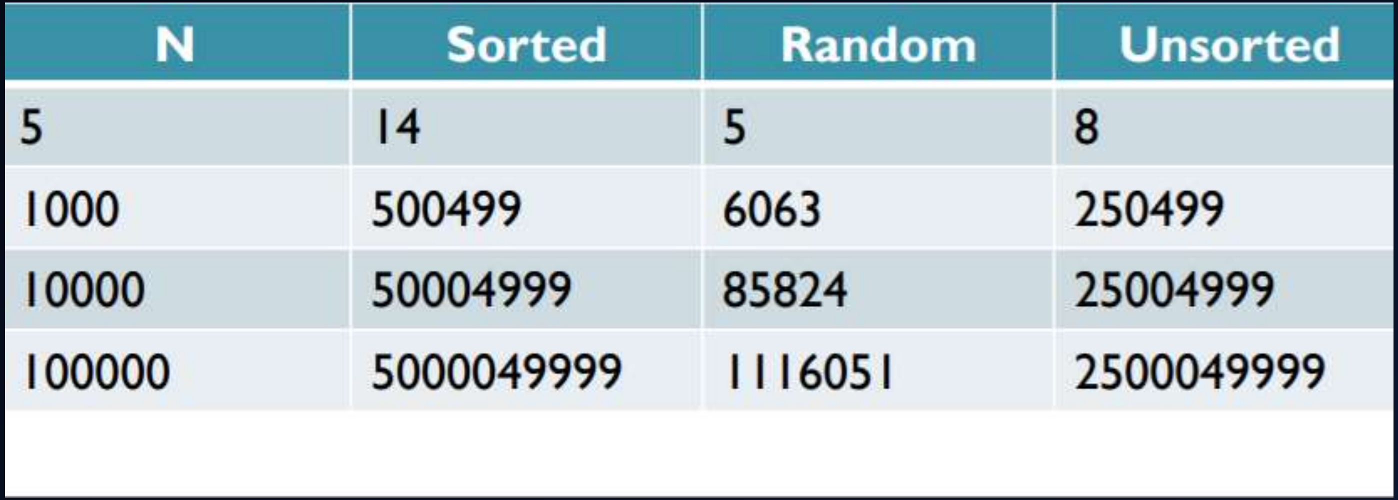
```cpp
int main()
{
    int n=6;
    int arr[n];
    for(int i=0;i<n;i++)
    {
        cin>>arr[i];
    }
    Quicksort(arr,0,n-1);
    for(int i=0;i<n;i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl<<swaps<<" "<<comp;
    return 0;
}
```

Comparisons



N	Sorted	Random	Unsorted
5	10	7	10
1000	499500	10128	499500
10000	49995000	156071	49995000
100000	4999950000	2008655	4999950000

Swaps

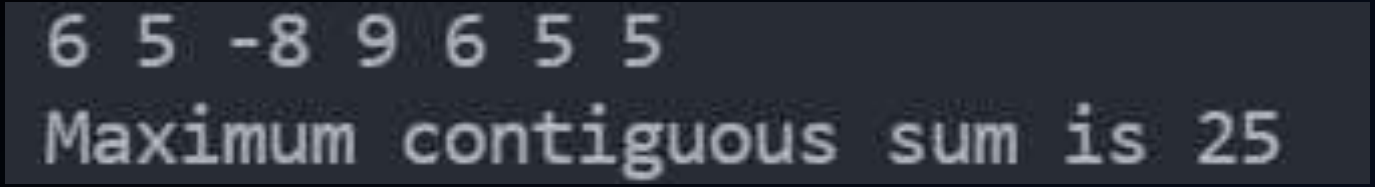


N	Sorted	Random	Unsorted
5	14	5	8
1000	500499	6063	250499
10000	50004999	85824	25004999
100000	5000049999	1116051	2500049999

Maxsubarray algo:

```
#include <limits.h>
#include <stdio.h>
#include<iostream>
using namespace std;

int max(int a, int b) { return (a > b) ? a : b; }

int max(int a, int b, int c) { return max(max(a, b), c); }


int maxCrossingSum(int arr[], int l, int m, int h)
{
    // Include elements on left of mid.
    int sum = 0;
    int left_sum = INT_MIN;
    for (int i = m; i >= l; i--) {
        sum = sum + arr[i];
        if (sum > left_sum)
            left_sum = sum;
    }


    sum = 0;
    int right_sum = INT_MIN;
    for (int i = m + 1; i <= h; i++) {
        sum = sum + arr[i];
        if (sum > right_sum)
            right_sum = sum;
    }


    return max(left_sum + right_sum, left_sum, right_sum);
}

// Returns sum of maximum sum subarray in aa[l..h]
int maxSubArraySum(int arr[], int l, int h)
{
    if (l == h)
        return arr[l];

    // Find middle point
    int m = (l + h) / 2;

    return max(maxSubArraySum(arr, l, m),
            maxSubArraySum(arr, m + 1, h),
            maxCrossingSum(arr, l, m, h));
}
```

```
int main()
{
    int no_of_input;
    cin>>no_of_input;
    int arr[no_of_input];
    for (int i=0;i<no_of_input;i++)
    {
        cin>>arr[i];
    }
    int n = sizeof(arr) / sizeof(arr[0]);
    int max_sum = maxSubArraySum(arr, 0, n - 1);
    printf("Maximum contiguous sum is %d\n",
max_sum);
    getchar();
    return 0;
}
```



```
6 5 -8 9 6 5 5
Maximum contiguous sum is 25
```

Recurrence Relation:- 2T(n/2) + cn

Time Complexity:- O(nlogn)