DAA Assignment 1

Nitin Raj Singh IIT2016132
Chirag Meel ISM2016006.
Lekhika Dugtal ITM2016008
Saurabh Kumar IWM2016502

1 Problem

Partition an unsorted list of integers into adjacent sorted subsets. Let n be the size of the original unsorted list and N be the number of partitions generated from it. Plot a graph to show the probability values Prob(N=2,3,... etc.) for different values of n.

2 Introduction and Literature Survey

In this problem we have to count the minimum number of adjacent sorted partitions (ascending or descending) in a given set of numbers. Then we are supposed to find the probability of forming N sorted partitions for different number of elements n. At last we have to plot the graph of getting N number of sorted partitions against the probability of getting those many sorted partitions.

This problem uses the fact that two numbers are always sorted either in ascending or in descending order. The approach we could take is that we can first check the first two numbers whether they are in ascending or descending order and we can continue checking further till we find the numbers to be in the ascending or descending order as found earlier. Whenever the condition becomes false, we increase the counter by 1 and start the new sequence from that number and go on in this way till the end of the set

3 Algorithm Design

3.1 Algorithm Approach

When confronted with a problem of counting the number of adjacent sorted subsets in the given set of numbers, we will simply consider given number of set as a array of particular size.

***rand()** function is used for allocating random numbers*

A sorted subset is a set of numbers which is an increasing or decreasing sequence.

To divide the list into adjacent sorted subsets we can start with first number check its relationship with the second number and accordingly continue traversing till the increasing or decreasing property is satisfied in the subset of numbers.

When the loop breaks start a new set and continue to do the same to generate all the sorted subsets.In case there is only one number left in the end consider it as a sorted subset.

Whenever the loop breaks it indicates the end of a sorted sequence.So we can maintain a counter to count number of sorted subsets and increment it whenever the loop breaks.

Eg : Elements are

1 2 5 7 3 2 9 10

Let this be the set. We know that two consecutive elements are always sorted either in ascending or descending order.
So taking the first two elements we come to know that they are in ascending order.

1 2 5 7 3 2 9 10

We keep moving the pointer forward till we find the numbers in ascending order or the index is less than or equal to n.

1 2 5 7 3 2 9 10

1 2 5 7 3 2 9 10

Next when the pointer reaches the element at 5th element, we see that the order is not followed.
So here we break the loop and the counter is increased by 1, i.e. we have formed one sorted set.
Again in the same way, we keep on forming the subsets and if at last only one element is left we take that element as a separate set.

1 2 5 7 **3** 2 9 10

1 2 5 7 **3 2** 9 10

Again the loop breaks and the counter is increased by 1.

1 2 5 7 3 2 **9** 10

1 2 5 7 3 2 **9 10**

3.2 Algorithm code

```
for $i = 0$ to $n - 1$ do
    if $i$ = $n$-$1$ then
        $counter \leftarrow counter + 1$;
    end
    if numbers are in descending order then
        traversing till it is descending ;
        $counter \leftarrow counter + 1$;
    else
        traversing till it is ascending ;
        $counter \leftarrow counter + 1$;
    end
end
```

3.3 C Implmentation

```c
#include <stdio.h>
#include<math.h>
#include<time.h>
int main(void)
{
        for(int n=10;n<=100;n+=10)
        {
            printf("n value is %d\n",n);
                int b[510]={0},t;
            for(t=0;t<30;t++)
            {
                int a[1000]={0};
                int i=0,counter=0;
                for(int j=0;j<n;j++)
                {
                    a[j]=rand()%100;
                }
                for(int i=0;i<n;i++)
                {
                        if(i==n-1)
                        {
                               counter++;
                               i++;
                        }
                        else if(a[i]<=a[i+1])
                        {
                                counter++;
                                while(i+1<n&&a[i]<=a[i+1])
                                {
                                        i++;
                                }
                        }
                        else if(a[i]>a[i+1])
                        {
                                counter++;
                                while(i+1<n&&a[i]>=a[i+1])
                                {
                                        i++;
                                }
                        }
                }

                b[counter]++;
                for(int j=0;j<n;j++)
                {
                    printf("%d ",a[j]);
                }
                printf("\n");
                printf("Number of sorted subsets is %d\n",counter);
            }
            int sum=0;
            for(int k=2;k<=n/2;k++)
            {
                float pro=(b[k]*1.0)/(t*1.0);
                printf("The probability of %d sorted subsets is %f\n",k,pro);
            }
            printf("\n");

        }
    return 0;
}
```

4 Analysis

```
                                        TIME        FREQUENCY
//traversing the unsorted list
For i from 0 to n-1                      3             m+1
 If i=n-1                                1             m
   Increment subset count                1             n11
 Else if a[i]<=a[i+1])                   3             m
   While a[i]<=a[i+1] and i+1<n          4             n1+1
     Increment i                         1             n1
       Increment subset count            1             n12
 Else
   While a[i]>a[i+1] and i+1<n           4             n2+1
     Increment i;                        1             n2
       Increment subset count            1             n13
```

$$n11+n12+n13=k=n$$

$$m+n1+n2=n$$

For worst case:
When the number of sets formed is n/2 and after every two element the order of the numbers is changed. In this case the program would enter the for loop get into the else if or else part the while loop will execute for two times, then it will break the while loop increase the counter and again it will start with the for loop for the next index and this will continue for n/2 times. The time complexity in this case would be:-

$$T \ \alpha \ (3+1+3+4+1+1)*n/2+3$$

$$T \ \alpha \ 13*n/2+3$$

For the best case: The best case would occur when the whole entered number is in a sorted order. So once the for loop starts it will enter the else if or else condition and the corresponding while loop will run for (n-1) times and then it will break the while loop increase the counter and then exit the for loop. So the complexity will be :-

$$T \ \alpha \ 3+1+3+5*(n-1)+1+3$$

$$T \ \alpha \ 5*n-5+11$$

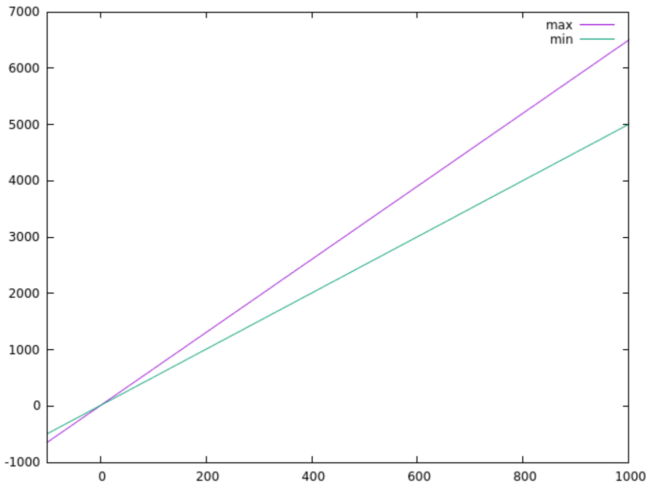
$$T \ \alpha \ 5*n+6$$

In this case we see that the best case complexity and the worst case complexity evaluates to be equal.

$$O(n) = \ \Omega(n)$$

Thus, here we represent the complexity as

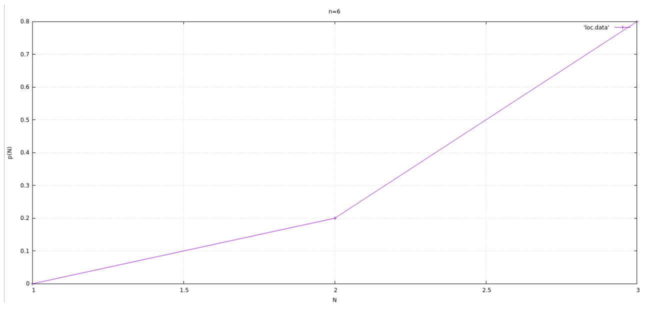
$$\theta(n)$$



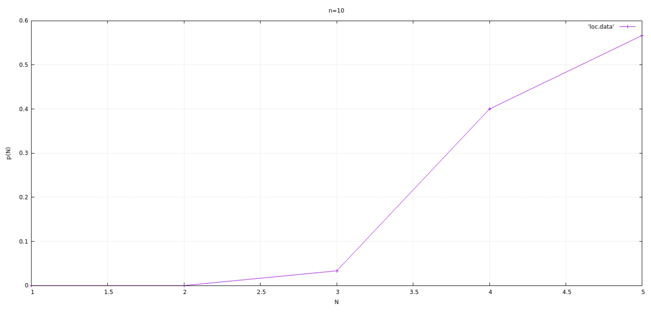
5 Experimental Study

These graphs are plotted with the help of gnu-plots.

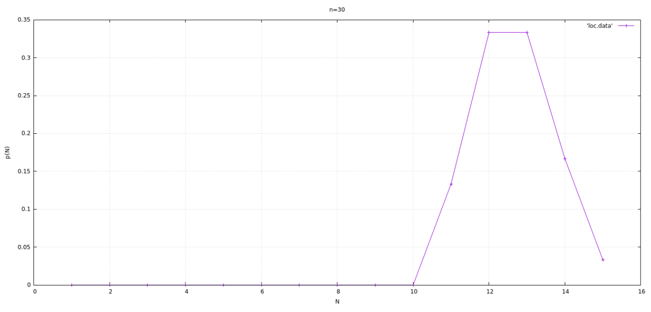
5.1 P(n) vs n plots



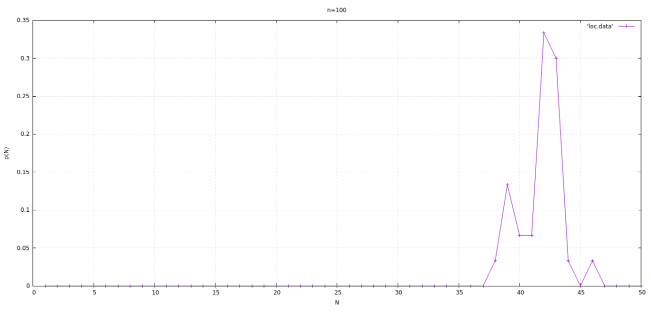
$Fig1 : when \ \ n=5$



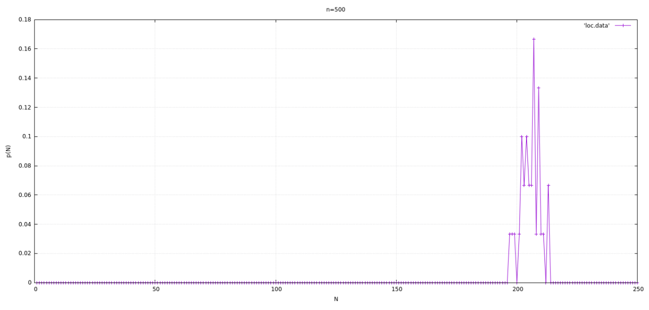
$Fig2 : when \ \ n=10$



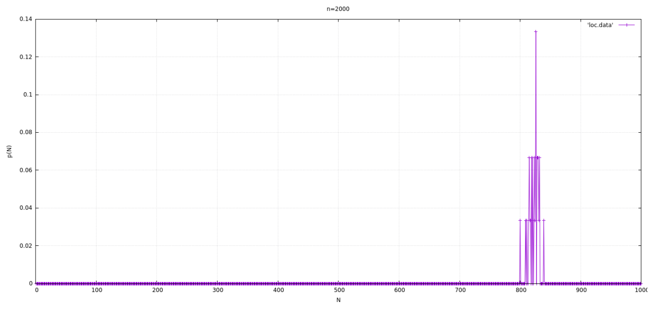
$Fig3 : when \ \ n=30$



$Fig4 : when \ \ n=100$

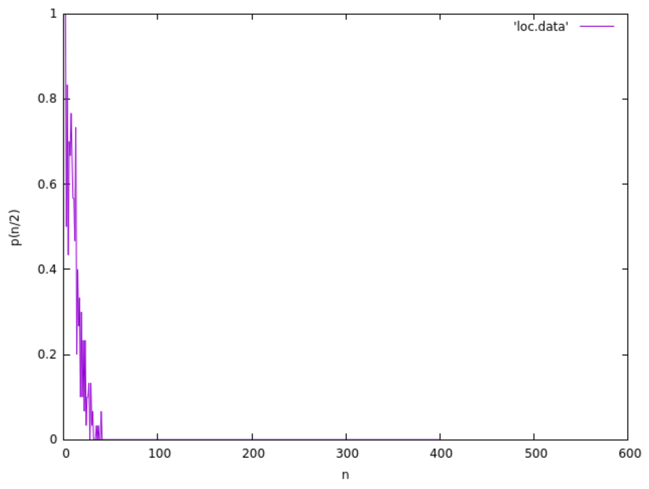


$$Fig5 : when \;\; n = 500$$



$$Fig6 : when \;\; n = 2000$$

5.2 P(n/2) vs n plot



6 Discussion and Conclusion

6.1 Trends in p(N) vs N graph

- The peak in the p(N) vs N graph for different values of n is shifting from the point n/2 on the x-axis towards left gradually as n value is increasing.This implies that the sorted subsets generated are mostly close to n/2 but are decreasing as n value is increasing.

- The concentration of significant probabilities start getting concentrated in a small region which is evident from the increase in impulsive behavior in the probability distribution graph for large values of n.

- The peak value reduces as n value increases indicating the decrease in highest probability value.

6.2 Trends in p(n/2) vs n graph

- It is evident from the p(n/2) vs n graph that the probability of generating n/2 sorted subsets from the unsorted list is decreasing drastically.The probability decreases from 1 at n=2 to nearly 0 at large values of n which can be seen in the table.

n	2	4	10	20	35	100
P(n/2)	1.000	0.833	0.566	0.100	0.033	0.0001

6.3 Conclusion

Most of the significant probabilities are close to n/2 and almost equal for large values of n and in case of small values of n the peak lies at n/2 and the probabilities are spread across.

7 References

- How to solve it by Computer By R.G. Dromney

- Introduction to Algorithm 3rd Edition By Thomas H.Cormen

- overleaf.com

- tex.stackexchange.com

- latex.org