Name: Vishal Shashikant Salvi.
UID: 2019230069
Batch: C
Class: SE Comps

Experiment No 4

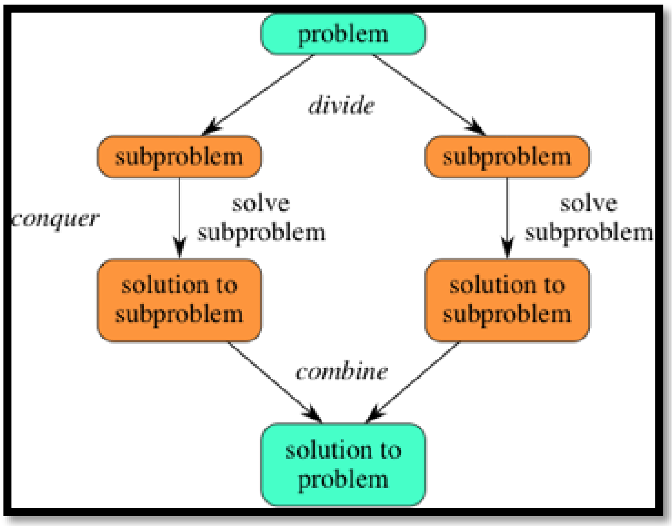
Aim: Min-Max Using Divide and Conquer

Theory:

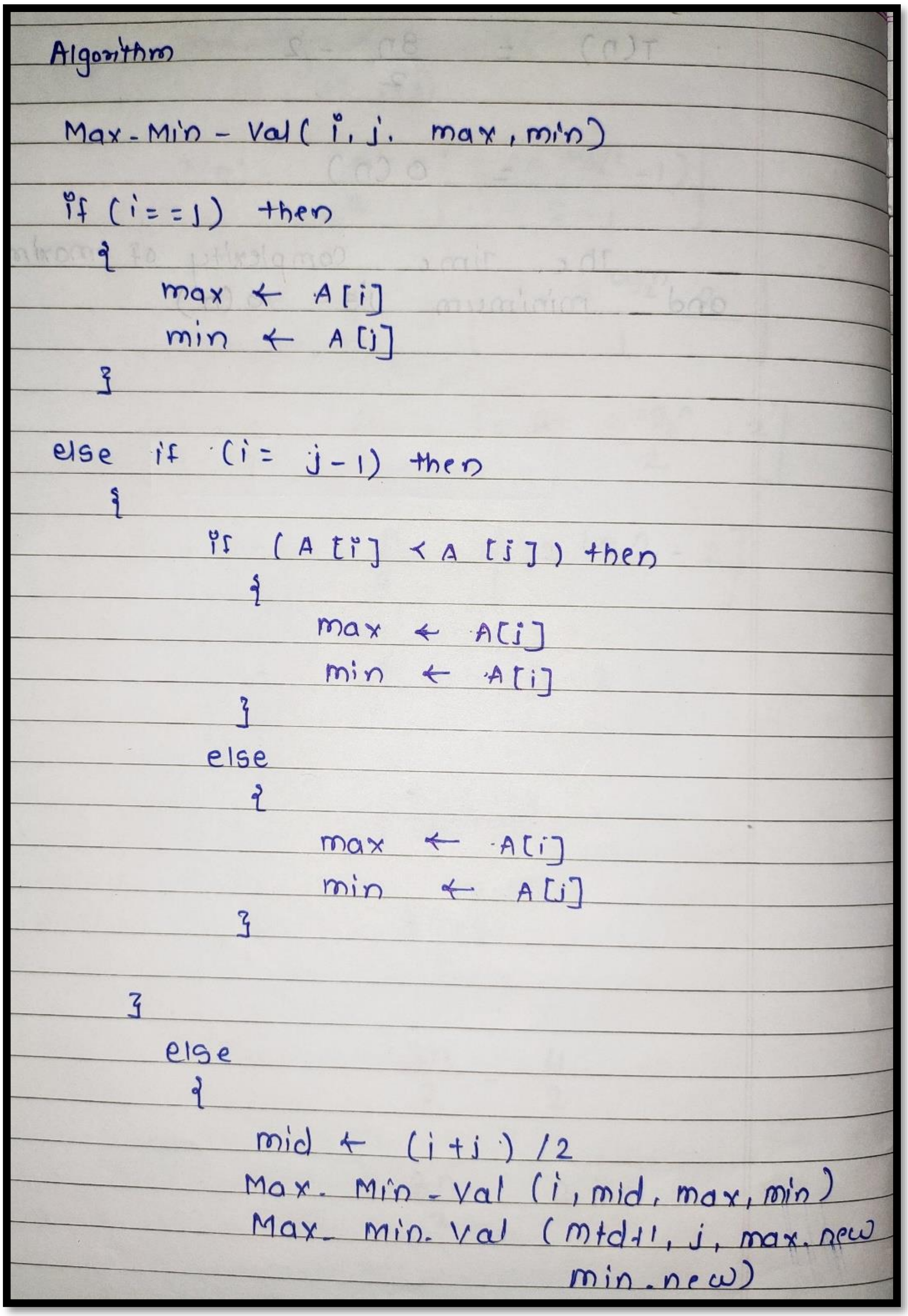
Discuss divide and Conquer strategy in general.

1. **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
2. **Conquer** the subproblems by solving them recursively. If they are small enough, solve the subproblems as base cases.
3. **Combine** the solutions to the subproblems into the solution for the original problem.

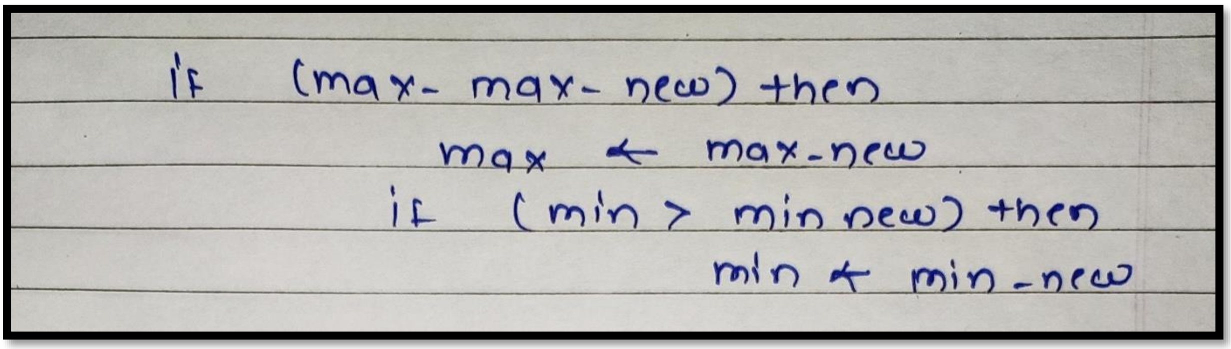
You can easily remember the steps of a divide-and-conquer algorithm as *divide, conquer, combine*. Here's how to view one step, assuming that each divide step creates two subproblems (though some divide-and-conquer algorithms create more than two):



Algorithm:

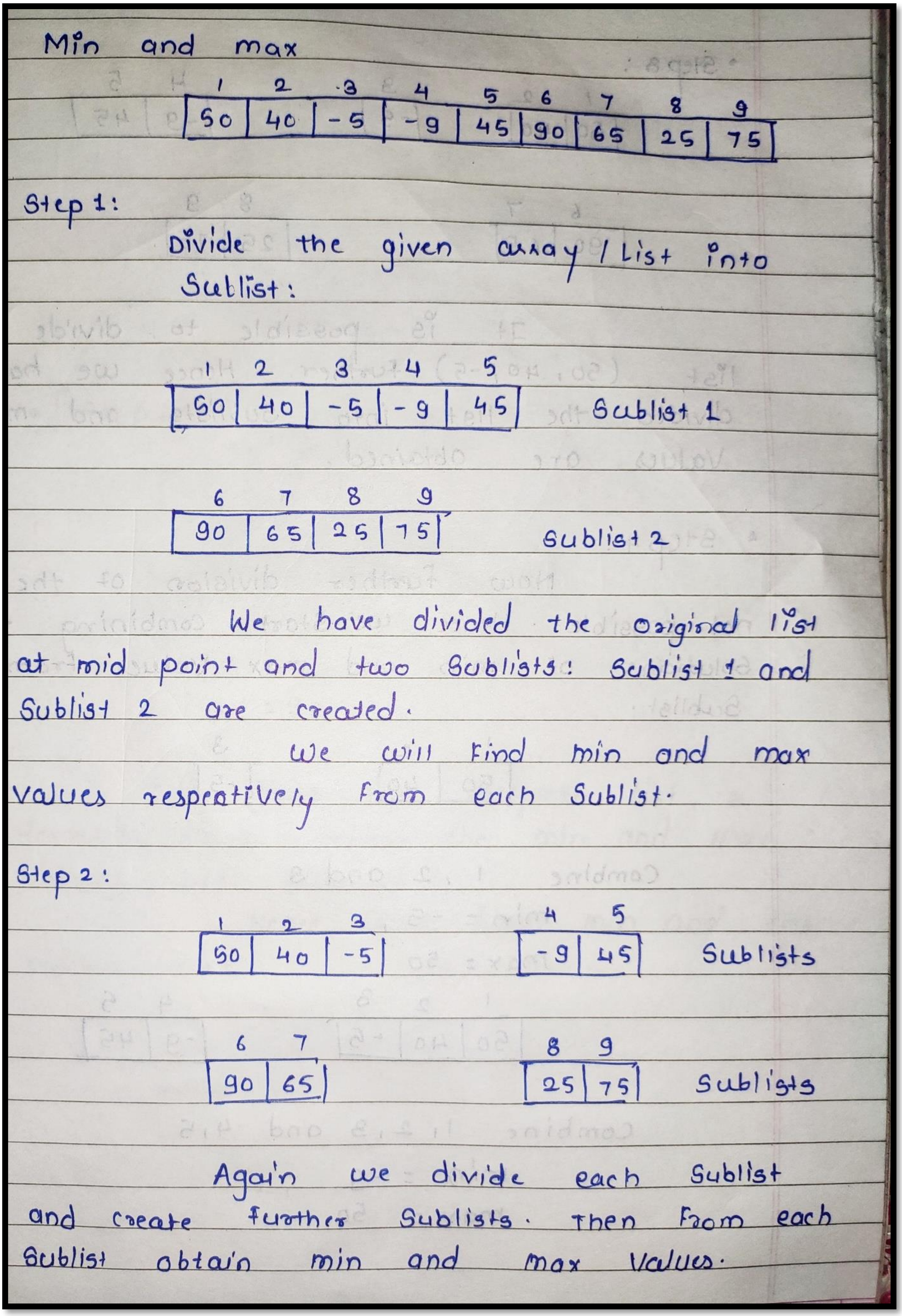


```
Algorithm

Max-Min - Val( i, j. max, min)

if (i==j) then
{
        max ← A[i]
        min ← A[j]
}

else if (i= j-1) then
{
        if (A [i] < A [j]) then
        {
                max ← A[j]
                min ← A[i]
        }
        else
        {
                max ← A[i]
                min ← A[j]
        }

}
    else
    {
        mid ← (i+j) /2
        Max- Min - Val (i, mid, max, min)
        Max- Min. Val (mid+1, j, max. new
                      min.new)
```



```
if (max- max- new) then
        max ← max-new
    if (min > min new) then
        min ← min-new
```

Explain the problem:



Min and max

1	2	3	4	5	6	7	8	9
50	40	-5	-9	45	90	65	25	75

Step 1:

Divide the given array / List into Sublist:

1	2	3	4	5
50	40	-5	-9	45

Sublist 1

6	7	8	9
90	65	25	75

Sublist 2

We have divided the original list at mid point and two sublists: sublist 1 and Sublist 2 are created.

We will find min and max values respectively from each sublist.

Step 2:

1	2	3
50	40	-5

4	5
-9	45

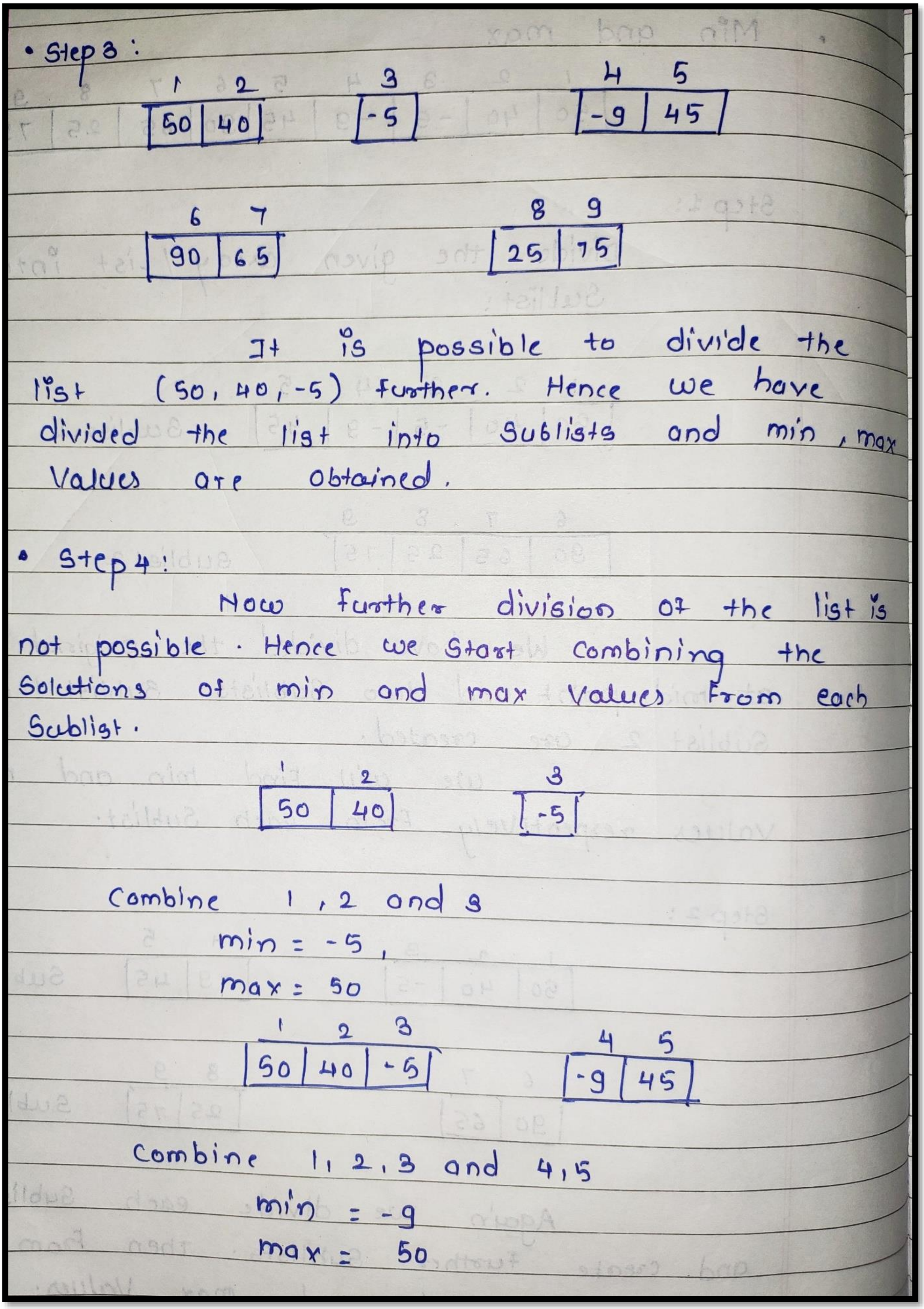
Sublists

6	7
90	65

8	9
25	75

Sublists

Again we divide each sublist and create further sublists. Then from each sublist obtain min and max values.



- Step 3 :

1	2
50	40

3
-5

4	5
-9	45

6	7
90	65

8	9
25	75

It is possible to divide the list (50, 40, -5) further. Hence we have divided the list into Sublists and min, max Values are obtained.

- Step 4 :

Now further division of the list is not possible. Hence we Start combining the Solutions of min and max values from each Sublist.

1	2
50	40

3
-5

Combine 1, 2 and 3

min = -5,

max = 50

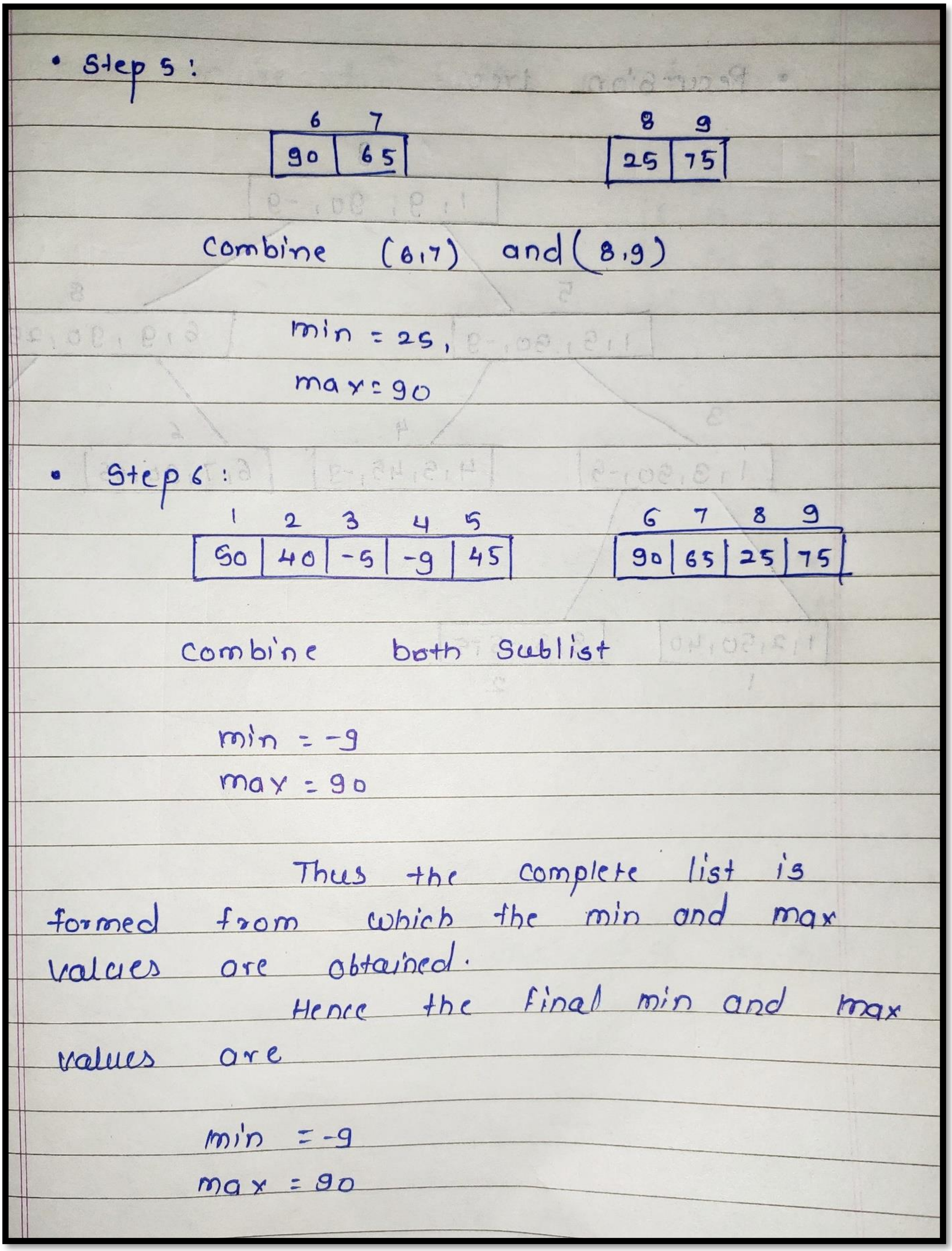
1	2	3
50	40	-5

4	5
-9	45

Combine 1, 2, 3 and 4, 5

min = -9

max = 50



- Step 5 :

6	7
90	65

8	9
25	75

Combine (6,7) and (8,9)

min = 25,

max = 90

- Step 6 :

1	2	3	4	5
50	40	-5	-9	45

6	7	8	9
90	65	25	75

Combine both Sublist

min = -9

max = 90

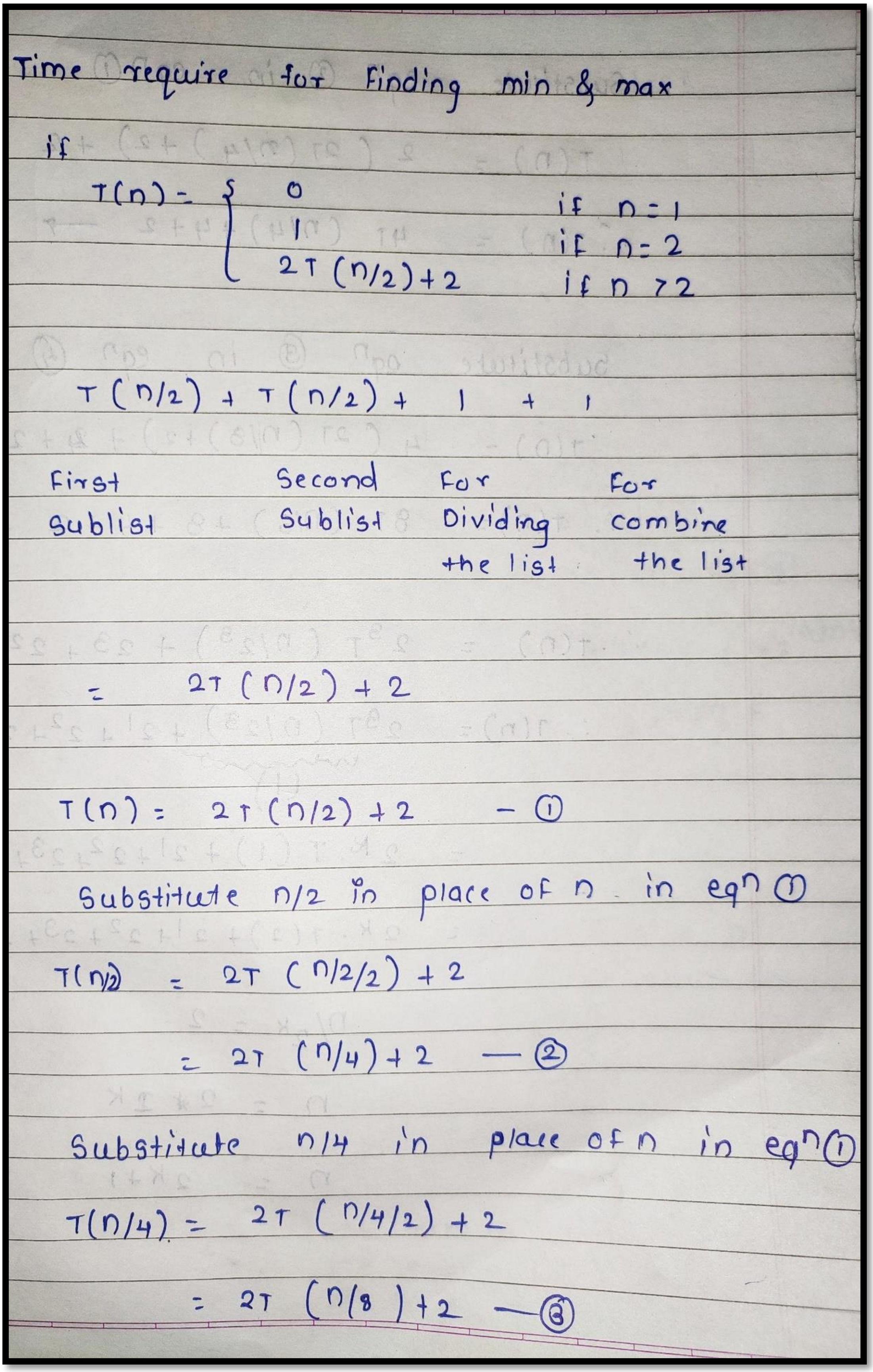
Thus the complete list is formed from which the min and max values are obtained.

Hence the final min and max values are

min = -9

max = 90

Discuss the time complexity of straight Minmax :



Time require for Finding min & max

if

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \\ 2T(n/2) + 2 & \text{if } n > 2 \end{cases}$$

$$T(n/2) + T(n/2) + 1 + 1$$

First sublist	Second sublist	For Dividing the list	For combine the list

$$= 2T(n/2) + 2$$

$$T(n) = 2T(n/2) + 2 \quad — ①$$

Substitute n/2 in place of n in eqn ①

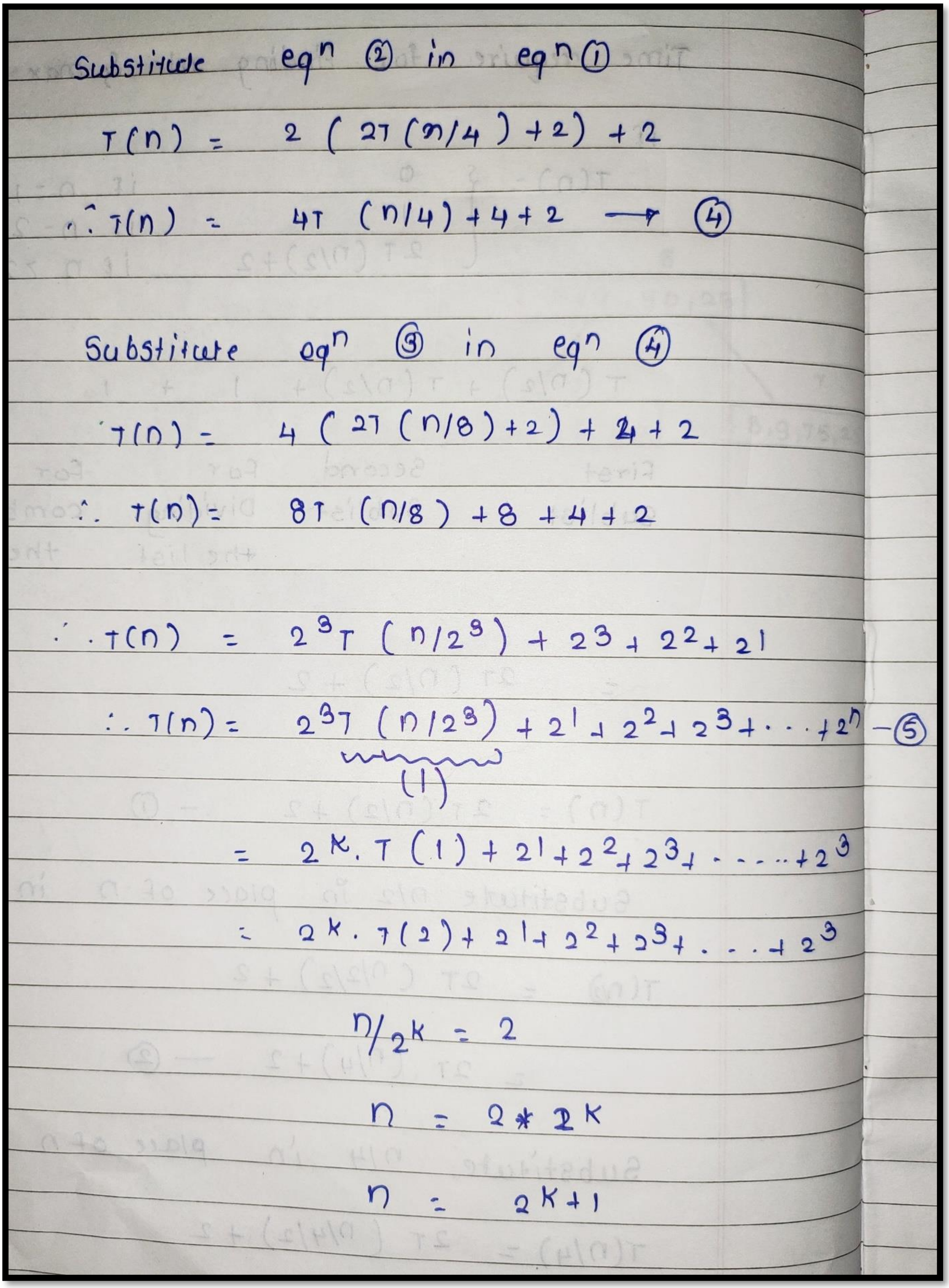
$$T(n/2) = 2T(n/2/2) + 2$$

$$= 2T(n/4) + 2 \quad — ②$$

Substitute n/4 in place of n in eqn ①

$$T(n/4) = 2T(n/4/2) + 2$$

$$= 2T(n/8) + 2 \quad — ③$$



Substitude eq^n ② in eq^n ①

$T(n) = 2\ (\ 2T\ (n/4\ ) + 2)\ + 2$

$\therefore T(n) = 4T\ (n/4) + 4 + 2 \longrightarrow$ ④

Substitute eq^n ③ in eq^n ④

$T(n) = 4\ (\ 2T\ (n/8) + 2)\ + 4 + 2$

$\therefore\ T(n) = 8T\ (n/8\ ) + 8\ + 4 + 2$

$\therefore T(n)\ =\ 2^3T\ (\ n/2^3)\ + 2^3 + 2^2 + 2^1$

$\therefore T(n) = \underbrace{2^3T\ (n/2^3)}_{(1)} + 2^1 + 2^2 + 2^3 + \dots + 2^n$ — ⑤

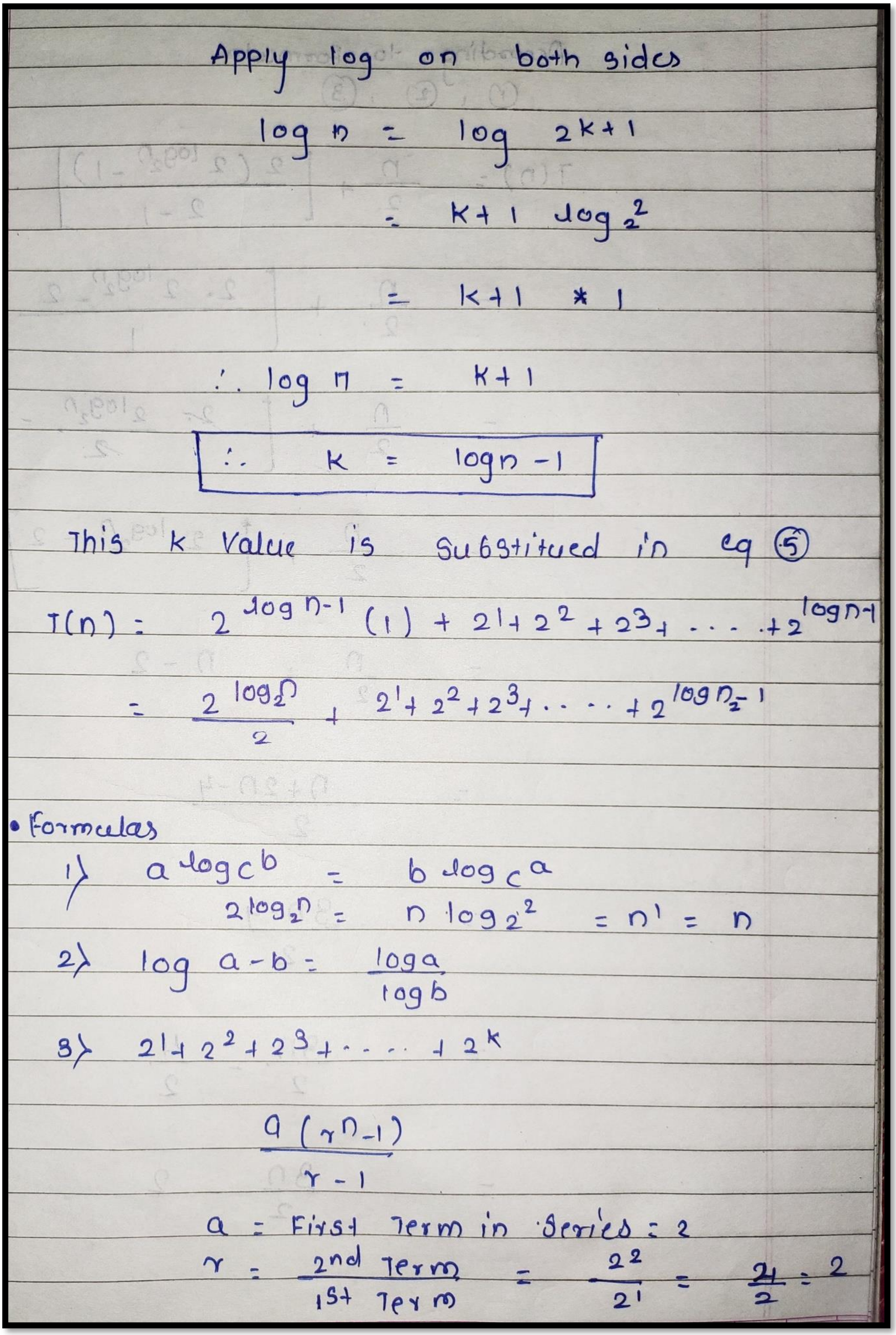
$= 2^k . T\ (1) + 2^1 + 2^2 + 2^3 + \dots + 2^3$

$= 2^k . T(2) + 2^1 + 2^2 + 2^3 + \dots + 2^3$

$n/2^k = 2$

$n = 2 * 2^k$

$n = 2^{k+1}$



Apply log on both sides

$$\log n = \log 2^{k+1}$$

$$= k+1 \log_2^2$$

$$= k+1 * 1$$

$$\therefore \log n = k+1$$

$$\boxed{\therefore \quad k = \log n - 1}$$

This k value is substitued in eq ⑤

$$T(n) = 2^{\log n-1}(1) + 2^1 + 2^2 + 2^3 + \dots + 2^{\log n-1}$$

$$= \frac{2^{\log_2 n}}{2} + 2^1 + 2^2 + 2^3 + \dots + 2^{\log n_2 - 1}$$

- Formulas

1) $a^{\log_c b} = b^{\log_c a}$

$2^{\log_2 n} = n^{\log_2 2} = n^1 = n$

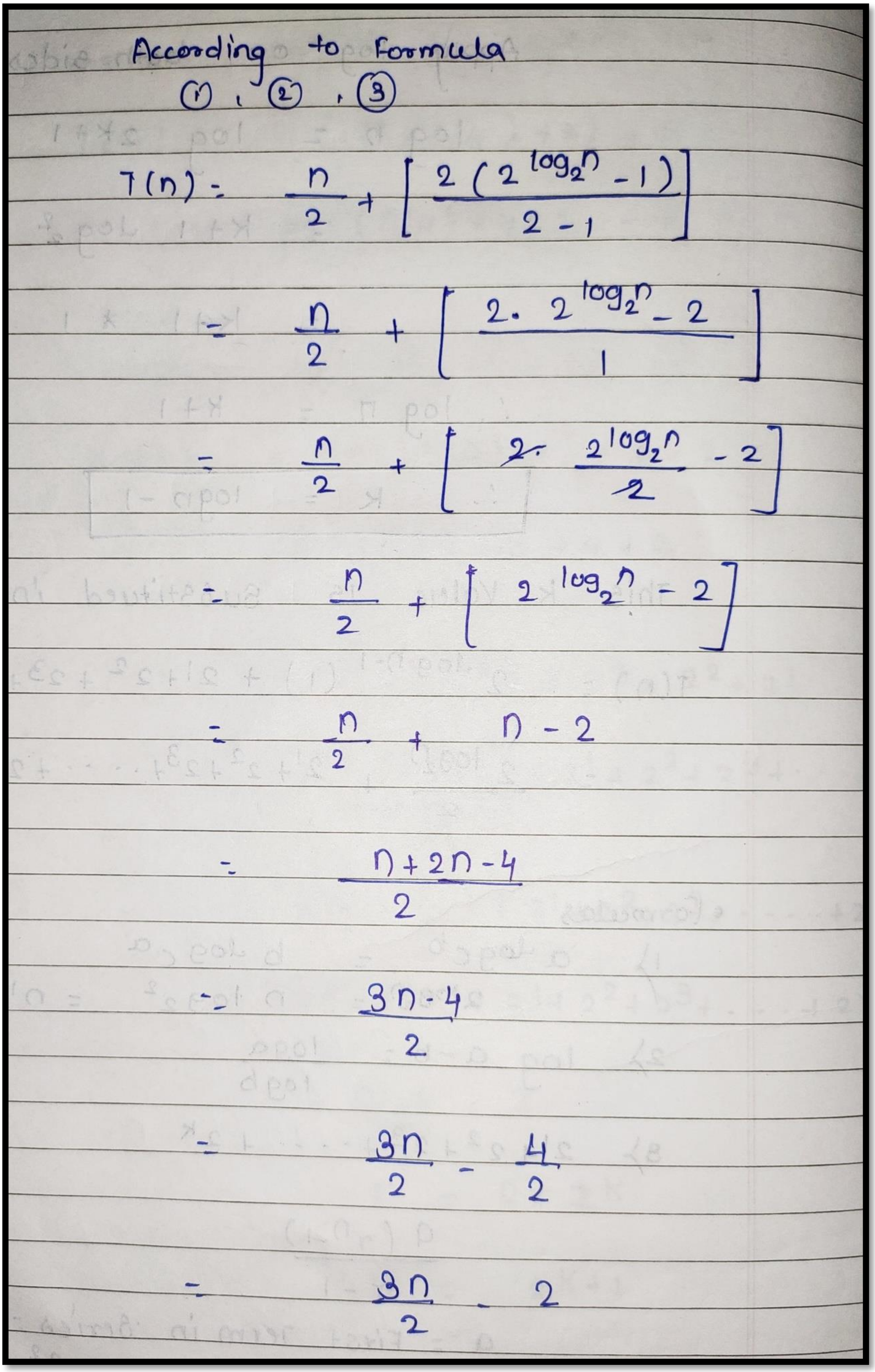
2) $\log a - b = \frac{\log a}{\log b}$

3) $2^1 + 2^2 + 2^3 + \dots + 2^k$

$$\frac{a(r^n - 1)}{r - 1}$$

a = First Term in Series = 2

$$r = \frac{2^{nd}\ \text{Term}}{1^{st}\ \text{Term}} = \frac{2^2}{2^1} = \frac{4}{2} = 2$$



According to formula ①, ②, ③

$$T(n) = \frac{n}{2} + \left[\frac{2(2^{\log_2 n} - 1)}{2-1}\right]$$

$$= \frac{n}{2} + \left[\frac{2 \cdot 2^{\log_2 n} - 2}{1}\right]$$

$$= \frac{n}{2} + \left[\cancel{2} \cdot \frac{2^{\log_2 n}}{\cancel{2}} - 2\right]$$

$$= \frac{n}{2} + \left[2^{\log_2 n} - 2\right]$$

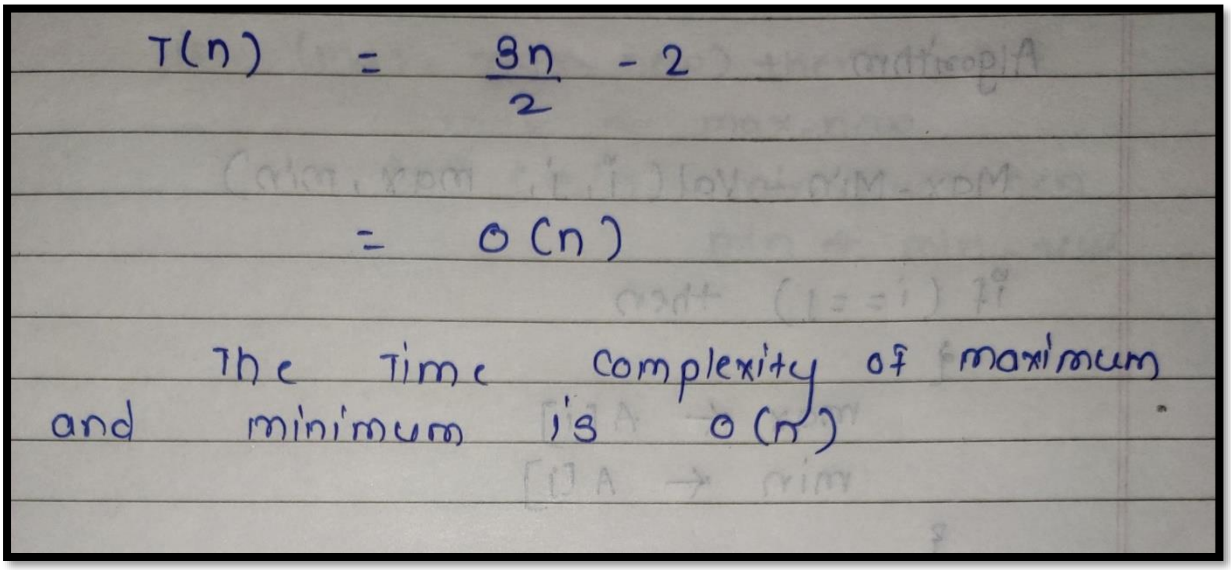
$$= \frac{n}{2} + n - 2$$

$$= \frac{n + 2n - 4}{2}$$

$$= \frac{3n - 4}{2}$$

$$= \frac{3n}{2} - \frac{4}{2}$$

$$= \frac{3n}{2} - 2$$



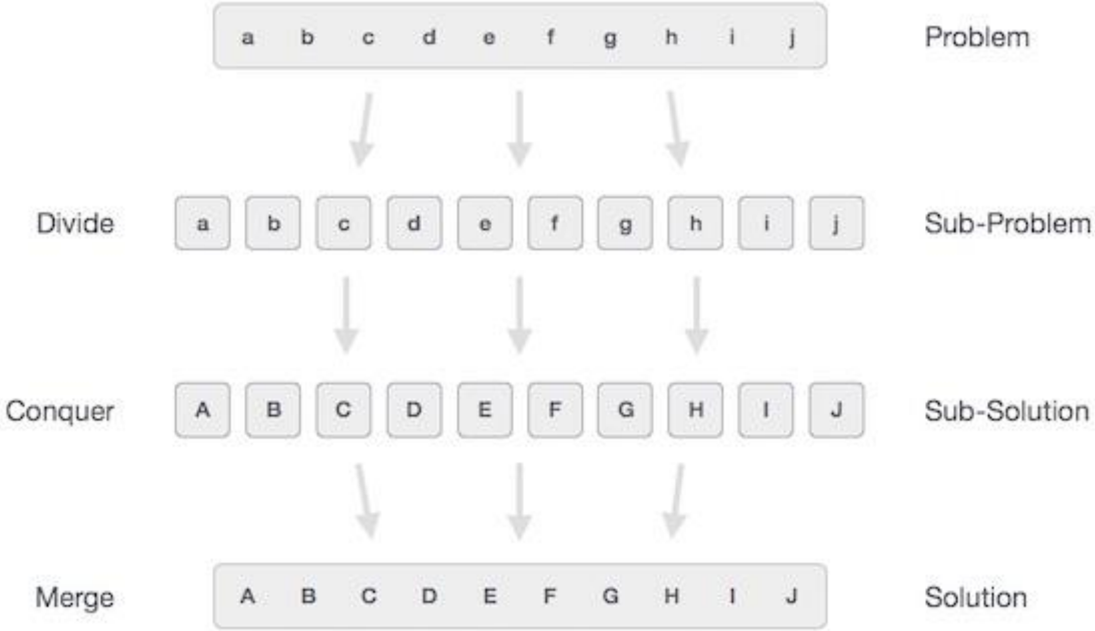
$$T(n) = \frac{3n}{2} - 2$$

$$= O(n)$$

The Time complexity of maximum and minimum is O(n).

Divide and Conquer approach for finding MinMax:

In divide and conquer approach, the problem in hand, is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing the subproblems into even smaller sub-problems, we may eventually reach a stage where no more division is possible. Those "atomic" smallest possible sub-problem (fractions) are solved. The solution of all sub-problems is finally merged in order to obtain the solution of an original problem.



Broadly, we can understand **divide-and-conquer** approach in a three-step process.

Divide/Break

This step involves breaking the problem into smaller sub-problems. Sub-problems should represent a part of the original problem. This step generally takes a recursive approach to divide the problem until no sub-problem is further divisible. At this stage, sub-problems become atomic in nature but still represent some part of the actual problem.

Conquer/Solve

This step receives a lot of smaller sub-problems to be solved. Generally, at this level, the problems are considered 'solved' on their own.

Merge/Combine

When the smaller sub-problems are solved, this stage recursively combines them until they formulate a solution of the original problem. This algorithmic approach works recursively and conquer & merge steps works so close that they appear as one.

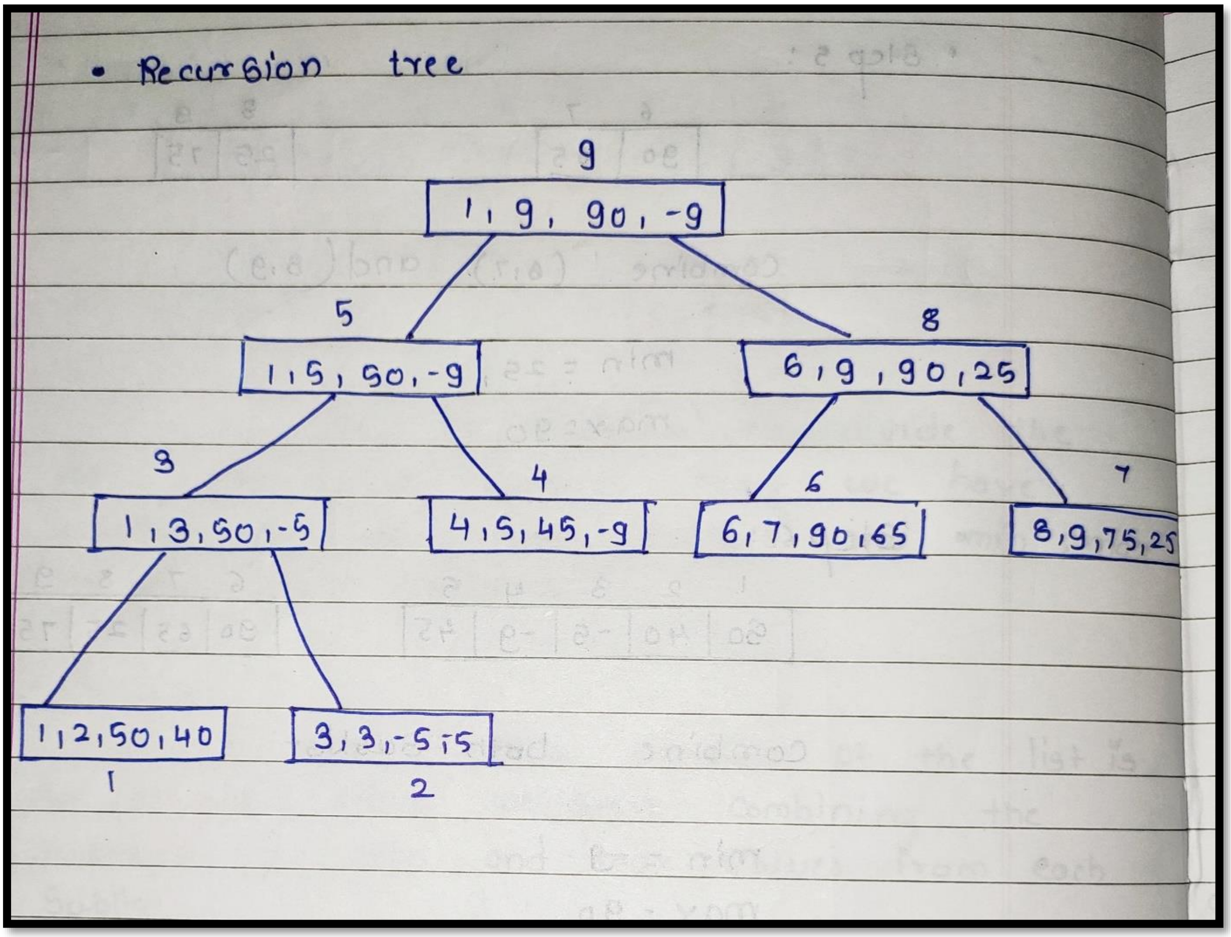
Comment On Time Complexity:

Number of comparisons requires applying the divide and conquering algorithm on n elements/items = $\frac{3n}{2}-2$

Number of comparisons requires applying general approach on n elements = (n-1) + (n-1) = 2n-2

Time Complexity: O(n)

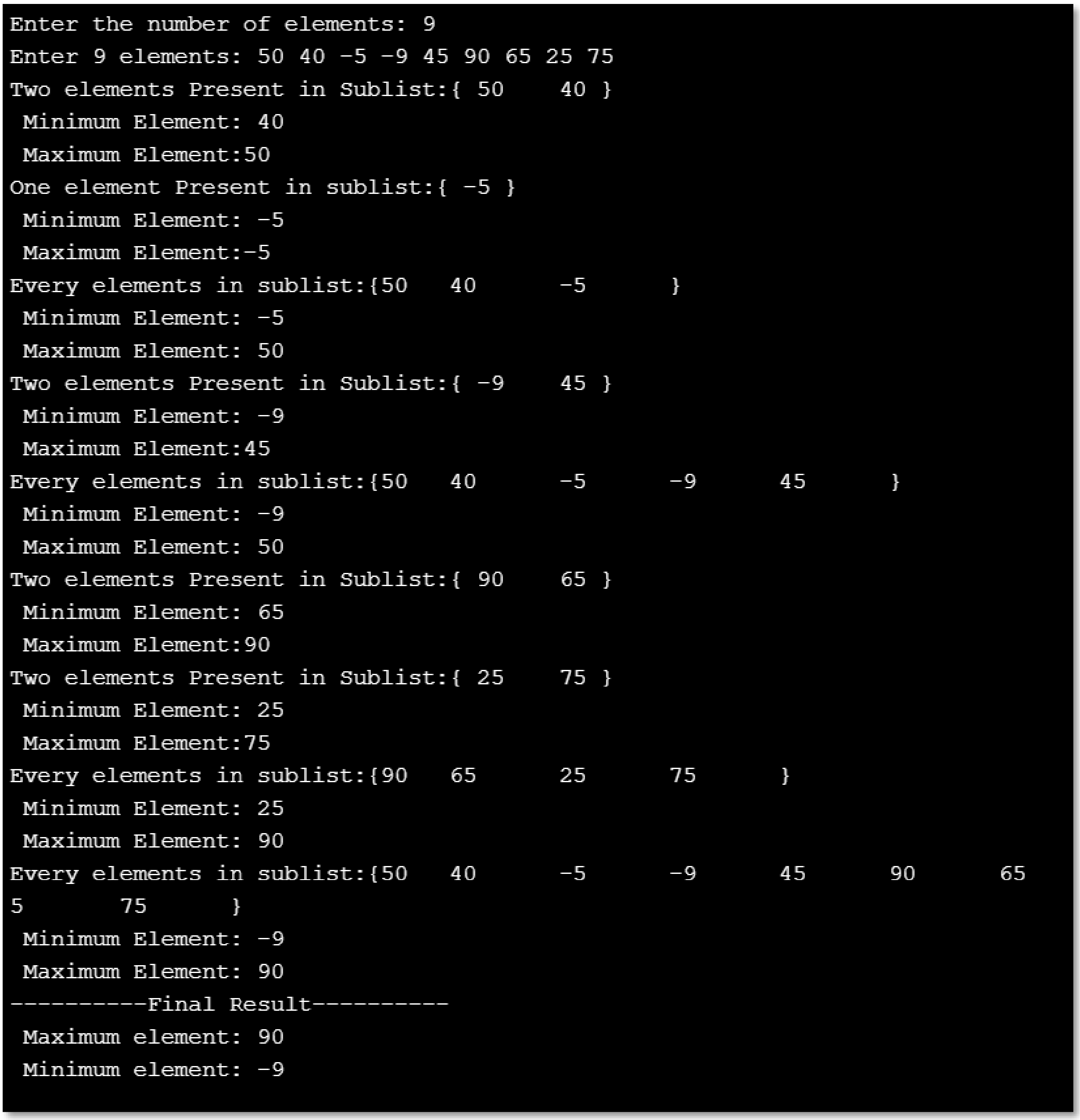
Draw Recursion tree showing calculation for your input numbers.



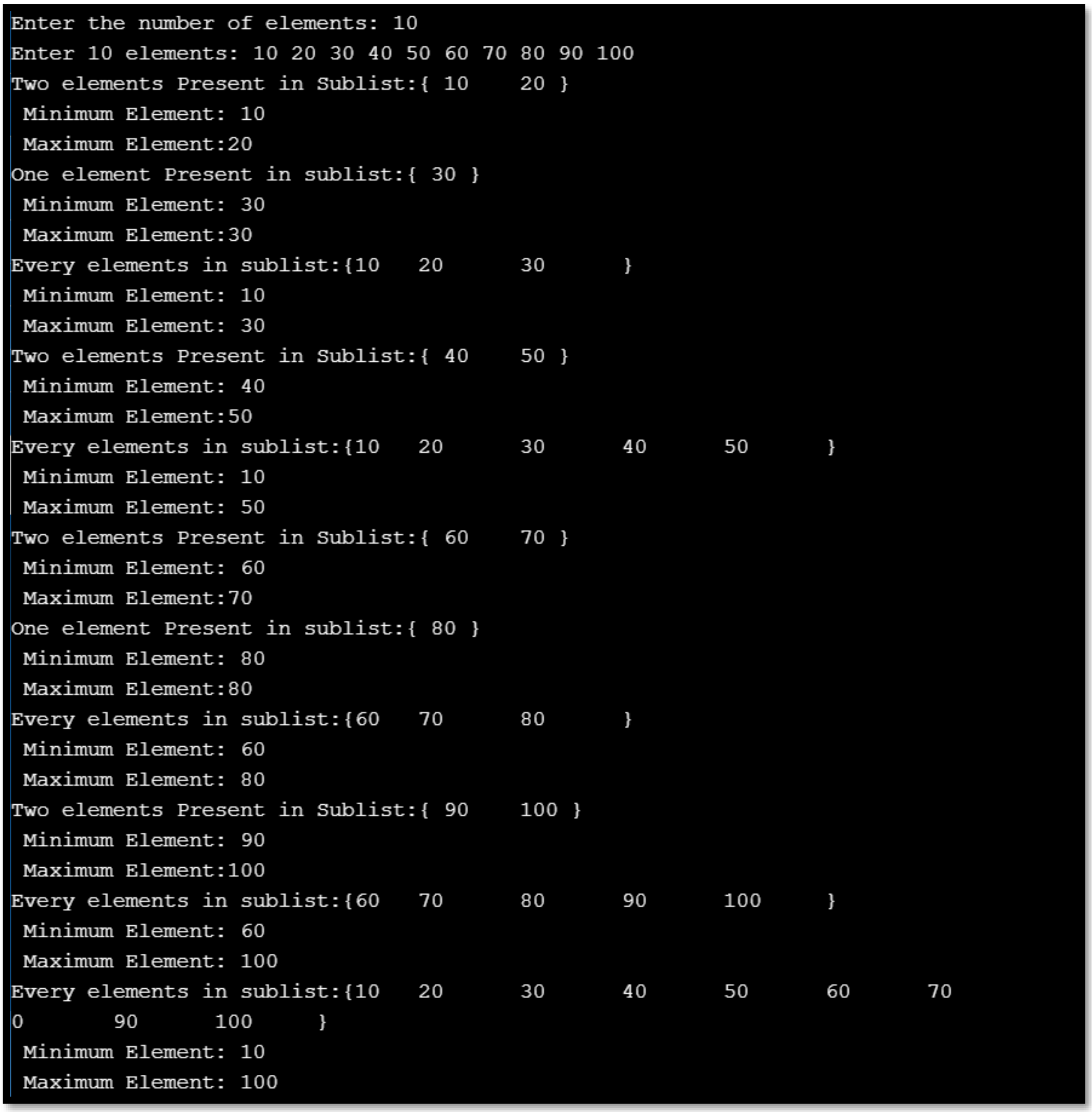
Code:

```
#include<stdio.h>
#include<stdlib.h>
void findmaxmin( int , int , int * , int *);
int a[100] , n ;
int main(){
    int l, h, i , min , max ;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Enter %d elements: ",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    l=0;
    h=n-1;
    findmaxmin(l , h , &max , &min );
    printf("----------Final Result----------");
    printf("\n Maximum element: %d",max);
    printf("\n Minimum element: %d",min);
}
void findmaxmin( int l , int h , int *max ,int *min )
{
  int lmax , rmax , lmin , rmin , mid ;
    if( l == h )
    {
        *max = a[h];
        *min = a[l];
         printf("One element Present in sublist:{ %d } \n Minimum Element: %d\t\n Maximum Element:%d\n",a[l],*min,*max);
         return;
    }
    else if ( l + 1 == h )
    {
        if ( a[l] >= a[h] )
        {
            *max = a[l];
            *min = a[h];
            printf("Two elements Present in Sublist:{ %d\t%d } \n Minimum Element: %d\t\n
Maximum Element:%d\n",a[l],a[h],*min,*max);
            return;
        }
        else
        {
            *max = a[h];
            *min = a[l];
            printf("Two elements Present in Sublist:{ %d\t%d } \n Minimum Element: %d\t\n
Maximum Element:%d\n",a[l],a[h],*min,*max);
            return;
        }
    }
    else
    {
      mid = (l + h) / 2 ;
      findmaxmin( l , mid , &lmax , &lmin );
      findmaxmin( mid + 1 , h , &rmax , &rmin ) ;
      if ( lmax > rmax )
          *max = lmax;
      else
          *max = rmax;
      if ( lmin > rmin )
          *min = rmin ;
      else
          *min = lmin ;
    printf("Every elements in sublist:");
    printf("{");
    for(int i=l;i<h+1;i++){
        printf("%d\t",a[i]);
    }
    printf("}");
    printf("\n Minimum Element: %d\t\n Maximum Element: %d\n",*min,*max);
    return;
        }
    }
```

Output:



```
Enter the number of elements: 9
Enter 9 elements: 50 40 -5 -9 45 90 65 25 75
Two elements Present in Sublist:{ 50    40 }
 Minimum Element: 40
 Maximum Element:50
One element Present in sublist:{ -5 }
 Minimum Element: -5
 Maximum Element:-5
Every elements in sublist:{50   40      -5      }
 Minimum Element: -5
 Maximum Element: 50
Two elements Present in Sublist:{ -9    45 }
 Minimum Element: -9
 Maximum Element:45
Every elements in sublist:{50   40      -5      -9      45      }
 Minimum Element: -9
 Maximum Element: 50
Two elements Present in Sublist:{ 90    65 }
 Minimum Element: 65
 Maximum Element:90
Two elements Present in Sublist:{ 25    75 }
 Minimum Element: 25
 Maximum Element:75
Every elements in sublist:{90   65      25      75      }
 Minimum Element: 25
 Maximum Element: 90
Every elements in sublist:{50   40      -5      -9      45      90      65
5       75      }
 Minimum Element: -9
 Maximum Element: 90
----------Final Result----------
 Maximum element: 90
 Minimum element: -9
```



```
Enter the number of elements: 10
Enter 10 elements: 10 20 30 40 50 60 70 80 90 100
Two elements Present in Sublist:{ 10    20 }
 Minimum Element: 10
 Maximum Element:20
One element Present in sublist:{ 30 }
 Minimum Element: 30
 Maximum Element:30
Every elements in sublist:{10   20      30      }
 Minimum Element: 10
 Maximum Element: 30
Two elements Present in Sublist:{ 40    50 }
 Minimum Element: 40
 Maximum Element:50
Every elements in sublist:{10   20      30      40      50      }
 Minimum Element: 10
 Maximum Element: 50
Two elements Present in Sublist:{ 60    70 }
 Minimum Element: 60
 Maximum Element:70
One element Present in sublist:{ 80 }
 Minimum Element: 80
 Maximum Element:80
Every elements in sublist:{60   70      80      }
 Minimum Element: 60
 Maximum Element: 80
Two elements Present in Sublist:{ 90    100 }
 Minimum Element: 90
 Maximum Element:100
Every elements in sublist:{60   70      80      90      100     }
 Minimum Element: 60
 Maximum Element: 100
Every elements in sublist:{10   20      30      40      50      60      70
0       90      100     }
 Minimum Element: 10
 Maximum Element: 100
```

Conclusion: In minimum maximum the list of elements is divided at the mid in order to obtain two sublists. From both the sublist maximum and minimum elements are chosen. Two maxima and minima are compared and from them real maximum and minimum elements are determined. Also learn understood how to reduce the number of comparison while finding the minimum and maximum value from a set of numbers.