BURCU İSKENDER
21328103

BBM 204 PROGRAMMING LAB.
ASSIGNMENT 1 - REPORT

Time Table

Algorithm/ (n)	unit	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
Matrix Multiplication	s	0,01	0,11	0,76	2,81	29,13	53,17	96,82	168,9	237	323,5	430,8	580
Bubble Sort	µs	125	383	539	749	1386	1834	2420	3057	3723	4513	5396	6644
Find Max. Element	ns	2526	7144	10342	11921	16223	19973	23092	26389	27279	32526	34890	38250
Merge Sort	µs	46	206	385	574	1163	1416	1710	2147	2556	2697	3000	3994
Binary Search	ns	1282	1427	1763	2059	3157	3256	3269	3401	3552	3559	3679	3657

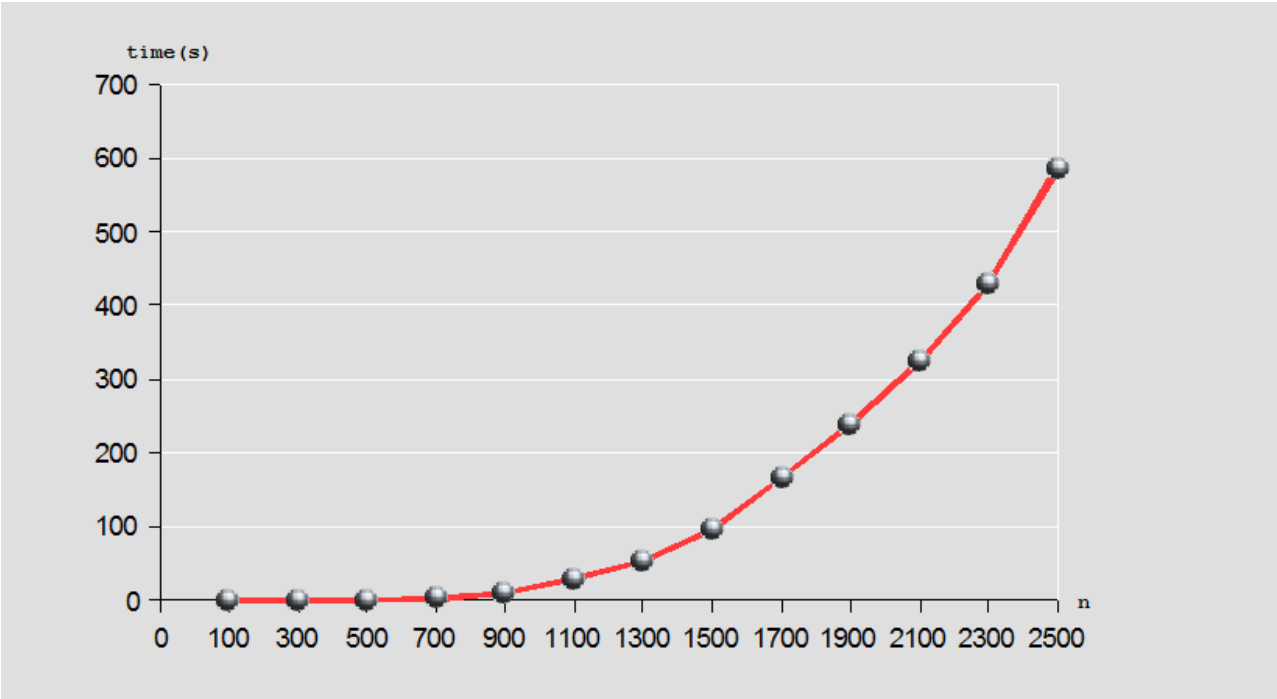
ANALYSİS OF ALGORİTHMS and GRAPHS

1) Matrix Multiplication Algorithm

```
for i from 1 to N                       ->N times
        for j from 1 to N               ->N times
                c(i,j)=0                ->N*N times
        for k from 1 to N               ->N times
                c(i,j) += a(i,k) * b(k,j)   ->N*N*N times
```

Time Complexity: Best case O (N^3)
Average case O (N^3)
Worst case O (N^3)
Because of the nested 3 loops runs N*N*N times

Space complexity: O(N^2)

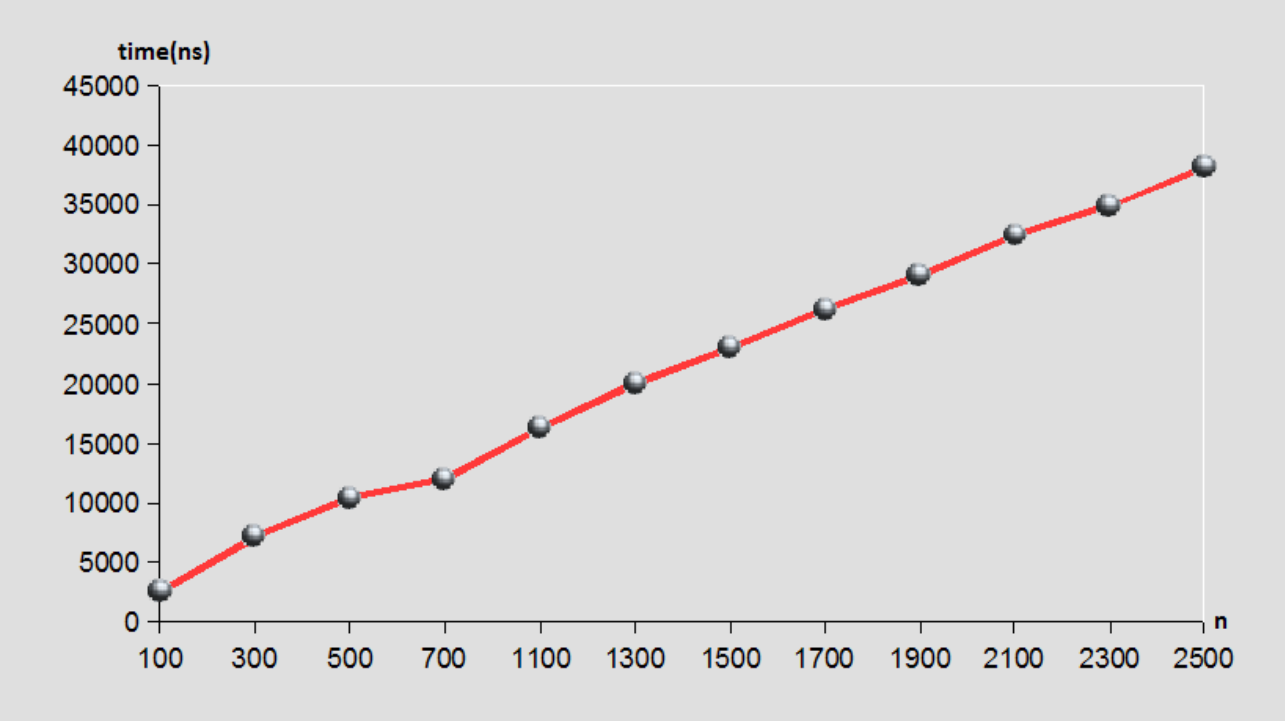


2) Finding Maximum Element Algorithm

```
for i from 1 to N                       ->N times
        if maxElement < a[j]            ->N times
                maxElement = a[j]
```

Time complexity : Best Case O (N)
Average case O(N)
Worst case O(N)

Memory requirement/space complexity : O(N)
Because of the array size of N.24bytes+ 4N (4->int)



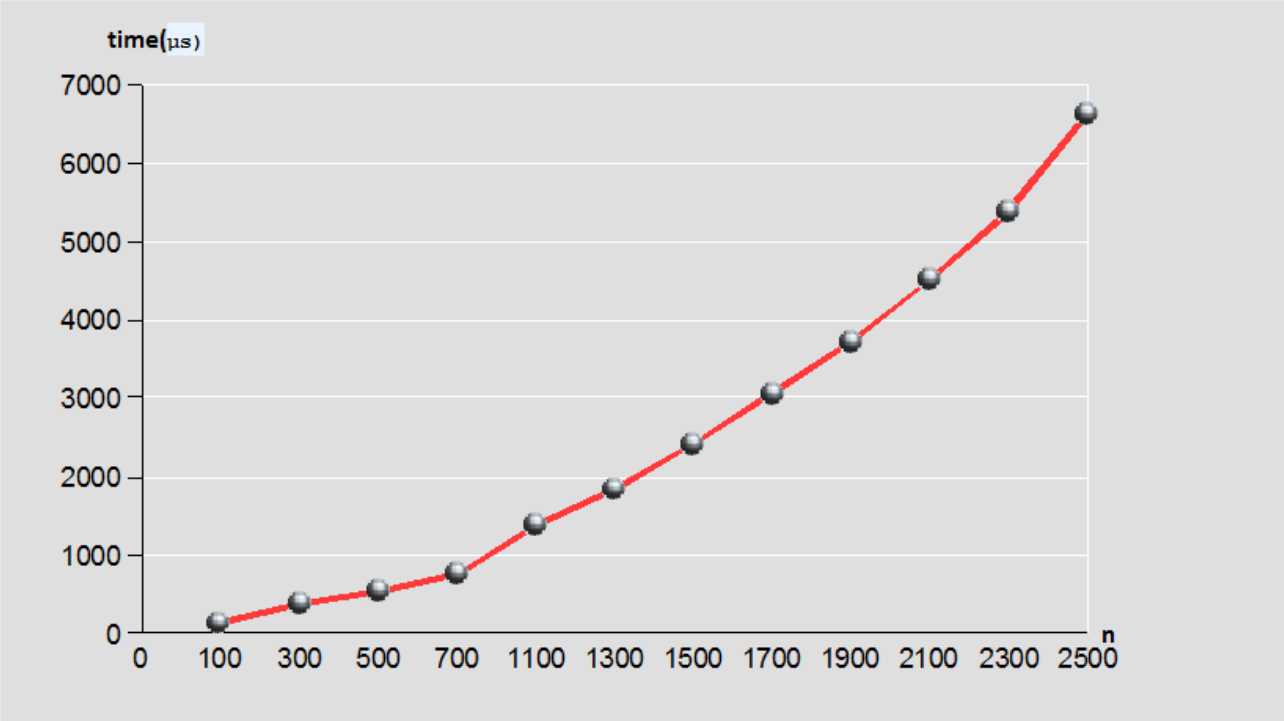
3) BubbleSort Algorithm

```
for i from 1 to N                       ->N times
        swaps=0                         ->N times
        for j from 0 to N-i             -> N+(N-1)+(N-2)+...+1 times
                swaps(a[j],a[j+1])
                swaps +=1
        if swaps = 0
                break
```

Time Complexity : Best Case: O(N)
Average Case: O(N^2)
Worst Case: O(N^2)

If all elements already sorted before run N times.Else for loop run N+(N-1)+(N-2)+...+1 times.This is equal N*(N+1)/2 .

Space complexity/ Memory requirement: O(1)
24+4N (4->integer)



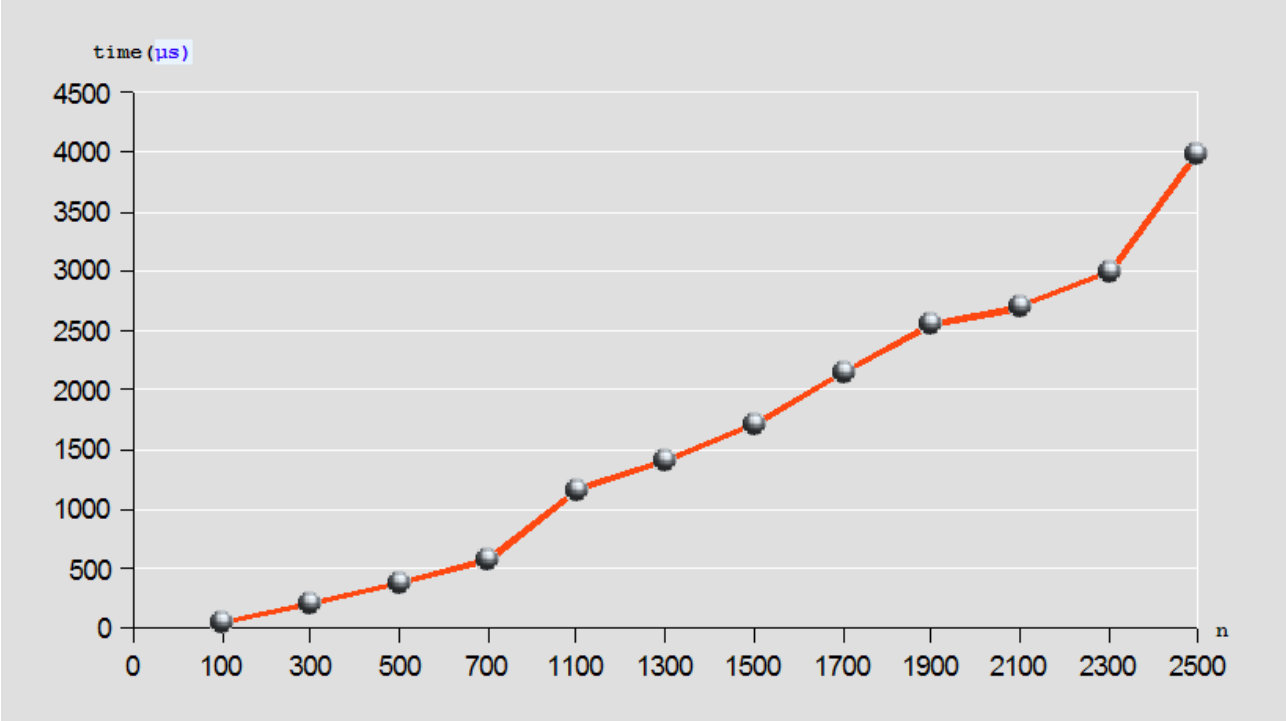
4) Merge Sort Algorithm

```
func mergesort(a[])                                          ->N+N/2+(N/2)/2+..
      if left < right
            int middle = left + (right - left) / 2;
            mergesort(a,left, middle,n);
            mergesort(a,middle + 1, right,n);
            merge(a,left, (middle+1), right,n);              ->N times

function merge{
        temp[];
        left_end = (middle - 1);
        tmp_pos = left;
        num_elements = (right - left + 1);

        while ((left <= left_end) && (middle <= right))
            if a[left] <= a[middle]
                temp[tmp_pos++] = a[left++];
            else
                temp[tmp_pos++] = a[middle++];

        while left <= left_end
            temp[tmp_pos++] = a[left++];

        while middle <= right
            temp[tmp_pos++] = a[middle++];

        for i from 0 to num_elements
            a[right] = temp[right];
            right--;
```

Time Complexity: Best case O(NlogN)
Average case O(NlogN)
Worst case O(NlogN)

time mergesort N elements = time mergesort N/2 elements +time to merge two arrays each N/2 elements. For recurrence relation mergesort function run lgN times and call merge function for all mergesort function N times.this is divide and conquer.

Memory requirement/Space complexity: O(N)
for array a = N In merge sort create an extra array.
for array c = N 3*(24+4N) -> memory requrie



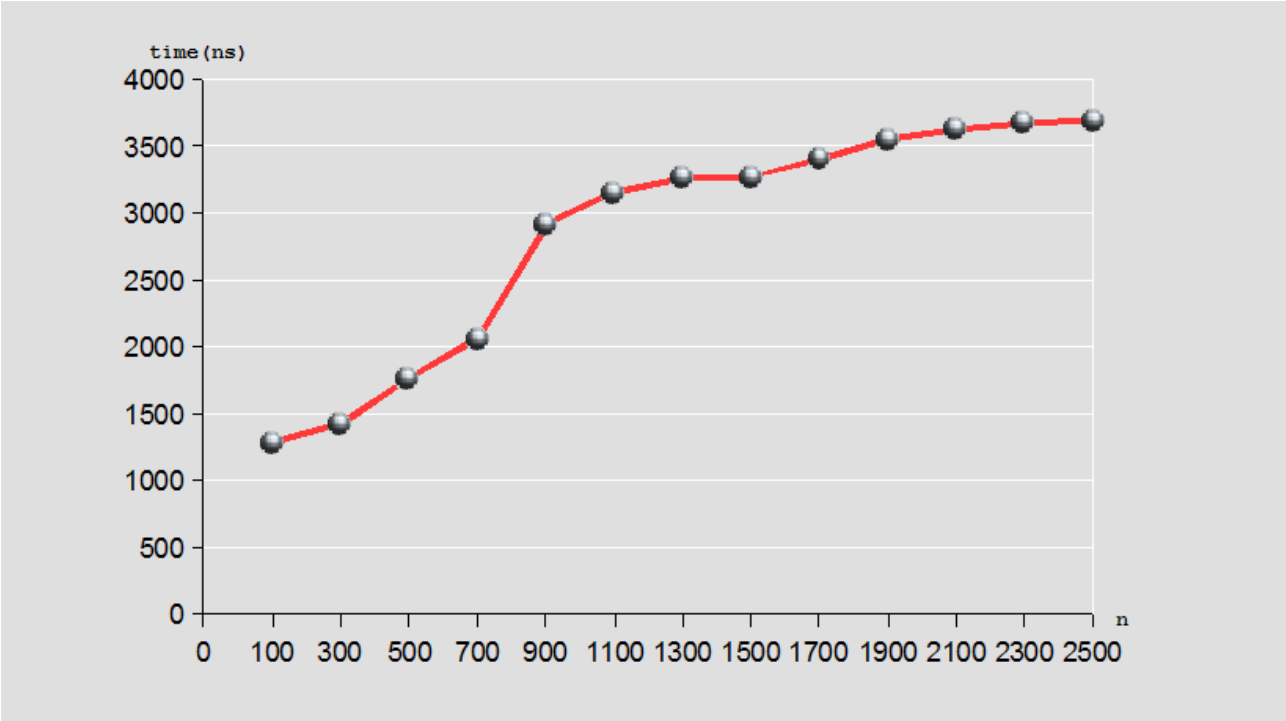
5) Binary Search Algorithm

```
low = 0                                  ->1 times
high = N – 1                             ->1 times
while low <= high
        middle =  low +((high – low) / 2 ->(N/2)+(N/2)/2+((N/2)/2)/2+....
        if  A[mid] > value
                high = mid - 1
        else if
                low = mid + 1
        else return true
return false
```

Time complexity : Best Case: O(1) (searched element in the middle of the array)
Average Case: O(log(n))
Worst Case: O(log(n))

while loop runs lgN time,ecause of mid value is halved each loop time. If searched element in the middle of array while runs 1 time and return true.

Memory Requirement/Space complexity : O(N)
For the array size of N. 24 + 4N-> memory requirement



Sources :

- For bubble sort and matrix multiplication algorithms codes: www.programmingsimplified.com
- For merge sort algorithm http://www.softwareandfinance.com/