Hacettepe University

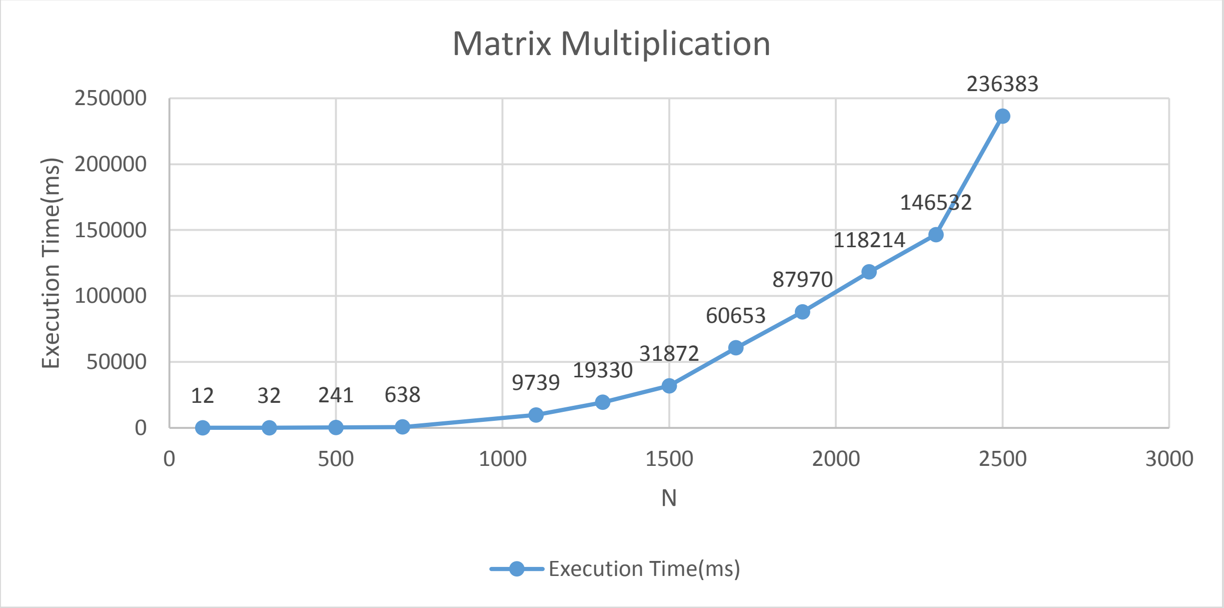
Computer Science and Engineering Departmen

NAME & SURNAME	Candaş Nasıf
ID	21328232
COURSE	BBM204
EXPERIMENT	Assignment I
SUBJECT	Analysis of Algorithms
DUE DATE	01.03.2016
ADVISORS	
E-MAIL	candas.nasif@hacettepe.edu.tr

1.Matrix Multiplication

```
for i = 1 to N
    for j = 1 to N
        c(i,j) = 0
        for k = 1 to N
            $c(i,j) = c(i,j) + a(i,k) * b(k,j)$
        end
    end
end
```

N	Execution Time(ms)
100	12
300	32
500	241
700	638
1100	9739
1300	19330
1500	31872
1700	60653
1900	87970
2100	118214
2300	146532
2500	236383

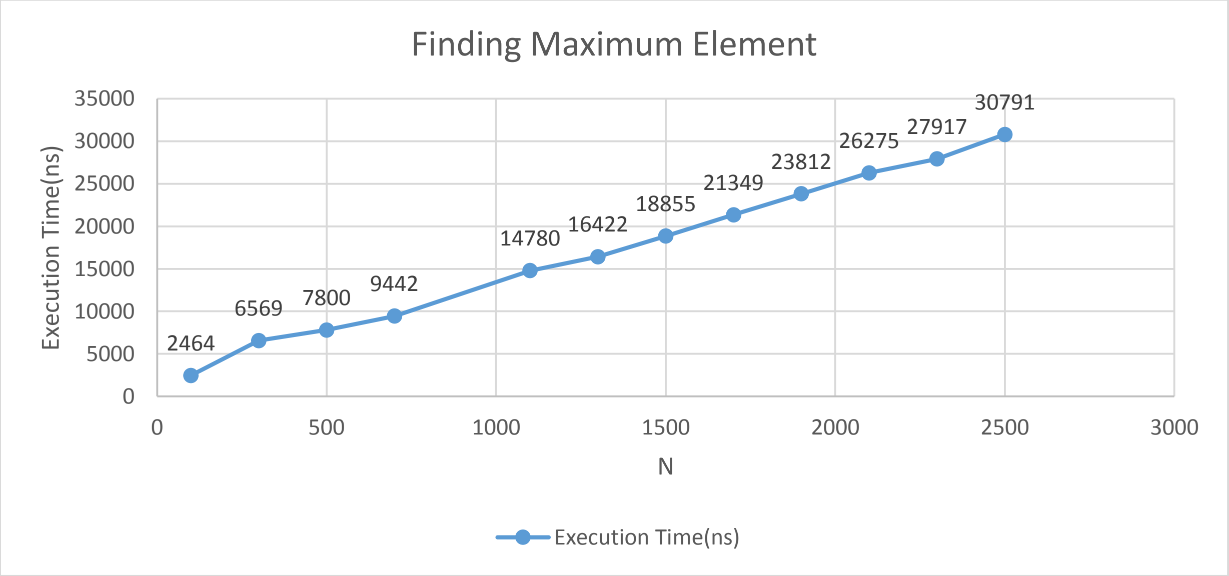


I tried this algorithm different N values and calculate execution times for each N.This table show us matrix multiplication complexity.Matrix multiplication complexity is O(n^3). because on each of the loop, N is multiplied by N, since you have a nested loop 3 times which completely process the entire N, that will be N X N X N = N^3 .I calculated the execution time in millisecond.Because times are huge numbers for nanosecond.Graph show us execution time grows up with N proportional and complexity of matrix multiplication.Best case, worst case and averege case sane for this algorithm.Because we use three nested loops and all three loops travel the 1 to N for all inputs.This algorithm run slowly because of complexity.At the begining growth of execution time is less but then it grow up faster and big difference.

2.Finding Maximum Element

```
Func Find Max Element(var a as array)
For i from i to N
        İf a[i]> max
                Max=a[i]
End func
```

N	Execution Time(ns)
100	2464
300	6569
500	7800
700	9442
1100	14780
1300	16422
1500	18855
1700	21349
1900	23812
2100	26275
2300	27917
2500	30791

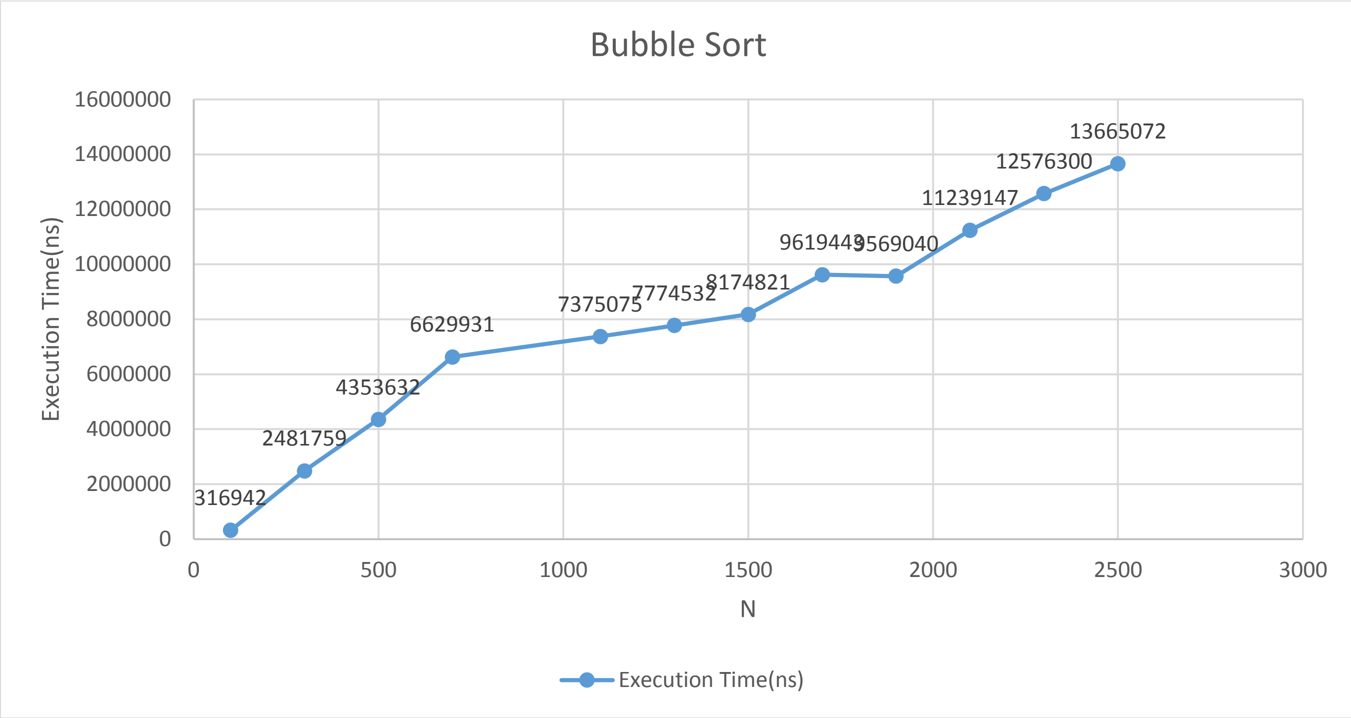


This algorithm find the maximum element in an array.Travel an array and compare elements of array, always keep bigger element and at last return the biggest element.Algorithms complexity is O(n) Because we use just one loop 0 to N.Best, worst and averege case same for this algorithm.Because we travel all array for all inputs.We have similar a y=x line on the graph because of complexity.We have stable growth for execution time.I calculeted execution time with nanosecond.Because I saw zero when i calculeted with milisecond.

3.Bubble Sort

```
func Bubble sort(var a as array)
 for i from 1 to N
        swaps = 0
        for j from 0 to N - 1
                if a[j] > a[j+1]
                        swap(a[j], a[j+1])
                        swaps = swaps + 1
        if swaps = 0 break
end func
```

N	Execution Time(ns)
100	316942
300	2481759
500	4353632
700	6629931
1100	7375075
1300	7774532
1500	8174821
1700	9619443
1900	9569040
2100	11239147
2300	12576300
2500	13665072



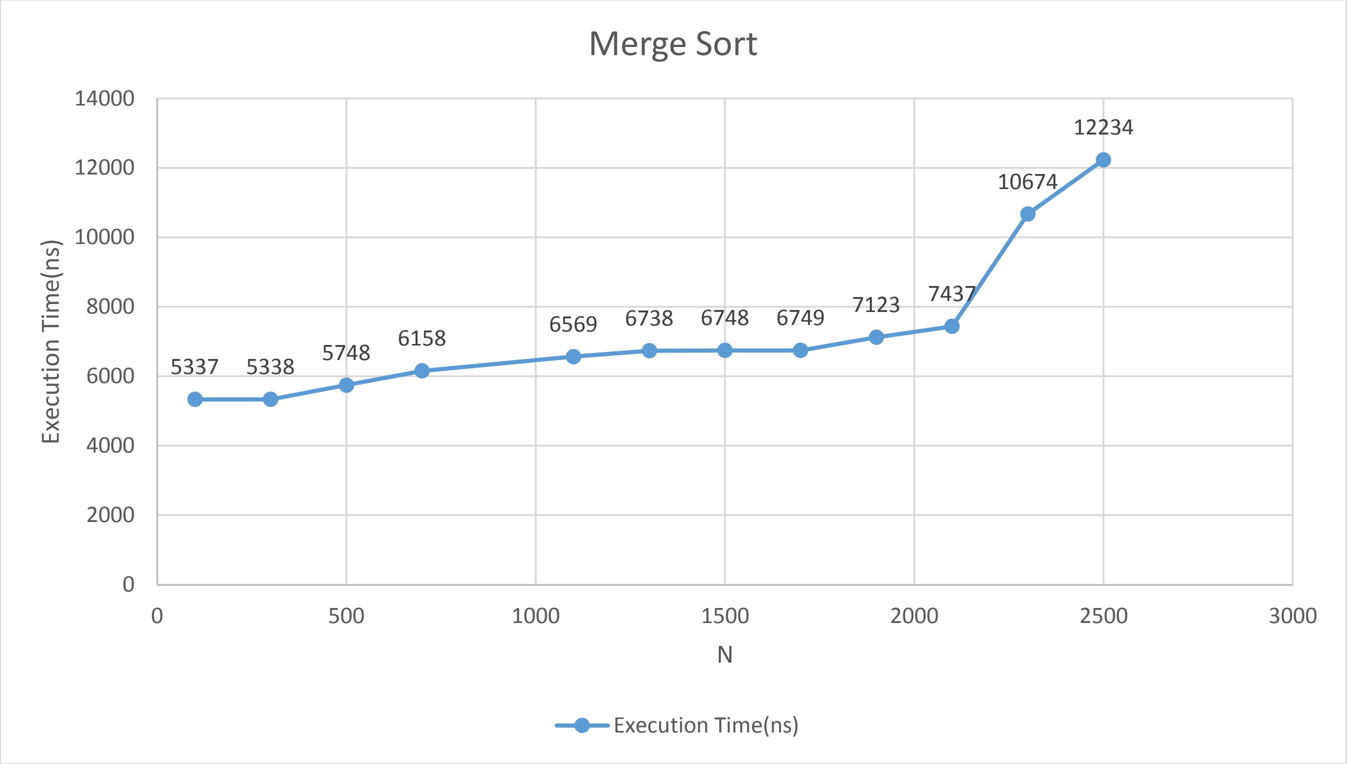
This algorithm sort an array with comparison.We compare to two elements of array and if first bigger than second one swap them.This operation continued as array sorting completed.Algorithm complexity is O(n^2).Because we use two loop in algorithm.First loop 0 to N second loop 0 to N-1. N*(N-1)=N^2-N.So complexity is O(n^2).Best case for this algorithm if we can sort the array just first iterate and complexity will be O(n).Worst case for this algorithm if array have sorted last iterate then complexity will be O(n^2).Graph show us execution time grow nonlinearity but growings are not big from n^3.I calculated execution time in nanosecond because millisecond too small for see the execution time.

4.Merge Sort

```
func mergesort(var a as array)
    if(n==1)
        return a
    varl1asarray=a[0]...a[n/2]
    varl2asarray=a[n/2+1]...a[n]
    l1=mergesort(l1)
    l2=mergesort(l2)
    returnmerge(l1,l2)
end func
```

```
func merge(var a as array,var b as array)
    var c as array
    while(a and b have elements)
        if(a[0]>b[0])
            add b[0] to the end of c
            remove b[0] from b
        else
            add a[0] to the end of c
            remove a[0] from a
        while(a has elements)
            add a[0] to the end of c
            remove a[0] from a
        while(b has elements)
            add b[0] to the end of c
            remove b[0] from b
     return c
end func
```

N	Execution Time(ns)
100	5337
300	5338
500	5748
700	6158
1100	6569
1300	6738
1500	6748
1700	6749
1900	7123
2100	7437
2300	10674
2500	12234



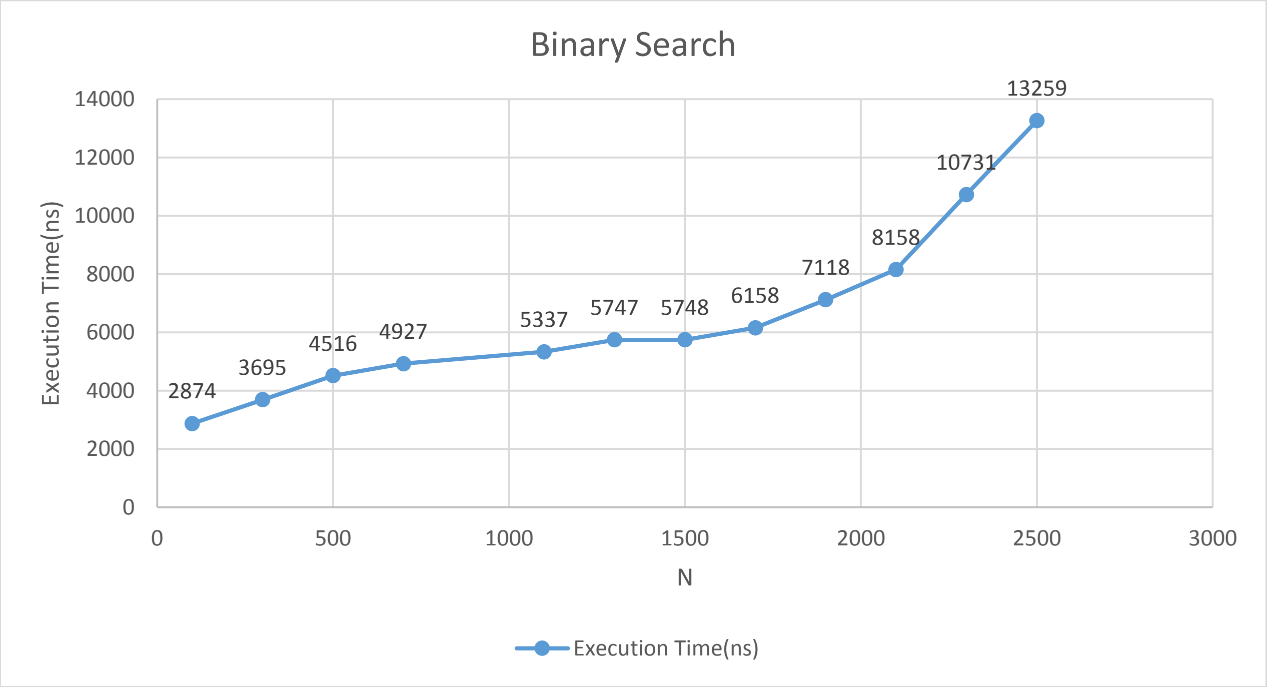
Merge sort algorithm first of all divide two equal part and keep them part.Then compare elements of array .Whic one is bigger one add to final sorted array this element.Remove the appended element from the array.Then algorithm continue this operation when array sorted completely.Merge sort algorithm faster than bubble sort algorithm(graph show us that).Because merge sort complexity is O(nlogn) dir.Complexity same for all inputs.Because algorithm divide the array two equal parts all inputs and do same operations.I calculated execution time in nanosecond .My graph does not like nlogn graph.I iterate the running and calculation operate but I could not take better results.But growth on end of graph line looks like nlogn graph.

5.Binary Search

```
funcBinarySearch(a,value,left,right)
    while left<=right
        mid=floor((right−left)/2)+left
        ifa[mid]==value
            return mid
```

```
                if value < a[mid]
                        right = mid−1
                else left = mid+1
        return not found
end func
```

N	Execution Time(ns)
100	2874
300	3695
500	4516
700	4927
1100	5337
1300	5747
1500	5748
1700	6158
1900	7118
2100	8158
2300	10731
2500	13259



Binary search using for search an element in an sorted array.Binary search divide the sorted array and compare with middle element and searching element.If searching elment smaller than middle element same operation doing with right part of array else same operation doing left part of array.This operation over when searching element has been found.Algorithms complexity is O(logn).Best case for this algorithm if searching element is first middle elemetn this algorithm run with O(1).Worst case for this algorithm if searching element is last middle element this algorithm run with O(logn).We can see binary search graph looks like logn graph.Binary search algorithm run slowly for big array .Execution time grows much when N will grow.I calculate execution in nanosecond.