HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BBM204 SOFTWARE PRACTICUM

Taha BASKAK
21228104

Contents

1 RUNNING TIME ANALYSIS **2**

1.1 Analyze the algorithm , total cost and find tilde notation 2

1.2 Five Different algorithm and analyze 3

1.2.1 Finding Second Largest Element Algorithm 4

1.2.2 Stooge Sort Algorithm 5

1.2.3 Radix Sort Algorithm 6

1.2.4 Shaker Sort Algorithm 7

1.2.5 Maximum Subarray Algorithm 8

1.3 Question 9

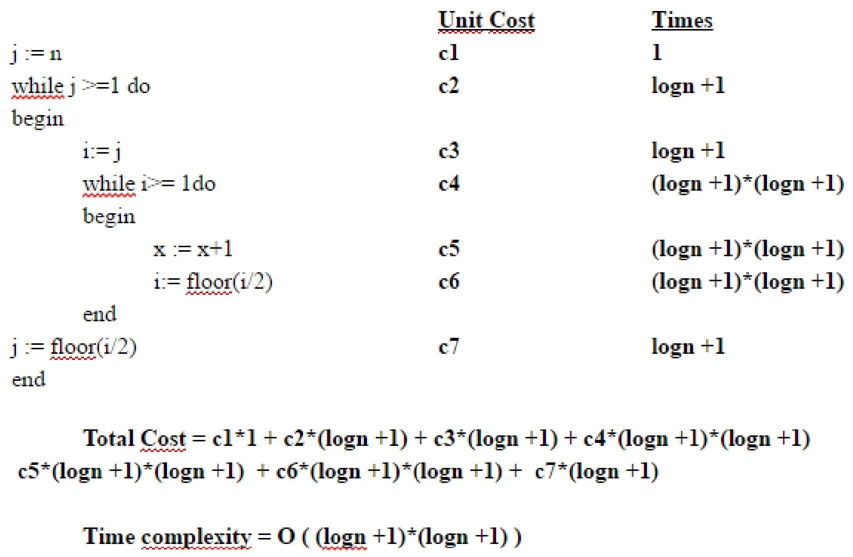
2 FINDING Nth NEAREST TOURISTIC PLACES **10**

3 REFERENCES **11**

1 RUNNING TIME ANALYSIS

Analysis of the algorithms given in this section

1.1 Analyze the algorithm , total cost and find tilde notation



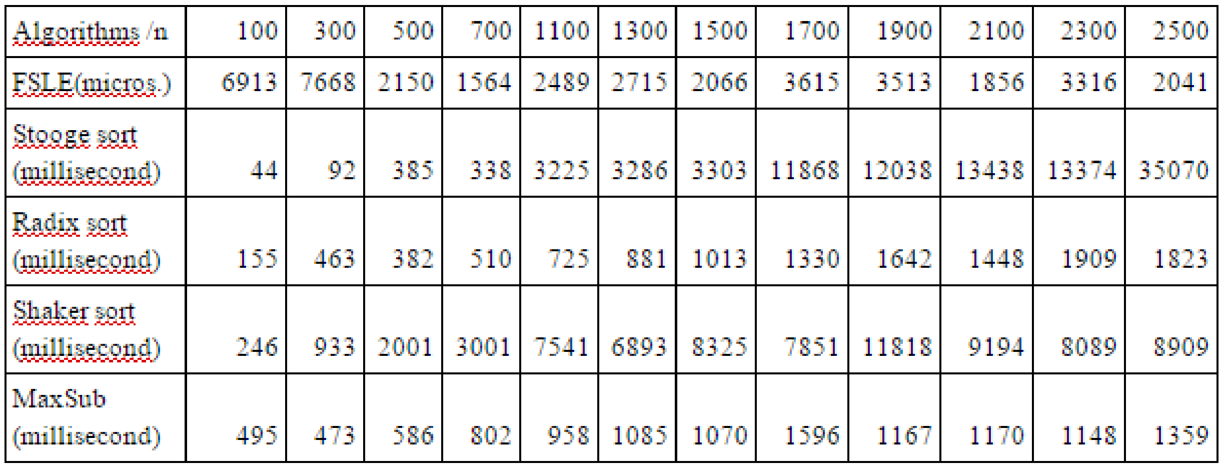
```
                                          Unit Cost        Times
j := n                                    c1               1
while j >=1 do                            c2               logn +1
begin
    i:= j                                 c3               logn +1
    while i>= 1do                         c4               (logn +1)*(logn +1)
    begin
        x := x+1                          c5               (logn +1)*(logn +1)
        i:= floor(i/2)                    c6               (logn +1)*(logn +1)
    end
j := floor(i/2)                           c7               logn +1
end
```

Total Cost = c1*1 + c2*(logn +1) + c3*(logn +1) + c4*(logn +1)*(logn +1) c5*(logn +1)*(logn +1) + c6*(logn +1)*(logn +1) + c7*(logn +1)

Time complexity = O ((logn +1)*(logn +1))

1.2 Five Different algorithm and analyze

The running time of 5 different algorithms given in this part was calculated according to the algorithm complexities and the number of elements in 13 different sizes (100-300 -...- 2500).



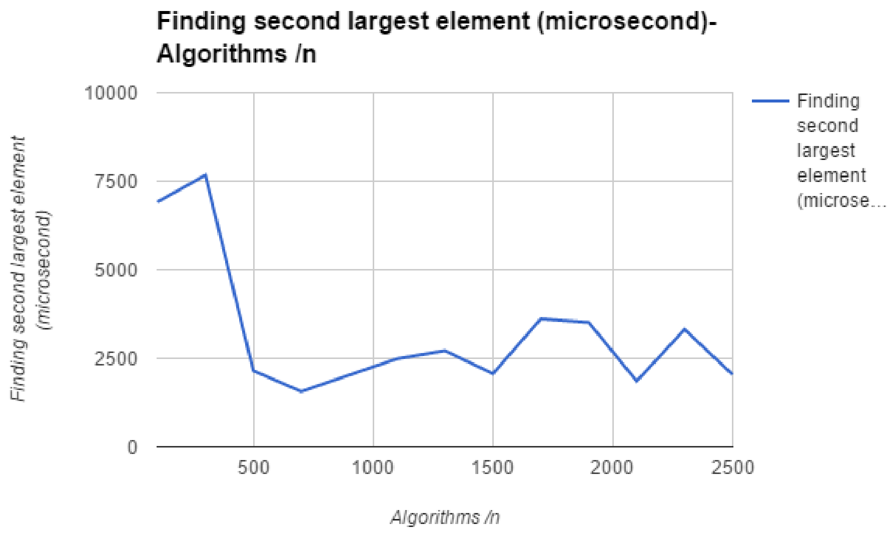
Algorithms /n	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
FSLE(micros.)	6913	7668	2150	1564	2489	2715	2066	3615	3513	1856	3316	2041
Stooge sort (millisecond)	44	92	385	338	3225	3286	3303	11868	12038	13438	13374	35070
Radix sort (millisecond)	155	463	382	510	725	881	1013	1330	1642	1448	1909	1823
Shaker sort (millisecond)	246	933	2001	3001	7541	6893	8325	7851	11818	9194	8089	8909
MaxSub (millisecond)	495	473	586	802	958	1085	1070	1596	1167	1170	1148	1359

ALGORITHM'S TIME

Units of held times are next to the name of the algorithm. It is defined in the interface in the sent code.
Chose one and input number
1.Finding Second Largest
2.Stooge Sort
3.Radix Sort
4.Shaker Sort
5.Maximum Subarray
6.Exit Program

1.2.1 Finding Second Largest Element Algorithm

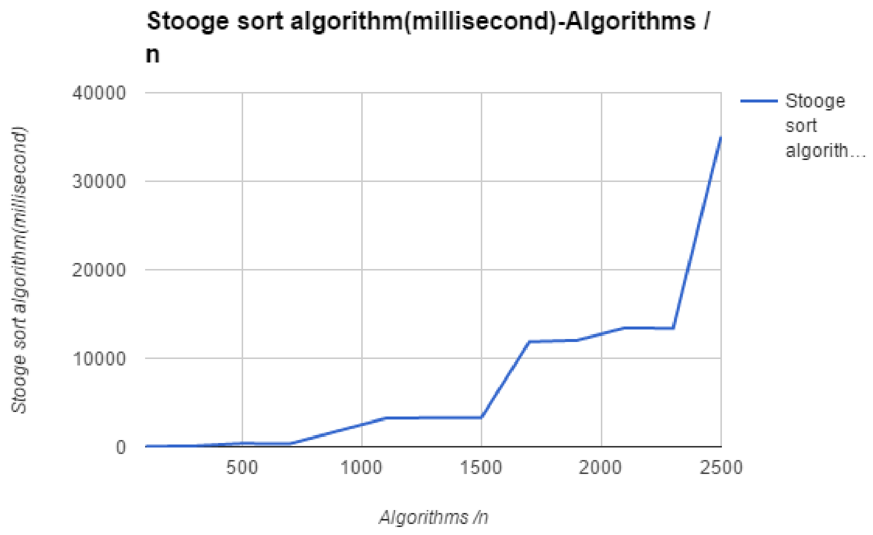
The algorithm simply selects the maximum and the second maximum among the first two elements. It then changes the first and second maximum values according to the values that are passed through all the other elements.



As you can see in the graph, the working time was higher than the others at first. The reason for this is primarily due to randomly generated numbers. Later, the reason for this fall is due to the fact that the random numbers generated by all the sizes work in succession, resulting in faster memory counts due to cache memory.

1.2.2 Stooge Sort Algorithm

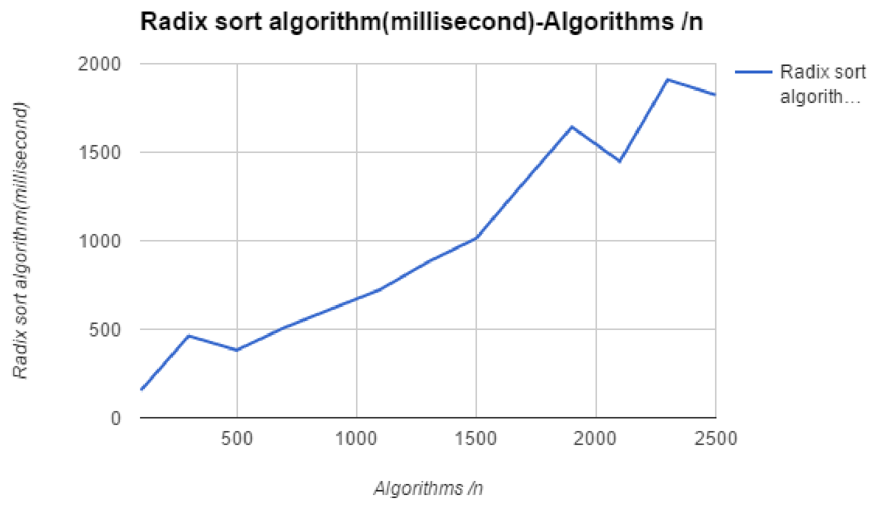
The algorithm simply compares and replaces the first and last element of the submitted index. Then, taking the difference of max and min value which is a fixed value according to the difference of the maximum and minimum values sent to be greater than 1, it takes the difference to 3 and recursively executes the function recursively.



As can be seen from the graph, as the number N increases, the working time increases. The reason for this is that the algorithm is not very effective. The reason for the sudden rise of intermediate passes also originates from the size of the numbers produced.

1.2.3 Radix Sort Algorithm

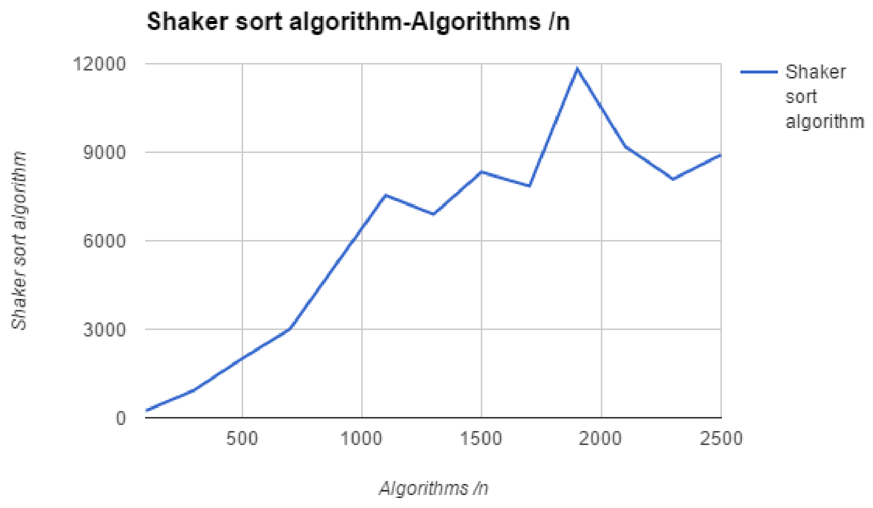
The algorithm first finds the largest element in the sequence. The value is then multiplied by 10 in each cycle. Then the intermediate sequence is added to the elements of the first directory within certain rules and all the elements of the last intermediate directory are added to the original sequence.



Compared to other graphs, the run time of the radix sort algorithm is much shorter. The reason for this is that it performs faster by dividing small pieces in the sorting process in its algorithm.

1.2.4 Shaker Sort Algorithm

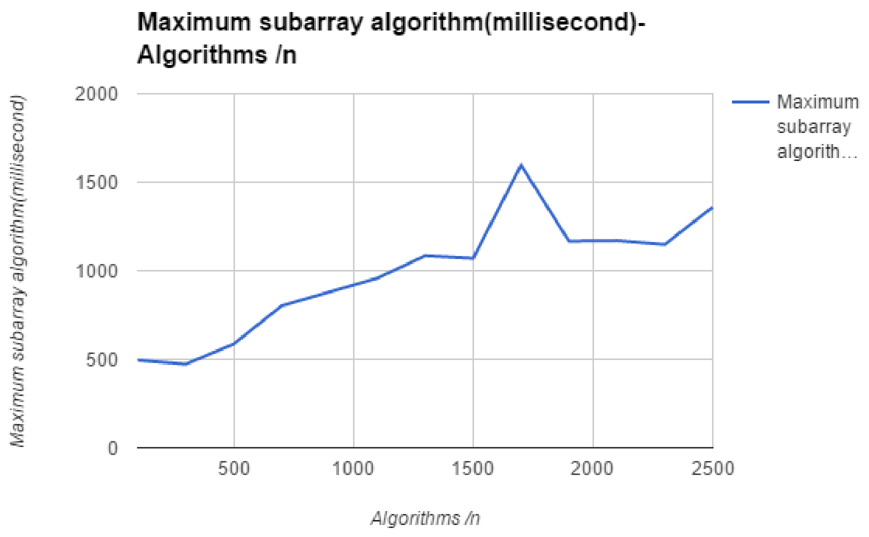
The algorithm simply enters two different loops, rotating the first half of the size of the submitted array. In the first half of the return value, the value of the index is subtracted from the value of the index, and the whole value is rotated. In the other loop, starting from the value of the index when it returns from the first half of the size, it rotates from the first half to the return value. This line is sequenced.



As seen in the graph, as the N dimension increases, the work increases over time. The reason for the ups and downs is that cache memory gets faster access to the previously held random numbers.

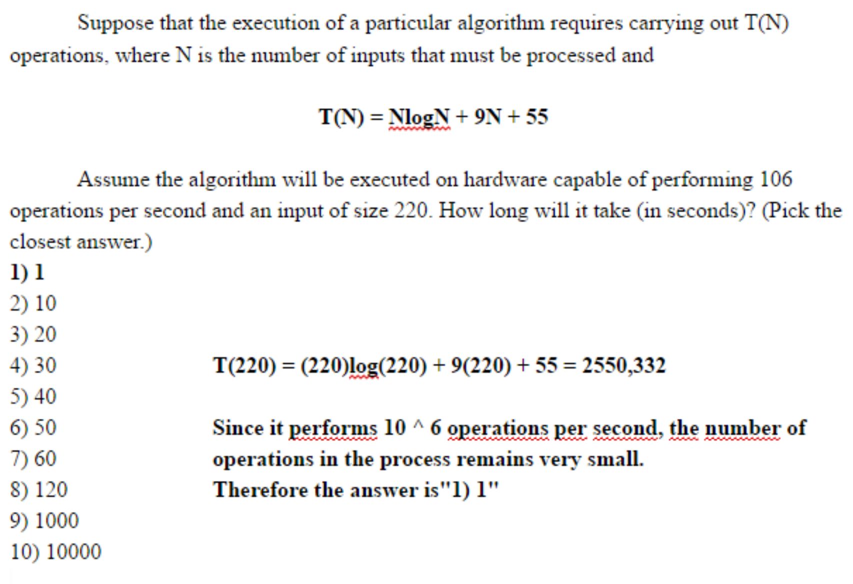
1.2.5 Maximum Subarray Algorithm

The main purpose of this algorithm is to find the subsequence which is the largest sum in the whole array. The general operation divides the sequence into two, and tries to find the subsequence which is the largest sum in the left and right subsequences.



As the graph shows, as the number N increases, the working time increases. This is due to an increase in the number of products produced and an increase in the number N.

1.3 Question



Suppose that the execution of a particular algorithm requires carrying out T(N) operations, where N is the number of inputs that must be processed and

T(N) = NlogN + 9N + 55

Assume the algorithm will be executed on hardware capable of performing 106 operations per second and an input of size 220. How long will it take (in seconds)? (Pick the closest answer.)

1) 1
2) 10
3) 20
4) 30
5) 40
6) 50
7) 60
8) 120
9) 1000
10) 10000

T(220) = (220)log(220) + 9(220) + 55 = 2550,332

Since it performs 10 ^ 6 operations per second, the number of operations in the process remains very small.
Therefore the answer is"1) 1"

2 FINDING Nth NEAREST TOURISTIC PLACES

Alice wants to visit Spain and she has one day. There are one hundred places to visit, and it's important for Alice to pay for it. In the code I wrote, the coordinates of Alice and one hundred places were randomly generated. The booster distances were then calculated. If booster distances are less than two hundred, priority is given to the cost of the place to visit. If booster distances are greater than two hundred, priority is given to the shortness of the trip distance. The program initially prompts the user for the number of places to visit and can be visited up to one hundred places. If more space is required, the program will alert you. After completing the operations, the user navigates places according to priority order.

3 REFERENCES

http://stackoverflow.com/questions/2615712/finding-the-second-highest-number-in-array
http://www.geeksforgeeks.org/radix-sort/
http://www.programming-algorithms.net/article/40270/Shaker-sort
http://www.geeksforgeeks.org/divide-and-conquer-maximum-sum-subarray
http://www.geeksforgeeks.org/largest-sum-contiguous-subarray/