# HACETTEPE UNIVERSITY SOFTWARE LABORATORY II ASSIGNMENT 1

# ONUR TOLGA KESEMEN 21228511

March 25, 2017

#### 1 PART 1: RUNNING TIME ANALYSIS

#### 1.1 Total Cost of Algorithm

I found costs for all units in the algorithm given below. Then calculated total cost as c1 + (n+1)c2 + n\*c3 + n\*(n+1)\*c4 + n\*n\*c5+n\*n\*c6 + n\*c7. The tilde notation for that algorithm is n\*n

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

#### 1.2 Measurement Of Runtime

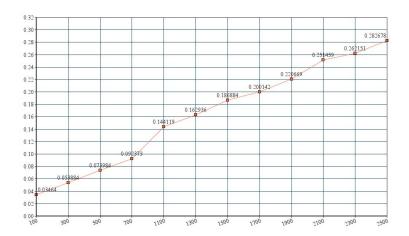
Algorithms/n	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
Finding second largest	0,03464	0,05388	0,07398	0,09237	0,144119	0,162936	0,186884	0,200142	0,220669	0,251459	0,26215	0,282678
element		4	4	3							1	
Stooge sort algorithm	8,20965	29,9951	198,776	199,512	1701,556	1547,838	1729,484	4008,901	4328,165	3963,396	4031,80	13776,24
	3	46	485	048	015	321	742	595	861	762	4546	3882
Radix algorithm	0,11760	0,35709	0,93698	1,28210	1,568203	1,794004	2,081387	2,791291	2,636907	2,884519	3,23433	3,416091
	5		8	4		100			1.03		9	100
Shaker sort algorithm	0,14625	1,20170	2,80241	3,31259	4,929555	5,94523	7,153349	8,510293	8,851132	9,814206	9,39510	10,13109
	7	5		9							6	7
Maximum subarray	0,11161	0,52344	0,82708	1,22950	2,31403	2,74938	3,468693	5,469253	6,39897	7,067392	7,45655	6,570886
algorithm	8	7	1	2							6	
												-

<sup>\*</sup>Time measurement is in milisecond

#### 1.2.1 Finding Second Largest Element In An Array

Algorithms	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
/n												
Finding	0,034	0,0538	0,0739	0,0923	0,1441	0,1629	0,1868	0,2001	0,2206	0,2514	0,2621	0,2826
second	64	84	84	73	19	36	84	42	69	59	51	78
largest												
element												

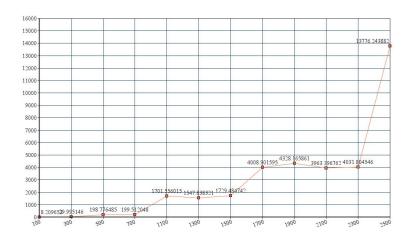
<sup>\*</sup>Time measurement is in milisecond



#### 1.2.2 Stooge Sort Algorithm

Algorithms	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
/n												
Stooge sort	8,20	29,995	198,7	199,51	1701,	1547,8	1729,	4008,9	4328,1	3963,	4031,8	13776,
algorithm	9653	146	76485	2048	55601	38321	48474	01595	65861	39676	04546	243882
					5		2			2		

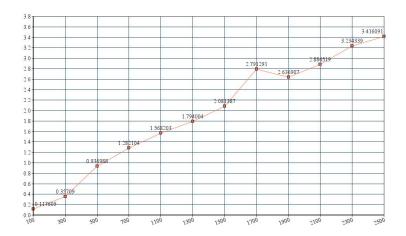
<sup>\*</sup>Time measurement is in milisecond



#### 1.2.3 Radix Sort Algorithm

Algorithm	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
s/n												
Radix	0,1176	0,357	0,9369	1,2821	1,5682	1,7940	2,0813	2,7912	2,6369	2,8845	3,2343	3,4160
algorithm	05	09	88	04	03	04	87	91	07	19	39	91

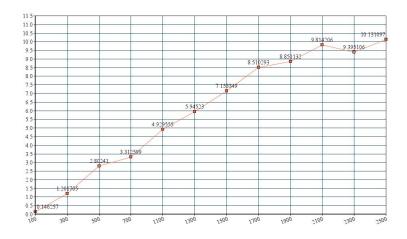
<sup>\*</sup>Time measurement is in milisecond



## 1.2.4 Shaker Sort Algorithm

Algorithm	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
s/n												
Shaker	0,1462	1,2017	2,802	3,3125	4,9295	5,945	7,1533	8,5102	8,8511	9,8142	9,3951	10,1310
sort	57	05	41	99	55	23	49	93	32	06	06	97
algorithm									9			

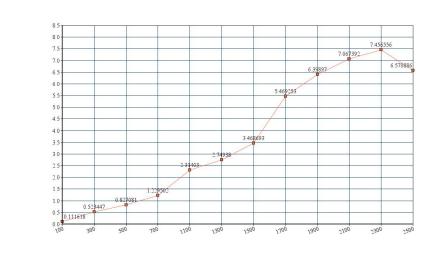
<sup>\*</sup>Time measurement is in milisecond



## 1.2.5 Maximum Subarray Algorithm

Algorithm s/n	100	300	500	700	1100	1300	1500	1700	1900	2100	2300	2500
Maximum	0,1116	0,5234	0,8270	1,2295	2,314	2,749	3,4686	5,4692	6,398	7,0673	7,4565	6,5708
subarray algorithm	18	47	81	02	03	38	93	53	97	92	56	86

<sup>\*</sup>Time measurement is in milisecond



## 2 REFERENCES

 $http://web.cs.hacettepe.edu.tr/\ bbm202/slides/06-heaps.pdf$