ANALYSIS OF ALGORITHMS

REPORT

written by Buğra Ekuklu

You may compile the codes with following commands

(the current working directories are the folders of different applications):

```
g++ AoAHW3ApplicationDelegate.cpp Bootstrapping/Application.cpp Containers/Array.cpp
Strategies/AlgorithmicSortStrategy.cpp Strategies/RadixSortStrategy.cpp Models/DOMElement.cpp
Supporting\ Files/main.cpp -O3 -std=c++11 -o main
```

```
g++ AoAHW3ApplicationDelegate.cpp Bootstrapping/Application.cpp Containers/Array.cpp
Models/DOMElement.cpp Models/Player.cpp Strategies/AlgorithmicSortStrategy.cpp Strategies/
HeapSortStrategy.cpp Supporting\ Files/main.cpp -O3 -std=c++11 -o main
```

A. Sorting In Linear Time

1. The complexity we expected to obtain was $O(n\ k)$, however since memory allocations took too much time, this could not become visible until we reach $n = 10000$. The parameter $k = 10$ did not change since it indicates the number of digits of the input. The performance graph of the algorithm with respect to N over runtime is included at the end of the report.

2. The gnome sort is stable since the cursors (i and j) in the algorithm moves with respect to the compared values. However, freezing sort is stable under a circumstance, where it sorts decreasing. In incremental sorting, freezing sort needs to be modified, where freezing blocks need to be interchanged. The implementation given behaves stable in decremental sorting, but needs to be modified in incremental sorting.

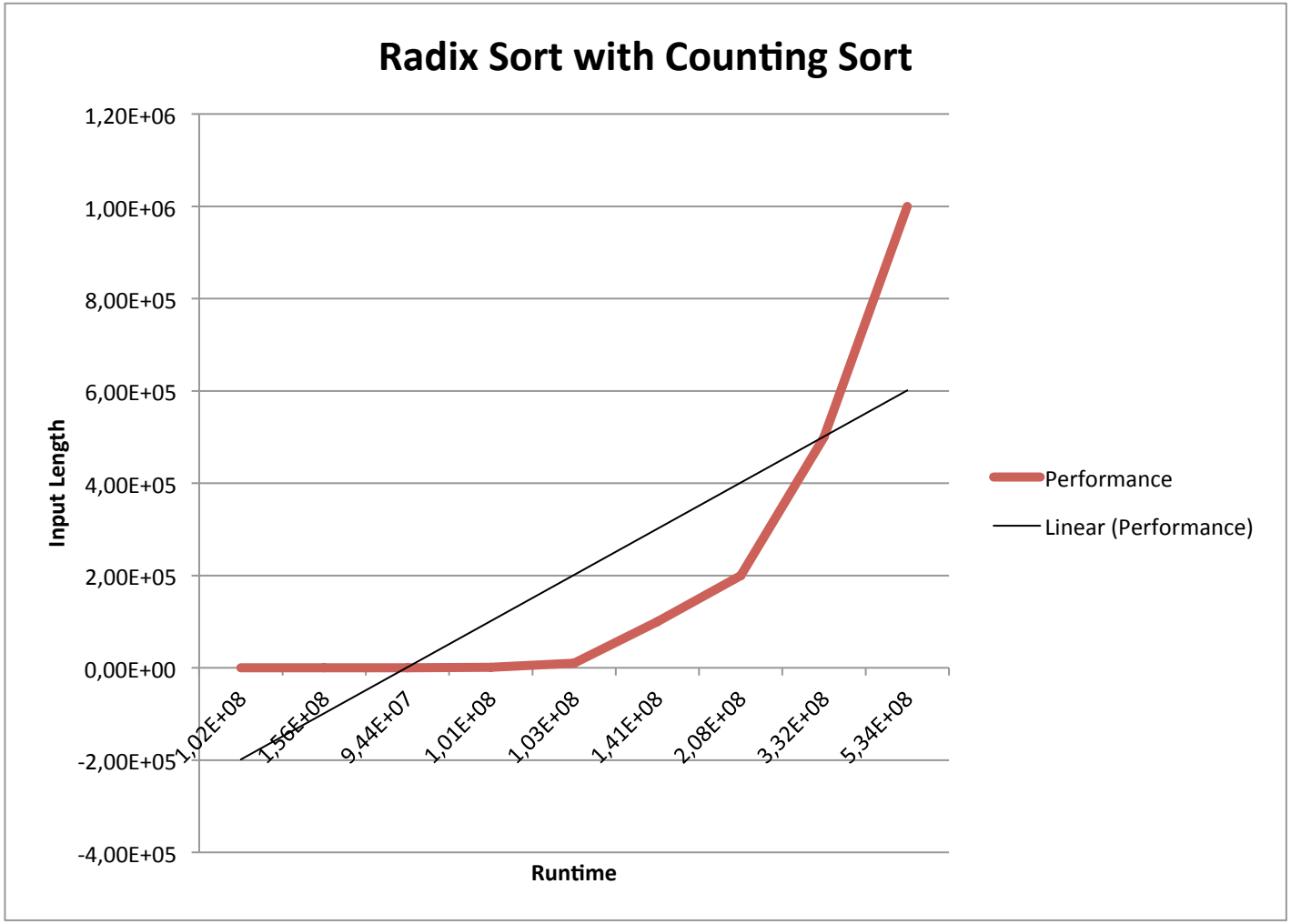
The radix sort needs to have a stable sorting algorithm in intermediate layer. To make algorithm work properly, we may either use gnome sort or freezing sort if we are sorting decremental.

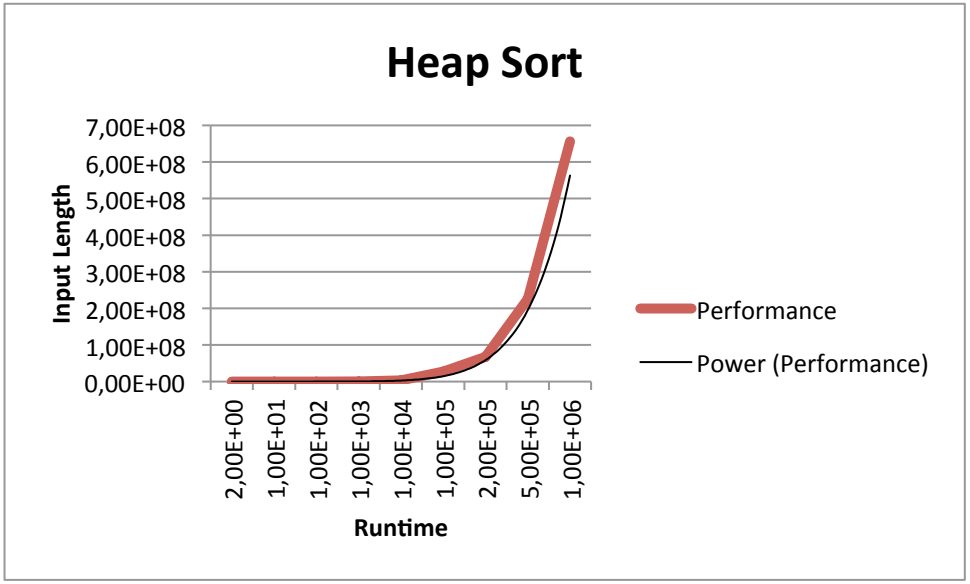
B. Heap Sort

1. The complexity we expected to obtain was $O(n\ lgn)$, which fits with the trend analysis of the results. The performance graph of the algorithm with respect to N over runtime is also included at the end of the report.

Heap Sort and Radix Sort based on Counting Sort

		Complexity	Benchmark nr.	Input Type	Input Length	Runtime			Runtime Difference Relative To Base Run	Theoretical Runtime Estimation	Deviation of Theoretical vs. Practical	Logarithmic Success Exponent
Units	Sub-SI	N/A	N/A	N/A	N/A	ns	s	min	N/A	N/A	N/A	N/A
	SI	N/A	N/A	N/A	N/A	sE-09	s	s * 60^(-1)	N/A	N/A	N/A	N/A
Heap Sort		*O(n lgn)*	1	uint_least64_t	2,00E+00	2,00E+00	2,00E-09	3,33E-11	N/A	N/A	N/A	N/A
			2		1,00E+01	1,46E+03	1,46E-06	2,44E-08	7,31E+02	6,64E+01	9,09E-02	-1,04E+00
			3		1,00E+02	9,48E+03	9,48E-06	1,58E-07	4,74E+03	9,71E+05	2,05E+02	2,31E+00
			4		1,00E+03	1,41E+05	1,41E-04	2,34E-06	7,03E+04	9,45E+07	1,34E+03	3,13E+00
			5		1,00E+04	3,08E+06	3,08E-03	5,13E-05	1,54E+06	1,87E+10	1,21E+04	4,08E+00
			6		1,00E+05	2,72E+07	2,72E-02	4,53E-04	1,36E+07	5,11E+12	3,76E+05	5,58E+00
			7		2,00E+05	6,76E+07	6,76E-02	1,13E-03	3,38E+07	9,58E+13	2,83E+06	6,45E+00
			8		5,00E+05	2,26E+08	2,26E-01	3,77E-03	1,13E+08	6,40E+14	5,66E+06	6,75E+00
			9		1,00E+06	6,56E+08	6,56E-01	1,09E-02	3,28E+08	4,51E+15	1,37E+07	7,14E+00
Radix Sort		*O(n k)*	10	uint_least64_t	2,00E+00	1,02E+08	1,02E-01	1,71E-03	N/A	N/A	N/A	N/A
			11		1,00E+01	1,56E+08	1,56E-01	2,60E-03	1,53E+08	5,10E+08	3,33E+00	5,22E-01
			12		1,00E+02	9,44E+07	9,44E-02	1,57E-03	9,26E+07	5,10E+09	5,51E+01	1,74E+00
			13		1,00E+03	1,01E+08	1,01E-01	1,69E-03	9,93E+07	5,10E+10	5,13E+02	2,71E+00
			14		1,00E+04	1,03E+08	1,03E-01	1,71E-03	1,01E+08	5,10E+11	5,06E+03	3,70E+00
			15		1,00E+05	1,41E+08	1,41E-01	2,36E-03	1,39E+08	5,10E+12	3,68E+04	4,57E+00
			16		2,00E+05	2,08E+08	2,08E-01	3,46E-03	2,04E+08	1,02E+13	5,01E+04	4,70E+00
			17		5,00E+05	3,32E+08	3,32E-01	5,53E-03	3,25E+08	2,55E+13	7,84E+04	4,89E+00
			18		1,00E+06	5,34E+08	5,34E-01	8,90E-03	5,23E+08	5,10E+13	9,74E+04	4,99E+00


Radix Sort with Counting Sort
1,20E+06
1,00E+06
8,00E+05
6,00E+05
4,00E+05
2,00E+05
0,00E+00
-2,00E+05
-4,00E+05
Input Length
1,02E+08
1,56E+08
9,44E+07
1,01E+08
1,03E+08
1,41E+08
2,08E+08
3,32E+08
5,34E+08
Runtime
Performance
Linear (Performance)


Heap Sort
7,00E+08
6,00E+08
5,00E+08
4,00E+08
3,00E+08
2,00E+08
1,00E+08
0,00E+00
Input Length
2,00E+00
1,00E+01
1,00E+02
1,00E+03
1,00E+04
1,00E+05
2,00E+05
5,00E+05
1,00E+06
Runtime
Performance
Power (Performance)