Fundamental Algorithms - Spring 2018
Homework 3

Daniel Rivera Ruiz
Department of Computer Science
New York University
drr342@nyu.edu

1.

$$2n^4 - 11n + 98 = \Theta(n^4)$$
$$6n + 43n\lg(n) = \Theta(n\lg(n))$$
$$63n^2 + 14n\lg^5(n) = \Theta(n^2)$$
$$3 + \frac{5}{n} = \Theta(1)$$

2. The following table shows the steps of `RADIX-SORT`:

Step 0	Step 1	Step 2	Step 3
COW	SE**A**	T**A**B	**B**AR
DOG	TE**A**	B**A**R	**B**IG
SEA	MO**B**	E**A**R	**B**OX
RUG	TA**B**	T**A**R	**C**OW
ROW	DO**G**	S**E**A	**D**IG
MOB	RU**G**	T**E**A	**D**OG
BOX	DI**G**	D**I**G	**E**AR
TAB	BI**G**	B**I**G	**F**OX
BAR	BA**R**	M**O**B	**M**OB
EAR	EA**R**	D**O**G	**N**OW
TAR	TA**R**	C**O**W	**R**OW
DIG	CO**W**	R**O**W	**R**UG
BIG	RO**W**	N**O**W	**S**EA
TEA	NO**W**	B**O**X	**T**AB
NOW	BO**X**	F**O**X	**T**AR
FOX	FO**X**	R**U**G	**T**EA

3. The following table shows the steps of `BUCKET-SORT`:

Array A	Array B (linked lists)	Array B (sorted lists)	Sorted array
1: 0.79	0: $\emptyset$	$\emptyset$	0.13
2: 0.13	1: 0.13 $\rightarrow$ 0.16 $\rightarrow \emptyset$	0.13 $\rightarrow$ 0.16 $\rightarrow \emptyset$	0.16
3: 0.16	2: 0.20 $\rightarrow \emptyset$	0.20 $\rightarrow \emptyset$	0.20
4: 0.64	3: 0.39 $\rightarrow \emptyset$	0.39 $\rightarrow \emptyset$	0.39
5: 0.39	4: 0.43 $\rightarrow \emptyset$	0.43 $\rightarrow \emptyset$	0.43
6: 0.20	5: 0.53 $\rightarrow \emptyset$	0.53 $\rightarrow \emptyset$	0.53
7: 0.89	6: 0.64 $\rightarrow \emptyset$	0.64 $\rightarrow \emptyset$	0.64
8: 0.53	7: 0.79 $\rightarrow$ 0.71 $\rightarrow \emptyset$	0.71 $\rightarrow$ 0.79 $\rightarrow \emptyset$	0.71
9: 0.71	8: 0.89 $\rightarrow \emptyset$	0.89 $\rightarrow \emptyset$	0.79
10: 0.43	9: $\emptyset$	$\emptyset$	0.89

4. (a) `COUNTING-SORT` takes time $O(\max(N, K))$, where N is the length of the array, and K is the maximum value an element $A[I]$ can take. Therefore:

$$T(n) = O(\max(N, K))$$
$$T(n) = O(\max(N, N^N))$$
$$T(n) = O(N^N)$$

(b) `RADIX-SORT` takes time $O(D\max(N, K))$, where N is the length of the array, and K^D is the maximum value an element $A[I]$ can take. Therefore:

$$T(n) = O(D\max(N, K))$$
$$T(n) = O(N\max(N, N))$$
$$T(n) = O(N^2)$$

(c) Finally, using `RADIX-SORT` with a different base:

$$T(n) = O(D\max(N, K))$$
$$T(n) = O(\sqrt{N}\max(N, N^{\sqrt{N}}))$$
$$T(n) = O(N^{\sqrt{N}+0.5})$$

5. The numbers in the sequence seem to coincide with the stops of the C, E, F, M trains, excepting for 13, which should actually be 14^{th} St. Under this assumption, the next number should be 59 for the C/E lines, 57 for the F, or 53 for the M.

6. (a) There are two nested `for` loops, each running from 1 to N. Since there is only one operation to be performed (i.e. `x++`), each inner loop will take time $O(N)$, and there are N such loops:

$$T(N) = O(N) \cdot N$$
$$T(N) = O(N^2)$$

(b) There is only one operation inside the `while` loop, which duplicates the index I each time. At the K^{th} iteration, we will have $I = 2^K$, and therefore the algorithm will run as long as $2^K < N \rightarrow K < \lg(N)$:

$$T(N) = O(\lg N)$$

(c) The inner `while` loop will run as long as $J^2 < I$ and the value of J will be increased by one inside of it. Under these conditions, let's use the following table to explore the number of times the inner loop will be executed as I grows:

I	$\max(J)$ such that $J^2 < I$	Iterations in `while`
1	0	0
4	1	$1 \cdot (4-1)$
9	2	$2 \cdot (9-4)$
16	3	$3 \cdot (16-9)$
...	...	...

Looking at the table, it is clear that the total amount of operations will be the sum of all the iterations such that $I \leq N \rightarrow J^2 < N \rightarrow J < \sqrt{N}$:

$$1 \cdot (2^2 - 1^2) + 2 \cdot (3^2 - 2^2) + 3 \cdot (4^2 - 3^2) + \ldots = \sum_{J=1}^{\sqrt{N}} \left((J-1)(J^2 - (J-1)^2)\right)$$

Simplifying the summation we get $\sum_{J=1}^{\sqrt{N}} \left(2J^2 - 3J + 1\right)$. Since we will be using O notation, the only term that we care about is the quadratic one. Considering that $\sum_{i=1}^{n}(i^2) = O(n^3)$,

we can finally conclude:

$$T(N) = O\left(\sum_{J=1}^{\sqrt{N}} J^2\right)$$
$$T(N) = O\left(N^{\frac{3}{2}}\right)$$

(d) At the K^{th} iteration of the inner `while` loop, the condition $J = 2^K I < N$ must be met. Therefore, for each value of $I = 1 \dots N$ there will be $K = \lg\left(\frac{N}{I}\right)$ iterations. With this, we can calculate the total running time of the algorithm as follows:

$$T(N) = O\left(\sum_{I=1}^{N} \lg\left(\frac{N}{I}\right) = \sum_{I=1}^{N} (\lg N - \lg I) = \sum_{I=1}^{N} \lg N - \sum_{I=1}^{N} \lg I\right)$$
$$T(N) = O\left(N\lg N - \lg\left(\prod_{I=1}^{N} I\right)\right)$$
$$T(N) = O\left(N\lg N - \lg(N!)\right)$$

From Stirling's formula we know that $\lg(N!) \sim N\lg N$, and therefore $T(N) = O(N\lg N)$.

7. (a) Professor Squander's selection for the number of buckets is not ideal. With n^2 buckets, the last step of `BUCKET-SORT` where all the linked lists are concatenated will take $O(n^2)$ time, since *all* the lists must be visited once.

(b) In the case of Ima, the concatenation step will actually be quite fast since there are only $\sqrt{n}$ buckets, but the problem will arise when sorting the elements in each one of the buckets. Considering that there are n elements to sort in total, the expected value for the number of elements in each bucket will be $E = \sqrt{n}$, and we can use a Poisson distribution with $\lambda = \sqrt{n}$ to estimate the running time of the sorting process for all the buckets.
The expected number of buckets N with m elements in them is given by the following equation:

$$N = \sqrt{n} \cdot \frac{(\sqrt{n})^m e^{-\sqrt{n}}}{m!} = \frac{(\sqrt{n})^{m+1} e^{-\sqrt{n}}}{m!}$$

According to the problem definition, each one of these buckets will take $O(m^2)$ time to be sorted. To calculate the overall time of the algorithm, we add an m^2 factor to the previous equation and sum over all values of m[1]:

$$T(n) = O\left(\sum_{m=0}^{\infty} \frac{(\sqrt{n})^{m+1} e^{-\sqrt{n}} m^2}{m!}\right)$$
$$T(n) = O\left(n^{\frac{3}{2}} + n\right)$$
$$T(n) = O\left(n^{\frac{3}{2}}\right)$$

Ima's selection of the number of buckets yields better results than that of the professor ($O(n^{\frac{3}{2}})$ vs. $O(n^2)$), but still it is worse than the optimal value of $O(n)$ achieved when the number of buckets equals the number of elements to be sorted.

(c) The amount of space S used by the algorithm is given by multiplying the number of buckets N by the expected number of elements in each bucket E.
For the case with n buckets we have:

$$S = n \cdot 1 = n$$

And for the case with $\sqrt{n}$ buckets:

$$S = \sqrt{n} \cdot \sqrt{n} = n$$

Therefore, the amount of space used by both algorithms will be roughly the same.

[1] *Wolfram Mathematica* was used to reduce the resulting summation.