Fundamental Algorithms, Problem Set 3 Solutions

1. Write each of the following functions as $\Theta(g(n))$ where $g(n)$ is one of the standard forms: $2n^3 - 11n + 98$; $6n + 43n \lg n$; $63n^2 + 14n \lg^5 n$; $3 + \frac{5}{n}$
   `Solution:`In order, $\Theta(n^4), \Theta(n \lg n), \Theta(n^2), \Theta(1)$.

2. Illustrate the operation of `RADIX-SORT` on the list: COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX following the Figure in the Radix-Sort section. (Use alphabetical order and sort one letter at a time.)
   `Solution:`From left to right:

COW	SEA	TAB	BAR
DOG	TEA	BAR	BIG
SEA	MOB	EAR	BOX
RUG	TAB	TAR	COW
ROW	DOG	SEA	DIG
MOB	RUG	TEA	DOG
BOX	DIG	DIG	EAR
TAB	BIG	BIG	FOX
BAR	BAR	MOB	MOB
EAR	EAR	DOG	NOW
TAR	TAR	COW	ROW
DIG	COW	ROW	RUG
BIG	ROW	NOW	SEA
TEA	NOW	BOX	TAB
NOW	BOX	FOX	TAR
FOX	FOX	RUG	TEA

3. Illustrate the operation of `BUCKET-SORT` on the array
   $A = (.79, .13, .16, .64, .39, .20, .89, .53, .71, .43)$
   following the Figure in the Bucket-Sort section.
   `Solution:`We first construct the auxiliary list B:

$\emptyset$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\emptyset$
	.13	.20	.39	.43	.53	.64	.79	.89	
	$\downarrow$						$\downarrow$		
	.16						.71		

We then sort each bucket with any sort, giving us:

$\emptyset$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\emptyset$
	.13	.20	.39	.43	.53	.64	.71	.89	
	$\downarrow$						$\downarrow$		
	.16						.79		

Finally, we concatenate each of the buckets together in order:

.13	.16	.20	.39	.43	.53	.64	.71	.79	.89

4. Given $A[1 \cdots N]$ with $0 \leq A[I] < N^N$ for all I.

   (a) How long will `COUNTING-SORT` take?
   `Solution:`$\Theta(N^N)$ since you have to go through array C of length N^N. If you wrote $O(N^N + N)$ it is not technically wrong but it misses the point. In asymptotics we ignore the lower order terms to put things in the right form. Thus we want to write $O(N^N)$, in this case $\Theta(N^N)$ since you do indeed need to go through array C.

   (b) How long will `RADIX-SORT` take using base N?
   `Solution:`Now C has length N do for each digit this is a linear sort, $\Theta(N)$. However, there are N digits so that the total time is $\Theta(N^2)$.

   (c) How long will `RADIX-SORT` take using base $N^{\sqrt{N}}$? (Assume $\sqrt{N}$ integral.)
   `Solution:`There are now $\sqrt{N}$ digits, not too bad, but each digit takes time $N^{\sqrt{N}}$ as that is the length of C so the total time is $\Theta(\sqrt{N}N^{\sqrt{N}})$ which is awful.

   (d) (*) Argue that for no base K will `RADIX-SORT` do as well as `HEAP-SORT`.
   `Solution:`Here is one way. If the base Khas $K \geq N^2$ then it takes time N^2 or more just to go through the array C, which is worse than the $\Theta(N \lg N)$ `HEAP-SORT`. On the other side, if the base K has $K \leq N^2$ then the number of digits is at least $N/2$ (the smaller the base, the greater the number of digits and with base N^2 there are $N/2$ digits) and as each digit takes linear time the total time is $\Omega(N^2)$ which is worse than `HEAP-SORT`.

5. `Just For Fun:` What is the next term is the sequence

$$4, 14, 23, 34, 42, 50, \ldots$$

`Solution:`Its the A Train! Next stop, Columbus Circle.

6. Write the time $T(N)$ (don't worry about the output!) for the following algorithms in the form $T(N) = \Theta(g(N))$ for a standard $g(N)$. *Assume* (for this problem only!) that addition and multiplication take one time unit.

(a)
```
X=0
FOR I=1 TO N
        do FOR J=1 TO N
                X ++
```
`Solution:`Double-loop, time $\Theta(N^2)$.

(b)
```
I=1
WHILE I < N
        do I = 2*I
```
`Solution:`Only $\lg N$ doublings to get to N and each doubling (by our assumption) takes one time unit so total time $\Theta(\lg N)$.

(c)
```
FOR I=1 TO N
        do J=1
        WHILE J*J < I
                do J=J+1
```
`Solution:`For each I the subloop takes $O(\sqrt{I})$ as after that $J * J > I$. So we need look at $\sqrt{1} + \ldots + \sqrt{N}$. This is at most $N\sqrt{N}$ as there are N terms and each is at most $\sqrt{N}$. As a lower bound cut it off at the middle. (This often works!) We have $\sqrt{N/2} + \ldots + \sqrt{N}$. There are $N/2$ terms here and each is at least $\sqrt{N/2}$ so the total is at least $N\sqrt{N}/(2\sqrt{2})$. In Theta-World (as your lecturer likes to call it) constants don't count so the answer is $\Theta(N^{3/2})$.

(d)
```
FOR I = 1 to N
        J=I
        WHILE J < N
                do J=2*J
```
`Solution:`For a given I the subloop starts at I and double until

reaching N. We double t times, where t is the smallest integer $I2^t \geq N$, so $t = \lceil \lg(N/I) \rceil$. So the total time is

$$TOTAL = O\left(\sum_{i=1}^{n} \lceil \lg(n/i) \rceil\right) \tag{1}$$

This is challenging.
`Approach One:` We "get rid" of the ceiling by noting that the ceiling can only affect each term by 1 and therefore the sum by at most n and so

$$TOTAL = O(n) + O\left(\sum_{i=1}^{n} \lg(n/i)\right) \tag{2}$$

Now there are a variety of approaches. One is via Stirling's Formula. The object in parentheses is precisely $\ln(n^n/n!)$ and by Stirling $n^n/n! \sim e^n(2\pi n)^{-1/2}$. The square root term is inconsequential and $\lg(n^n/e^n) \sim n \lg 2 = O(n)$. Thus

$$TOTAL = O(n) + O(n) = O(n) \tag{3}$$

this is a linear time algorithm! Note that the ceiling actually does have an effect on the constants buried in the O as both parts are linear in n. `Comment:` How did we know that removing the ceilings would work? We didn't, we tried it and it turned out its effect was not dominant so we were OK. This is part of the *art* of doing asymptotics!
`Approach Two:` Similar to the analysis of BUILD-MAX-HEAP we have 1 doubling $n/2$ times, 2 doublings $n/4$ times ($n/4 < i \leq n/2$), 3 doublings $n/8$ times so the total doublings is $n \sum_u u 2^{-u}$ but that sum, even to infinity, is 2 so the total doublings is $\sim 2n$.

7. Prof. Ligate decides to do Bucket Sort on n items with n^2 buckets while his student Ima Hogg decides to do Bucket Sort on n items with $n^{1/2}$ buckets. Assume that the items are indeed uniformly distributed. Assume that Ima's algorithm for sorting inside a bucket takes time $O(m^2)$ when the bucket has m items.

   (a) Argue that Prof. Ligate has made a poor choice of the number of buckets by looking analyzing the time of Bucket Sort in his case. `Solution:`The time will be $O(n^2)$ since one has to pass through the buckets to link them up. Note that even though most of the buckets are empty you don't know which one's are empty so you have to check each one.

(b) Argue that Ima has made a poor choice of the number of buckets by looking analyzing the time of Bucket Sort in her case.
`Solution:`Lets say each of the $n^{1/2}$ buckets had around $n^{1/2}$ items. (Indeed, with high probability that will be the case.) Then sorting each bucket (under our assumption) takes $O((n^{1/2})^2) = O(n)$ so the total time for bucket sorting would be $O(n^{3/2})$.

(c) Argue that Ima uses roughly the same amount of *space* as someone using n buckets.
`Solution:`She didn't save any space. While the array is only of length $n^{1/2}$ she has all the items in the linked lists in the array so the total space used up by the array is still $\Theta(n)$.

If you take a number and double it and double it again and then double it a few more times, the number gets bigger and bigger and goes higher and higher and only arithmetic can tell you what the number is when you quit doubling.
from *Arithmetic* by Carl Sandburg