COMS 311: Homework 2
Due: Feb 23, 11:59pm
Total Points: 50

Late submission policy. Any assignment submission that is late by not more than two business days from the deadline will be accepted with 20% penalty for each business day. That is, if a homework is due on Friday at 11:59 PM, then a Monday submission gets 20% penalty and a Tuesday submission gets another 20% penalty. After Tuesday no late submissions are accepted.

Submission format. Homework solutions will have to be typed. You can use word, LaTeX, or any other type-setting tool to type your solution. Your submission file should be in pdf format. Do **NOT** submit a photocopy of handwritten homework except for diagrams that can be hand-drawn and scanned. We reserve the right **NOT** to grade homework that does not follow the formatting requirements. Name your submission file: `<Your-net-id>-311-hw2.pdf`. For instance, if your netid is `asterix`, then your submission file will be named `asterix-311-hw2.pdf`. Each student must hand in their own assignment. If you discussed the homework or solutions with others, a list of collaborators must be included with each submission. Each of the collaborators has to write the solutions in their own words (copies are not allowed).

General Requirements

- When proofs are required, do your best to make them both clear and rigorous. Even when proofs are not required, you should justify your answers and explain your work.
- When asked to present a construction, you should show the correctness of the construction.

Some Useful (in)equalities

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- $2^{\log_2 n} = n$, $a^{\log_b n} = n^{\log_b a}$, $n^{n/2} \leq n! \leq n^n$, $\log x^a = a \log x$
- $\log(a \times b) = \log a + \log b$, $\log(a/b) = \log a - \log b$
- $a + ar + ar^2 + ... + ar^{n-1} = \frac{a(r^n-1)}{r-1}$
- $1 + \frac{1}{2} + \frac{1}{2^2} + ... + \frac{1}{2^n} = 2(1 - \frac{1}{2^{n+1}})$
- $1 + 2 + 4 + ... + 2^n = 2^{n+1} - 1$

---

1. (**10 pts**) Using a table size of 11, and a hash function $h(x) = 3x + 2$ for positive integers, draw the resulting hash table array when the integers

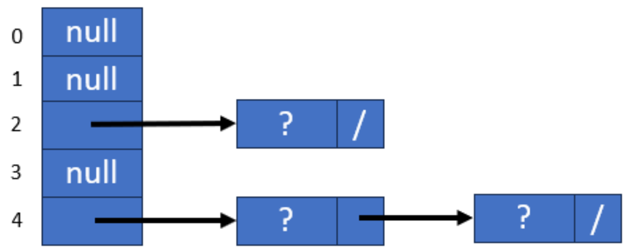
$$\mathbf{1, 17, 10, 15, 14, 43, 4, 21, 23, 24, 19, 41, 42, 9, 44}$$

are added to an empty hash table in this order. Note that for this hash function, x is stored in the table at index $h(x)$ % n, where n is the table size.

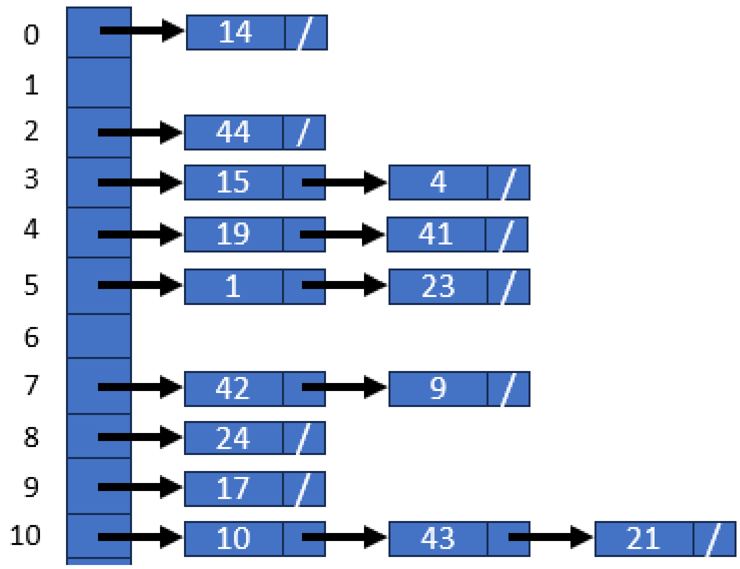
(a) Using a chaining hash table that is not enlarged when elements are added.

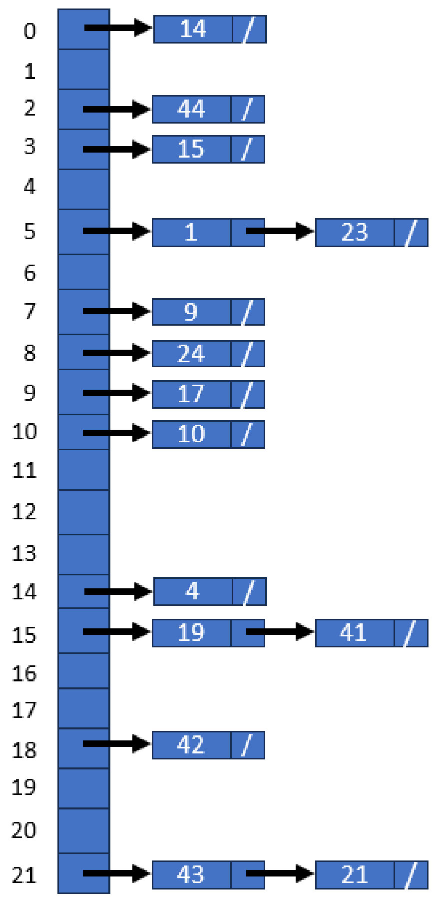
(b) Using a chaining hash table that is doubled in size when the table reaches a load factor of $\alpha > 0.75$.

An example drawing of a hash table of table size 5 with 3 elements is shown below.



x	$h(x)$	$h(x)\%11$	$h(x)\%22$
1	5	5	5
17	53	9	9
10	32	10	10
15	47	3	3
14	44	0	0
43	131	10	21
4	14	3	14
21	65	10	21
23	71	5	5
24	74	8	8
19	59	4	15
41	125	4	15
42	128	7	18
9	29	7	7
44	134	2	2


0
1
2
3
4
5
6
7
8
9
10
14
44
15
4
19
41
1
23
42
9
24
17
10
43
21
(a)


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
14
44
15
1
23
9
24
17
10
4
19
41
42
43
21
(b)

2. (**10 pts**) Consider the following recurrence relation:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 7 \cdot T(\frac{n}{2}) + 1 & \text{if } n > 1 \end{cases}$$

(a) Use the Master theorem to show that $T(n) \in \Theta(n^{\log_2(7)})$.

(b) Use induction to prove that $T(n) = \frac{1}{6}(7n^{\log_2(7)} - 1)$.

(a) $T(n) = 7 \cdot T(\frac{n}{2}) + 1$ with $a = 7$, $b = 2$, and $f(n) = 1$ as in the Master Theorem. Since $f(n) \in O(n^{\log_2 7 - \varepsilon})$ for $\varepsilon = 1$, $T(n) \in \Theta(n^{\log_2 7})$.

(b) We prove $T(n) = \frac{1}{6}(7n^{\log_2 7} - 1)$ is the solution to the recurrence above by induction.

Base case: $T(1) = \frac{1}{6}(7n^{\log_2 7} - 1) = \frac{1}{6}(7 - 1) = \frac{1}{6}(6) = 1$.

Induction Step: Assume $T(K) = \frac{1}{6}(7K^{\log_2 7} - 1)$ for $K < n$ where K and n are powers of 2. We show the result for $T(n)$:

$$\begin{aligned} T(n) &= 7 \cdot T(\frac{n}{2}) + 1 \\ &= 7(\frac{1}{6}(7\left[\frac{n}{2}\right]^{\log_2 7} - 1)) + 1 \quad \text{By Induction Hypothesis} \\ &= \frac{7}{6}(7\left[\frac{n}{2}\right]^{\log_2 7} - 1) + 1 \\ &= \frac{7}{6}(7\left[\frac{n^{\log_2 7}}{2^{\log_2 7}}\right] - 1) + 1 \\ &= \frac{7}{6}(\frac{7}{7}n^{\log_2 7} - 1) + 1 \\ &= \frac{7}{6}(n^{\log_2 7} - 1) + 1 \\ &= \frac{7}{6}n^{\log_2 7} - \frac{7}{6} + \frac{6}{6} \\ &= \frac{7}{6}n^{\log_2 7} - \frac{1}{6} \\ &= \frac{1}{6}(7n^{\log_2 7} - 1) \end{aligned}$$

3. (**10 pts**) Without using the Master Theorem, give the asymptotic upper bounds on the following recurrence relations. Note that methods found in chapter 4 of the text

may be useful.

(a) $T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 4 \cdot T(\frac{n}{2}) + n & \text{if } n \geq 2 \end{cases}$

(b) $T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 2 \cdot T(\frac{n}{2}) + n \log n & \text{if } n \geq 2 \end{cases}$

(c) $T(n) = \begin{cases} 1 & \text{if } n < 3 \\ T(\frac{n}{3}) + n & \text{if } n \geq 3 \end{cases}$

(d) $T(n) = \begin{cases} 1 & \text{if } n < 3 \\ 9 \cdot T(\frac{n}{3}) + n^{2.5} & \text{if } n \geq 3 \end{cases}$

(e) $T(n) = \begin{cases} 1 & \text{if } n < 3 \\ T(n-2) + n^2 & \text{if } n \geq 3 \end{cases}$

(a)

Level	Problem Size	# Nodes	Work / Node	Level Total
1	$\frac{n}{2^0}$	$4^0 = 1$	n	n
2	$\frac{n}{2^1}$	4^1	$\frac{n}{2^1}$	$2^1 n$
...	...	...	...	...
i	$\frac{n}{2^{i-1}}$	4^{i-1}	$\frac{n}{2^{i-1}}$	$2^{i-1} n$

Assuming at level l when problem size $= 1$, then

$$\frac{n}{2^{l-1}} = 1 \Rightarrow l = \log_2(n) + 1$$

So for all i from 1 to $\log_2(n)$, the work done at level i is $2^{i-1}n$. At level $\log_2(n)+1$, the work done at each node is 1, and there are $4^{\log_2(n)+1-1} = n^2$ nodes, so total work at level $\log_2(n) + 1$ is n^2.

$$\begin{aligned}
\text{Total cost} &= \left[\sum_{i=1}^{\log_2(n)} 2^{i-1} n\right] + n^2 \\
&= \frac{n}{2}\left[\sum_{i=1}^{\log_2(n)} 2^i\right] + n^2 \\
&= \frac{n}{2}\left[\frac{2^{\log_2(n)+1} - 1}{2 - 1} - 1\right] + n^2 \\
&= \frac{n}{2}\left[2n - 1\right] + n^2 \\
&= 2n^2 - \frac{n}{2} \\
&\in O(n^2)
\end{aligned}$$

(b)

Level	Problem Size	# Nodes	Work / Node	Level Total
1	$\frac{n}{2^0}$	$2^0 = 1$	$n\log(n)$	$n\log n$
2	$\frac{n}{2^1}$	2^1	$\frac{n}{2^1}\log(\frac{n}{2^1})$	$n\log(\frac{n}{2^1})$
...	...	...	...	...
i	$\frac{n}{2^{i-1}}$	2^{i-1}	$\frac{n}{2^{i-1}}\log(\frac{n}{2^{i-1}})$	$n\log(\frac{n}{2^{i-1}})$

Assuming at level l when problem size = 1, then

$$\frac{n}{2^{l-1}} = 1 \Rightarrow l = \log_2(n) + 1$$

So for all i from 1 to $\log_2(n)$, the work done at level i is $n\log_2(\frac{n}{2^{i-1}})$. At level $\log_2(n)+1$, the work done at each node is 1, and there are $2^{\log_2(n)+1-1} = n$ nodes, so total work at level $\log_2(n)+1$ is n.

$$\begin{aligned}
\text{Total cost} &= \left[\sum_{i=1}^{\log_2(n)} n\log_2(\frac{n}{2^{i-1}})\right] + n \\
&= \left[\sum_{i=1}^{\log_2(n)} n(\log_2(n) - \log_2(2^{i-1}))\right] + n \\
&= \left[\sum_{i=1}^{\log_2(n)} n\log_2(n)\right] - \left[\sum_{i=1}^{\log_2(n)} n(i-1)\right] + n \\
&= n\log^2(n) - n\left[\sum_{i=1}^{\log_2(n)} i - \sum_{i=1}^{\log_2(n)} 1\right] + n \\
&= n\log^2(n) - n\left[\frac{\log_2(n)(\log_2(n)+1)}{2} - \log_2(n)\right] + n \\
&= n\log^2(n) - \frac{n\log^2(n)}{2} - \frac{n\log_2(n)}{2} + n\log_2(n) + n \\
&= \frac{n\log^2(n)}{2} + \frac{n\log_2(n)}{2} + n \\
&\in O(n\log^2(n))
\end{aligned}$$

(c)

Level	Problem Size	# Nodes	Work / Node	Level Total
1	$\frac{n}{3^0}$	1	n	n
2	$\frac{n}{3^1}$	1	$\frac{n}{3^1}$	$\frac{n}{3^1}$
...	...	...	...	...
i	$\frac{n}{3^{i-1}}$	1	$\frac{n}{3^{i-1}}$	$\frac{n}{3^{i-1}}$

Assuming at level l when problem size = 3, then

$$\frac{n}{3^{l-1}} = 3 \Rightarrow l = \log_3(n)$$

So for all i from 1 to $\log_3(n)$, the work done at level i is $\frac{n}{3^{i-1}}$. At level $\log_3(n)+1$, the work done at each node is 1, and there is 1 node, so total work at level $\log_3(n)+1$ is 1.

$$\begin{aligned}
\text{Total cost} &= \left[\sum_{i=1}^{\log_3(n)} \frac{n}{3^{i-1}}\right] + 1 \\
&= n \cdot \left[\sum_{i=1}^{\log_3(n)} \frac{1}{3^{i-1}}\right] + 1 \\
&= n \cdot \left[\frac{-3+3n}{2n}\right] + 1 \quad \text{[Geometric Series]} \\
&= \frac{3}{2}n - \frac{1}{2} \\
&\in O(n)
\end{aligned}$$

(d)

Level	Problem Size	# Nodes	Work / Node	Level Total
1	$\frac{n}{3^0}$	$9^0 = 1$	$n^{2.5}$	$n^{2.5}$
2	$\frac{n}{3^1}$	9^1	$(\frac{n}{3^1})^{2.5}$	$\frac{n^{2.5}}{\sqrt{3}^1}$
3	$\frac{n}{3^2}$	9^2	$(\frac{n}{3^2})^{2.5}$	$\frac{n^{2.5}}{\sqrt{3}^2}$
...	...	...	...	...
i	$\frac{n}{3^{i-1}}$	9^{i-1}	$(\frac{n}{3^{i-1}})^{2.5}$	$\frac{n^{2.5}}{\sqrt{3}^{i-1}}$

Assuming at level l when problem size $= 3$, then

$$\frac{n}{3^{l-1}} = 3 \Rightarrow l = \log_3(n)$$

So for all i from 1 to $\log_3(n)$, the work done at level i is $\frac{n^{2.5}}{\sqrt{3}^{i-1}}$. At level $\log_3(n)+1$, the work done at each node is 1, and there is $9^{\log_3(n)+1-1} = n^2$ nodes, so total work at level $\log_3(n)+1$ is n^2.

$$\begin{aligned}
\text{Total cost} &= \left[\sum_{i=1}^{\log_3(n)} \frac{n^{2.5}}{\sqrt{3}^{i-1}}\right] + n^2 \\
&= n^{2.5}\left[\sum_{i=1}^{\log_3(n)} \frac{1}{\sqrt{3}^{i-1}}\right] + n^2 \\
&= n^{2.5}\left[-\frac{\sqrt{3}}{2}n^{-\frac{1}{2}} + \frac{\sqrt{3}}{2} - \frac{3}{2}n^{-\frac{1}{2}} + \frac{3}{2}\right] + n^2 \\
&= -\frac{\sqrt{3}}{2}n^2 + \frac{\sqrt{3}}{2}n^{2.5} - \frac{3}{2}n^2 + \frac{3}{2}n^{2.5} + n^2 \\
&\in O(n^{2.5})
\end{aligned}$$

(e) This recurrence relation decreases n by 2 until it reaches 1 or 2. Thus, the recurrence is bounded by $\frac{n}{2}+1$ in depth. Each recurrence costs n^2 work, so we guess that a bound is n^3, and hence $T(n) \in \Theta(n^3)$. This is easily checked by a routine induction.

4. (**10 pts**) If possible, use the Master theorem to give bounds on the following recurrence relations.

$$\text{(a)}\ T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 4 \cdot T(\frac{n}{2}) + n & \text{if } n \geq 2 \end{cases}$$

$$\text{(b)}\ T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 2 \cdot T(\frac{n}{2}) + n \log n & \text{if } n \geq 2 \end{cases}$$

$$\text{(c)}\ T(n) = \begin{cases} 1 & \text{if } n < 3 \\ 3 \cdot T(\frac{n}{3}) + n & \text{if } n \geq 3 \end{cases}$$

$$\text{(d)}\ T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 2 \cdot T(\frac{n}{4}) + \sqrt{n} & \text{if } n \geq 2 \end{cases}$$

(a) $n^{\log_b a} = n^{\log_2 4} = n^2$ and $f(n) = n \in O(n^{2-\varepsilon})$ for $\varepsilon = 0.5$, so $T(n) \in \Theta(n^2)$. (case 1)

(b) $n^{\log_b a} = n^{\log_2 2} = n$ and $f(n) = n \log^1 n$. Since $f(n) = \Theta(n^{\log_b a} \log^k n)$ for $k = 1$, case 2 of Master Theorem applies. Therefore, $T(n) = \Theta(n \log^2 n)$.

(c) $n^{\log_b a} = n^{\log_3 3} = n$ with $f(n) = n$. This is not case 1 since $f(n) \notin O(n^{1-\varepsilon})$ for $\varepsilon > 0$. For case 2, $f(n) = n = n^{\log_b a}$, so $T(n) \in \Theta(n \log n)$.

(d) $n^{\log_b a} = n^{\log_4 2} = n^{\frac{1}{2}}$ with $f(n) = n^{\frac{1}{2}}$. $f(n) \notin O(n^{\frac{1}{2}-\varepsilon})$, so not case 1. $f(n) \in \Theta(n^{\frac{1}{2}})$, so case 2 applies. Thus, $T(n) \in \Theta(\sqrt{n}\log n)$.

5. (**10 pts**) The algorithm below computes nothing useful. It takes as a parameter an array A of integers. Note that $A.length$ returns the length of the array, and $A.sub(s, l)$ returns a new array (with elements copied) of length l with values copied from A starting at index s.

Algorithm 1 Wacky(A)

```
1: if A.length == 1 then
2:     A[0] += 1;
3: else
4:     int m = ⌊A.length/2⌋;
5:     Wacky (A.sub(0, m));
6:     Wacky (A.sub(m, m));
7:     Wacky (A.sub(0, 1));
```

(a) Write a recurrence relation that gives the running time of **Wacky**.

(b) Use the Master theorem to give a bound on the running time in terms of n.

(a) We assume the following cost for each line of code:

- Line 1 takes 1 step.
- Line 2 takes 1 step.
- Line 4 takes 1 step.
- Line 5: call is 1 step + recursion.
- Line 6: call is 1 step + recursion.
- Line 7: call is 1 step + recursion, is terminal case = 2 steps (3 steps total)

Thus, a recurrence relation for the running time is bounded by:

$$T(n) = \begin{cases} 2 & \text{if } n = 1 \\ 2 \cdot T(\frac{n}{2}) + c & \text{otherwise} \end{cases}$$

where c is some constant.

(b) $n^{\log_2 2} = n$ with $f(n) = c$ where c is a constant, so case 1 applies (with $\varepsilon = 1$) and $T(n) = \Theta(n)$.