Kushagr,Gupta & Rayan,Dasoriya
`kgupta7,rdasori@ncsu.edu`

December 31, 2019

1. [8 points] *Purpose: Understanding asymptotic notation*

Consider the list of nine functions below:

$$n^{\lg n} \quad (\lg n)^2 \quad n! \quad 2^n \quad n^{0.0001} \quad n^n \quad (n-1)! \quad n^2 \quad 2^{(\lg n)^2}$$

Sort them into a sequence $g_1(n), \ldots, g_{10}(n)$ so that for $i = 1, \ldots, 8$, you have either $g_i \in \mathrm{o}(g_{i+1})$ or $g_i \in \Theta(g_{i+1})$. Prove all eight relationships using what you know about little-oh and big-theta.

1 For $\lg^2 n$ and $n^{0.0001}$

$\because$ For any constant a>0

$$\lg^b n = \mathrm{o}(n^a)$$

$$\therefore \lg^2 n = \mathrm{o}(n^{0.0001})$$

2 For n^2

$$\lim_{n\to\infty} \frac{n^{0.0001}}{n^2} = 0$$

$$\therefore n^{0.0001} = \mathrm{o}(n^2)$$

3 For $n^{\lg n}$

$$\lim_{n\to\infty} \frac{n^2}{n^{\lg n}} = 0 \because \lim_{n\to\infty} \frac{2}{\lg n} = 0$$

$$\therefore n^2 = \mathrm{o}(n^{\lg n})$$

4 For $2^{\lg^2 n}$

$$\because n^{\log_2 n} = n^{\log_2 n * \log_2 2} = 2^{\lg^2 n}$$

$$\therefore 2^{\lg^2 n} = \Theta(n^{\lg n})$$

5 For 2^n

$$\because \lg^b n = \mathrm{o}(n^a)$$

$$\lim_{n\to\infty} \frac{2^{\lg^2 n}}{2^n} = 0$$

$$\therefore n^{\lg n} = \mathrm{o}(2^n)$$

6 For $(n-1)!$

$$\lim_{n\to\infty} \frac{2^n}{(n-1)!} = \lim_{n\to\infty} \frac{n * 2^n}{(n)!} = \lim_{n\to\infty} \frac{2^{n+\lg n}}{(n)!}$$

By Sterling's approximation for all n >1,

$$n! = \sqrt{2\pi n} \left\{\frac{n}{e}\right\}^n e^{\alpha_n}$$

$$\lim_{n\to\infty} \frac{2^{n+\lg n}}{(n)!} = 0$$

$$\because \lim_{n\to\infty} \frac{2^{n+\lg n}}{\left\{\frac{n}{e}\right\}^n} = 0$$

$$\therefore 2^n = \mathrm{o}((n-1)!)$$

7 For $(n)!$

$$\lim_{n\to\infty} \frac{(n-1)!}{n!} = \lim_{n\to\infty} \frac{1}{n} = 0$$

$$\therefore (n-1)! = \mathrm{o}(n!)$$

8 For n^n

$$\because n! = \sqrt{2\pi n} \left\{\frac{n}{e}\right\}^n e^{\alpha_n}$$

$$\lim_{n\to\infty} \frac{n!}{n^n} = \lim_{n\to\infty} \frac{c}{e^n} = 0$$

$$\therefore n! = \mathrm{o}(n^n)$$

Also, by transitivity
If $f(n) = \mathrm{o}(g(n))$ and $g(n) = \mathrm{o}(h(n))$
$f(n) = \mathrm{o}(h(n)$
Hence,

$$\lg^2 n < n^{0.0001} < n^2 < n^{\lg n} = 2^{\lg^2 n} < 2^n < (n-1)! < n! < n^n$$

2. [6 points] *Purpose: Understanding asymptotic notation in the context of summations*

   Let

$$f(n) = \sum_{i=1}^{n} i^d (\lg i)^m$$

   where d is any real number > 0 and m is an integer ≥ 1. Give a simple (one term) function $g(n)$ such that $f(n) \in \Theta(g(n))$. Prove your answer using the definitions of big-oh, big-omega, anything you have learned about bounds on summations and, if appropriate, limits.

   By gross upper bound on value of summation

$$\sum_{i=1}^{n} F(i) \leq n * F(n)$$

   where F(i) = $i^d(\lg i)^m$

   and n*F(n) = $n^{d+1}(\lg n)^m$

   hence $\sum_{i=1}^{n} F(i) = O(n^{d+1}(\lg n)^m)$ with c1 = 1

   By gross lower bound on value of summation

$$\sum_{i=1}^{n} F(i) \geq (n/2) * F(n/2)$$

   where F(i) = $i^d(\lg i)^m$

   and (n/2)*F(n) = $(n/2)^{d+1}(\lg n/2)^m = (n/2)^{d+1}(\lg n - 1)^m$

   expanding $(\lg n - 1)^m = (\lg n)^m \ldots \pm 1$

   hence $\sum_{i=1}^{n} F(i) = \Omega(n^{d+1}(\lg n)^m)$ with c2 = 0.5^{d+1}

   Now,

$$f(n) = \Omega(n^{d+1}(\lg n)^m) = O(n^{d+1}(\lg n)^m) = \Theta(()n^{d+1}(\lg n)^m)$$

   Hence,

$$f(n) \in \Theta((n^{d+1}(\lg n)^m)$$

3. [8 points, not evenly distributed] *Purpose: application of the Master Theorem and ability to recognize and deal with situations where it does not apply*

   Give Θ bounds for $T(n)$ in each of the following recurrences. Assume $T(n)$ is constant for small values of n.

   (a). T(n) $= 17 \cdot T(n/4) + n^2 \lg n$

   Comparing it with T(n) $= a \cdot T(n/b) + f(n)$

   a =17 and b = 4

   f(n) $= n^2 \lg n$

   Now, $n^{\log_4 17} = n^{2.043}$

   f(n) $= O(n^{\log_4 17-\epsilon})$ for any $0 < \epsilon < 0.043$

   So, By Master's theorem version 1, we can state that

   T(n) $= \Theta(n^{\log_4 17})$

   (b). T(n) $= 2 \cdot T(n/2) + n\lg^2 n$

   Comparing it with $T(n) = a \cdot T(n/b) + f(n)$

   a =2 and b = 2

   f(n) $= n\lg^2 n$

   Assuming $n = 2^k$

Level	# of instances	Instance size	Cost per instance	Total cost
0	1	n	$n\lg^2 n$	$n\lg^2 n$
1	2	$\frac{n}{2}$	$\frac{n}{2}\lg^2\frac{n}{2}$	$n\lg^2\frac{n}{2}$
i	2^i	$\frac{n}{2^i}$	$\frac{n}{2^i}\lg^2\frac{n}{2^i}$	$n\lg^2\frac{n}{2^i}$
k	2^k	1	c_0	$c_0 2^k$

$$\begin{aligned} \text{T(n)} &= c_0 2^k + \sum_{i=0}^{k-1} n\lg^2\frac{n}{2^i} \\ &= c_0 2^k + n\sum_{i=0}^{k-1}(\lg n - i)^2 \\ &= c_0 n + n\sum_{i=0}^{k-1}(k-i)^2 \\ &= c_0 n + n\sum_{j=1}^{k}(j)^2 \\ &= c_0 n + \frac{(n)(k)(k+1)(2k+1)}{6} \\ &= c_0 n + \frac{(n)(\lg n)(\lg n)(2\lg n + 1)}{6} \end{aligned}$$

$\therefore T(n) = \Theta(n\lg^3 n)$

   (c). T(n) $= 4 \cdot T(n/2) + \frac{n^2}{\lg n}$

   Comparing it with $T(n) = a \cdot T(n/b) + f(n)$

   a =4 and b = 2

   f(n) $= n^2/\lg n$

Assuming $n = 2^k$

Level	# of instances	Instance size	Cost per instance	Total cost
0	1	n	$n^2/\lg n$	$n^2/\lg n$
1	4	$\frac{n}{2}$	$\frac{n^2}{2^2}\lg\frac{n}{2}$	$n^2/\lg n - 1$
i	4^i	$\frac{n}{2^i}$	$\frac{n^i}{4^i}\lg\frac{n}{2^i}$	$n^2/\lg n - i$
k	4^k	1	c_0	$c_0 4^k$

$$\begin{aligned}
\text{T(n)} &= c_0 4^k + \sum_{i=0}^{k-1} \frac{n^2}{\lg n - i} \\
&= c_0 4^k + n^2 \sum_{i=0}^{k-1} \frac{1}{\lg n - i} \\
&= c_0 4^k + n^2 \sum_{i=0}^{k-1} \frac{1}{k - i} \\
&= c_0 4^k + n^2 \sum_{i=1}^{k} \frac{1}{k} \\
&= c_0 4^k + n^2 \ln k \\
&= c_0 n^2 + n^2 \ln(\lg n)
\end{aligned}$$

$\therefore T(n) = \Theta(n^2 \ln(\lg n))$

(d). T(n) $= 5 \cdot T(n/4) + \frac{n^{5/4}}{\lg^2 n}$
Comparing it with $T(n) = a \cdot T(n/b) + f(n)$
a =5 and b = 4
f(n) $= \frac{n^{\frac{5}{4}}}{\lg^2 n}$
Now, $n^{\lg_b a} = n^{\lg_4 5} = n^{1.16}$
$\frac{n^{1.25}}{\lg^2 n} = \Omega(n^{1.16+E})$ $\therefore$ Master theorem case 3

To check Regularity condition:
a·f(n/b) $\leq$ cf(n)

$\frac{5\cdot(\frac{n}{4})^{5/4}}{\lg^2(n/4)} \leq c \cdot \frac{n^{5/4}}{\lg^2 n}$

$\frac{5}{4^{5/4}} * \frac{\lg^2 n}{(\lg n-2)^2} \leq c$

$0.884 * (\frac{\lg n}{\lg n - 2})^2 \leq c$

Since n $= 4^k \therefore \lg n = 2k$

$0.884 * (\frac{2k}{2k-2})^2 \leq c$

Taking $n_0 = 4^{5000001}$ 2k = 10000002

$0.884 * (\frac{10000002}{10000000})^2 \leq c$

Hence Regularity condition holds for c = 0.885 and $n_0 = 4^{5000001}$
$\therefore T(n) = \frac{n^{\frac{5}{4}}}{\lg^2 n}$

4. [credited to Gary Miller in Jeff Erickson's book] [8 points]

   Consider the following sorting algorithm. The parameter $A[1..n]$ in both procedures can refer to any sequence of contiguous elements of A. So, for example, when UNUSUAL($A[1..n]$) calls UNUSUAL($A[n/4+1..3n/4]$), $n/2$ plays the role of n, $A[n/4+1]$ plays the role of A[1], and $A[3n/4]$ plays the role of $A[n]$.

```
CRUEL($A[1..n]$) is          ▷ sorts the array $A$
    if $n \geq 2$ then
        CRUEL($A[1..n/2]$)
        CRUEL($A[n/2+1..n]$)
        UNUSUAL($A[1..n]$)
    endif
end CRUEL
```

and

```
UNUSUAL($A[1..n]$) is       ▷ sorts $A$ assuming both halves have been sorted
    if $n = 2$ then
        if $A[1] > A[2]$ then swap $A[1] \leftrightarrow A[2]$ endif    ▷ the only comparison
    else
        for $i \leftarrow 1$ to $n/4$ do
            swap $A[i+n/4] \leftrightarrow A[i+n/2]$ end do
        UNUSUAL($A[1..n/2]$)                 ▷ recurse on left half
        UNUSUAL($A[n/2+1..n]$)               ▷ recurse on right half
        UNUSUAL($A[n/4+1..3n/4]$)            ▷ recurse on middle half
    endif
end UNUSUAL
```

   This sorting algorithm is ***oblivious*** – it's behavior does not depend on the values of the numbers stored in the array. You may assume that the input size n is a power of 2.

   (a) [4 points] Prove by induction that CRUEL sorts an array correctly.

   - Prove the Base Case for CRUEL:
     For the base case where n = 1, the recursion terminates.
     Let us assume a sub-array of size 2 is sent to CRUEL. CRUEL will make a recursive call to itself with size 1, which as shown above terminates the recursion. CRUEL will also call UNUSUAL with a size of 2.
     Hence the base case for UNUSUAL is n=2.

   - Prove the Base Case for UNUSUAL:
     Let us assume a sub-array of size 2 is sent to UNUSUAL.
     If the values are out of order, they will be swapped, while if they are in order the values are not altered.
     These being the only two possible cases, which will both result in expected outcome process that Unusual is valid in the base case.

   - Proving UNUSUAL(A[n])
     For UNUSUAL(A[1..n]), with the following assumption :
     A[0] $\leq$ A[2] ... $\leq$ A[n/2] and
     A[n/2+1] $\leq$ A[n/2 + 2] $\leq$ A[n]
     We swap A[n/4+1.. n/2] with A[n/2+1..3n/4] to give us $A_1[0..n]$,
     in which strictly any number in $A_1[0...n/4] \leq A_1[n/2+1..3n/4]$ and
     $A_1[n/4+1..n/2] \leq A_1[3n/4+1..n]$

i.e $A_1[0..n/2]$ contains the smaller (since this algorithm sorts in ascending order) halves of the two sub-arrays of the inputs A[0..n/2] and A[n/2+1..n]
Conversely, $A_1[n/2+1..n]$ contains the larger halves of the two sub-arrays of the inputs A[0..n/2] and A[n/2+1..n]
In the next 2 steps we sort $A_1[0..n/2]$ and $A_1[n/2+1..n]$ to give us
$A_2[0..n/2]$ and $A_2[n/2+1..n]$
Which means strictly
$A_2[0..n/4] \leq A_2[n/4+1..n/2]$ and
$A_2[n/2+1..3n/4] \leq A_2[3n/4+1..n]$
also
$A_2[0..n/4] \leq A_2[n/4+1..n]$ and
$A_2[0..3n/4] \leq A_2[3n/4+1..n]$
In the final step we sort $A_2[n/4+1..3n/4]$ to get $A_3[0..n]$ where
$A_3[n/4+1..n/2] \leq A_3[n/2+1..3n/4]$
Hence,
$A_3[0..n/4] \leq A_3[n/4+1..n/2] \leq A_3[n/2+1..3n/4] \leq A_3[3n/4+1..n]$
Which proves that UNUSUAL(A[0..n]) results in sorted array

- Proving CRUEL(A[n])
  Assuming CRUEL(A[0..n/2]) and CRUEL(A[n/2+1..n]) are correct, and UNUSUAL(A[n]) sorting array correctly as shown above, CRUEL(A[n]) sorts an array properly.

(b) [1 point] Prove that CRUEL would *not* sort correctly if we removed the **for** loop from UNUSUAL.
Taking the following list : 6421

```
CRUEL(6 4 2 1)
    CRUEL(6 4)
        CRUEL(6)
            end CRUEL(6)
        CRUEL(4)
            end CRUEL(4)
        UNUSUAL(6 4)
            end UNUSUAL(4 6)
    CRUEL(2 1)
        CRUEL(2)
            end CRUEL(2)
        CRUEL(1)
            end CRUEL(1)
        UNUSUAL(2 1)
            end UNUSUAL(1 2)
    UNUSUAL(4 6 1 2)
        UNUSUAL(4 6)
            end UNUSUAL(4 6)
        UNUSUAL(1 2)
            end UNUSUAL(1 2)
        UNUSUAL(6 1)
            end UNUSUAL(1 6)
        end UNUSUAL(4 1 6 2)
    end CRUEL(4 1 6 2)
FINAL RESULT : 4,1,6,2
```

Hence, we can prove that CRUEL will not sort correctly if we remove the **for** loop from UNUSUAL.

(c) [1 point] Prove that CRUEL would *not* sort correctly if we swapped the last two lines of UNUSUAL.

Taking the following list : 6,4,2,1

```
CRUEL(6 4 2 1)
    CRUEL(6 4)
        CRUEL(6)
            end CRUEL(6)
        CRUEL(4)
            end CRUEL(4)
        UNUSUAL(6 4)
            end UNUSUAL(4 6)
    CRUEL(2 1)
        CRUEL(2)
            end CRUEL(2)
        CRUEL(1)
            end CRUEL(1)
        UNUSUAL(2 1)
            end UNUSUAL(1 2)
    UNUSUAL(4 6 1 2)
        swap(6 1)
        UNUSUAL(4 1)
            end UNUSUAL(1 4)
        UNUSUAL(4 6)
            end UNUSUAL(4 6)
        UNUSUAL(6 2)
            end UNUSUAL(2 6)
        end UNUSUAL(1 4 2 6)
    end CRUEL(1 4 2 6)
FINAL RESULT : 1,4,2,6
```

Hence, we can prove that CRUEL will not sort correctly if we swap the last two lines of UNUSUAL.

(d) [2 points] Give a Θ bound for the number of comparisons and swaps performed by UNUSUAL when applied to an array of n elements? Prove your answer.

For every call to UNUSUAL with size n, UNUSUAL calls itself recursively 3 times, each with size n/2.
Also, for every call of size n, it is guaranteed that n/2 values are swapped using the pre-recursion for loop.
Also in the base case where n=2 , the worst case guarantees 1 swap.
Hence the equation is of form:
T(n) = 3.T(n/2) + n/2
By Master Theorem Case 1:

$$T(n) = \Theta(n^{\log_2 3})$$

(e) [2 points] Give a Θ bound on the number of comparisons and swaps performed by CRUEL when applied to an array of n elements? Prove your answer.

For every call to CRUEL with size n, CRUEL calls itself recursively 2 times, each with size n/2.
Also, for every call of size n, UNUSUAL is called for the size n, which is of complexity $\Theta(n^{\log_2 3})$
Hence the equation is of form:
T(n) = 2.T(n/2) + $\Theta(n^{\log_2 3})$
By Master Theorem Case 3:

$$T(n) = \Theta(n^{\log_2 3})$$