CS5800 ALGORITHMS S'22 — ABHI SHELAT
DUE WED FEB 2, 2022 AT 11.59P VIA GRADESCOPE

H1.A.SHELAT

- Use Latex to prepare your answers. Submit a PDF to Gradescope. Handwritten solutions will not be graded.
- You are permitted to study with friends and discuss the problems; however, *you must write up you own solutions, in your own words*. Do not submit anything you cannot explain. If you collaborate with any of the other students on any problem, please list all your collaborators in your submission for each problem.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class for answers is not allowed.
- Please use *exactly 1 page* for your answers by placing `\newpage` after your answer. You can delete any portion of the question if you need more space. You will lose points if you ignore this requirement because our grading system depends on it.

PROBLEM 1 *Asymptotic Notation Review (15)*

Rank the following functions by order of growth; that is, write the functions below in order $f_1, f_2, \ldots, f_{12}$ so that $f_i = O(f_{i+1})\ \forall i \in \{1, \ldots, 11\}$. You do not need to provide justification. Hint: use logarithms to simplify the functions.

$$\begin{array}{cccccc} 8^{\sqrt{\log_3 n}} & (\log_2 n)^{(\log_2 n)/(\log_2 \log_2 n)} & n! & n^{1/(\log_9 n)} & \log_2(\log_2(n)) & 1.0005^n \\ 5^{\log_7 n} & \log_2(n!) & n^4 & \sqrt{n} & 2^{\log_5 n} & 2^{(\log_2 n)^5} \end{array}$$

Solution: We state without proof the following relations: $n \geq \sqrt{n} \geq \log n$ and $n/\log n > \log\log n$ for $n \geq 2$. Once can take logs of these functions to compare them, or alternatively, express each one in the form of 2^y and compare exponents:

1. $n^{1/\log_9 n} = 2^{log_2 n/(\log_2 n/\log_2 9)} = O(2^{\log_2 9}) = O(1)$
2. $\log_2 \log_2 n = 2^{\log_2 \log_2 \log_2 n}$
3. $8^{\sqrt{\log_3 n}} = 2^{3 \cdot \sqrt{\log_2 n/\log_2(3)}} = O(2^{3\sqrt{\log_2 n/\log_2 3}})$
4. $2^{\log_5 n} = 2^{\log_2 n/\log_2 5}$
5. $\sqrt{n} = 2^{\log_2 n/2}$
6. $5^{\log_7 n} = 2^{\log_2 5 \log_2 n/\log_2 7} = O(2^{(\log_2 5/\log_2 7)\log_2 n})$
7. $(\log_2 n)^{(\log_2 n)/(\log_2 \log_2 n)} = 2^{(\log_2 \log_2 n * \log_2 n/\log_2 \log_2 n)}$
   $= O(2^{log_2 n}) = O(n)$
8. $\log_2 n! = 2^{\log_2 \sum_i^n \log_2 i} \leq 2^{\log_2 n \log n} = O(2^{\log_2 n + \log\log n})$

9. $n^4 = 2^{4 \log_2 n}$

10. $1.0005^n = 2^{(\log_2 1.0005)n} = O(2^{(\log_2 1.0005)n})$

11. $2^{(\log_2 n)^5}$

12. This last function requires using *Stirling's approximation* to understand. Factorial grows extremely quickly, and it is hard to understand this using a graphing method. Stirling's approximation states that $\log(n!) \sim \Theta(n \log n)$. Therefore, we have

$$n! = 2^{\Theta(n \log n)}$$

And thus, we have that $\log_2 n^5 = O(n \log n)$, which makes this function the largest. Most attempts at graphing $n!$ will fail for numerical overflows. Moreover, the cross-over point for when $n!$ becomes larger than $2^{\log n^5}$ is also quite large. However, at $n = 2^{100}$, you can see that $\log(2^{100})^5 = 10^{10}$, which is immediately smaller than $2^{100} \cdot \log(2^{100})$.

Since this analysis required Stirling's approximation, we won't take points off if you missed this one, or missed number 8.

PROBLEM 2 *Induction (15)*

In class, we saw an informal argument for why

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$$

Prove this formula using induction. State a base case, then a hypothesis, and then, that the hypothesis holds for the next larger case.

Solution: Let $n = 2$, we can see that

$$\begin{aligned} f(2) &= \sum_{i}^{2} a^i = a^0 + a^1 + a^2 = 1 + a + a^2 \\ &= \frac{(a-1)(1+a+a^2)}{(a-1)} \\ &= \frac{(a^3-1)}{(a-1)} \end{aligned}$$

Assume that the formula holds for n, then for $n+1$ we have

$$\begin{aligned} f(n+1) &= \sum_{i}^{n+1} a^i = \left(\sum_{i}^{n} a^i \right) + a^{n+1} \\ &= \frac{(a^{n+1}-1)}{(a-1)} + \frac{(a-1)(a^{n+1})}{(a-1)} \\ &= \frac{(a^{n+1}-1) + (a^{n+2} - a^{n+1})}{(a-1)} \\ &= \frac{(a^{n+2}-1)}{(a-1)} \end{aligned}$$

thus the formula holds for $n+1$.

PROBLEM 3 *Recurrences (25)*

Solve the following recurrences by obtaining a Θ bound. You may assign a standard value for the base case terms $T(1), T(2), \ldots, T(k)$ for some small constant k. Prove your answer.

1. $T(n) = T(n-5) + n$

   Solution: By guess and check, we show $T(n) \leq 2n^2$. Suppose $T(1) = T(2) = T(3) = T(4) = 1$. The formula holds for these base cases. Inductive step: suppose the property is true for all $n < n_0$, then when $n = n_0$,

$$\begin{aligned} T(n) = T(n-5) + n &< 2(n-5)^2 + n \\ &< 2n^2 - 20n + 50 + n \\ &< 2n^2 \text{ when } n \geq 3 \end{aligned}$$

   For the lower bound, we show $T(n) > (1/20)n^2$, which holds for all of the base cases. Suppose that is true for all $n < n_0$. Then when $n = n_0$, we have

$$\begin{aligned} T(n) &= T(n-5) + n \\ &> \frac{1}{20}(n-5)^2 + n = \frac{n^2}{20} + \frac{n}{2} + \frac{25}{20} \\ &> \frac{n^2}{20} \end{aligned}$$

2. $T(n) = 27T(\lceil n/19 \rceil) + n$

   Solution: From case 1 of the Master's theorem, it follows that

$$T(n) = \Theta(n^{\log_{19}(27)})$$

3. $T(n) = 2T(\lceil \sqrt{n} \rceil) + 2$

   Solution: Let $n = 2^m$ and $S(m) = T(2^m)$, for $m \geq 0$, then the recurrence is simplified as $S(m) = 2S(m/2) + 2$. Fo r the recurrence $S(m)$, we have $a = 2, b = 2, f(n) = 2$, and $n^{\log_b a} = n = O(n)$. Since $f(n) = O(n^{\log_2 2 - \epsilon})$, where $\epsilon = 0.1$, we have $S(m) = \Theta(m)$ by case 1 of Master Theorem. Change back from $S(m)$ to $T(n)$, we obtain $T(n) = \Theta(\log n)$.

4. $T(n) = T(\lceil n/11 \rceil) + T(\lfloor 6n/7 \rfloor) + n$

   Solution: Suppose $T(0) = T(1) = 1$. There are many ways to get the initial guess, e.g. make the recursion tree and notice that at level k, the amount of work looks like the binomial formula for $((1/11) + (6/7))^k$. We guess that $T(n) = \Theta(n)$.

Proof. We aim to show $T(n) \leq 100n - 99 = O(n)$. The base case holds for $T(1)$. Inductive step: suppose the guess is true for all $n < n_0$, then when $n = n_0$,

$$\begin{aligned}
T(n_0) &= T(\lceil n_0/11 \rceil) + T(\lfloor 6n_0/7 \rfloor) + n_0 \\
&\leq (100\lceil n_0/11 \rceil - 99) + (100\lfloor 6n_0/7 \rfloor - 99) + n_0 \\
&\leq (100(n_0/11 + 1) - 99) + (100 \cdot 6n_0/7 - 99) + n_0 \\
&= \frac{100}{11}n_0 + 100 - 99 + \frac{600}{7}n_0 - 99 + n_0 \\
&= \left(\frac{700 + 6600 + 77}{77}\right) n_0 - 98 \\
&\leq 96n_0 - 98 \\
&\leq 96n_0 - 98 + (4n_0 - 1) = 100n_0 - 99 \text{ when } n_0 > 1
\end{aligned}$$

The last line in the derivation follows because when $n_0 > 1$, then $4n_0 > 1$ and thus we are adding a quantity to the right side that is positive. Note here the choice of constants 100 and 99 ensure that the formula holds for the base case $T(1) = 1$.

For the lower bound, a linear lower bound follows directly from the n term in the recurrence: $T(n) \geq n = \Omega(n)$ because

$$T(n) = T(\lceil n/11 \rceil) + T(\lfloor 6n/7 \rfloor) + n \geq n$$

Therefore, $T(n) = \Theta(n)$. □

PROBLEM 4 *Master theorem not applicable (20)*

Consider the recurrence $T(n) = 2T(n/2) + f(n)$ in which

$$f(n) = \begin{cases} n^3 & \text{if } \lceil \log(n) \rceil \text{ is even} \\ n^2 & \text{otherwise} \end{cases}$$

Show that $f(n) = \Omega(n^{\log_b(a)+\epsilon})$. Explain why the third case of the Master's theorem does not apply. Prove a Θ-bound for the recurrence.

Solution: Observe that $f(n) = \Omega(n^2) = \Omega(n^{\log_2(2)} = n)$. However, the regularity condition of Case 3, which requires that there exists a constant $c < 1$ such that $af(n/b) < cf(n)$ does not hold. Consider an n such that $\log(n)$ is odd, and therefore $\log(n/2)$ is even. In this case, $2f(n/2) = n^3/4$, but $f(n) = n^2$. Therefore, no $c < 1$ can make the inequality hold for large n.
However, $T(n) = \Theta(n^3)$ which can be established by guess and check.

As an upper-bound, $T(n) < 2n^3$ holds for small cases; suppose it holds for all $n < n_0$. Now consider n_0.

$$\begin{aligned} T(n_0) &= 2T(n_0/2) + f(n_0) \\ &\leq 2 \cdot 2(n_0/2)^3 + f(n_0) \\ &\leq n_0^3 \left(\frac{1}{2}\right) + f(n_0) < 2n_0^3 \end{aligned}$$

which completes the upper-bound. The last inequality follows because f is upper-bounded by n^3.

For the lower-bound, show that $T(n) > n^3/8$. As before, consider the case n_0, but expand the recursive formula twice:

$$\begin{aligned} T(n_0) &= 2T(n_0/2) + f(n_0) = 2[2T(n_0/4) + f(n_0/2)] + f(n_0) \\ &= 4T(n_0/4) + 2f(n_0/2) + f(n_0) \\ &\geq 2f(n_0/2) + f(n_0) \\ &> n_0^3/8 \end{aligned}$$

The last line follows because either $\log(n_0)$ is even and thus $f(n_0) = n_0^3$, or $\log(n_0/2)$ is even, in which case $2f(n_0/2) = n_0^3/4$; both are greater than $n_0^3/8$.

PROBLEM 5 *Approximate Square Root*

Present and analyze an algorithm that on input $n \in \mathbb{N}$, outputs $\lfloor \sqrt{n} \rfloor$ using $O(\log(n))$ integer ops.

Solution: Perform a binary search over the list of numbers from 1 to n to find the approximate root. Start at $m = \frac{n}{2}$. At each step, check if m is the best integer approximation to $\sqrt{n}$, or $m * m > n$, or $m * m < n$ and then repeat the operation for the lower or upper half of the list, respectively, or just terminate in the first case. As a base case, when either the input is ≤ 3, we return an answer. The following pseudocode run as APPROXROOT$(n, 1, n, 1)$ executes the algorithm:

```
ApproxRoot($n, low, high$)
1  if $n \leq 3$ return 1
2  $m \leftarrow \lfloor \frac{low+high}{2} \rfloor$
3  if $m^2 \leq n$ and $(m+1)^2 > n$ return $m$
4  if $m^2 < n$ return ApproxRoot($n, m, high$)
5  if $m^2 > n$ return ApproxRoot($n, low, m$)
```

The recurrence for counting the number of calls to integer ops that our algorithm makes is $T(n) = T(\lfloor n/2 \rfloor) + \theta(1)$, which solves to $\Theta(\log n)$ by the second case of the Master's theorem.