Homework 0

Gopal Pandurangan

Instructions:
Read the Academic honesty policy posted on Blackboard which is repeated here: All submitted work should be your own. Copying or using other people's work (including from the Web) will result in $-$MAX points, where MAX is the maximum possible number of points for that assignment. Repeat offense will result in getting a failure grade in the course and reporting to the Chair. If you have any questions regarding any assignment, please contact me. The best way is to ask on Piazza.

By submitting this homework, **you affirm that you have followed the Academic Dishonest Policy.**

Justify your answers Show **appropriate work**.

Write **clearly** and **legibly**.

Please **start to work on the problems early** as these generally require some time.

We recommend the use of LaTeX for typing solutions. LaTeX is a software system for typesetting documents and has found widespread use in academia and is considered the standard way to create mathematical documents. To get started using LaTeX, the use of Overleaf is recommended. A short tutorial on the use of LaTeX can be found here.

Out: Tuesday, August 23
Due: Sunday, September 4, 11:59PM
Submissions will not be accepted after the deadline.

The exercises below are from the book available at my homepage. Please always use the latest version of the book which is posted at this site, since the book is updated periodically.
Reading: Chapters 2-4, Appendices A, B, and E. In particular, several worked exercises with solutions are given. Trying to solve the worked exercises **before** seeing their solutions is a good learning technique.

Exercises refer to the corresponding chapter exercises in the book.
2.4 (part c), 3.4 (part d), 4.1 (part a), 4.3 (part 6), 4.8, Exercises A.6 (in Appendix A), Exercise B.6 (in Appendix B).

Exercise 2.4 (c)

Rank the following functions by order of growth. That is, find an arrangement $f_1, f_2, \ldots, f_k$ of the functions satisfying

$$\begin{aligned} f_1 &= \mathcal{O}(f_2) \\ f_2 &= \mathcal{O}(f_3) \\ &\vdots \\ f_{k-1} &= \mathcal{O}(f_k) \end{aligned}$$

Justify your ordering. Note that $\log^k n$ is the usual way of writing $(\log n)^k$.

$$n^3,\ \frac{n}{\log^2 n},\ n\log n, 1.1^n,\ \frac{1}{n^3},\ \log^6 n,\ \frac{1}{n},\ 2^{\lg n},\ n!,\ n^{\lg\lg n},\ 2^{\sqrt{\log n}},\ n^{\frac{1}{\log n}}$$

Solution. The ordering is

$$\frac{1}{n^3}, \frac{1}{n}, n^{\frac{1}{\log n}}, \log^6 n, 2^{\sqrt{\log n}}, \frac{n}{\log^2 n}, 2^{\log n}, n\log n, n^3, n^{\lg\lg n}, 1.1^n, n!$$

Simply take the limits of these functions in order and show that they all exist and are finite:

- $\frac{1}{n^3} = \mathcal{O}\left(\frac{1}{n}\right)$:

$$\begin{aligned} \lim_{n\to\infty} \frac{\frac{1}{n^3}}{\frac{1}{n}} &= \lim_{n\to\infty} \frac{n}{n^3} \\ &= \lim_{n\to\infty} \frac{1}{n^2} \\ &= 0 \end{aligned}$$

- $\frac{1}{n} = \mathcal{O}\left(n^{\frac{1}{\log n}}\right)$ (note that $n^{\frac{1}{\log n}} = 2$):

$$\begin{aligned} \lim_{n\to\infty} \frac{\frac{1}{n}}{2} &= \lim_{n\to\infty} \frac{1}{2n} \\ &= 0 \end{aligned}$$

- $n^{\frac{1}{\log n}} = \mathcal{O}\left(\log^6 n\right)$ (note that $n^{\frac{1}{\log n}} = 2$):

$$\lim_{n\to\infty} \frac{2}{\log^6 n} = 0$$

- $\log^6 n = \mathcal{O}\left(2^{\sqrt{\log n}}\right)$. We show that the limit of their logs is 0:

$$\begin{aligned} \lim_{n\to\infty} \frac{\log\left(\log^6 n\right)}{\log\left(2^{\sqrt{\log n}}\right)} &= \lim_{n\to\infty} \frac{6\log\log n}{\sqrt{\log n}} \\ &= \lim_{n\to\infty} \frac{\frac{6}{n\ln 2\ln n}}{\frac{1}{2n\sqrt{\ln 2\ln n}}} \\ &= \lim_{n\to\infty} \frac{12}{\sqrt{\ln 2\ln n}} \\ &= 0 \end{aligned}$$

- $2^{\sqrt{\log n}} = \mathcal{O}\left(\frac{n}{\log^2 n}\right)$. We show that the limit of their logs is 0:

$$\begin{aligned} \lim_{n\to\infty} \frac{\log\left(2^{\sqrt{\log n}}\right)}{\log\left(\frac{n}{\log^2 n}\right)} &= \lim_{n\to\infty} \frac{\sqrt{\log n}}{\log n - 2\log\log n} \\ &= 0 \end{aligned}$$

- $\frac{n}{\log^2 n} = \mathcal{O}\left(2^{\log n}\right)$ (note that $2^{\log n} = n$):

$$\begin{aligned}\lim_{n\to\infty} \frac{\frac{n}{\log^2 n}}{n} &= \lim_{n\to\infty} \frac{1}{\log^2 n} \\ &= 0\end{aligned}$$

- $2^{\log n} = \mathcal{O}(n \log n)$ (note that $2^{\log n} = n$):

$$\begin{aligned}\lim_{n\to\infty} \frac{n}{n \log n} &= \lim_{n\to\infty} \frac{1}{\log n} \\ &= 0\end{aligned}$$

- $n \log n = \mathcal{O}\left(n^3\right)$:

$$\begin{aligned}\lim_{n\to\infty} \frac{n \log n}{n^3} &= \lim_{n\to\infty} \frac{\log n}{n^2} \\ &= \lim_{n\to\infty} \frac{\frac{1}{n \ln 2}}{2n} \\ &= \lim_{n\to\infty} \frac{1}{2n^2 \ln 2} \\ &= 0\end{aligned}$$

- $n^3 = \mathcal{O}\left(n^{\lg \lg n}\right)$. We show that the limit of their logs is 0:

$$\begin{aligned}\lim_{n\to\infty} \frac{\log\left(n^3\right)}{\log\left(n^{\lg \lg n}\right)} &= \lim_{n\to\infty} \frac{3 \log n}{\lg \lg n \log n} \\ &= \lim_{n\to\infty} \frac{3}{\lg \lg n} \\ &= 0\end{aligned}$$

- $n^{\lg \lg n} = \mathcal{O}(1.1^n)$. We show that the limit of their logs is 0:

$$\begin{aligned}\lim_{n\to\infty} \frac{\log\left(n^{\lg \lg n}\right)}{\log\left(1.1^n\right)} &= \lim_{n\to\infty} \frac{\lg \lg n \log n}{n \log 1.1} \\ &= 0\end{aligned}$$

- $1.1^n = \mathcal{O}(n!)$. A basic induction proof works here. Notice that $1.1^2 < 2!$ (this is our base case). Suppose we have that $1.1^n < n!$ for some n. Then

$$\begin{aligned}1.1^{n+1} &= 1.1^n \times 1.1 \\ &\leq n! \times 1.1 \\ &< n! \times n \\ &= (n+1)!\end{aligned}$$

  A more elegant proof that applies for any $a > 1$ shows that $a^n < n!$ for all sufficiently large n. Consider the series

$$\sum_{n=1}^{\infty} \frac{a^n}{n!}$$

  And notice that

$$\begin{aligned}\lim_{n\to\infty} \frac{\frac{a^{n+1}}{(n+1)!}}{\frac{a^n}{n!}} &= \lim_{n\to\infty} \frac{a}{n = 1} \\ &= 0\end{aligned}$$

  Thus, by the ratio test, this series converges. In particular, this forces

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

□

Exercise 3.4 (d)

Prove by induction that $3^n < n!$ for $n \geq 7$.

Solution. Proceed by induction on n. The base case is trivial:

$$3^7 = 2187 < 5040 = 7!$$

Suppose that $3^{n-1} < (n-1)!$. Then

$$\begin{aligned} 3^n &= 3 \times 3^{n-1} \\ &< 3 \times (n-1)! \text{ by the induction hypothesis} \\ &< n \times (n-1)! \text{ since } n \geq 7,\ 3 < n \\ &= n! \end{aligned}$$

as desired. □

Exercise 4.1 (a)

Prove that the asymptotic bound for the following recurrence

$$T(n) \leq 2T(n/2) + n^2$$

is

$$T(n) = \mathcal{O}\big(n^2 \log n\big)$$

by induction. Assume the base cases of the recurrence are constants i.e., $T(n) < k$ for $n < n_0$ where n_0 and k are some constants.

Solution. Begin by induction on n. The base case is given. Our induction hypothesis is that, for all $n_0 \leq k < n$, $T(k) \leq ck^2 \log k$ for some $c > 0$. Then

$$\begin{aligned} T(n) &\leq 2T\Big(\frac{n}{2}\Big) + n^2 \text{ by definition of } T(n) \\ &\leq 2c\Big(\frac{n}{2}\Big)^2 \log \frac{n}{2} + n^2 \text{ by our induction hypothesis} \\ &= 2c\frac{n^2}{4} \log \frac{n}{2} + n^2 \\ &= 2c\frac{n^2}{4} \log n - 2c\frac{n^2}{4} \log 2 + n^2 \\ &= c\frac{n^2}{2} \log n - c\frac{n^2}{2} + n^2 \\ &= c\frac{n^2}{2} \log n + n^2\Big(-\frac{c}{2} + 1\Big) \\ &\leq c\frac{n^2}{2} \log n \text{ for } c \geq 2 \\ &< cn^2 \log n \text{ for all c } \geq 2 \end{aligned}$$

□

Exercise 4.3 (6)

Solve the following recurrence

$$T(n) = 4T(n/2) + n^3$$

Give the answer in terms of *Big-Theta* notation. Solve up to constant factors, i.e., your answer must give the correct function for $T(n)$, up to constant factors. You can assume constant base cases, i.e., $T(1) = T(0) = c$, where c is a positive constant. You can ignore floors and ceilings.

You can use the DC Recurrence Theorem if it applies.

Solution. In this case, $a = 4$, $b = 2$, and $f(n) = n^3$. Then

$$\begin{aligned} af\left(\frac{n}{b}\right) &= 4\left(\frac{n}{2}\right)^3 \\ &= 4\frac{n^3}{8} \\ &= \frac{1}{2}n^3 \end{aligned}$$

hence $c = 1/2 < 1$ and, by the DC Recurrence Theorem, $T(n) = \Theta\left(n^3\right)$. □

Exercise 4.8

Give a recursive algorithm to compute b^n for a given integer n. Your algorithm should perform only $\mathcal{O}(\log n)$ integer multiplications.

Solution. Notice that, if $n = b_0 b_1 \dots b_k$ is the binary representation of n, then $2^n = 2^{b_0 b_1 \dots b_k} = 2^{2^k b_0} 2^{2^{k-1} b_1} \cdots 2^{b_n}$. Whenever these b_i is 0, $2^{2^{k-i} b_i} = 1$.

Algorithm `pow`(b, n): Returns b^n

```
1: procedure POW(b, n)
2:     if n = 0 then
3:         return 1
4:     else
5:         if n is even then
6:             return POW(b^2, n/2)
7:         else
8:             return b × POW(b^2, ⌊n/2⌋)
```

Note that this takes $\mathcal{O}(\log n)$ steps, as each step we divide n by 2, and thus the algorithm terminates in $\lfloor \log n \rfloor$ steps. □

Exercise A.6

Consider the set of first $2n$ positive integers, i.e., $A = \{1, 2, \dots, 2n\}$. Take any subset S of $n + 1$ distinct numbers from set A. Show that there two numbers in S such that one divides the other.

Solution. Partition A into sets of the form $S_m = \{m, 2m, 4m, \dots\}$, where m is odd. Notice that this forms a partition of $\{1, 2, \dots, 2n\}$[1] and that, for any a, b in a subset S_m, we must have $a \mid b$ or $b \mid a$. The number of such sets is exactly the number of odd numbers between 1 and $2n$, which is n. Thus, by the pigeonhole principle, given $n + 1$ numbers, at least two must fall in the same subset S_m, and we are done. □

Exercise B.6

A *strange* number is one whose only prime factors are in the set $\{2, 3, 5\}$. Give an efficient algorithm (give pseudocode) that uses a **binary heap** data structure to output the n-th strange number. Explain why your algorithm is correct and analyze the run time of your algorithm. (Hint: Consider generating the strange numbers in increasing order, i.e., 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, etc. Show how to generate efficiently the next strange number using a heap.)

[1] A proof of this is fairly straightforward, so we leave it to the reader.

Solution. Begin with a min-heap with a root node of 1. At each step, pop the heap and continue popping until the root value changes. Then, insert $2r$, $3r$, and $5r$ into the the heap. Perform this n times to determine the n-th strange number.

Clearly, the heap will contain every strange number at some step, for if $2^{p_1}3^{p_2}5^{p_3}$ is some strange number, then, when the root is any of $2^{p_1-1}3^{p_2}5^{p_3}$, $2^{p_1}3^{p_2-1}5^{p_3}$, or $2^{p_1}3^{p_2}5^{p_3-1}$, we will push $2^{p_1}3^{p_2}5^{p_3}$. At each step, the root is the smallest strange number in the heap. Since duplicates are removed, the value returned is the n-th strange number. This argument can be formalized by induction.

Algorithm 2 Outputs the n-th strange number.

```
1: function STRANGE-NUMBER(n)
2:     H = BINARY-HEAP()
3:     H.push(1)
4:     for i = 1 to n do
5:         r = H.pop()
6:         while H is nonempty and H[0] == r do ▷ Check for duplicate values of r.
7:             H.pop()
8:         H.push(2r, 3r, 5r)
9:     return H[0]
```

□