Week 1 Tutorial Sheet

(To complete at home during week 1)

Problem 1. Show, using elementary properties of the logarithm function, that the following identities are true

(a) $\log_2\left(\frac{k+1}{2}\right)+1=\log_2(k+1)$

(b) $a^{\log_b(n)}=n^{\log_b(a)}$ for any base $b>1$

Solution

(a) We will make use of the facts that $\log_2(a)+\log_2(b)=\log_2(ab)$, and that $\log_2(2)=1$. Using these, we find

$$\begin{aligned}\log_2\left(\frac{k+1}{2}\right)+1&=\log_2\left(\frac{k+1}{2}\right)+\log_2(2),\\&=\log_2\left(\frac{k+1}{2}\times 2\right),\\&=\log_2(k+1),\end{aligned}$$

as required.

(b) Let's use the fact that for any positive real a, we can write $a=b^{\log_b(a)}$ since log and exponentiation are inverses. We can therefore write

$$a^{\log_b(n)}=\left(b^{\log_b(a)}\right)^{\log_b(n)}.$$

Now we dig into our exponent laws and use the fact that $(x^{p_1})^{p_2}=x^{p_1p_2}$ to write

$$\left(b^{\log_b(a)}\right)^{\log_b(n)}=b^{\log_b(a)\log_b(n)}=\left(b^{\log_b(n)}\right)^{\log_b(a)}.$$

Finally, since $b^{\log_b(n)}=n$, we have

$$\left(b^{\log_b(n)}\right)^{\log_b(a)}=n^{\log_b(a)},$$

and hence $a^{\log_b(n)}=n^{\log_b(a)}$ as required.

Problem 2. Using mathematical induction, prove the following identity:

$$\sum_{i=1}^{n} i \coloneqq 1+2+\ldots+n=\frac{n(n+1)}{2},\qquad \text{for } n\geq 1.$$

Solution

Define the following:

$$L(n)=1+2+\ldots+n,\qquad R(n)=\frac{n(n+1)}{2}.$$

We want to show that $L(n)=R(n)$ for all $n\geq 1$.

Base Case:

Let $n=1$. We have

$$L(1)=1=\frac{1(1+1)}{2}=R(1),$$

as required.

Inductive Case:

Assume that $L(k) = R(k)$ for some $k \geq 1$, i.e. assume that

$$1 + 2 + \ldots + k = \frac{k(k+1)}{2}.$$

We want to prove that $L(k+1) = R(k+1)$. Beginning with the left hand side, we wish to express the value of $L(k+1)$ in terms of $L(k)$ so that we may make use of the inductive hypothesis. We can do so by writing

$$\begin{aligned} L(k+1) &= 1 + 2 + \ldots + k + (k+1), \\ &= L(k) + (k+1). \end{aligned}$$

Now for the most important step, we apply the inductive hypothesis $L(k) = R(k)$ to write

$$L(k) + (k+1) = R(k) + (k+1).$$

From here, we simply use algebra to clean up the resulting expression and obtain the desired result.

$$\begin{aligned} R(k) + (k+1) &= \frac{k(k+1)}{2} + (k+1), \\ &= \frac{k(k+1)}{2} + \frac{2(k+1)}{2}, \\ &= \frac{k(k+1) + 2(k+1)}{2}, \\ &= \frac{(k+1)(k+2)}{2}, \\ &= R(k+1), \end{aligned}$$

and therefore $L(k+1) = R(k+1)$ as required. Therefore, by induction on n, it is true that

$$\sum_{i=1}^{n} i \coloneqq 1 + 2 + \ldots + n = \frac{n(n+1)}{2}, \qquad \text{for } n \geq 1.$$

Problem 3. Using mathematical induction, prove the following identity:

$$\sum_{i=0}^{n} r^i \coloneqq 1 + r + r^2 + r^3 + \ldots + r^n = \frac{r^{n+1} - 1}{r - 1}, \qquad \text{for all } n \geq 0, r \neq 1.$$

Solution

Define the following:

$$L(n) = 1 + r + r^2 + r^3 + \ldots + r^n, \qquad R(n) = \frac{r^{n+1} - 1}{r - 1}.$$

We want to show that $L(n) = R(n)$ for all $n \geq 0$, where $r \neq 1$.

Base Case:

Let $n = 0$. We have

$$R(0) = \frac{r^{0+1} - 1}{r - 1} = \frac{r - 1}{r - 1} = 1 = L(0),$$

as required.

Inductive Case:

Assume that $L(k) = R(k)$ for some $k \geq 0$, i.e. assume that

$$1 + r + r^2 + r^3 + \ldots + r^k = \frac{r^{k+1} - 1}{r - 1}.$$

We want to prove that $L(k+1) = R(k+1)$. Beginning with the left hand side, we'd like to express $L(k+1)$ in terms of $L(k)$ so that we can make use of the inductive hypothesis. We can do so by writing

$$\begin{aligned} L(k+1) &= 1 + r + r^2 + r^3 + \ldots + r^k + r^{k+1}, \\ &= L(k) + r^{k+1}. \end{aligned}$$

Now we apply the inductive hypothesis that $L(k) = R(k)$ to write

$$L(k) + r^{k+1} = R(k) + r^{k+1}.$$

Now we use algebra to express the result in terms of $R(k+1)$.

$$\begin{aligned} R(k) + r^{k+1} &= \frac{r^{k+1} - 1}{r - 1} + r^{k+1}, \\ &= \frac{r^{k+1} - 1}{r - 1} + \frac{r^{k+1}(r-1)}{r-1}, \\ &= \frac{r^{k+1} + r^{k+1}(r-1) - 1}{r-1}, \\ &= \frac{r^{k+1}(1 + (r-1)) - 1}{r-1}, \\ &= \frac{r^{k+1} r - 1}{r-1}, \\ &= \frac{r^{(k+1)+1} - 1}{r-1}, \\ &= R(k+1), \end{aligned}$$

and therefore $L(k+1) = R(k+1)$ as required. Therefore, by induction on n, it is true that

$$\sum_{i=0}^{n} r^i \coloneqq 1 + r + r^2 + r^3 + \ldots + r^n = \frac{r^{n+1} - 1}{r - 1}, \qquad \text{for all } n \geq 0, r \neq 1.$$

Problem 4. Using Problem 3, show that the following inequality is true for all integers $n \geq 1$

$$\sum_{i=0}^{n} \frac{1}{2^i} \coloneqq 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{2^n} < 2$$

Solution

Notice that $\displaystyle\sum_{i=0}^{n} \frac{1}{2^i}$ can be rewritten as $\displaystyle\sum_{i=0}^{n} \left(\frac{1}{2}\right)^i$, and that by setting $r = \frac{1}{2}$ we can rewrite this as $\displaystyle\sum_{i=0}^{n} r^i$, so

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1} = \frac{\left(\frac{1}{2}\right)^{n+1} - 1}{\frac{1}{2} - 1} = \frac{\left(\frac{1}{2}\right)^n - 2}{-1} = 2 - \left(\frac{1}{2}\right)^n.$$

Since $\left(\frac{1}{2}\right)^n > 0$ for all $n \geq 1$, we have $2 - \left(\frac{1}{2}\right)^n < 2$, and hence it is true that $\sum_{i=0}^{n} \frac{1}{2^i} < 2$ as required.

Problem 5. The harmonic numbers are defined as the partial sums of the harmonic sequence:

$$H(n) := \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

We want to determine how this quantity behaves asymptotically, as it frequently arises in algorithm analysis.

(a) Prove that $H(n) > \log_e(n)$

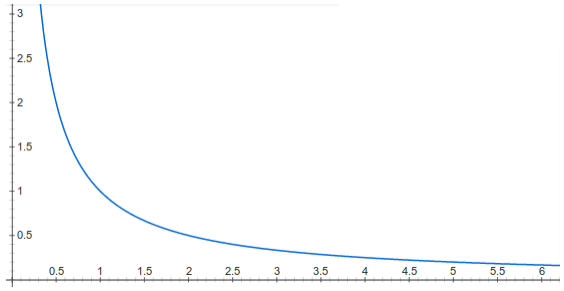
(b) Prove that $H(n) \leq \log_e(n) + 1$

(c) Hence deduce that $H(n) = \Theta(\log(n))$ [1]

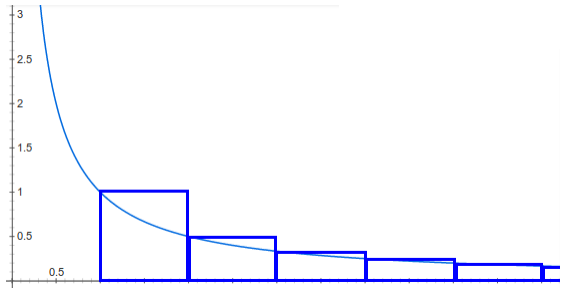
[Hint: Consider the area under the function $f(x) = \frac{1}{x}$ using calculus and compare this with $H(n)$ geometrically.]

Solution

Consider the function $f(x) = \frac{1}{x}$ for $x \geq 1$. A plot of f is depicted below.



We notice that the value of the sum $H(n) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ is equal to the area of the rectangles depicted on the following plot.



Since the rectangles are above the curve, their total area is bigger than the area underneath the curve (from $x = 1$ onwards). The area underneath the curve is given by

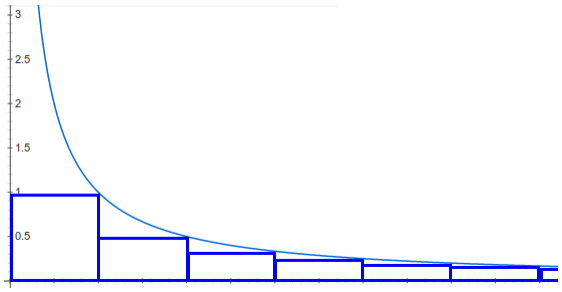
$$\int_1^{n+1} \frac{1}{x} dx = \log_e(n+1),$$

and therefore we have

$$H(n) > \log_e(n+1) > \log_e(n),$$

[1] Remember that the notation $\Theta(f(n))$ means that $H(n)$ is both upper bounded by $c_1 f(n)$ for some c_1 (it is $O(f(n))$ and lower bounded by $c_2 f(n)$ for some c_2 (it is $\Omega(f(n))$).

and therefore $H(n) > \log_e(n)$. Now, consider the same rectangles but all shifted 1 unit to the left.



The leftmost rectangle has area 1, and the others all are all below the curve. The area underneath the curve from 1 onwards is given by

$$\int_1^n \frac{1}{x} = \log_e(n),$$

and therefore we have

$$H(n) \leq 1 + \log_e(n),$$

and therefore we have $\log_e(n) < H(n) \leq \log_e(n) + 1$. Together, we deduce that $H(n) = \Theta(\log(n))$.

Problem 6. Prove that for all $N \geq 0$,

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1.$$

Solution

Define the following:

$$L(n) = \sum_{i=0}^{N} 2^i = 1 + 2 + 4 + \ldots + 2^N, \qquad R(n) = 2^{N+1} - 1.$$

We want to show that $L(n) = R(n)$ for all $n \geq 0$.

Base Case:

Let $n = 0$. We have

$$R(0) = 2^{0+1} - 1 = 2 - 1 = 1 = L(0),$$

as required.

Inductive Case:

Assume that $L(k) = R(k)$ for some $k \geq 0$, i.e. assume that

$$1 + 2 + 4 + \ldots + 2^k = 2^{k+1} - 1.$$

We want to prove that $L(k+1) = R(k+1)$. Beginning with the left hand side, we try to write $L(k+1)$ in terms of $L(k)$ so that we can make use of the inductive hypothesis.

$$\begin{aligned} L(k+1) &= 1 + 2 + 4 + \ldots + 2^k + 2^{k+1}, \\ &= L(k) + 2^{k+1}. \end{aligned}$$

Now we apply the inductive hypothesis that $L(k) = R(k)$ to write

$$L(k) + 2^{k+1} = R(k) + 2^{k+1}.$$

Finally, we clean up the algebra to obtain the result.

$$\begin{aligned} R(k)+2^{k+1} &= 2^{k+1}-1+2^{k+1}, \\ &= 2\times 2^{k+1}-1, \\ &= 2^{(k+1)+1}-1, \\ &= R(k+1), \end{aligned}$$

and hence $L(k+1)=R(k+1)$ as required. Therefore, by induction on n, it is true that

$$\sum_{i=0}^{N} 2^i = 2^{N+1}-1, \qquad \text{for all } N \geq 0.$$

Problem 7. For each of the following expressions, write a tight upper bound as $n \to \infty$ in big-O notation:

(a) $3n^3+100n^2+n$

(b) $n^3\log(n)+0.5n^4+100n^2$

(c) $5\sqrt{n}+10\log(n)$

(d) $\log(n)+\log(\log(n))+\log(\log(\log(n)))$

(e) $2n\log(n)+8n\log(n^2)$

(f) $3n\log(n)+5n(\log(n))^2$

(g) $\log(n)+2\log^2(n)$

(h) $3\log(n)+(\log(\log(n))^2$

Solution

(a) As n gets large, n^3 dominates n^2 and n, so this is $O(n^3)$

(b) First, n^4 dominates n^2. Second, we know that n dominates $\log(n)$, so consequently n^4 dominates $n^3\log(n)$, so this is $O(n^4)$

(c) $\sqrt{n}$ is equivalent to $n^{\frac{1}{2}}$. We know that for any value of $k>0$, n^k dominates $\log(n)$ (if you've taken `MTH1030`, try to prove this formally – that is $\lim_{n\to\infty}\frac{\log(n)}{n^k}=0$), so this is $O(\sqrt{n})$

(d) Since $\log(n)$ is dominated by n, having many nested logs is smaller than a single log, therefore this is $O(\log(n))$

(e) Since $\log(n^2)=2\log(n)$, both terms are the same magnitude $O(n\log(n))$, hence the answer is $O(n\log(n))$ ($O(n\log(n^2))$ is also correct, but longer to write and unnecessary)

(f) Since $\log(n)$ is not $O(1)$, $(\log(n))^2$ dominates $\log(n)$, hence this is $O(n(\log(n))^2)$

(g) The notation $\log^2(n)$ is short for $(\log(n))^2$, so by the same reasoning as above, this is $O(\log^2(n))$. Note that this notation does not mean $\log(\log(n))$

(h) Intuitively, we understand that $\log(n)$ is the inverse exponential, hence it grows slower than the square function grows fast. This should lead you to believe that $\log(n)$ dominates $(\log(\log(n)))^2$. To see this more concretely, substitute $n=2^{2^k}$ and we obtain $3\cdot 2^k+k^2$, and it is clear that the first term dominates, hence this is $O(\log(n))$

Problem 8. The Fibonacci sequence is defined by the recurrence relation $F(n)=F(n-1)+F(n-2)$, with $F(1)=F(2)=1$.

(a) Write a Python function that, given a non negative integer n, computes the n^{th} Fibonacci number iteratively (using a loop). Analyse the time and space complexity of your implementation.

(b) Write a Python function that, given a non negative integer n, computes the n^{th} Fibonacci number recursively. Analyse the time and space complexity of your implementation.

Think carefully about the assumptions that you make when analysing the time and space complexity of your algorithms.