Warm up

Question 1

Prove $\log_b(xy) = \log_b(x) + \log_b(y)$

Proof

Let x, y, and b be positive real numbers such that $x > 0$, $y > 0$, and $b > 1$.

Then for some real numbers X and Y, we define the following:

$$X = \log_b(x) \tag{1}$$
$$Y = \log_b(y) \tag{2}$$

From the definition of the inverse of the logarithm, we can rearrange equations (1) and (2) to get:

$$x = b^X$$
$$y = b^Y$$

Given x and y we can write,

$$\begin{aligned} xy &= b^X \cdot b^Y \\ &= b^{X+Y} \end{aligned} \tag{3}$$

Now we apply the general logarithm of base b to (3) to get:

$$\begin{aligned} \log_b(xy) &= \log_b(b^{X+Y}) && (4)\\ &= \log_b(b) \cdot (X+Y) && (5)\\ &= X + Y \\ &= \log_b(x) + \log_b(y) \end{aligned}$$

On line (4), we used the following identity: $\log_a(b^c) = \log_a(b) \cdot c$ for some real numbers a, b, and c.

In addition, on line (5), we used the trivial identity: $\log_b(b) = 1$ for some positive real number b.

Now, we have achieved the following result:

$$\log_b(xy) \equiv \log_b(x) + \log_b(y)$$

which completes the proof.

Question 2

Prove $\log_b(x) = \dfrac{\log_a(x)}{\log_a(b)}$

Proof

Let x, a, and b be positive real numbers such that $x > 0$, $a > 1$, and $b > 1$.

Then for some real numbers A and B, we define the following:

$$\begin{aligned} A &= \log_a(x) \\ B &= \log_b(x) \end{aligned}$$

From the definition of the inverse of the logarithm, we can rearrange these equations to get:

$$\begin{aligned} x &= a^A \\ x &= b^B \end{aligned}$$

Observe that

$$\begin{aligned} A &= \log_a(x) \\ &= \log_a(b^B) \end{aligned} \tag{1}$$

Consider for the moment that for some real number A'

$$\begin{aligned} A' = \log_a(b) &\Leftrightarrow a^{A'} = b \\ &\Leftrightarrow (a^{A'})^B = b^B \\ &\Leftrightarrow a^{B \cdot A'} = b^B \\ &\Leftrightarrow B \cdot A' = log_a(b^B) \\ &\Leftrightarrow B \cdot \log_a(b) = log_a(b^B) \end{aligned}$$

Then we can use this to continue from (1)

$$\begin{aligned} A &= \log_a(b^B) \\ &= \log_a(b) \cdot B \\ &= \log_a(b) \cdot \log_b(x) \end{aligned}$$

It follows that

$$\log_a(x) = \log_a(b) \cdot \log_b(x) \Leftrightarrow \log_b(x) = \frac{\log_a(x)}{\log_a(b)}$$

as desired, which completes the proof.

Induction and Well Ordering

Question 3

Prove that for all natural numbers $n \geq 0$, $10^n - 1$ is an integer multiple of 9.

Proof

We proceed by induction on n. Since $10^0 - 1 = 1 - 1 = 0 = 9 \cdot 0$, the statement is true when $n = 0$.

Assume that

$$10^k - 1 = 9 \cdot c$$

for an arbitrary natural number k, and an arbitrary natural number c such that $c \geq 0$. We show that

$$10^{k+1} - 1 = 9 \cdot d$$

for some natural number d. Observe that by adding $9 \cdot (10^k - 1) + 9$ to both sides of the induction hypothesis, we get

$$\begin{aligned} 10^k - 1 + 9 \cdot (10^k - 1) + 9 &= 10^k - 1 + 9 \cdot 10^k - 9 + 9 \\ &= 10 \cdot 10^k - 1 \\ &= 10^{k+1} - 1 \end{aligned} \tag{1}$$

The result (1) from above is the new left hand side of the induction hypothesis, and combining this with the new right hand side becomes

$$10^{k+1} - 1 = 9 \cdot c + 9 \cdot (10^k - 1) + 9 = 9 \cdot (c + 10^k)$$

Since $c + 10^k$ is an integer, $10^{k+1} - 1$ is an integer multiple of 9 as desired.

By the Principle of Mathematical Induction, $10^n - 1$ is an integer multiple of 9 for all natural numbers $n \geq 0$.

Question 4

Prove that every rational number $0 < \frac{p}{q} < 1$ can be written as a finite sum $\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_k}$, where $0 < a_1 < a_2 < \cdots < a_k$ are all natural numbers for some positive integer k.

Proof

We proceed by induction on p. When $p = 1$, then we get $\frac{1}{q}$, which is the only term in the finite sum of unit fractions.

Assume that every rational number of the form $0 < \frac{p}{q} < 1$,

$$\frac{p}{q} = \frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_k}$$

where $0 < a_1 < a_2 < \cdots < a_k$ are all natural numbers for some positive integer k. We show that

$$\frac{p+1}{q} = \frac{1}{b_1} + \frac{1}{b_2} + \cdots + \frac{1}{b_j}$$

such that $0 < \frac{p+1}{q} < 1$, and where $0 < b_1 < b_2 < \cdots < b_j$ are all natural numbers for some positive integer j.

We choose the smallest positive integer c such that, or in other words, find the largest rational number $\frac{1}{c}$ such that $\frac{1}{c} < \frac{p+1}{q}$. Since $\frac{p+1}{q} < 1$, then $c > 1$, or succinctly, $c \geq 2$.

Observe that

$$0 < \frac{p+1}{q} - \frac{1}{c} = \frac{c \cdot (p+1) - q}{c \cdot q}$$

Let us assume for the moment that $c \cdot (p+1) - q > p$, then

$$\begin{aligned} c \cdot (p+1) - q > p &\Leftrightarrow c \cdot (p+1) - p > q \\ &\Leftrightarrow c \cdot p + c - p > q \\ &\Leftrightarrow p \cdot (c-1) + c > q \end{aligned}$$

We can safely infer that $p \cdot (c-1) + c > p \cdot (c-1) + c - 1 > q$, from which we can see that

$$\frac{p+1}{q} > \frac{1}{c-1} > \frac{1}{c}$$

which is a contradiction since we assumed $\frac{1}{c}$ to be the largest rational number such that $\frac{1}{c} < \frac{p+1}{q}$.

Thus, $c \cdot (p+1) - q \leq p$, which implies that

$$\frac{p+1}{q} < \frac{1}{c-1}$$

By the induction hypothesis, since $c \cdot (p+1) - q \leq p$, it follows that

$$\frac{p+1}{q} - \frac{1}{c} = \frac{c \cdot (p+1) - q}{c \cdot q}$$

can be written in the form

$$\frac{1}{d_1} + \frac{1}{d_2} + \cdots + \frac{1}{d_m}$$

where $0 < d_1 < d_2 < \cdots < d_m$ are all natural numbers for some positive integer m.

To ensure that the chosen value of c is distinct from d_m, let us assume, to the contrary, that $d_i = c$ for some positive integer i such that $1 \leq i \leq m$. Then we have

$$\frac{p+1}{q} = \frac{1}{d_1} + \frac{1}{d_2} + \cdots + \frac{2}{d_i} + \cdots + \frac{1}{d_m} > \frac{2}{c} \tag{1}$$

Observe that from $c \geq 2$,

$$\begin{aligned} c \geq 2 &\Leftrightarrow 2 \cdot c - 2 > c \\ &\Leftrightarrow 2 \cdot (c-1) > c \\ &\Leftrightarrow \frac{2}{c} > \frac{1}{c-1} \end{aligned}$$

From (1), we get $\frac{p+1}{q} > \frac{2}{c} > \frac{1}{c-1}$, which is a contradiction since we worked out that

$$\frac{p+1}{q} < \frac{1}{c-1}$$

Thus, c must be distinct from all values of d_m. Hence, we can safely choose values of c without prior knowledge of the values d_m.

By the Principle of Mathematical Induction, every rational number of the form $0 < \frac{p}{q} < 1$ can be written as

$$\frac{p}{q} = \frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_k}$$

where $0 < a_1 < a_2 < \cdots < a_k$ are all natural numbers for some positive integer k.

Question 5

Consider the following statement. $S(n)$: *"it is possible to tile a triangular grid with 2^n triangles on a side and one corner missing, using tiles that consist of three triangles in a line."*

Use the ***Principle of Well Ordering*** to prove that $\forall n \in \mathbb{N}, S(n)$.

Proof

We proceed by the Principle of Well Ordering. Assume, to the contrary, that the statement $S(i)$ is false for some positive integers i. We set up a nonempty subset of $\mathbb{N}$, that is, a set of counterexamples to the statement $S(n)$, $C = \{x \in \mathbb{N} \mid S(x) \text{ is false}\}$.

By the Well Ordering principle, C contains the smallest element c. Since $S(1)$ is true (see Figure 1), then $1 \notin C$.

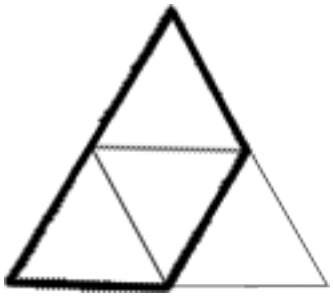

Figure 1: Base case

Thus, $c \geq 2$. From which we can infer that $c - 1 \notin C$, and that $S(c-1)$ is true. Let us define $T(n)$ to be a "geometric representation" of $S(n)$.

Given that $S(c-1)$ is true, then $T(c-1)$ is the associated geometric representation. Note that if $S(c)$ is false, then $T(c)$ would be the associated geometric counterexample. Then, let us attempt to augment shapes of $T(c-1)$ onto $T(c-1)$.

We shall augment three shapes of $T(c-1)$ onto $T(c-1)$ [See Figure 2]. Ignore the fact that the shapes being augmented are of $T(2)$, and think of them as arbitrary shapes for which $S(n-1)$ is true. Starting off with some arbitrary shape of $T(c-1)$ (shape 1), we arrange two shapes of $T(c-1)$ (shape 2 and 3) such that its missing corners make the "missing" tile defined in $S(n)$. Finally, place the third $T(c-1)$ shape (shape 4) above shape 2.

Since that $T(c-1)$ has 2^{c-1} triangles on a side, the new shape we just described, composing of four $T(c-1)$ shapes, has 2^c triangles on a side. Then this new shape must be a geometric representation of $S(c)$ We define this new shape to be

$$T(c) = 4 \cdot T(c-1) + T(1)$$

where $T(1)$ is the shape to fill in the missing "tile space".

Since $T(c)$ is a geometric representation of $S(c)$ with the required "missing corner", it follows that $S(c)$ is true. This contradicts our assumption that $c \in C$. Thus, C must be empty, that is, there are no counterexamples to the statement $S(n)$, which completes the proof.

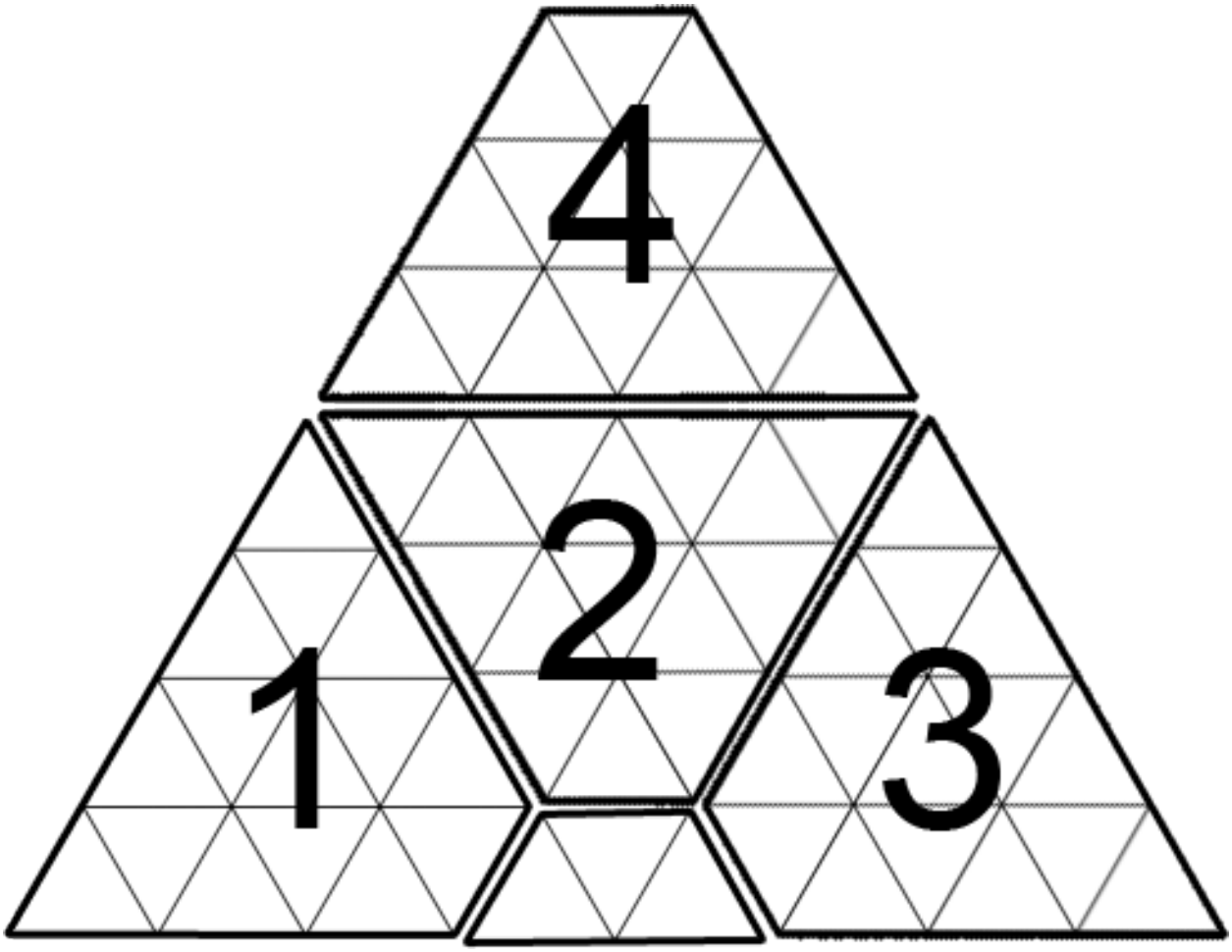

Figure 2: Construction of S(c)

Recurrence Relations

Question 6

Give a recurrence relation for the size of the set of n-bit binary strings that *do not contain* "111" and contain at least 1 bit.

Proof

Let us construct the recurrence relation $A(n)$ for the size of the set of n-bit binary strings that *do contain* "111". From this, we can derive the recurrence relation being asked for.

Let us consider n-bit strings of length n that do contain "111", and break the 'classes' of strings into cases by considering the first few sequence of bits:

Case 1 First bit is 0

Since the first bit is 0, then the remaining string of length $n-1$ contains "111".

Case 2 First two bits is 10

Since the first two bits is 10, then the remaining string of length $n-2$ contains "111".

Case 3 First three bits is 110

Since the first three bits is 001, then the remaining string of length $n-3$ contains "111".

Case 4 First three bits is 111

Since the first three bits is "111", then the remaining string of length $n-3$ can be of any combination of bits, which is 2^{n-3} possibilities.

The cases described above was simplified from cases where only first three bits were considered. To infer the simplification, consider that in case 1, the next two bits are treated as "don't-cares", and thus case 1 covers the possibility that the n-bit string may begin with "000", "010", "001", or "011". Likewise, in case 2, the next bit is also treated as "don't-care", and thus covers the possibility that the first three bits may begin with "100" or "101". In case 3, "110" is the only sequence left to consider in a 3-bit string arrangement aside from "111".

Then the recurrence relation is defined to be

$$A(n) = A(n-1) + A(n-2) + A(n-3) + 2^{n-3}$$

with initial conditions $A(0) = A(1) = A(2) = 0$.

Let us define the recurrence relation being asked for by defining it to be $S(n)$ (that is number of n-bit strings that do not contain "111"). Observe that

$$S(n) + A(n) = 2^n \tag{1}$$

with which that the combination of $S(n)$ and $A(n)$ makes up all possibilities of n-bit strings.

Then with (1), we can derive $A(n-1)$, $A(n-2)$, and $A(n-3)$

$$\begin{aligned} A(n-1) &= 2^{n-1} - S(n-1) \\ A(n-2) &= 2^{n-2} - S(n-2) \\ A(n-3) &= 2^{n-3} - S(n-3) \end{aligned}$$

Observe that from (1),

$$\begin{aligned}
S(n) &= 2^n - A(n) \\
&= 2^n - (A(n-1) + A(n-2) + A(n-3) + 2^{n-3}) \\
&= 2^n - A(n-1) - A(n-2) - A(n-3) - 2^{n-3} \\
&= 2^n - \left[2^{n-1} - S(n-1)\right] - \left[2^{n-2} - S(n-2)\right] - \left[2^{n-3} - S(n-3)\right] - 2^{n-3} \\
&= 2^n - 2^{n-1} + S(n-1) - 2^{n-2} + S(n-2) - 2^{n-3} + S(n-3) - 2^{n-3}
\end{aligned}$$

Using the fact that $2^{n-1} = 2^n - 2^{n-1}$, further simplifies $S(n)$ to

$$S(n) = S(n-1) + S(n-2) + S(n-3)$$

The initial conditions for $A(n)$ suggests the initial conditions for $S(n)$:

$$\begin{aligned}
S(0) &= 1 \\
S(1) &= 2 \\
S(2) &= 4
\end{aligned}$$

Question 7

Define a recurrence relation for a number of comparisons **Selection Sort** make between list items.

Proof

Let us define $S(n)$ to be the number of comparisons Selection Sort make between list items, where n is the size of the list being sorted.

Then for an empty list or a list containing only one element, no comparisons are made: $S(0) = S(1) = 0$. For a list of two elements, one comparison is performed: $S(2) = 1$.

Let us describe the behaviour of the selection sort in attempt to derive the recurrence relation. Given A, the list to be sorted of size n, the selection sort algorithm creates a list B which is considered to be the 'sorted' list. To begin sorting, an element x is chosen from A. This chosen element may be the first element of the list. Then, the algorithm would compare x with all other elements in A. There are $n - 1$ elements other than x. For each comparison, if the algorithm finds a new value such that it is less than (or greater than) x, then x is defined to be this new value.

Note that the behaviour of the comparison or how the initial value of x is chosen does not matter; the principle of the algorithm remains the same and such details are left to the implementor.

Once the algorithm has traversed through the rest of the list beyond the first chosen value of x, it has made $n - 1$ comparisons. Afterwards, x is removed from A and added to the list B (pushed into the right). The process repeats with a smaller list A of size $n - 1$.

From this description, we can define the recurrence relation to be

$$S(n) = S(n-1) + n - 1$$

with initial conditions $S(0) = S(1) = 0$.

Question 8

Given complexity of two algorithms to multiply two numbers in binary:

$$T_1(n) = 4 \cdot T_1(\frac{n}{2}) + a \cdot n$$

$$T_2(n) = 3 \cdot T_2(\frac{n}{2}) + b \cdot n$$

where a and b are constants.

Analyze how much of an improvement $T_2(n)$ has over $T_1(n)$.

Proof

To be able analyze, we need to to solve the recurrence relations by obtaining the closed-form solutions to $T_1(n)$ and $T_2(n)$.

Let us first solve

$$T_1(n) = 4 \cdot T_1(\frac{n}{2}) + a \cdot n$$

We solve by repeated substitution and infer a pattern.

$$\begin{aligned}
T_1(n) &= 4 \cdot T_1\left(\frac{n}{2}\right) + a \cdot n \\
&= 4 \cdot \left[4 \cdot T_1\left(\frac{n}{2^2}\right) + a \cdot \frac{n}{2}\right] + a \cdot n \\
&= 4^2 \cdot T_1\left(\frac{n}{2^2}\right) + a \cdot n \left(\frac{4}{2} + 1\right) \\
&= 4^2 \cdot \left[4 \cdot T_1\left(\frac{n}{2^3}\right) + a \cdot \frac{n}{2^2}\right] + a \cdot n \left(\frac{4}{2} + 1\right) \\
&= 4^3 \cdot T_1\left(\frac{n}{2^3}\right) + a \cdot n \left(\frac{4^2}{2^2} + \frac{4}{2} + 1\right) \\
&\vdots \\
&= 4^k \cdot T_1\left(\frac{n}{2^k}\right) + a \cdot n \cdot \left(\sum_{i=0}^{k-1} \frac{4^i}{2^i}\right) \qquad (1)
\end{aligned}$$

for some natural number k.

Using geometric series identity

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

for some constant a. We can simplify (1) to get

$$T_1(n) = 4^k \cdot T_1\left(\frac{n}{2^k}\right) + a \cdot n \cdot \left(2^k - 1\right)$$

Next, let us define an initial condition $T_2(\lambda) = c$ for some constants λ and c. Then the recurrence relation 'terminates' when $\frac{n}{2^k} = \lambda$. We solve for k to get $k = \log_2\left(\frac{n}{\lambda}\right)$. Substitute for k and $T_1(\lambda) = c$ to get

$$T_1(n) = 4^{\log_2\left(\frac{n}{\lambda}\right)} \cdot c + a \cdot n \cdot \left(2^{\log_2\left(\frac{n}{\lambda}\right)} - 1\right) \tag{2}$$

Next we solve

$$T_2(n) = 3 \cdot T_2(\frac{n}{2}) + b \cdot n$$

to obtain its closed-form solution.

Like before, we solve by repeated substitution and infer a pattern.

$$\begin{aligned}
T_2(n) &= 3 \cdot T_2\left(\frac{n}{2}\right) + b \cdot n \\
&= 3 \cdot \left[3 \cdot T_2\left(\frac{n}{2^2}\right) + b \cdot \frac{n}{2}\right] + b \cdot n \\
&= 3^2 \cdot T_2\left(\frac{n}{2^2}\right) + b \cdot n\left(\frac{3}{2} + 1\right) \\
&= 3^2 \cdot \left[3 \cdot T_2\left(\frac{n}{2^3}\right) + b \cdot \frac{n}{2^2}\right] + b \cdot n\left(\frac{3}{2} + 1\right) \\
&= 3^3 \cdot T_2\left(\frac{n}{2^3}\right) + b \cdot n\left(\frac{3^2}{2^2} + \frac{3}{2} + 1\right) \\
&\vdots \\
&= 3^k \cdot T_2\left(\frac{n}{2^k}\right) + b \cdot n \cdot \left(\sum_{i=0}^{k-1} \frac{3^i}{2^i}\right)
\end{aligned} \tag{3}$$

for some natural number k.

We again use the geometric series identity to simplify (3) to get

$$T_2(n) = 3^k \cdot T_2\left(\frac{n}{2^k}\right) + 2 \cdot b \cdot n \cdot \left[\left(\frac{3}{2}\right)^k - 1\right]$$

Similarly, let us define an initial condition $T_2(\gamma) = d$ for some constants γ and d. Then the recurrence relation 'terminates' when $\frac{n}{2^k} = \gamma$. We solve for k to get $k = \log_2\left(\frac{n}{\gamma}\right)$. Substitute for k and $T_2(\gamma) = d$ to get

$$\begin{aligned}
T_2(n) &= 3^{\log_2\left(\frac{n}{\gamma}\right)} \cdot d + 2 \cdot b \cdot n \cdot \left[\left(\frac{3}{2}\right)^{\log_2\left(\frac{n}{\gamma}\right)} - 1\right] \\
&= 3^{\log_2\left(\frac{n}{\gamma}\right)} \cdot d + 2 \cdot b \cdot n \cdot \left[3^{\log_2\left(\frac{n}{\gamma}\right)} \cdot \frac{\gamma}{n} - 1\right] \\
&= 3^{\log_2\left(\frac{n}{\gamma}\right)} \cdot (d + 2 \cdot b \cdot \gamma) - 2 \cdot b \cdot n
\end{aligned} \tag{4}$$

Now that we have obtained closed-form solutions for $T_1(n)$ and $T_2(n)$, let us simplify (2) and (5) by using $\gamma = \lambda = 1$

$$
\begin{aligned}
T_1(n) &= 4^{\log_2(n)} \cdot c + a \cdot n \cdot \left(2^{\log_2(n)} - 1\right) \\
&= n^2 \cdot (a + c) - a \cdot n \qquad (5) \\
T_2(n) &= 3^{\log_2(n)} \cdot (d + 2 \cdot b) - 2 \cdot b \cdot n \qquad (6)
\end{aligned}
$$

Let us further simplify (5) and (6) by assuming that $n = 2^m$ for some natural number m.

$$
\begin{aligned}
T_1(2^m) &= (2^m)^2 \cdot (a + c) - a \cdot 2^m \\
&= 4^m \cdot (a + c) - a \cdot 2^m \qquad (7) \\
T_2(2^m) &= 3^{\log_2(2^m)} \cdot (d + 2 \cdot b) - 2 \cdot b \cdot 2^m \\
&= 3^m \cdot (d + 2 \cdot b) - b \cdot 2^{m+1} \qquad (8)
\end{aligned}
$$

Now inspecting (7) and (8), it is easier to see which time complexity has the higher growth rate.

Let us formalize by saying that

$$
\lim_{m \to \infty} \frac{T_2(2^m)}{T_1(2^m)} = 0
$$

Thus, the algorithm associated with $T_2(n)$ has a much better time complexity than the algorithm associated with $T_1(n)$.