ps4

Siwei Tang, Yuanyue Qiao, Yuhan Zhang

March 2020

1 Analyzing nested loops

b) We will split the function as loop1 and loop2 to get our running time analysis

For the outer loop(loop 1), it's simply a loop that runs n times that starts from i = 1 to i = n
We can express the running time analysis of outer loop as:
$T(n) = \sum_{i=1}^{n}$ running time of inner loop

In the inner loop, the value of j will be:
$j = 1, 3, 9, 27....$ which is same thing as
$j = 3^0, 3^1, 3^2....3^k$, where k is the number of executions of loop 2.
Then assume the loop 2 stops when j = 3^k
$\Rightarrow j = 3^K \geq$i
$\Rightarrow k \geq \log_3 i$
$\Rightarrow k = \lceil \log_3 i \rceil$

In the loop there are two basic steps, so the total running time is
$\Rightarrow \sum_{i=1}^{n} 2 * \lceil \log_3 i \rceil$

c) By fact 1,$\forall x \in \mathbb{R}$, x$\leq$ $\lceil x \rceil$ $\leq$x+1
$\sum_{i=1}^{n} 2 * \log_3 i \leq \sum_{i=1}^{n} 2 * \lceil \log_3 i \rceil \leq \sum_{i=1}^{n} 2 * (\log_3 i + 1)$

Lower bound:$\sum_{i=1}^{n} 2 * \log_3 i$
$\Rightarrow \sum_{i=1}^{n} 2 * \log_3 i = 2 * \log_3(1 * 2 * 3 * 4.... * n) = 2 * \log_3(n!)$
$2\log_3(n!) = \frac{2\ln n!}{\ln 3}$
by part a) and fact 2
$n! \in \theta(e^{n \ln n - n + \frac{1}{2}\ln n})$
$\Rightarrow \ln n! \in \theta(n \ln n - n + \frac{1}{2}\ln n)$
$\Rightarrow \ln n! \in O(n \ln n - n + \frac{1}{2}\ln n) \wedge \ln n! \in \Omega(n \ln n - n + \frac{1}{2}\ln n)$
We want to show $\frac{2\ln n!}{\ln 3} \in O(n \ln n - n + \frac{1}{2}\ln n) \wedge \frac{2\ln n!}{\ln 3} \in \Omega(n \ln n - n + \frac{1}{2}\ln n)$

First, we want to show $\frac{2\ln n!}{\ln 3} \in O(n \ln n - n + \frac{1}{2}\ln n)$
That is $\exists c_1 \in \mathbb{R}^+, \exists n_1 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_1 \Rightarrow \frac{\ln n!}{\ln 3} \leq c_1 \cdot (n \ln n - n + \frac{1}{2}\ln n)$
by definition, $\ln n! \in O(n \ln n - n + \frac{1}{2}\ln n) \Rightarrow \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow \ln n! \leq c \cdot (n \ln n - n + \frac{1}{2}\ln n)$
$Let\ c_1 = \frac{2c}{\ln 3},\ n_1 = n_0, let\ n \in \mathbb{N},\ assume\ n \geq n_0$
$\Rightarrow \frac{2\ln n!}{\ln 3} \leq \frac{2c}{\ln 3} \cdot (n \ln n - n + \frac{1}{2}\ln n)$
$\Rightarrow \frac{2\ln n!}{\ln 3} \leq c_1 \cdot (n \ln n - n + \frac{1}{2}\ln n)$
$\Rightarrow \frac{2\ln n!}{\ln 3} \in O(n \ln n - n + \frac{1}{2}\ln n)$
Second, we want to show $\frac{2\ln n!}{\ln 3} \in \Omega(n \ln n - n + \frac{1}{2}\ln n)$
That is $\exists c_2 \in \mathbb{R}^+, \exists n_2 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_2 \Rightarrow \frac{2\ln n!}{\ln 3} \geq c_2 \cdot (n \ln n - n + \frac{1}{2}\ln n)$
by definition, $\ln n! \in \Omega(n \ln n - n + \frac{1}{2}\ln n) \Rightarrow \exists c' \in \mathbb{R}^+, \exists n_0' \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0' \Rightarrow \ln n! \geq c' \cdot (n \ln n - n + \frac{1}{2}\ln n)$

$Let\ c_2 = \frac{2c'}{\ln 3},\ n_2 = n_0', let\ n \in \mathbb{N},\ assume\ n \geq n_0'$
$\Rightarrow \frac{2\ln n!}{\ln 3} \geq \frac{2c'}{\ln 3} \cdot (n\ln n - n + \frac{1}{2}\ln n)$
$\Rightarrow \frac{2\ln n!}{\ln 3} \geq c_2 \cdot (n\ln n - n + \frac{1}{2}\ln n)$
$\Rightarrow \frac{2\ln n!}{\ln 3} \in \Omega(n\ln n - n + \frac{1}{2}\ln n)$

$\Rightarrow \frac{2\ln n!}{\ln 3} \in \theta(n\ln n - n + \frac{1}{2}\ln n)$

by asymptotic property, $-n \in O(n\ln n) \wedge \frac{1}{2}\ln n \in O(n\ln n) \Rightarrow -n + \frac{1}{2}\ln n \in O(n\ln n)$
$\Rightarrow n\ln n - n + \frac{1}{2}\ln n \in O(n\ln n) \Rightarrow \frac{2\ln n!}{\ln 3} \in O(n\ln n)$
Since $n\ln n - n + \frac{1}{2}\ln n \in \Omega(n\ln n)$
$\Rightarrow n\ln n - n + \frac{1}{2}\ln n \in \theta(n\ln n)$

Upper bound:$\sum_{i=1}^{n} 2 * (\log_3 i + 1)$
$\Rightarrow \sum_{i=1}^{n}(2 * \log_3 i + 2) = 2n + 2 * \log_3 n! = 2n + 2 * \frac{\ln n!}{\ln 3}$
by part b) and fact 2
$n! \in \theta(e^{n\ln n - n + \frac{1}{2}\ln n})$
$\Rightarrow$ (Similar to above)$\frac{2\ln n!}{\ln 3} \in \theta(n\ln n - n + \frac{1}{2}\ln n)$
by asymptotic property, since n $\in O(n\ln n - n + \frac{1}{2}\ln n)$
$\Rightarrow 2n + 2 * \frac{\ln n!}{\ln 3} \in O(n\ln n - n + \frac{1}{2}\ln n)$
Since $2n + 2 * \frac{\ln n!}{\ln 3} \in \Omega(n\ln n - n + \frac{1}{2}\ln n)$
$\Rightarrow 2n + 2 * \frac{\ln n!}{\ln 3} \in \theta(n\ln n - n + \frac{1}{2}\ln n)$
by asymptotic property, n $\in O(n\ln n)$, also we have proved $\frac{2\ln n!}{\ln 3} \in O(n\ln n)$
$\Rightarrow 2n + 2 * \frac{\ln n!}{\ln 3} \in O(n\ln n)$
Since we have proved $2 * \frac{\ln n!}{\ln 3} \in \Omega(n\ln n) \Rightarrow 2n + 2 * \frac{\ln n!}{\ln 3} \in \Omega(n\ln n)$
$\Rightarrow 2n + 2 * \frac{\ln n!}{\ln 3} \in \theta(n\ln n)$

Therefore the theta bound of print_three is $\theta(n\ln)$

2 Odds and Evens

a)
Let n $\in \mathbb{N}$, $Let\ nums\ be\ an\ arbitrary\ list\ of\ length\ n$
$The\ first\ step\ line\ 6\ counts\ as\ 1\ basic\ step.\ In\ loop\ 1,\ line\ 8,\ counts\ as\ 1\ basic\ step$
Loop 1(outside loop) will execute at most from i = n to i = 0 by 1, so it will execute at most n + 1 times
For the loop 2(inner loop), it will execute from i = 0 to i -1, so in total at most T(n) is
$T(n) = 1 + \sum_{i=0}^{n}(i + 1) = n + 2 + \sum_{i=1}^{n} i = n + 2 + \frac{n(n+1)}{2} = \frac{n^2+3n+4}{2}$
by asymptotic property, $\frac{3n}{2} \in O(n^2) \Rightarrow T(n)\frac{n^2+3n+4}{2} \in O(n^2)$
$\Rightarrow O(n^2)\ is\ the\ tight\ upper\ bound\ on\ the\ worst\ running\ time\ of\ longest\_even\_prefix$

b)
$Let\ n \in \mathbb{N}$
$Let\ nums = [2, 2, ..., 1, ..., 2]\ with\ length\ n$
Except the element at index $\lfloor \frac{n}{2} \rfloor$ is 1, the other elements are all 2
Since Loop 1, the outer loop starts from i = n to i = 1, loop 2 (the inner loop) starts from j= 0 to j = i -1
Since the integer at index $\lfloor \frac{n}{2} \rfloor$ is odd, so when the outer loop executes from i = n to i = $\frac{n}{2} + 1$
The range of j is from j = 0 to j = $\frac{n}{2}$, since the odd integer 1 is contained in the range,
so the inner loop will stops as soon as j reaches $\lfloor \frac{n}{2} \rfloor$
When the outer loop executes to i = $\frac{n}{2}$, the range of j is from 0 to $\frac{n}{2} - 1$, so all possible values of j are even integers
if statement will not executes, when loop 2 finishes, `found_odd` = False, then it goes into the return statement
Line 8 counts as 1 basic step inside loop 1, the return statement counts as 1 basic step
So the running time T(n) is $1 + \sum_{i=\frac{n}{2}+1}^{n}(\frac{n}{2}+1) + 1 + \frac{n}{2} + 1 = 2 + \frac{n}{2} + (\frac{n}{2}+1) * (n - \frac{n}{2} - 1 + 1) = 2 + \frac{n}{2} + (\frac{n}{2}+1) * \frac{n}{2}$
$\Rightarrow T(n) = \frac{n^2}{4} + n + 2$

by asymptotic property, since $\frac{n^2}{4} \in \Omega(n^2) \Rightarrow T(n) = \frac{n^2}{4} + n + 2 \in \Omega(n^2)$
$\Rightarrow O(n^2)$ *is the asymptotic lower bound on the worst running time of longest_even_prefix*

c)
Let n $\in \mathbb{N}$, *let nums be arbitrary lists of integers and* $len(nums) = n$
Consider the following proof by case analysis.
Case 1: all the integers are even
So at the first executions of loop 1 (the outer loop), i = n
the loop 2 (the inner loop), the range of j is from 0 to n - 1
since j are all even integers, if statement will not execute, the loop 2 will run n times
when loop 2 finishes, `found_odd` = False, goes into the return statement
$\Rightarrow T(n) = n + 1 \in \Omega(n)$

Case 2: assume nums has odd integers, and the first odd integer is at index k, where k $\in \mathbb{N}$
Since Loop 1, the outer loop starts from i = n to i = 1, loop 2 (the inner loop) starts from j= 0 to j = i -1
Since the integer at index k is odd, so when the outer loop executes from i = n to i = $k + 1$
The range of j is from j = 0 to j = k, since the odd integer is contained in the range of j,
so the inner loop will stops as soon as j reaches k, so loop 2 executes k + 1 times every execution of loop 1
When the outer loop executes to i = k, the range of j is from 0 to $k-1$, so all possible values of j are even integers
if statement will not execute, when loop 2 finishes, `found_odd` = False, then it goes into the return statement
Line 8 counts as 1 basic step inside loop 1, the return statement counts as 1 basic step
the running time for i = k is 1 + k - 1 - 0 + 1 + 1 = k+ 2
$\Rightarrow T(n) = \sum_{i=k+1}^{n}(k+2) + k + 2 = (k+2)(n-k) + k + 2 = kn - k^2 + 2n - 2k + k + 1 = (k+2)n - k^2 - k + 1$
Since $-k^2 - k + 1$ are all constant and $(k+2)n \in \Omega(n) \Rightarrow T(n) = (k+2)n - k^2 - k + 1 \in \Omega(n)$

$\Rightarrow$ the running time of `longest_even_prefix(nums)` *is* $\Omega(n)$

3 Unpredictable loop variables

a)
Let n $\in \mathbb{N}$, the input family should be all the lists of length n with all the elements are negative integers.

At first, i = 0, j = 1, i < n, since integer at index 0 is negative
so goes into the else branch,the first element is greater than 0, i = 0, j = 2
Then since the first element becomes greater than 0, the loop starts from i = 0 again, j = 2
so if statement is satisfied, i = 0 + j = 0 + 2 = 2,
then goes into else branch since the element at index 2 is smaller than 0
the element at index 2 becomes greater than 0, i = 0, j = 2 * 2 = 4
Then the loop will start again from i = 0, if statement is satisfied and i = 0 + 4 = 4, then goes into else branch since since the element at index 4 is smaller than 0
the element at index 4 becomes greater than 0, i = 0, j = 4 * 2 = 8
Then, similar like before, the loop will start over with i = 0 and larger j again and again

Since all the elements in the lst is smaller than 0, so before while loop ends(i $\geq$ n),
as soon as the if statement is satisfied, then it will goes into else branch, j becomes larger.

$First: (i = 0) \Rightarrow (else\ branch,\ i = 0) \Rightarrow (if\ statement,\ i = 2) \Rightarrow (else\ branch,\ i = 0) \Rightarrow (if\ statement,\ i = 4 = 2^2) \Rightarrow (else\ branch,\ i = 0) \Rightarrow (if\ statement,\ i = 8 = 2^3)$

It will loop over again and again until i $\geq$ n, ignore the steps that i goes back to 0, then it becomes $i = 0 \Rightarrow i = 2 \Rightarrow i = 4 \Rightarrow i = 8 \Rightarrow i = 16 \Rightarrow ...$

i will increases by j and j will multiply by 2 every two loop executions
then assume the while loop ends in 2k loop iterations, so i multiply by 2 in k times, $i = 2^k$
Since loop ends, $i = 2^k \geq n \Rightarrow k \geq \log n \Rightarrow k = \lceil \log n \rceil$
Each loop iteration takes at most 3 steps, so in total at most 2k *3 = 6k = $6\lceil \log n \rceil \leq 6 \log n + 6$ steps
$\Rightarrow$ Running time is $O(\log n)$.
Each loop iteration takes at least 1 step, so in total at least 2k *1 = 2k = $2\lceil \log n \rceil \geq 2 \log n$ steps
$\Rightarrow$ Running time is $\Omega(\log n)$.

In conclusion, the running time is $\theta(\log n)$.

b)
Let n $\in \mathbb{N}$, the input family should be all the lists of length n that the first half are all non-negative integers and the second half are negative integers

Assume n is the power of 2 $\Rightarrow \exists k \in \mathbb{Z}^+, n = 2^k$
The length of first and second half is $\frac{n}{2} = 2^k * \frac{1}{2} = 2^{k-1}$

For the first half, from index 0 to index $2^{k-1} - 1$, all the integers are greater than or equal to 0,
the loop always goes in to the if statement
i increases by 1 (j =1) until i reaches $2^{k-1}-1$, so the loop will iterate $\frac{n}{2}$ times in the first half, so in total $\frac{n}{2}$ steps

For the second half, from index 2^{k-1} *to* $2^k - 1$
when i = 2^{k-1}, the element at index i is negative integer, so it starts goes into else branch
so integer at index i becomes positive, i = 0, j = 2, and then when i $\leq 2^{k-1}$
i increases by 2, since the first half are all non-negative integers,so it will stay in if branch when i $\leq 2^{k-1}$, runs $\frac{n}{4}$ steps
When i increases to greater than 2^{k-1}, i = 0, j = 4, then like before, i increases by 4 (j =1), runs $\frac{n}{8}$ steps
between index 0 and index 2^{k-1}, when i is greater than 2^{k-1}, i would becomes 0, j would be 8
the else branch will start over and over again until j $\geq 2^{k-1}$,
since j increases by multiplying by 2, so when j $< 2^{k-1}$, i will always increases by j, reaches into the second half of the list, then i becomes 0 and j becomes larger (multiplying by 2), after j becomes 2^{k-1},
i would first become 0, then i = i + j = 2^{k-1}, since element at index 2^{k-1} is positive, so i = i + j = $2^k = n$
The loop stops since i = n.

Since the number of executions of else branch is the same as powers of j when loop stops
assume power of j is x, j = 2^x
$\Rightarrow 2^x \geq 2^{k-1} = \frac{n}{2} \Rightarrow x \geq \log n \Rightarrow x = \lceil \log n \rceil \geq \log n \Rightarrow x \in \Omega(\log n)$
So the else branch will executes $\Omega(\log n)$ times.

For the total running time, the if branch will executes $\frac{n}{2}$ times in the first half
In the second half, the if branch will executes $\frac{n}{4} = 2^{k-2}$ after the first execution of else branch since j is 2, i increases by 2.
After the second execution of else branch j is 4, i increases by 4, so if branch executes $\frac{n}{8} = 2^{k-3}$ times
so the total times of if branch executes is
$\sum_{i=0}^{k-1} 2^i = \frac{1-2^k}{1-2} (by formula) = 2^k - 1 = n - 1$
Line 2 to line 4 counts as 3 basic steps
so the total run time $T(n) = 3 + 1 * (n-1) + 2 * \lceil \log n \rceil \leq n + 2 + 2(\log n + 1) = n + 4 + 2\log n$
by asymptotic property, $\log n \in O(n) \Rightarrow n + 4 + 2\log n \in O(n) \Rightarrow T(n) \in O(n)$
$T(n) = 3 + 1 * (n-1) + 2 * \lceil \log n \rceil \geq n + 2 + 2\log n \Rightarrow T(n) \in \Omega(n)$

$\Rightarrow T(n) \in \theta(n)$

c)
We want to show the worst case running time is O(n)
Let $n \in \mathbb{N}$, let lst be an arbitrary list of integers of length n

Since j increases by multiplying by 2, so assume when the while loop stops, j = 2^f $where$ $f \in \mathbb{N}$
Since it is worst case, assume that the else branch executes f times, stops when $2^f \geq n$
$\Rightarrow f = \lceil \log_2 n \rceil \Rightarrow f \leq \log_2 n + 1$
Every time the else branch executes, i = 0, the loop starts from i = 0 again
i will first increase by j = 1 = 2^0 in first execution of if branch
after second execution of else branch, i = 0, j = 2, then goes into the if branch
i will increase by 2, Similarly, in third execution, in if branch, i will increase from to to 4 by j = 4
So in the f'th execution, in if branch, i will increase by j = 2^f
Since lst may have positive integer elements, so the if branch may execute some times, assume it is c times
Since line 2 to line 4 counts as 3 basic steps
Since in if branch, i increases by j, so if branch runs at most $\lceil \frac{n}{j} \rceil$ every time after else branch executes
The else branch executes f times, so that is $3f$ steps, so in total
$T(n) = 3f+c+3+\sum_{i=0}^{f} \lceil \frac{n}{2^i} \rceil \leq 3f+c+3+\sum_{i=0}^{f} (\frac{n}{2^i}+1) = 4f+4+c+n*\sum_{i=0}^{f} \frac{1}{2^i} = 4f+4+c+n*\frac{1-(\frac{1}{2})^{f+1}}{1-\frac{1}{2}}$
$\Rightarrow T(n) \leq 4f+4+c+2n-2n*\frac{1}{2^{f+1}}$
Since $f \leq \log_2 n + 1 \Rightarrow T(n) \leq 4\log_2 n + 3 + 4 + c + 2n - 2n * \frac{1}{2^{\log_2 n + 1}}$
$\Rightarrow T(n) \leq 4\log_2 n + 7 + c + 2n - 2n * \frac{1}{2*2^{\log_2 n}} = 4\log_2 n + 7 + c + 2n - 2n * \frac{1}{2*n} = 4\log_2 n + 6 + c + 2n$
by asymptotic property, $4\log_2 n \in O(n)$ and c+6 is constant
$\Rightarrow T(n) = 4\log_2 n + 6 + c + 2n \in O(n)$

4 An average-case analysis

a) We want to prove $\forall n \in \mathbb{Z}^+, \forall k \in \mathbb{N}, \frac{n}{2^k} - \frac{2^k-1}{2^k} \leq x_k \leq \frac{n}{2^k},$ prove by induction on k
$Let\ n \in \mathbb{Z}^+,$ define P(k) as $\frac{n}{2^k} - \frac{2^k-1}{2^k} \leq x_k \leq \frac{n}{2^k},$ where k $\in \mathbb{N}$

Base Case:
Let k = 0, we want to show $\frac{n}{2^0} - \frac{2^0-1}{2} \leq x_0 \leq \frac{n}{2^0},$ that is $x_0 = n$
by code, $x = x_0 = n,$ the initial value of x is n
Let k = 1, we want to show $\frac{n}{2} - \frac{2-1}{2} \leq x_k \leq \frac{n}{2}$
by code, $x_1 = \lfloor \frac{n}{2} \rfloor$
by fact, $\frac{n-1}{2} \leq \lfloor \frac{n}{2} \rfloor \leq \frac{n}{2}$
Thus, the base case is proved

Inductive step: $Let\ m \in \mathbb{N},\ let\ m \geq 1,$ assume P(m) is True.
That is $\frac{n}{2^m} - \frac{2^m-1}{2^m} \leq x_m \leq \frac{n}{2^m}$
We want to show P(m+1) is True, that is $\frac{n}{2^{m+1}} - \frac{2^{m+1}-1}{2^{m+1}} \leq x_{m+1} \leq \frac{n}{2^{m+1}}$
by code, $x_{m+1} = \lfloor \frac{x_m}{2} \rfloor$
by fact, $\frac{x_m-1}{2} \leq (x_{m+1} = \lfloor \frac{x_m}{2} \rfloor) \leq \frac{x_m}{2}$
by Inductive Hypothesis, $x_m \leq \frac{n}{2^m}$
$\Rightarrow \frac{x_m}{2} \leq \frac{n}{2^{m+1}}$
$\Rightarrow \lfloor \frac{x_m}{2} \rfloor \leq \frac{x_m}{2} \leq \frac{n}{2^{m+1}}$

by Inductive Hypothesis, $x_m \geq \frac{n}{2^m} - \frac{2^m-1}{2^m}$
$\Rightarrow \frac{x_m}{2} \geq \frac{n}{2^{m+1}} - \frac{2^m-1}{2^{m+1}}$
$\Rightarrow \frac{x_m-1}{2} \geq \frac{n}{2^{m+1}} - \frac{2^m-1}{2^{m+1}} - \frac{1}{2} = \frac{n}{2^{m+1}} - \frac{2^m-1}{2^{m+1}} - \frac{2^m}{2^{m+1}} = \frac{n}{2^{m+1}} - (\frac{2^m-1}{2^{m+1}} + \frac{2^m}{2^{m+1}}) = \frac{n}{2^{m+1}} - \frac{2^{m+1}-1}{2^{m+1}}$
by fact, $\lfloor \frac{x_m}{2} \rfloor \geq \frac{x_m-1}{2} \geq \frac{n}{2^{m+1}} - \frac{2^{m+1}-1}{2^{m+1}}$

$\Rightarrow \frac{n}{2^{m+1}} - \frac{2^{m+1}-1}{2^{m+1}} \leq x_{m+1} = \lfloor \frac{x_m}{2} \rfloor \leq \frac{n}{2^{m+1}}$
$\Rightarrow$ P(m+1) holds
Thus, the statement is proved

b)
We want to show $\forall n \in \mathbb{Z}^+, \forall k \in \mathbb{N}, (\texttt{convert\_to\_binary}(n)\ \textit{takes exactly k loop iterations}) \Leftrightarrow 2^{k-1} \leq n \leq 2^k - 1$

Part 1: $\textit{Let } n \in \mathbb{Z}^+, \textit{ let } k \in \mathbb{N},$ assume $\texttt{convert\_to\_binary}(n)$ *takes exactly k loop iterations*
We want to show $2^{k-1} \leq n \leq 2^k - 1$

by part a) result, $x_{k-1} \leq \frac{n}{2^{k-1}} \Rightarrow n \geq x_{k-1} \times 2^{k-1}$
To avoid the (k+1)th iteration, x_k should be smaller than or equal to 0, since n > 0,
We know that a positive integer divides by 2 is still positive, so x_k cannot be smaller than 0, so $x_k = 0$
To make $x_k = 0$, x_{k-1} should smaller than 2,since if $x_{k-1} \geq 2, x_k > 0$, so $x_{k-1} = 1$
$\Rightarrow n \geq x_{k-1} \times 2^{k-1} = 2^{k-1}$

by part a) result, $x_{k-1} \geq \frac{n}{2^{k-1}} - \frac{2^{k-1}-1}{2^{k-1}} \Rightarrow n \leq 2^{k-1} \times x_{k-1} + 2^{k-1} - 1$
Since $x_{k-1} = 1 \Rightarrow n \leq 2^{k-1} \times 1 + 2^{k-1} - 1 = 2^k - 1$

$\Rightarrow 2^{k-1} \leq n \leq 2^k - 1$

Part 2: $\textit{Let } n \in \mathbb{Z}^+, \textit{ let } k \in \mathbb{N},$ assume $2^{k-1} \leq n \leq 2^k - 1$
We want to show $\texttt{convert\_to\_binary}(n)$ *takes exactly k loop iterations*
by code,if since every loop iteration n would be divides by 2 until the remaining x = 0
so if $\exists m \in \mathbb{Z}^+, m \geq n$
then if x = m, x will take at least the same number of loop iterations as when x = n
Similarly, if $\exists h \in \mathbb{Z}^+, h \leq n$
then if x = h, x will take at most the same number of loop iterations as when x = n

Check n = $x = 2^{k-1} \Rightarrow x_1 = 2^{k-2} \Rightarrow x_2 = 2^{k-3} \Rightarrow ... \Rightarrow x_{k-1} = 2^{k-k} = 1 \Rightarrow x_k = 0$
when n = $2^{k-1}, \texttt{convert\_to\_binary}(n)$ *takes exactly k loop iterations*

Check n = $x = 2^{k-1} - 1 \Rightarrow x_1 = \lfloor 2^{k-2} - \frac{1}{2} \rfloor = 2^{k-2} - 1 \Rightarrow x_2 = \lfloor 2^{k-3} - \frac{1}{2} \rfloor = 2^{k-3} - 1 \Rightarrow ... \Rightarrow x_{k-1} = \lfloor 2^{k-k} - \frac{1}{2} \rfloor = 2^{k-k} - 1 = 0$
when n = $2^{k-1} - 1, \texttt{convert\_to\_binary}(n)$ *takes exactly* $k-1$ *loop iterations*

Check n = $x = 2^k - 1 \Rightarrow x_1 = \lfloor 2^{k-1} - \frac{1}{2} \rfloor = 2^{k-1} - 1 \Rightarrow x_2 = \lfloor 2^{k-3} - \frac{1}{2} \rfloor = 2^{k-3} - 1 \Rightarrow ... \Rightarrow x_{k-1} = \lfloor 2^{k-k+1} - \frac{1}{2} \rfloor = 2^1 - 1 = 1 \Rightarrow x_k = 0$
when n = $2^k - 1, \texttt{convert\_to\_binary}(n)$ *takes exactly k loop iterations*

Check n = $x = 2^k \Rightarrow x_1 = 2^{k-1} \Rightarrow x_2 = 2^{k-2} \Rightarrow ... \Rightarrow x_{k-1} = 2^{k-k+1} = 2 \Rightarrow x_k = 1 \Rightarrow x_{k+1} = 0$
when n = $2^k, \texttt{convert\_to\_binary}(n)$ *takes exactly* $k+1$ *loop iterations*

We already check that when n = $2^{k-1} - 1,$ function takes k -1 loop iterations
so for positive integers smaller than or equal to $2^{k-1} - 1$
they will have at most the same number of loop iterations as $2^{k-1} - 1$,
so they take at most k -1 loop iterations
Also, we already check that when n = 2^k,
function takes k +1 loop iterations
so for positive integers greater than or equal to 2^k, they will have at least the same number of loop iterations as 2^k,
so they take at least k +1 loop iterations

For n between 2^{k-1} *and* $2^k - 1,$ since these two integers takes exactly k iterations,
so for any positive integers between them, they will take at least k loop iterations and at most k loop iterations
$\Rightarrow$ so they all take k loop iterations

$\Rightarrow \forall n \in \mathbb{Z}^+, \forall k \in \mathbb{N}, (\texttt{convert\_to\_binary}(n) \textit{ takes exactly k loop iterations}) \Leftrightarrow 2^{k-1} \leq n \leq 2^k - 1$

c)
by b) result, we know that there are $2^k - 1 - 2^{k-1} + 1 = 2^{k-1}$ integers such that takes k loop iterations
so the total running time for k loop iterations integers are $k \times 2^{k-1}$
Let $n \in \mathbb{Z}^+$, *let* $I_n = \{1, ..., 2^n - 1\}$, by part b), we know that $2^n - 1$ takes n loop iterations
So the total running time is $\sum_{k=1}^{n} k2^{k-1}$
(by formula) $= \frac{1-2^{n+1}}{(1-2)^2} - \frac{(n+1)2^n}{1-2} = (n+1)2^n + 1 - 2^{n+1} = n2^n + 1 + 2^n - 2 \cdot 2^n = n \cdot 2^n + 1 - 2^n = (n-1)2^n + 1$
So the average running time is $\frac{\sum_{k=1}^{n} k2^{k-1}}{2^n - 1} = \frac{(n-1)2^n + 1}{2^n - 1}$