Design and Analysis of Algorithms
Tutorial 2

童咏昕
北京航空航天大学 计算机学院
yxtong@buaa.edu.cn

Question 1

Give asymptotic upper bounds for T(n) by recursion tree approach. Make your bounds as tight as possible.

(a)

$$T(1) = 1$$
$$T(n) = T\left(\frac{n}{2}\right) + n \qquad if\ n > 1$$

(b)

$$T(1) = T(2) = 1$$
$$T(n) = T(n-2) + 1 \quad if\ n > 2$$

(c)

$$T(1) = 1$$
$$T(n) = T\left(\frac{n}{3}\right) + n \qquad if\ n > 1$$

Question 1

(d)

$$T(1) = 1$$
$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad\qquad if\ n > 1$$

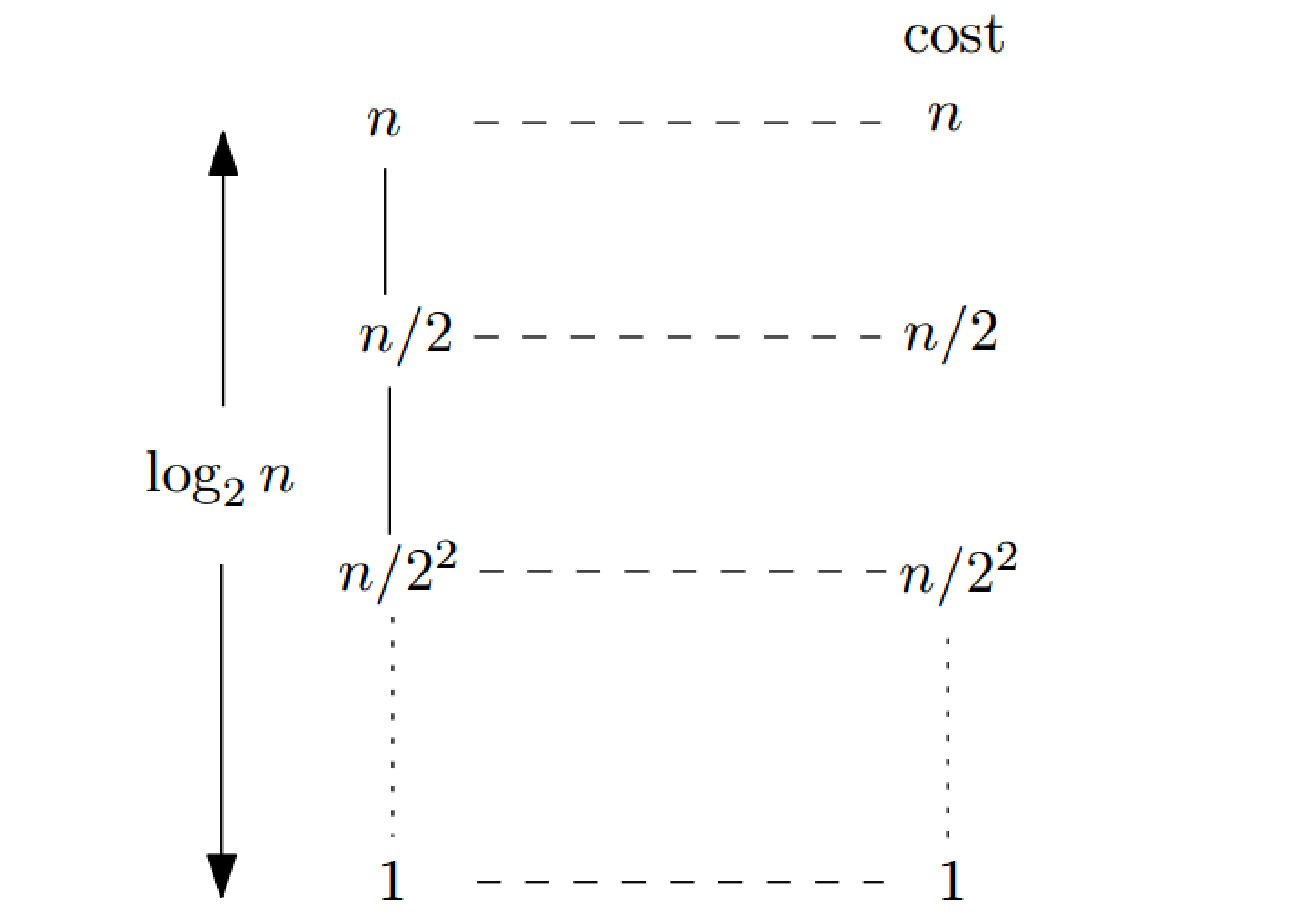
(e)

$$T(1) = 1$$
$$T(n) = 3T\left(\frac{n}{2}\right) + n^2 \qquad\qquad if\ n > 1$$

(f)

$$T(1) = 0, T(2) = 1$$
$$T(n) = T\left(\frac{n}{2}\right) + \log_2 n \qquad\qquad if\ n > 2$$

Solution 1(a)



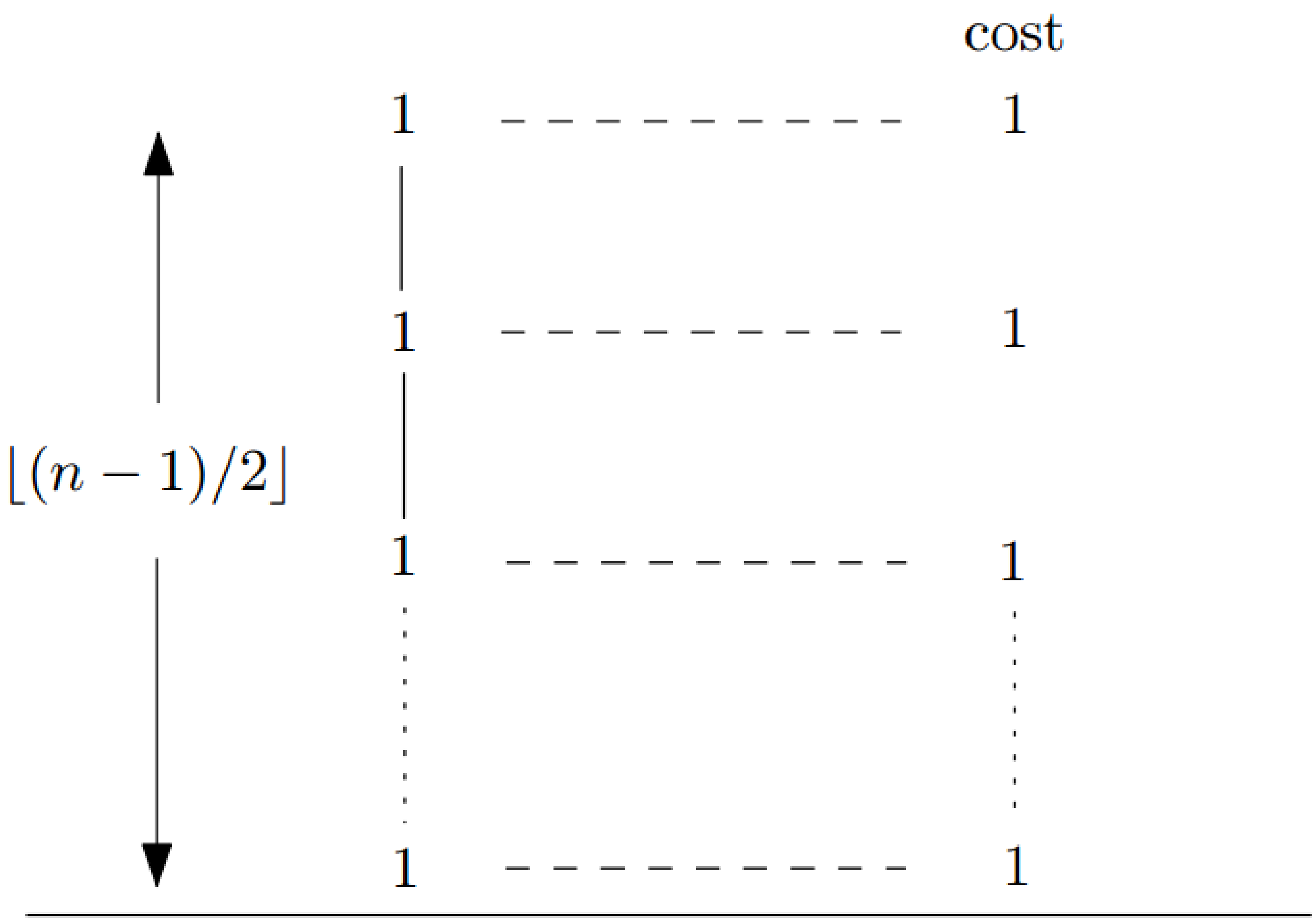
$$\text{total} = n + n/2 + n/2^2 \cdots + 1$$
$$= O(n)$$

Solution 1(a)

Set $h = \log_2 n$

$$
\begin{aligned}
T(n) &= n + T(n/2) \\
&= n + n/2 + T(n/2^2) \\
&= n + n/2 + n/2^2 + T(n/2^3) \\
&\dots \\
&= n + n/2 + n/2^2 + \cdots + n/2^{h-2} + n/2^{h-1} + T(n/2^h) \\
&= n(1 + 1/2 + 1/2^2 + \cdots + 1/2^{h-2} + 1/2^{h-1}) + T(n/2^h) \\
&\leq n(1 + 1/2 + 1/2^2 + \cdots + 1/2^{h-1} + \dots) + T(n/2^h) \\
&= 2 \cdot n + T(1) \\
T(n) &= O(n)
\end{aligned}
$$

Solution 1(b)

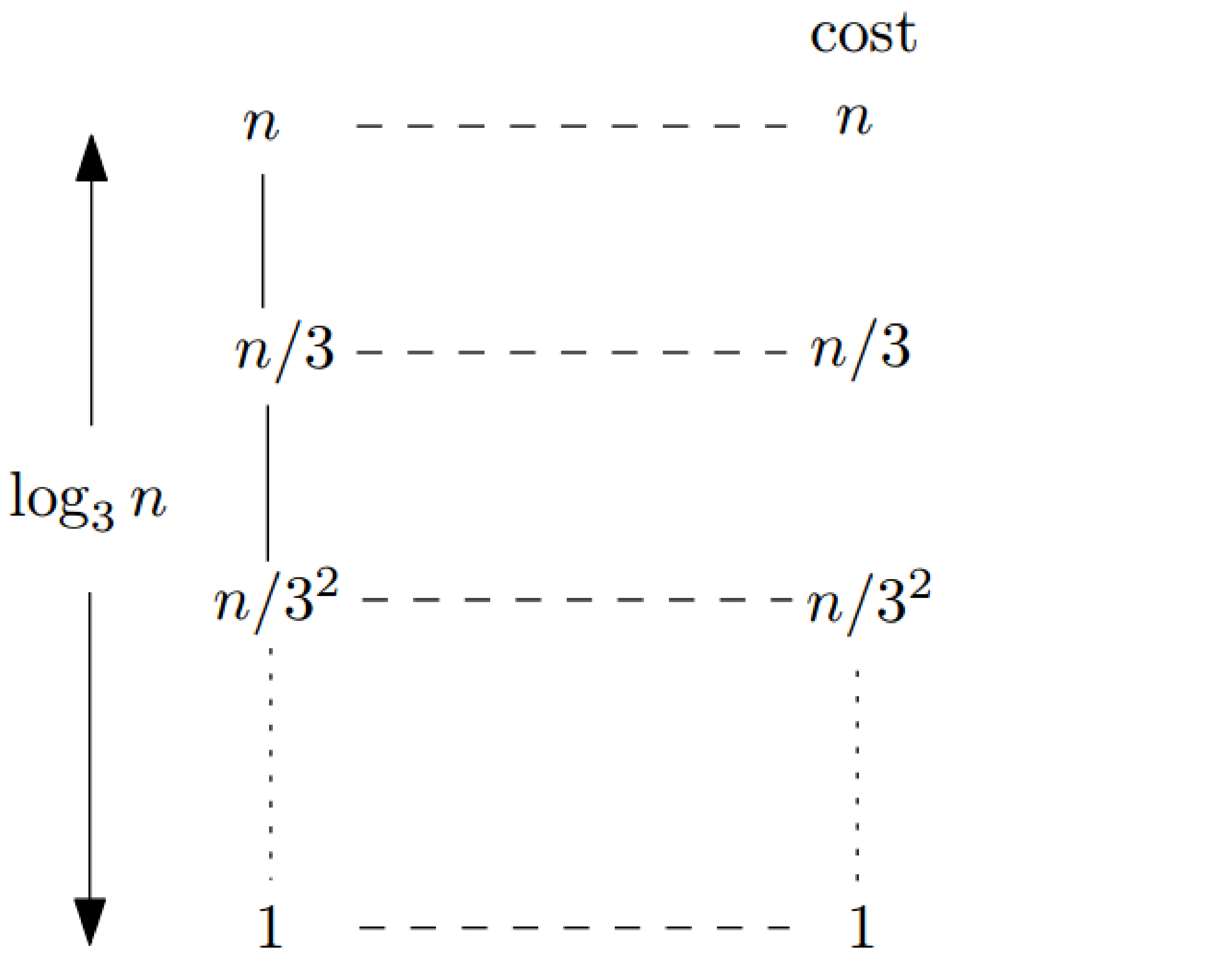


$$\text{total} = 1 + \lfloor (n-1)/2 \rfloor$$
$$= O(n)$$

Solution 1(b)

$$
\begin{aligned}
T(n) &= T(n-2)+1 \\
&= T(n-2\cdot 2)+2 \\
&= T(n-3\cdot 2)+3 \\
&\cdots \\
&= T(n-\lfloor (n-1)/2 \rfloor \cdot 2)+\lfloor (n-1)/2 \rfloor \\
T(n) &= 1+\lfloor (n-1)/2 \rfloor = \lceil (n/2) \rceil = O(n)
\end{aligned}
$$

Solution 1(c)



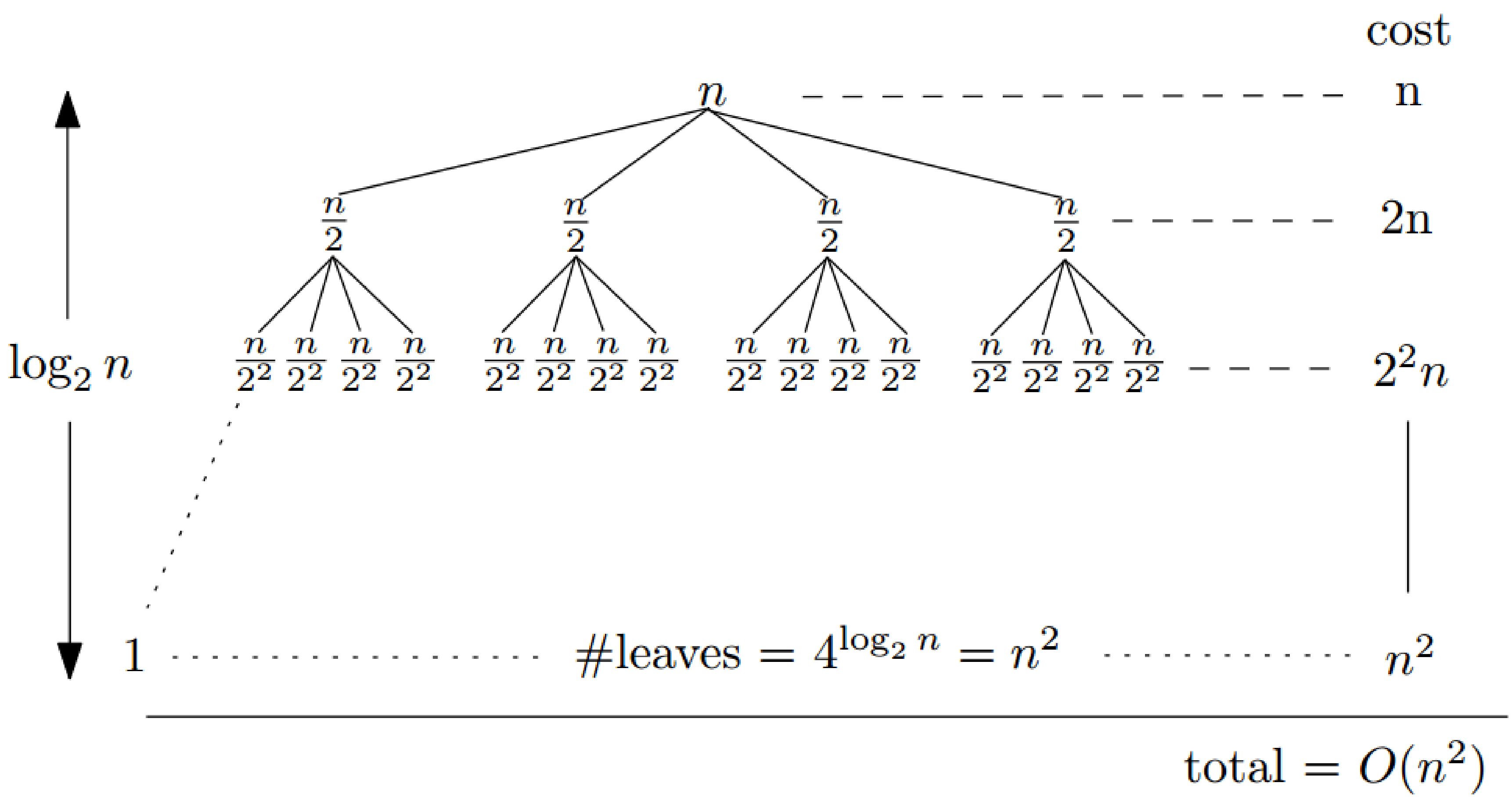
$$\text{total} = n + n/3 + n/3^2 \cdots + 1$$
$$= O(n)$$

Solution 1(c)

Set $h = \log_3 n$

$$
\begin{aligned}
T(n) &= n + T(n/3) \\
&= n + n/3 + T(n/3^2) \\
&= n + n/3 + n/3^2 + T(n/3^3) \\
&\dots \\
&= n + n/3 + n/3^2 + \cdots + n/3^{h-2} + n/3^{h-1} + T(n/3^h) \\
&= n(1 + 1/3 + 1/3^2 + \cdots + 1/3^{h-2} + 1/3^{h-1}) + T(n/3^h) \\
&\le n(1 + 1/3 + 1/3^2 + \cdots + 1/3^{h-1} + \dots) + T(n/3^h) \\
&= 3n/2 + T(1) \\
T(n) &= O(n)
\end{aligned}
$$

Solution 1(d)

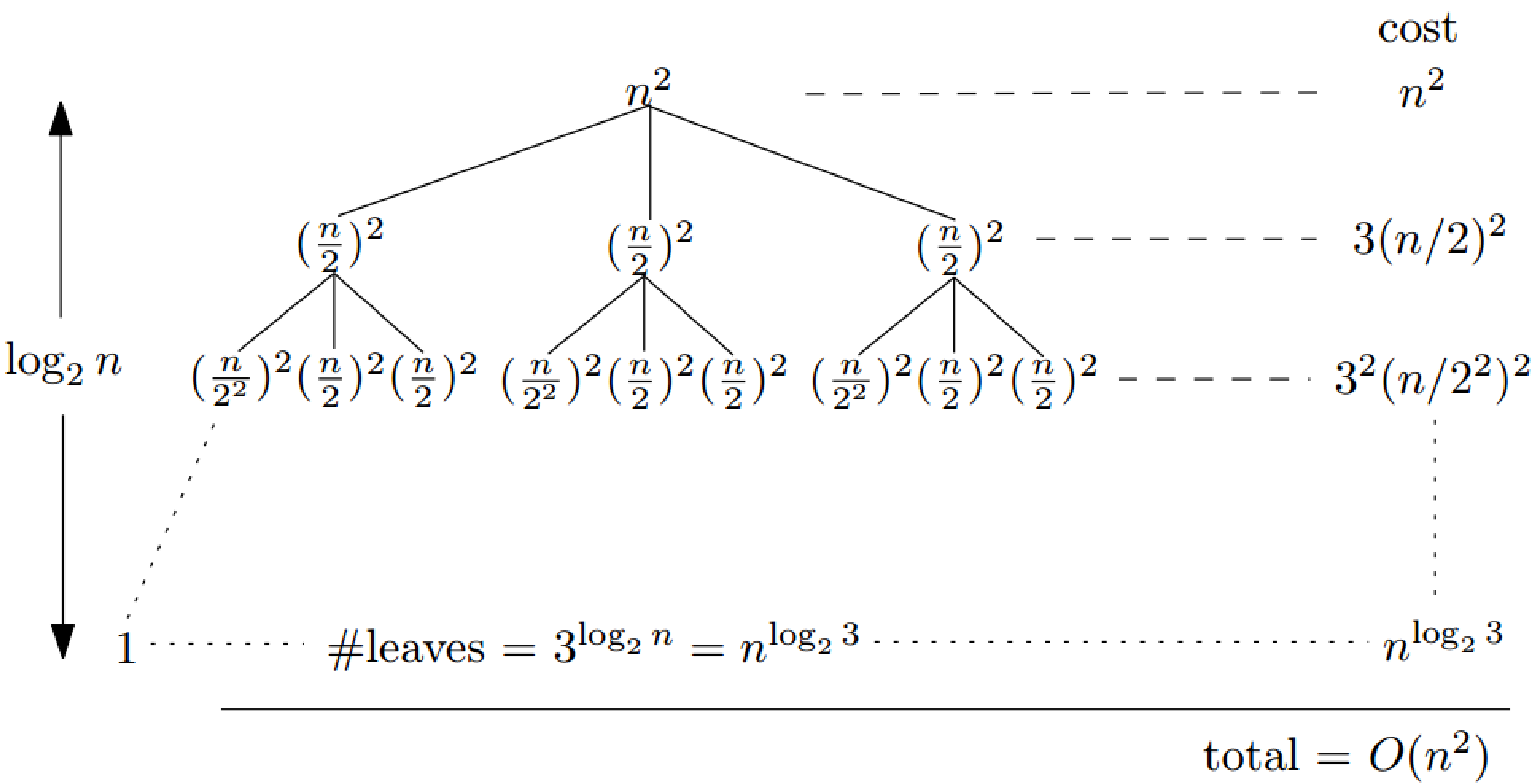

cost
n
n
$\frac{n}{2}$ $\frac{n}{2}$ $\frac{n}{2}$ $\frac{n}{2}$
2n
$\log_2 n$
$\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$ $\frac{n}{2^2}$
$2^2 n$
1
$\#\text{leaves} = 4^{\log_2 n} = n^2$
n^2
$\text{total} = O(n^2)$

Solution 1(d)

Set $h = \log_2 n$

$$
\begin{aligned}
T(n) &= n + 4T(n/2) \\
&= n + 2n + 4^2 T(n/2^2) \\
&= n + 2n + 2^2 n + 4^3 T(n/2^3) \\
&\dots \\
&= n + 2n + 2^2 n + \cdots + 2^{h-2} n + 2^{h-1} n + 4^h T(n/2^h) \\
&= n(1 + 2 + 2^2 + \cdots + 2^{h-1}) + 4^h T(n/2^h) \\
&= n \frac{2^h - 1}{2 - 1} + 4^h T(n/2^h) \\
T(n) &= n(n-1) + n^2 T(1) = O(n^2)
\end{aligned}
$$

Solution 1(e)



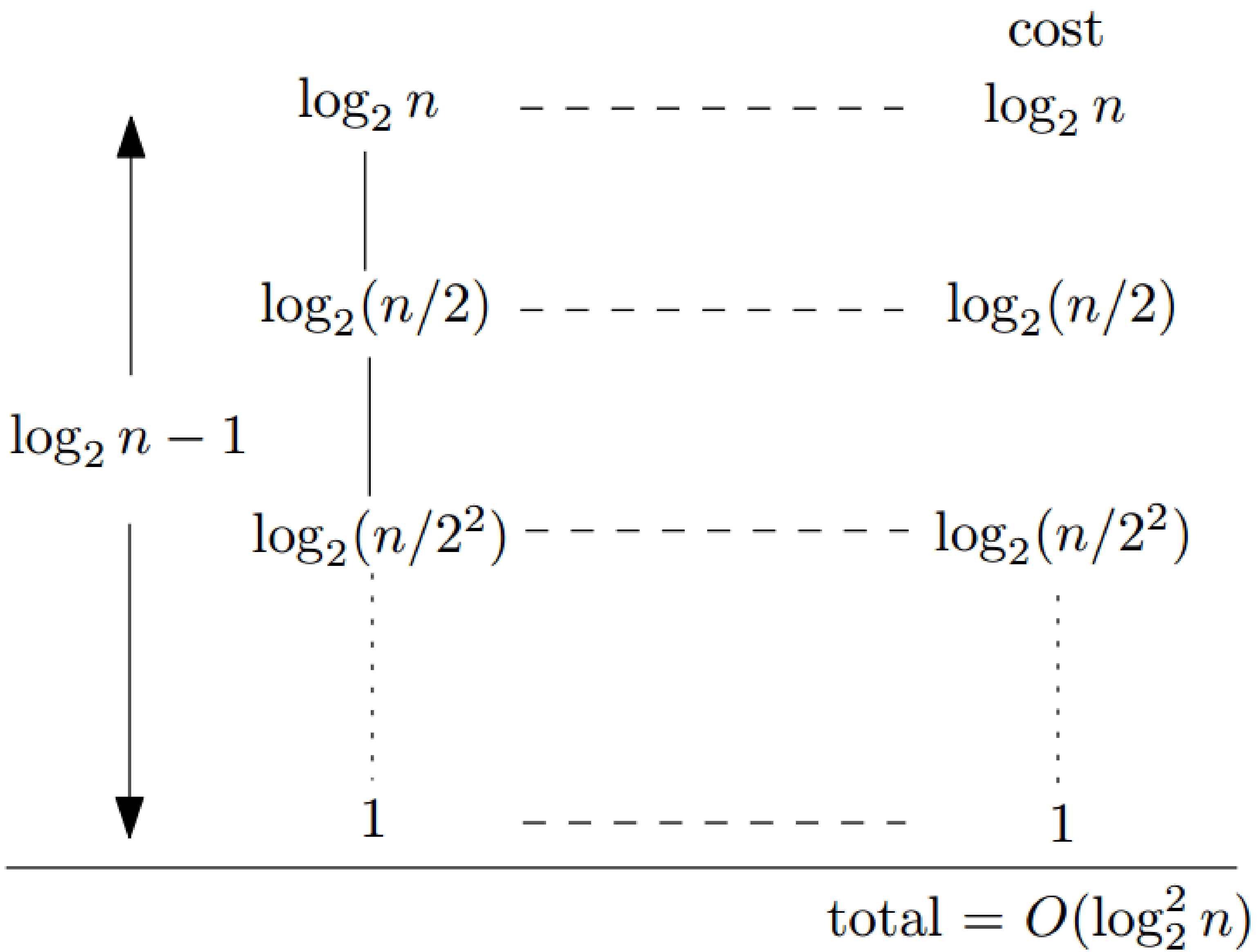
cost
n^2
n^2
$(\frac{n}{2})^2$ $(\frac{n}{2})^2$ $(\frac{n}{2})^2$
$3(n/2)^2$
$\log_2 n$
$(\frac{n}{2^2})^2 (\frac{n}{2})^2 (\frac{n}{2})^2$ $(\frac{n}{2^2})^2 (\frac{n}{2})^2 (\frac{n}{2})^2$ $(\frac{n}{2^2})^2 (\frac{n}{2})^2 (\frac{n}{2})^2$
$3^2(n/2^2)^2$
1
$\#\text{leaves} = 3^{\log_2 n} = n^{\log_2 3}$
$n^{\log_2 3}$
$\text{total} = O(n^2)$

Solution 1(e)

Set $h = \log_2 n$

$$
\begin{aligned}
T(n) &= n^2 + 3T(n/2) \\
&= n^2 + 3(n/2)^2 + 3^2 T(n/2^2) \\
&= n^2 + 3(n/2)^2 + 3^2(n/2^2)^2 + 3^3 T(n/2^3) \\
&\cdots \\
&= n^2 + 3(n/2)^2 + 3^2(n/2^2)^2 + \cdots + 3^{h-2}(n/2^{h-2})^2 \\
&\quad + 3^{h-1}(n/2^{h-1})^2 + 3^h T(n/2^h) \\
&= n^2[1 + 3/4 + (3/4)^2 + \cdots + (3/4)^{h-1}] + 3^h T(n/2^h) \\
&= n^2 \frac{1-(3/4)^h}{1-3/4} + 3^h T(n/2^h) \\
&= 4n^2(1 - n^{\log_2(3/4)}) + 3^h T(n/2^h) \\
&= 4n^2(1 - n^{(\log_2 3 - \log_2 4)}) + 3^h T(n/2^h) \\
&= 4n^2 - 4n^{\log_2 3} + 3^h T(n/2^h) \\
T(n) &= 4n^2 - 4n^{\log_2 3} + n^{\log_2 3} T(1) = O(n^2)
\end{aligned}
$$

Solution 1(f)


cost
$\log_2 n$
$\log_2 n$
$\log_2(n/2)$
$\log_2(n/2)$
$\log_2 n - 1$
$\log_2(n/2^2)$
$\log_2(n/2^2)$
1
1
total $= O(\log_2^2 n)$

Solution 1(f)

Set $h = \log_2 n - 1$

$$
\begin{aligned}
T(n) &= \log_2 n + T(n/2) \\
&= \log_2 n + \log_2(n/2) + T(n/2^2) \\
&= \log_2 n + \log_2(n/2) + \log_2(n/2^2) + T(n/2^3) \\
&\cdots \\
&= \log_2 n + \log_2(n/2) + \log_2(n/2^2) + \cdots + \log_2(n/2^{h-2}) \\
&\qquad + \log_2(n/2^{h-1}) + T(n/2^h) \\
&= h \cdot \log_2 n - [\log_2(2) + \cdots + \log_2(2^{h-2}) + \log_2(2^{h-1})] \\
&\qquad + T(n/2^h) \\
&= h \cdot \log_2 n - [1 + 2 + \cdots + (h-1)] + T(n/2^h) \\
&= h^2 + h - h \cdot (h-1)/2 + T(n/2^h) \\
&= h^2/2 + 3h/2 + T(n/2^h) \\
T(n) &= \frac{(\log_2 n - 1)^2}{2} + 3\frac{\log_2 n - 1}{2} + T(2) = O(\log_2^2 n)
\end{aligned}
$$

Question 2

Prove the following problems by induction.

(a) Given

$$T(1) = 1$$
$$T(n) = T\left(\frac{n}{2}\right) + n \qquad if\ n > 1$$

Prove $T(n) \le c \cdot n$ for some c.

(b) Given

$$T(1) = T(2) = 1$$
$$T(n) = T(n-2) + 1 \qquad if\ n > 1$$

Prove $T(n) \le c \cdot n$ for some c.

(c) Given

$$T(1) = 1$$
$$T(n) = T\left(\frac{n}{3}\right) + n \qquad if\ n > 1$$

Prove $T(n) \le c \cdot n$ for some c.

Question 2

(d) Given

$$T(1) = 1$$
$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad if\ n > 1$$

Prove $T(n) \leq c_1 \cdot n^2 - c_2 \cdot n$ for some c_1 and c_2.

(e) Given

$$T(1) = 1$$
$$T(n) = 3T\left(\frac{n}{2}\right) + n^2 \qquad if\ n > 1$$

Prove $T(n) \leq c_1 \cdot n^2$ for some c.

(f) Given

$$T(1) = 0, T(2) = 1$$
$$T(n) = T\left(\frac{n}{2}\right) + \log_2 n \qquad if\ n > 2$$

Prove $T(n) \leq c \cdot \log^2 n$ for some c.

Solution 2(a)

Base case $n = 1$: $T(1) = 1 \leq c \cdot 1$ for any $c \geq 1$.
Induction:

$$\begin{aligned}
T(n) &= T(n/2) + n \\
&\leq c \cdot n/2 + n \\
&= c \cdot n - c \cdot n/2 + n \\
&= c \cdot n - (c/2 - 1) \cdot n \\
&\leq c \cdot n \qquad \text{for } c \geq 2
\end{aligned}$$

Therefore, $T(n) \leq c \cdot n$ for $n \geq 1$ and $c \geq 2$.

Solution 2(b)

Base case $n = 1$: $T(1) = 1 \le c \cdot 1$ for any $c \ge 1$.
Base case $n = 2$: $T(2) = 1 \le c \cdot 2$ for any $c \ge 1/2$.
Induction:

$$\begin{aligned} T(n) &= T(n-2) + 1 \\ &\le c \cdot (n-2) + 1 \\ &= c \cdot n - 2c + 1 \\ &\le c \cdot n \qquad \text{for } c \ge 1/2 \end{aligned}$$

Therefore, $T(n) \le c \cdot n$ for $n \ge 1$ and $c \ge 1$ (due to the base case $n = 1$).

Solution 2(c)

Base case $n = 1$: $T(1) = 1 \leq c \cdot 1$ for any $c \geq 1$.
Induction:

$$\begin{aligned} T(n) &= T(n/3) + n \\ &\leq c \cdot (n/3) + n \\ &= c \cdot n - 2cn/3 + n \\ &= c \cdot n - (2c/3 - 1)n \\ &\leq c \cdot n \qquad \text{for } c \geq 3/2 \end{aligned}$$

Therefore, $T(n) \leq c \cdot n$ for $n \geq 1$ and $c \geq 3/2$.

Solution 2(d)

Base case $n = 1$: $T(1) = 1 \leq c_1 \cdot 1 - c_2 \cdot 1$ for any $c_1 \geq c_2 + 1$.
Induction:

$$\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\leq 4(c_1 \cdot (n/2)^2 - c_2 \cdot n/2) + n \\
&= c_1 \cdot n^2 - 2c_2 \cdot n + n \\
&= c_1 \cdot n^2 - c_2 \cdot n - (c_2 - 1) \cdot n \\
&\leq c \cdot n^2 - c_2 \cdot n \qquad \text{for } c_2 \geq 1
\end{aligned}$$

Therefore, $T(n) \leq c_1 \cdot n^2 - c_2 \cdot n$ for $n \geq 1$, $c_2 \geq 1$ and $c_1 \geq c_2 + 1$.

Solution 2(e)

Base case $n = 1$: $T(1) = 1 \le c \cdot 1$ for any $c \ge 1$.
Induction:

$$\begin{aligned}
T(n) &= 3T(n/2) + n^2 \\
&\le 3c \cdot (n/2)^2 + n^2 \\
&= 3c \cdot n^2/4 + n^2 \\
&= c \cdot n^2 - (c/4 - 1)n^2 \\
&\le c \cdot n^2 \qquad \text{for } c \ge 4
\end{aligned}$$

Therefore, $T(n) \le c \cdot n$ for $n \ge 1$ and $c \ge 4$.

Solution 2(f)

Base case $n = 1$: $T(1) = 0 \leq c \cdot 0$ for any c.
Base case $n = 2$: $T(2) = 1 \leq c \cdot 1$ for any $c \geq 1$.
Induction:

$$\begin{aligned}
T(n) &= T(n/2) + \log_2 n \\
&\leq c \cdot \log_2^2(n/2) + \log_2 n \\
&= c \cdot (\log_2 n - 1)^2 + \log_2 n \\
&= c \cdot (\log_2^2 n - 2 \cdot \log_2 n + 1) + \log_2 n \\
&= c \cdot \log_2^2 n - 2c \cdot \log_2 n + c + \log_2 n \\
&= c \cdot \log_2^2 n - (c - 1) \cdot \log_2 n - c(\log_2 n - 1) \\
&\leq c \cdot \log_2^2 n \qquad \text{for } c \geq 1 \text{ and } n \geq 2
\end{aligned}$$

Therefore, $T(n) \leq c \cdot \log_2^2 n$ for $n \geq 1$ and $c \geq 1$.

Question 3

Let A[1..n] be an array of positive integers.

Design a divide-and-conquer algorithm for computing the maximum value of $A[j] - A[i]$ with $j \geq i$.

Analyze your algorithm running time.

Solution 3: Idea of the algorithm

Similar to the MCS Maximum Contiguous Subarray Problem If we divide A into two roughly equal size sub-arrays, each with approximately n/2 elements, we observe that the maximum value of A[j] − A[i] must be one of the following:

- The maximum value of A[j] − A[i] in $A[1..\lfloor\frac{n}{2}\rfloor]$ where $1 \leq i \leq j \leq \lfloor\frac{n}{2}\rfloor$
- The maximum value of A[j] − A[i] in $A[\left\lfloor\frac{n}{2}\right\rfloor + 1..n]$ where $\left\lfloor\frac{n}{2}\right\rfloor + 1 \leq i \leq j \leq n$
- The maximum value of A[j] − A[i] where A[i] is the minimum value of $A\left[1..\left\lfloor\frac{n}{2}\right\rfloor\right]$ and A[j] is the maximum value of $A[\left\lfloor\frac{n}{2}\right\rfloor + 1..n]$

Solution 3: Details of the algorithm

Input

A is an array of positive integers
p is the start index of A
r is the end index of A

Output

An integer array $[MaxAll, MinAll, ValAll]$ where
$MaxAll$ and $MinAll$ are the maximum and minimum value of the input array respectively
$ValAll$ is the maximum value of $A[j] - A[i]$ with $j \geq i$

Solution 3: Algorithm FMD: Find Max Diff

FMD(array A, int p, int r)

begin
  if $p = r$ **then** $MaxAll = A[p]; MinAll = A[p]; ValAll = 0;$ // $O(1)$
  else
    $m = \lfloor \frac{p+r}{2} \rfloor;$
    $[MaxL, MinL, ValL] = FMD(A, p, m);$ // $T(\lfloor \frac{n}{2} \rfloor)$
    $[MaxR, MinR, ValR] = FMD(A, m+1, r);$ // $T(\lceil \frac{n}{2} \rceil)$
    $ValM = MaxR - MinL;$ // $O(1)$
    $ValAll = \max(ValL, ValR, ValM);$ // $O(1)$
    $MaxAll = \max(MaxL, MaxR);$ // $O(1)$
    $MinAll = \min(MinL, MinR);$ // $O(1)$
  end
  return $[MaxAll, MinAll, ValAll];$
end

Solution 3: Running time analysis

Let T(n) be the worst-case number of comparisons for problem of size n.

When n = 1, FMD takes constant time. So, T(1) = O(1).

For n > 1, as what we shown in the pseudo-code, FMD performs two recursive calls on partitions of half of the total size of the array and takes constant time for the operations in the combining step. Therefore we have the following recurrence.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$$

To simplify our analysis, we assume n is a power of 2.

Solution 3: Running time analysis

To solve this recurrence equation:

$$T(n) = \begin{cases} O(1), & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + O(1), & n > 1 \end{cases}$$

We use expansion method:

Set $h = \log_2 n$

$$\begin{aligned} T(n) &= 2 \cdot T\left(\frac{n}{2}\right) + c \\ &= 2 \cdot \left(2 \cdot T\left(\frac{n}{4}\right) + c\right) + c \\ &= 4 \cdot T\left(\frac{n}{4}\right) + 2c + c \\ &\dots \\ &= 2^h \cdot T\left(\frac{n}{2^h}\right) + 2^{h-1}c + 2^{h-2}c + \cdots + c \\ &= 2^h \cdot T\left(\frac{n}{2^h}\right) + \frac{c\left(2^h - 1\right)}{2 - 1} \end{aligned}$$

$$T(n) = n \cdot T(1) + (n-1)c = O(n)$$