Problem 1. Find the upper and the lower bounds for $\sum_{k=0}^{n} k^3$ using the integral method. Show your work.

Solution:

$\sum_{k=0}^{n} k^3 \leq \int_0^{n+1} x^3 dx = \frac{x^4}{4}|_0^{n+1} = \frac{(n+1)^4}{4} - \frac{0^4}{4} = \frac{(n+1)^4}{4}$

$\sum_{k=0}^{n} k^3 = \sum_{k=1}^{n} k^3 \geq \int_0^{n} x^3 dx = \frac{x^4}{4}|_0^{n} = \frac{(n)^4}{4} - \frac{0^4}{4} = \frac{n^4}{4}$

Problem 2. Consider that an algorithm has the following recurrence equation:

$$T(n) = 3T\left(\frac{n}{8}\right) + 3n - 1, \quad T(1) = 7$$

Answer the following.

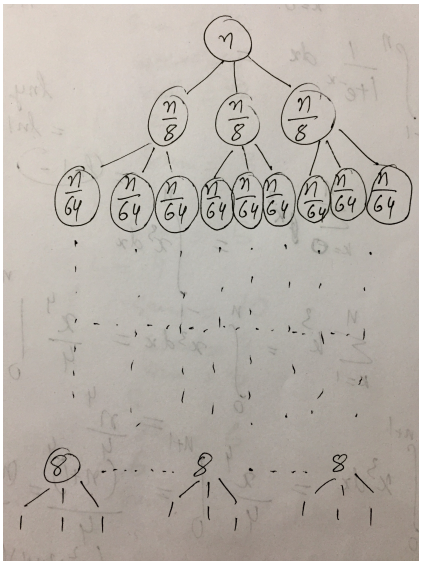
1. Calculate T(64) by hand. Show your work.

Solution:

$T(8) = 3T(1) + 3 \cdot 8 - 1 = 3 \cdot 7 + 24 - 1 = 21 + 24 - 1 = 44$
$T(64) = 3T(8) + 3 \cdot 64 - 1 = 3 \cdot 44 + 192 - 1 = 132 + 192 - 1 = 323$

2. Use the tree method to solve the recurrence exactly, assuming n is a power of 8. For each of the following parts, briefly justify and/or show your work when appropriate.
   (a) Draw the tree. You should show at least three levels at the top and at least two levels at the bottom.

Solution:



(b) What is the height of the tree? (Note that a tree with one node has height of 0, a tree with a root and some children has a height 1, etc.)

Solution: $lg_8 n$

(c) How many leaves are there?

Solution: $3^{lg_8 n}$

(d) What is the total work done by the leaves?

Solution: $3^{lg_8 n} \cdot 7$

(e) What is the size of each subproblem at level i? (Note that the root is at level 0, its children are at level 1, etc.)?

Solution: $\frac{n}{8^i}$

(f) How much work does each subproblem at level i (above the leaves) do?

Solution: $3 \cdot \frac{n}{8^i} - 1$

(g) What is the total work for level i (above the leaves)?

Solution: $3^i \cdot \left(3 \cdot \frac{n}{8^i} - 1\right)$

(h) Write a summation for the total work not including the leaves?

Solution:

$$\sum_{i=0}^{lg_8 n - 1} 3^i \cdot \left(3 \cdot \frac{n}{8^i} - 1\right)$$

(i) Simplify the summation.

Solution:

$$\begin{aligned}
\sum_{i=0}^{lg_8 n-1} 3^i \cdot \left(3 \cdot \frac{n}{8^i} - 1\right) &= 3n \sum_{i=0}^{lg_8 n-1} \left(\frac{3}{8}\right)^i - \sum_{i=0}^{lg_8 n-1} 3^i \\
&= 3n\left(\frac{1-\left(\frac{3}{8}\right)^{lg_8 n}}{1-\frac{3}{8}}\right) - \left(\frac{3^{lg_8 n}-1}{3-1}\right) \\
&= 3n \cdot \frac{8}{5}\left(1 - \frac{3^{lg_8 n}}{8^{lg_8 n}}\right) - \left(\frac{n^{lg_8 3}-1}{2}\right) \\
&= \frac{24n}{5}\left(1 - \frac{n^{lg_8 3}}{n^{lg_8 8}}\right) - \left(\frac{n^{lg_8 3}-1}{2}\right) \\
&= \frac{24n}{5}\left(\frac{n - n^{lg_8 3}}{n}\right) - \left(\frac{n^{lg_8 3}-1}{2}\right) \\
&= \frac{24}{5} \cdot n - \frac{24}{5} \cdot n^{lg_8 3} - \frac{n^{lg_8 3}}{2} + \frac{1}{2} \\
&= -\frac{53}{10} \cdot n^{lg_8 3} + \frac{24}{5} \cdot n + \frac{1}{2}
\end{aligned}$$

(j) What is the total work for the entire algorithm?

Solution: $7 \cdot n^{lg_8 3} - \frac{53}{10} \cdot n^{lg_8 3} + \frac{24}{5} \cdot n + \frac{1}{2} = \frac{17}{10} \cdot n^{lg_8 3} + \frac{24}{5} \cdot n + \frac{1}{2}$

Problem 3. Suppose there is an array of length n. It contains values in the range of $[1 \ldots n+1]$. However, exactly one value out of $\{1, \ldots, n+1\}$ is missing from this array. Find this missing number as efficiently as possible in the following two cases.

Write pseudo code to find the missing number for the two cases. Analyze the algorithms exactly.

1. The numbers in the array are stored in a random order.

Solution:

`Subtract the sum of the values in the array from the sum of the numbers` $[1, \ldots, n+1]$ `to find the missing value.`

`Pseudocode:`

```
sum = A[1]
for i = 2:n:
    sum = sum + A[i]
missing_val = n^2+3n+2/2 − sum
```

Analysis:

$$\sum_{i=2}^{n} 1 = n - 1$$

Runtime: $O(n)$

2. The array is sorted such that the value stored at *index* 1 < value at *index* 2 < ... < value at *index* n.

Solution:

Use binary search to find the smallest value that is **not** equal to its index.
Pseudocode:

```
findMissingVal(A,missing_val,start,end):
    if start > end:
        return missing_val - 1
    else:
        mid = (start + end)/2
        if A[mid] == mid:
            return findMissingVal(A,missing_val,mid+1,end)
        else if A[mid] > mid:
            if A[mid] < missing_val:
                missing_val = mid + 1
            return findMissingVal(A,missing_val,start,mid-1)
```

Analysis:

$$\sum_{i=1}^{lgn} 1 = lgn$$

Runtime: $O(lgn)$