Midterm 1 Solution

Q1

$T(n) = 4n^3 + 2^n + 5n^2 logn$ is $O(n^3)$.

Answer: False

Q2

$T(n) = 4n^3 + 2^n + 5n^2 logn$ is $O(2^n)$.

Answer: True

Q3

$T(n) = 4n^3 + 2^n + 5n^2 logn$ is $O(n^2 log)$.

Answer: True

Q4

Prove that $\Sigma_{i=1}^{n}(2i-1) = n^2$ using weak induction.

Answer:

Base case: When $n = 1$, $2 \times n - 1 = 1$ and $n^2 = 1$.

Inductive hypothesis: Let $k \geq 1$ be an arbitrary integer. Assume that $\Sigma_{i=1}^{k}(2i-1) = k^2$.

Induction step:

$$\Sigma_{i=1}^{k+1}(2i-1)$$
$$= \Sigma_{i=1}^{k}(2i-1) + (2 \times (k+1) - 1)$$
$$= k^2 + 2k + 1$$
$$= (k+1)^2$$

Q5

Let $T(n) = 4n^3 + 2n + nlogn$. Find the tightest Big-O and prove it. You may use any method you'd like.

Answer:

$T(n)$ is $O(n^3)$.

Using limit lemma theorem:

$$\lim_{n\to\infty} \frac{4n^3 + 2n + nlogn}{n^3}$$
$$= \lim_{n\to\infty} 4 + \frac{2}{n^2} + \frac{logn}{n^2}$$
$$= 4$$

$T(n)$ is $\Theta(n^3)$ so the tightest Big-O of $T(n)$ is $O(n^3)$.

Q6

Let $A[0...n-1]$ represent an array with length n and indices 0 to $n-1$. If $i > j$ in $A[i...j]$, then we say A is empty.

Define a function $foo()$ as follows:

```
foo(A[0...n-1]) {
    if n < 2:
        return 1

    i = 1
    while i <= n:
        boo(A[0...i])
        i = i * 2

    a = foo(A[0...n/4])
    b = foo(A[n/4+1...n/2])
    c = foo(A[n/2+1...n-1])
    return a + b + c
}
```

Suppose the function `boo()` 's runtime is $\Theta(m)$, where `m` is the size of the input array for `boo()` .

Questions:

1. (10 points) What is the recurrence for `foo()` ?

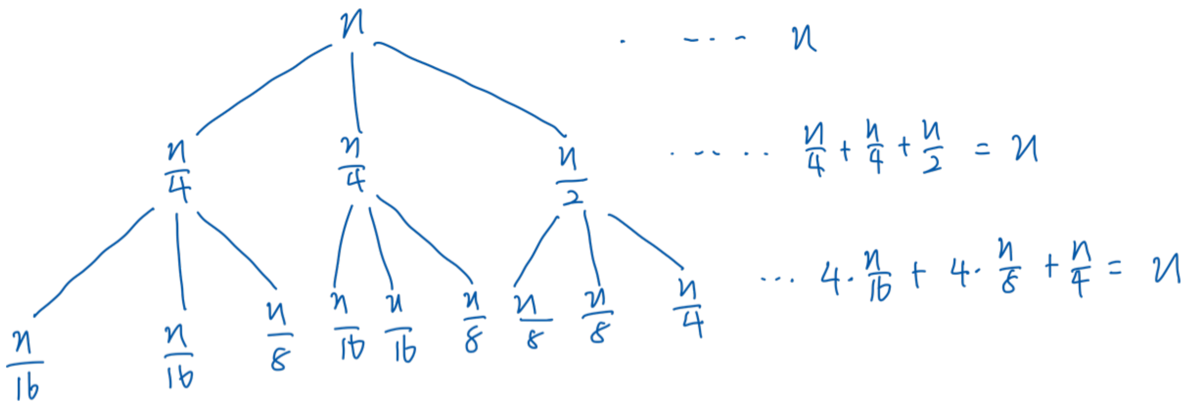
   Answer: $T(n) = 2T(\frac{n}{4}) + T(\frac{n}{2}) + n$

   Explanation:

   - There are 2 recursive calls to `foo()` with subproblem size $\frac{n}{4}$
   - There is 1 recursive call to `foo()` with subproblem size $\frac{n}{2}$
   - Runtime for `boo()` :
     - $$1 + 2 + 4 + 8 + ... + n$$
       $$= 2^0 + 2^1 + 2^2 + ... + 2^{logn}$$
       $$= \frac{1 - 2^{logn+1}}{1-2}$$
       $$= n \times 2 - 1$$

2. (15 points) Solve the recurrence in #1 using recursion tree and specify the Big-O as tight as possible.

Answer:



The longest branch: $n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow ...$

Length of the longest branch: $logn$

$T(n) = nlogn$, so $T(n)$ is $O(nlogn)$

3. (15 points) Verify your Big-O in #2 using substitution method.

   Answer:

   Let $k \geq 1$ be an arbitrary integer. Assume that for all $1 \leq n < k$, $T(n)$ is $O(nlogn)$, i.e., there exists a constant $c > 0$ such that $T(n) \leq cnlogn$.

$$
\begin{aligned}
&T(k) \\
&= 2T(\frac{k}{4}) + T(\frac{k}{2}) + k \\
&\leq 2c\frac{k}{4}log\frac{k}{4} + c\frac{k}{2}log\frac{k}{2} + k \\
&= \frac{ck}{2}logk - ck + \frac{ck}{2}logk - \frac{ck}{2} + k \\
&= cklogk - \frac{3}{2}ck + k
\end{aligned}
$$

   To show $T(k) \leq cklogk$:

$$
\begin{aligned}
&cklogk - \frac{3}{2}ck + k \leq cklogk \\
&k \leq \frac{3}{2}ck \\
&1 \leq \frac{3}{2}c \\
&\frac{2}{3} \leq c
\end{aligned}
$$

   Pick $c = 1$, so $T(k) \leq klogk$ for all $k \geq 1$. $T(n)$ is $O(nlogn)$.

4. (15 points) Can we solve the recurrence using master theorem? If yes, solve it using master theorem. If no, please explain.

   Answer:

   No, the recurrence doesn't follow the form $T(n) = aT(\frac{n}{b}) + f(n)$.