Course: CUS715, Instructor: Dr. Christoforos Christoforou

Problem Set 5 - Solving Recurrences - Substitution methods.

Important: When answering the questions below, make sure you follow the proof-format covered in class. That is, CLEARLY identify each step in the proof, show all algebraic steps, and explain any non-trivial steps.

Problems to solve:

1. Solve the following recurrence using the **Substitution method**. Show all steps.

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2.T(\tfrac{n}{2}) + 1000n \;\; \forall n \geq 2 \end{aligned}$$

Answer:

1st guess

Base case ->

O(n)?

n = 1

T(1) = c * n

1 = c * (1)

c can be 1!

Inductive step ->

For some k, k < n, T(k) <= c * k

T(n) = 2T($\frac{n}{2}$) + 1000n

<= 2(c * ($\frac{n}{2}$)) + 1000n

<= cn + 1000n

cn + 1000n is NOT less than cn (1000n can never be negative)!

2nd guess

Base case ->

O(n^2)?

n = 1

T(1) = c * n^2

1 = c * $(1)^2$

1 = c * 1

c can be 1!

Inductive step ->

For some k, k < n, T(k) <= c * k^2

T(n) = 2T($\frac{n}{2}$) + 1000n

<= 2(c * $(\frac{n}{2})^2$) + 1000n

<= 2(c * ($\frac{n^2}{4}$)) + 1000n

<= ($\frac{cn^2}{2}$) + 1000n

<= cn^2 - $\frac{cn^2}{2}$ + 1000n <= cn^2

If c = 2000, then

2000n^2 - $1000n^2$+ 1000n <= 2000n^2

$1000n^2$+ 1000n <= 2000n^2

$1000n^2$+ 1000n <= 1000n^2 $+ 1000n^2$

The inequality is satisfied, so T(n) is O(n^2).

DONE!

2. Solve the following recurrence using the **Substitution method**. Show all steps.

$$
\begin{aligned}
T(1) &= 1 \\
T(n) &= 7.T(\tfrac{n}{2}) + 18n^2 \ \ \forall n \geq 2
\end{aligned}
$$

Answer:

1st guess

O(n^3)?

Base case ->

n = 1

T(1) = c * n^3

1 = c * $(1)^3$

1 = c * (1)

c can be 1!

Inductive step ->

For some k, k < n, T(k) <= c * k^3

T(n) = 7T($\frac{n}{2}$) + 18n^2

    <= 7(c * ($\frac{n}{2}$)^3)+ 18n^2

    <= $\frac{7cn^3}{8}$+ 18n^2

    <= cn^3 - $\frac{cn^3}{8}$ + 18n^2 <= cn^3

If c = 144, then

144n^3- 18n^3 + 18n^2 <= 144n^3

126n^3 + 18n^2 <= 144n^3

126n^3 + 18n^2 <= 144n^3

126n^3 + 18n^2 <= 126n^3 + 18n^3

The inequality is satisfied, so T(n) is O(n^3).

DONE!

3. Calculate the running time of the following recursive functions. Solve any recurrences that arise using the substitution method.

Answer:

For any input of n, the recursive function will call itself $\log_2 n$ times, and each time it is called it will run through the for loop instructions in, at most, n steps. With each increasing n, the recursive function will call another time and run about another n steps. Thus, T(n) is approximately T($\frac{n}{2}$) + n, and the algorithm is O(n$\log_2 n$).

Substitution proof:

Base case ->

n = 1

T(1) = c * $n \log_2 n$

1 = c * $(1)\log_2 1$

1 = c * (0)

c can be 1 (or greater)!

Inductive step ->

For some k, k < n, T(k) <= c *k$\log_2 k$

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

<= c * $(\frac{n}{2}) \log_2(\frac{n}{2})$ + n <= n$\log_2 n$

<= $(\frac{cn}{2})(\log_2(n) - \log_2(2))$ + n <= n$\log_2 n$

<= $(\frac{cn}{2})(\log_2(n) - 1)$ + n <= n$\log_2 n$

<= $(\frac{cn}{2})(\log_2(n)) - (\frac{cn}{2})$ + n <= n$\log_2 n$

<= $(\frac{c}{2}) \log_2(n) - (\frac{c}{2})$ + 1 <= $\log_2 n$

If c = 2, then

$\log_2(n) - 1 + 1 <= \log_2 n$

$\log_2(n) <= \log_2 n$

The inequality is satisfied, so T(n) is O($n \log_2(n)$).

DONE!

```
   public int recursive(int n)
 {      int sum=0;      for (int
 i=1; i<= n; i++){}        sum=
 sum +1;
      }      if (n>1)
 {         return
 recursive(n/2);
      }
 else
 {        return
 1
      }
   }
```

Deliverable

If you have not already done so, create a GitHub repository that will host all your assignment for this course. In the repository should create a folder titled `cus715_problem_set_05` . Within that folder, add a `pdf` or `markdown` document with your answers to the problem set.