CSE 310 Assignment 3 Solutions

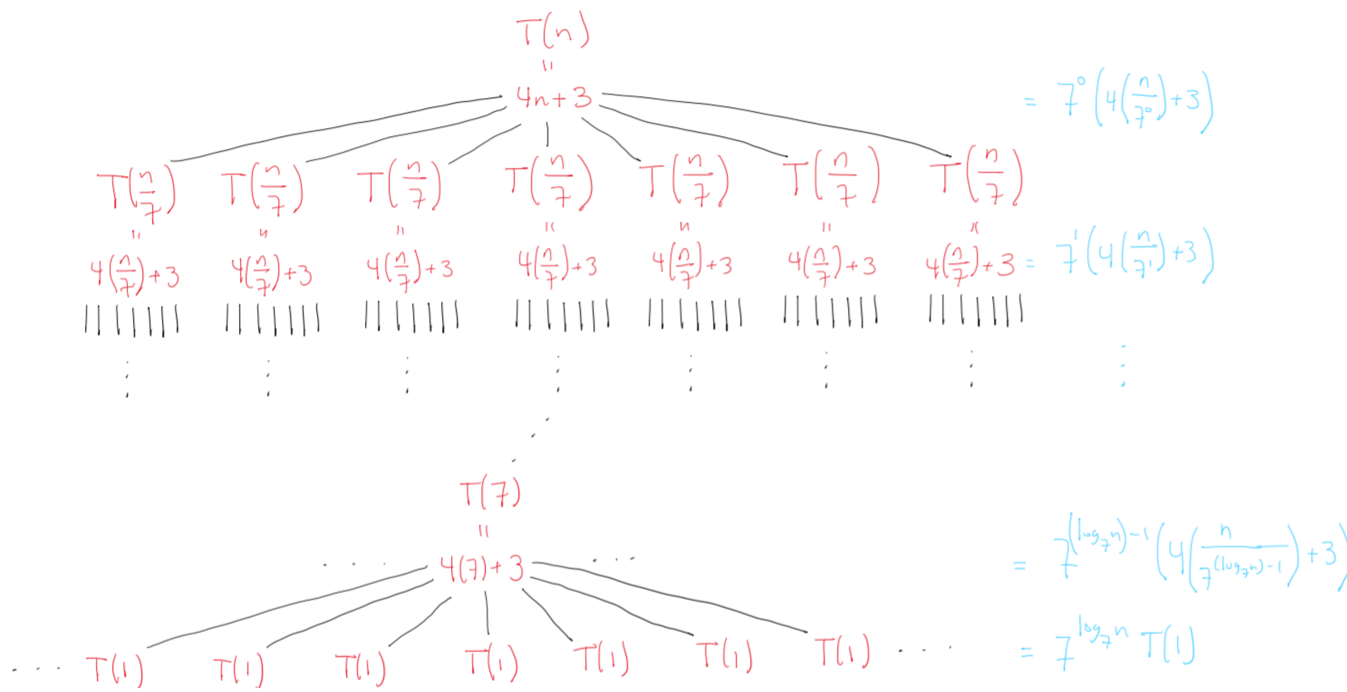
Taman Truong

February 1st, 2022

Problem 1

Suppose that $T(1) = 5$ and, for all $n \geq 2$, where n is a power of 7, $T(n) = 7T\left(\frac{n}{7}\right) + 4n + 3$. Solve this recurrence by drawing a recursion tree.

Solution



Thus,

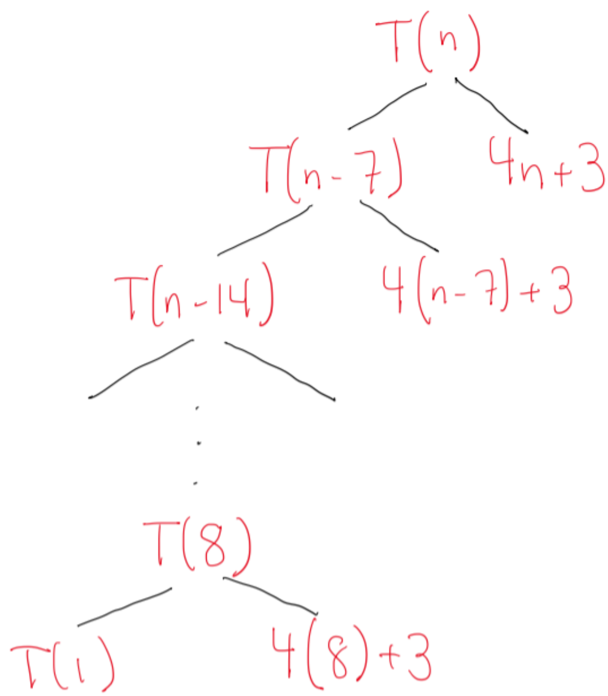
$$
\begin{aligned}
T(n) &= 7^0\left(4\left(\frac{n}{7^0}\right)+3\right)+7^1\left(4\left(\frac{n}{7^1}\right)+3\right)+\cdots+7^{(\log_7 n)-1}\left(4\left(\frac{n}{7^{(\log_7 n)-1}}\right)+3\right)+7^{\log_7 n}T(1) \\
&= \sum_{i=0}^{(\log_7 n)-1} 7^i\left(4\left(\frac{n}{7^i}\right)+3\right)+5n \\
&= \sum_{i=0}^{(\log_7 n)-1} (4n+3(7^i))+5n \\
&= \sum_{i=0}^{(\log_7 n)-1} (4n)+\sum_{i=0}^{(\log_7 n)-1} 3(7^i)+5n \\
&= 4n((\log_7 n-1)-0+1)+3\left(\frac{7^{((\log_7 n)-1)-0+1}-1}{7-1}\right)+5n \\
&= 4n\log_7 n+\frac{1}{2}(n-1)+5n \\
&= 4n\log_7 n+\frac{11}{2}n-\frac{1}{2}
\end{aligned}
$$

Therefore, for $n = 7^i$, $i = 0, 1, 2, \ldots$, $\boxed{T(n) = 4n\log_7 n+\frac{11}{2}n-\frac{1}{2}}$.

Problem 2

Suppose that $T(1) = 5$ and, for all $n \geq 2$, $T(n) = T(n-7) + 4n + 3$. Solve this recurrence by drawing a recursion tree.

Solution



Thus,

$$\begin{aligned}
T(n) &= (4n+3) + (4(n-7)+3) + \cdots + (4(8)+3) + T(1) \\
&= \frac{n-1}{7}\left(\frac{(4(8)+3)+(4n+3)}{2}\right) + 5 \\
&= \frac{1}{7}(n-1)(2n+19) + 5 \\
&= \frac{2}{7}n^2 + \frac{17}{7}n + \frac{16}{7}
\end{aligned}$$

Therefore, for $n = 7i+1$, $i = 0, 1, 2, \ldots$, $\boxed{T(n) = \frac{2}{7}n^2 + \frac{17}{7}n + \frac{16}{7}}$.

Problem 3

Suppose that $T(1) = 5$ and, for all $n \geq 2$, $T(n) = T(n-7) + 4n + 3$. Solve this recurrence by using the mathematical induction method.

Solution

Proof: We will show that $T(n) = \frac{2}{7}n^2 + \frac{17}{7}n + \frac{16}{7}$ is the solution to the recurrence relation $T(n) = T(n-7) + 4n + 3$ for all $n \geq 2$ with $T(1) = 5$ and $n = 7i+1$, $i = 0, 1, 2, \ldots$.

Base Case: The case where $n = 1$ is satisfied because $T(1) = 5 = \frac{2}{7}(1)^2 + \frac{17}{7}(1) + \frac{16}{7} = \frac{2}{7} + \frac{17}{7} + \frac{16}{7} = \frac{35}{7} = 5$.

Inductive Step: Assume that for all $n \geq 2$ with $T(1) = 5$ and $n = 7i+1$, $i = 0, 1, 2, \ldots$, $T(n) = \frac{2}{7}n^2 + \frac{17}{7}n + \frac{16}{7}$ is the solution to the recurrence relation $T(n) = T(n-7) + 4n + 3$. Then,

$$\begin{aligned}
T(n+7) &= T(n) + 4(n+7) + 3 \\
&= \left(\frac{2}{7}n^2 + \frac{17}{7}n + \frac{16}{7}\right) + 4n + 31 && \text{Use Inductive Hypothesis} \\
&= \frac{2}{7}n^2 + \frac{45}{7}n + \frac{233}{7} \\
&= \frac{2}{7}n^2 + 4n + 14 + \frac{17}{7}n + 17 + \frac{16}{7} \\
&= \frac{2}{7}(n^2 + 14n + 49) + \frac{17}{7}(n+7) + \frac{16}{7} \\
&= \frac{2}{7}(n+7)^2 + \frac{17}{7}(n+7) + \frac{16}{7}.
\end{aligned}$$

Thus, the inductive step holds. Therefore, $T(n) = \frac{2}{7}n^2 + \frac{17}{7}n + \frac{16}{7}$ is the solution to the recurrence relation $T(n) = T(n-7) + 4n + 3$ for all $n \geq 2$ with $T(1) = 5$ and $n = 7i+1$, $i = 0, 1, 2, \ldots$.

Problem 4

Use the master method to give tight asymptotic bounds for the following recurrences.

(a) $T(n) = 2T\left(\frac{n}{4}\right) + 5n^2$

(b) $T(n) = 16T\left(\frac{n}{4}\right) + 3n^2$

(c) $T(n) = 9T\left(\frac{n}{3}\right) + 4n$

Solution

(a) Suppose that $a = 2$, $b = 4$, and $f(n) = 5n^2$. Then, $f(n) = 5n^2 = \Omega(n^{\log_b a+\epsilon}) = \Omega(n^{\log_4 2+\epsilon}) = \Omega(n^{\frac{1}{2}+\epsilon}) = \Omega(n^2)$. Let $\epsilon = \frac{3}{2} > 0$. Then, $f(n) = 5n^2 = \Omega(n^{\frac{1}{2}+\epsilon}) = \Omega(n^2)$ is satisfied.

Consider $2f\left(\frac{n}{4}\right)$. Then, $2f\left(\frac{n}{4}\right) = 2\left(5\left(\frac{n}{4}\right)^2\right) = \frac{5}{8}n^2 \leq cf(n) = 5cn^2$. Let $c = \frac{1}{8} < 1$. Then, for large n, $\frac{5}{8}n^2 \leq cf(n) = 5cn^2 = \frac{5}{8}n^2$ is satisfied.

Therefore, by Case 3 of the Master Method, $\boxed{T(n) = \theta(n^2)}$.

(b) Suppose that $a = 16$, $b = 4$, and $f(n) = 3n^2$. Then, $f(n) = 3n^2 = \theta(n^{\log_b a}) = \theta(n^{\log_4 16}) = \theta(n^2)$. Therefore, by Case 2 of the Master Method, $\boxed{T(n) = \theta(n^2 \log_2 n)}$.

(c) Suppose that $a = 9$, $b = 3$, and $f(n) = 4n$. Then, $f(n) = 4n = O(n^{\log_b a-\epsilon}) = O(n^{\log_3 9-\epsilon}) = O(n^{2-\epsilon}) = O(n)$. Let $\epsilon = 1 > 0$. Then, $f(n) = 4n = O(n^{2-\epsilon}) = O(n)$ is satisfied. Therefore, by Case 1 of the Master Method, $\boxed{T(n) = \theta(n^2)}$.

Problem 5

Complete the program ompProb.cpp by writing code to compute the count of larger integers among randomly generated numbers by

(a) completing the sumOfDivisibleBy3WithLoop function to compute the sum of integers in the array that are divisible by 3 using a for loop sequentially. Include your code for this function in the file that you will be submitting.

(b) completing the sumOfDivisibleBy3WithLoop_OMP function to compute the sum of integers in the array that are divisible by 3 using a for loop in parallel (using OMP). Include your code for this function in the file that you will be submitting.

(c) Using 4 threads, execute your program using 100, 10000, 1000000, and 100000000 as the values of n (the number of randomly generated numbers and also the array size). Then record the execution time for each of the four functions above in a table shown below.

Execution Time of Function	$n = 100$	$n = 10000$	$n = 1000000$	$n = 100000000$
sumOfDivisibleBy3WithLoop				
sumOfDivisibleBy3WithLoop_OMP				

Solution

(a) The code for the sumOfDivisibleBy3WithLoop function is shown below.

```
int sumOfDivisibleBy3WithLoop(int * A, int n) {
  int divisibleBy3Sum = 0;
  for(int i = 0; i < n; i++) {
    if (A[i] % 3 == 0) {
      divisibleBy3Sum += A[i];
    }
  }
  return divisibleBy3Sum;
}
```

(b) The code for the sumOfDivisibleBy3WithLoop_OMP function is shown below.

```
int sumOfDivisibleBy3WithLoop_OMP(int * A, int n) {
  int max_threads = omp_get_max_threads();
  int divisibleBy3Sum = 0;
```

```

  # pragma omp parallel for shared (n) num_threads(max_threads)
  reduction (+: divisibleBy3Sum)
  for(int i = 0; i < n; i++) {
    if (A[i] % 3 == 0) {
      divisibleBy3Sum += A[i];
    }
  }
  return divisibleBy3Sum;
}
```

(c) The table of runtimes for each function for $n = 100, 10000, 1000000$ and 100000000 is shown below.

Execution Time of Function	$n = 100$	$n = 10000$	$n = 1000000$	$n = 100000000$
sumOfDivisibleBy3WithLoop	0.000002 seconds	0.000101 seconds	0.015033 seconds	0.951673 seconds
sumOfDivisibleBy3WithLoop_OMP	0.000212 seconds	0.008357 seconds	0.009276 seconds	0.401607 seconds

Source Code for Problem 5

```
/* This program generates a list of n random integers.
   First, it reads an integer n. Then, it then generates
   a sequence n random integers. */

#include <ctime>
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <omp.h>

using namespace std;

int sumOfDivisibleBy3WithLoop(int * A, int n);
int sumOfDivisibleBy3WithLoop_OMP(int * A, int n);

int main() {
  double wtime;
  int n;
  printf ( "How many random numbers do you want to generate?\n" );
  scanf ( "%d", &n );
  int * A;
  A = new int[n];

  //generate n of random numbers
  srand( (unsigned) time(NULL) );

  for(int i = 0; i < n; i++) {
    A[i] = rand() % 100;
  }

  int num_procs = omp_get_num_procs ( );
  int max_threads = omp_get_max_threads ( );
  printf ( " Number of processors available = %d\n", num_procs );
  printf ( " Number of threads = %d\n", max_threads );

  //executing sumOfDivisibleBy3WithLoop and also mesuring its execution time
```

```c
  wtime = omp_get_wtime ( );
  int sum1 = sumOfDivisibleBy3WithLoop(A, n);
  wtime = omp_get_wtime ( ) - wtime;
  printf( "\n The first sum is %d\n", sum1 );
  printf ( " time  %14f\n\n", wtime );

  //executing sumOfDivisibleBy3WithLoop_OMP and also mesuring its execution time
  wtime = omp_get_wtime ( );
  int sum2 = sumOfDivisibleBy3WithLoop_OMP(A, n);
  wtime = omp_get_wtime ( ) - wtime;
  printf( " The second sum is %d\n", sum2 );
  printf ( " time  %14f\n\n", wtime );
}

int sumOfDivisibleBy3WithLoop(int * A, int n) {
  int divisibleBy3Sum = 0;
  for(int i = 0; i < n; i++) {
    if (A[i] % 3 == 0) {
      divisibleBy3Sum += A[i];
    }
  }
  return divisibleBy3Sum;
}

int sumOfDivisibleBy3WithLoop_OMP(int * A, int n) {
  int max_threads = omp_get_max_threads();
  int divisibleBy3Sum = 0;

  # pragma omp parallel for shared (n) num_threads(max_threads) reduction (+: divisibleBy3Sum)
  for(int i = 0; i < n; i++) {
    if (A[i] % 3 == 0) {
      divisibleBy3Sum += A[i];
    }
  }
  return divisibleBy3Sum;
}
```