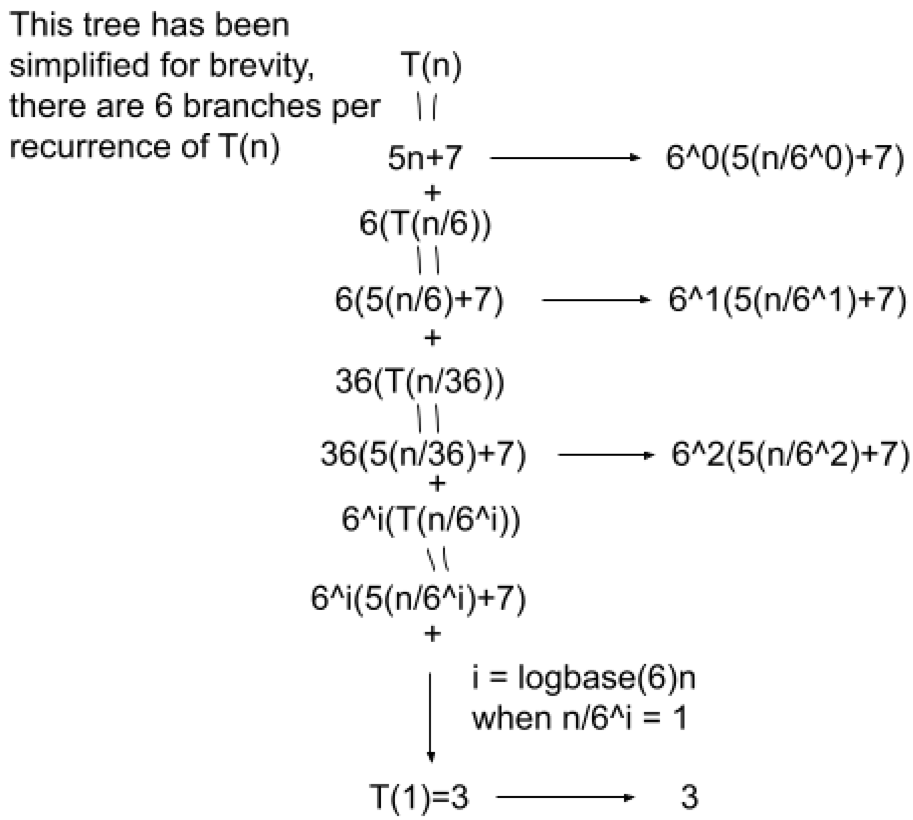
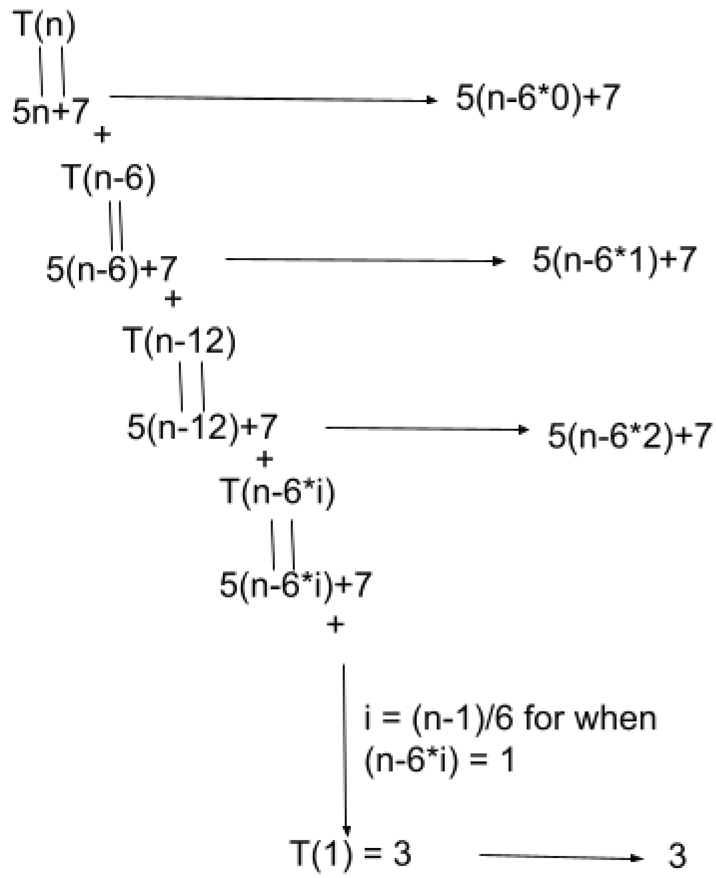
Vincent Latona
September 7, 2021

Assignment 3



1.

The recurrence tree branches 6 times recursively per case where $n \geq 2$, making the recurrence pattern a power of 6. Each level of the recurrence tree can be represented by a power of 6 through the non-negative variable i.The tree can be represented by the equation: $T(n) = \sum_{i=0}^{log_6 n-1} [6^i(5(\frac{n}{6^i}) + 7)] + 3$,which simplifies to the polynomial function $T(n) = 5nlog_6 n + \frac{7}{5}n + \frac{8}{5}$for all $n \geq 1$.



2.

The recurrence is self-defined with a single branch at each level. The reduction of the recurrence is always a multiple of 6 through the non-negative variable i.The tree can be represented by the equation: $T(n) = \sum_{i=0}^{(n-1)/6-1} [5(n - 6i) + 7] + 3$,which simplifies to the polynomial function $T(n) = \frac{5}{12}n^2 + \frac{44}{12}n - \frac{13}{12}$for all $n \geq 1$.

3. We desire to prove the recurrence $T(n) = 3$ when $n = 1$and $T(n) = T(n - 6) + 5n + 7$for $n \geq 2$. We will solve this proof by mathematical induction. Our Inductive Hypothesis is as follows: $T(n) = \frac{5}{12}n^2 + \frac{44}{12}n - \frac{13}{12}$for all $n \geq 1$.We must show this to be true for the base case, $T(1) = \frac{5}{12} + \frac{44}{12} - \frac{13}{12} = 3$, which is true. For the Inductive Step, we must assume the Inductive Hypothesis is true

for some arbitrary value $n = k$ and prove that it is true for $n = k + 6$.

$T(k + 6) = T((k + 6) - 6) + 5(k + 6) + 7$ is obtained from the recurrence and can be expanded by the assumption to $T(k + 6) = \frac{5}{12}k^2 + \frac{44}{12}k - \frac{13}{12} + 5k + 37$.

This matches the result of applying $n = k + 6$to the Inductive Hypothesis directly:

$T(k + 6) = \frac{5}{12}(k + 6)^2 + \frac{44}{12}(k + 6) - \frac{13}{12} = \frac{5}{12}k^2 + \frac{44}{12}k - \frac{13}{12} + 5k + 37$.

Thus, the Inductive Hypothesis, $T(n) = \frac{5}{12}n^2 + \frac{44}{12}n - \frac{13}{12}$for all $n \geq 1$,has been proven by mathematical induction.

4. a. Given $T(n) = 8T(\frac{n}{2}) + 7n$,then $a = 8,\ b = 2,\ f(n) = 7n$.Using the Master Method, $7n = O(n^{log_2 8} = n^3)$then by case 1, there exists $\varepsilon = 1$such that $7n = O(n^{3-\varepsilon} = n^2)$which is true. Therefore, by case 1 of the Master Method, $T(n) = \Theta(n^3)$.

   b. Given $T(n) = 3T(\frac{n}{9}) + 4n^2$,then $a = 3,\ b = 9,\ f(n) = 4n^2$.Using the Master Method, $4n^2 = \Omega(n^{log_9 3})$,since $log_9 3$< 1. The following also holds true for $\varepsilon = 1 - log_3 9$:$4n^2 = \Omega(n^{log_9 3+\varepsilon} = n)$.This is a case 3, which means $af(\frac{n}{b}) \leq cf(n)$ must be proven with constants $a, b, 0 < c < 1$. $3(4(\frac{n}{9})^2) = \frac{4}{27}n^2 \leq c(4n^2)$will hold true for $c = \frac{1}{27}$.Therefore, by case 3 of the Master Method, $T(n) = \Theta(4n^2)$.

   c. Given $T(n) = 9T(\frac{n}{3}) + 5n^2$,then $a = 9,\ b = 3,\ f(n) = 5n^2$.Using the Master Method, $5n^2 = \Theta(n^{log_3 9} = n^2)$,then by case 2 of the Master Method, $T(n)$=$\Theta(n^2 log_2 n)$.

5. a.

```
/*
* This function computes the sum of all elements using only a single loop
*/
int largerCountWithLoop(int* A, int n)
{
   int sum = 0;

   //Iterator for sum calculation
   for (int i = 0; i < n; i++)
   {
      sum = sum + A[i]; //Add element values together
   }
   return sum; //Return final sum
}
```

b.

```
/*
* This function computes the sum of all elements using parallel processing
*/
int largerCountWithLoop_OMP(int* A, int n)
{
   //Initialize local variables for parallel computation
   int sum = 0;
   int totalSum = 0;
   int i;

   //Establish parallel variable visibilities
   #pragma omp parallel shared(A, n) private(sum, i) num_threads(4)
   {
      #pragma omp for reduction(+:totalSum) //Reduce all sums into totalSum
      for (i = 0; i < n; i++)
      {
         sum = sum + A[i]; //Individual sum for thread
      }
      totalSum = totalSum + sum; //Total sum for given thread
   }
   return totalSum; //Return final overall sum from all threads
}
```

c.

Execution Time	n = 100	n = 10000	n = 1000000	n = 100000000
LargerLoop	0.000002	0.000032	0.004999	0.319713
LargerLoopOMP	0.000188	0.000246	0.005638	0.155752