CSCI 532 – Algorithm Design
Assignment 1

Name: Badarika Namineni CWID: 50233279

Question 1: Show that the solution of T(n) = T (n − 1) + n is O(n2).

Solution:

1a) $T(n) = T(n-1) + n$ — ①

we have to prove $T(n) = O(n^2)$

Lets prove it using substitution method

$T(n-1) = T(n-1-1) + n-1 = T(n-2) + n-1$ — ②

$T(n-2) = T(n-2-1) + n-2 = T(n-3) + n-2$ — ③

$T(n-3) = T(n-3-1) + n-3 = T(n-4) + n-3$ — ④

Substitute ② in ①, we get

$T(n) = T(n-2) + (n-1) + n$ — ⑤

Substitute ③ in ⑤

$T(n) = T(n-3) + (n-2) + (n-1) + n$ — ⑥

Substitute ④ in ⑥

$T(n) = T(n-4) + (n-3) + (n-2) + (n-1) + n$

For k^{th} iteration,

$T(n) = T(n-k) + \dots\dots (n-4) + (n-3) + (n-2) + (n-1) + n$

The above equation is true for $n>1$. To check at $n=1$, lets take $T(n-k)$ and equate to 1 to get the tight bound.

$n-k = 1$

$n = 1+k$

By substituting n value in the T(n) equation, we get

$T(n) = T(1+k-k) + \dots + (n-4) + (n-3) + (n-2) + (n-1) + n$

$= T(1) + \dots + (n-4) + (n-3) + (n-2) + (n-1) + n$

Since $T(1) = 1$. we get

$T(n) = 1 + \dots\dots + (n-4) + (n-3) + (n-2) + (n-1) + n$

which can be written as

$\frac{n(n-1)}{2}$

$T(n) = \frac{n^2-n}{2} = \frac{1}{2}(n^2-n) = \underline{O(n^2)}$

Question 2: Show that the solution of T(n) = T([n/2]) + 1 is O (lg n).

Solution:

(2) Given $T(n) = T(\frac{n}{2}) + 1$

To prove $T(n) = O(\log n)$, lets take a substitution method.

$$T(n) = T(\frac{n}{2}) + 1 \quad \text{—— (1)}$$

$$T(\frac{n}{2}) = T(\frac{n}{2^2}) + 1 \quad \text{—— (2)}$$

$$T(\frac{n}{2^2}) = T(\frac{n}{2^3}) + 1 \quad \text{—— (3)}$$

By substituting (2) in (1)

$$T(n) = T(\frac{n}{2^2}) + 1 + 1 = T(\frac{n}{2^2}) + 2 \quad \text{—— (4)}$$

By substituting (3) in (4)

$$T(n) = T(\frac{n}{2^3}) + 1 + 2 = T(\frac{n}{2^3}) + 3$$

For k^{th} iteration

$$T(n) = T(\frac{n}{2^k}) + k \quad \text{where } n > 1$$

At $n=1$, we can take the equation and equate it to 1.

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$$

$$= k = \log n$$

By substituting k value in the T(n), we get

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

Here, $\log n$ is greater than $\frac{n}{2^{\log n}}$. we get

$O(\log n)$.

Question 3: Use a recursion tree to give an asymptotically tight solution to the recurrence T(n) = T (n − a) + T(a) + cn, where a > 1 and c > 0 are constants.

Solution:

(3) The recursion tree representation for the recurrence

$T(n) = T(n-a) + T(a) + cn \qquad a > 1, \ c > 0$

Sum

$cn \longrightarrow cn$

$c(n-a) \quad ca \longrightarrow cn - ca + ca = cn$

$c(n-2a) \quad ca \quad ca \longrightarrow cn - 2ca + ca + ca$

$= cn - 2ca + 2ca = cn$

$c(n-3a) \quad ca \quad ca \quad ca \longrightarrow cn - 3ca + ca + ca + ca$

$= cn - 3ca + 3ca = cn$

For k^{th} iteration, ~~it is the sum of all the nodes~~

we get an expression

$n - ka$.

To get a value of k, we have to equate the equation to 1

$n - ka = 1$

$n - 1 = ka$

$k = \frac{n-1}{a} = \frac{n}{a} - \frac{1}{a}$

The total sum will be $\left(\frac{n}{a}\right) cn$

$\therefore \ T(n) = \left(\frac{n}{a}\right) cn = \frac{cn^2}{a}$

$= \underline{\underline{O(n^2)}}$

Question 4: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n)=T(n/2) + n^2$. Use the substitution method to verify your answer.

Solution:

(4) The recursive tree representation for the recurrence is

$T(n) = T\left(\frac{n}{2}\right) + n^2$ is

Sum

$n^2 \Rightarrow n^2$

$\left(\frac{n}{2}\right)^2 \rightarrow \left(\frac{n}{2}\right)^2$

$\frac{n^2}{16} \rightarrow \left(\frac{n}{4}\right)^2$

$\frac{n^2}{64} \rightarrow \left(\frac{n}{8}\right)^2$

At k iteration the cost would be $\left(\frac{n}{2^k}\right)^2$.

By equating $\frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$k = \log n$

$\therefore$ The run time of the tree is

$$T(n) = \sum_{k=0}^{\log n} c \cdot \left(\frac{n}{2^k}\right)^2$$

$$= cn^2 \cdot \sum_{k=0}^{\log n} \left(\frac{1}{4}\right)^k$$

$$= cn^2 \cdot \sum_{k=0}^{\infty} \left(\frac{1}{4}\right)^k$$

$$T(n) = O(n^2)$$

Question 5: Use the master method to give tight asymptotic bounds for the following recurrences.

a. T(n) = 2T(n/4) + 1.

b. T(n) = 2T(n/4) + √ n.

c. T(n) = 2T(n/4) + n

d. T(n) = 2T(n/4) + n2

Solution:

The master's theorem provides bounds for recurrences of the form

$T(n) = aT(n/b)+\Theta(n^k \log^p n)$ where $f(n) = \Theta(n^k \log^p n)$
a>=1, b>1, k>=0 and P is real number and
There are 3 cases to define the tight asymptotic bound for the recurrence

Case 1: If $a>b^k$, then $T(n) = \Theta(n^{\log_b a})$

Case 2: If $a=b^k$

a) If P>-1. Then $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$
b) If p = -1, then $T(n) = \Theta(n^{\log_b a} \log\log n)$
c) If p<-1, then $T(n) = \Theta(n^{\log_b a})$

Case 3: If $a<b^k$

a) If p>=0, then $T(n) = \Theta(n^k \log^p n)$
b) If p<0, then $T(n) = O(n^k)$

With the above rules, lets prove

a) T(n) = 2T(n/4) + 1
   a = 2, b = 4, k =0, p =0
   since $n^0 = 1$, we consider k =0 as it will be in n^k form inside Θ and we don't have anything in log so p =0
   $b^k = 4^0 = 1$
   which is a > bk which is case1
   $T(n) = \Theta\ (n^{log_4^2})$ = $\mathbf{\Theta(n^{1/2})}$

b) $T(n) = 2T(n/4) + \sqrt{n}$

$a = 2, b = 4, k = ½, p = 0$

we don't have anything in log so p =0

$b^k = 4^{1/2} = (2^2)^{1/2} = 2$

So, $a = b^k$ which is case 2.

Since p = 0 and p > -1, we choose instruction (a)

$T(n) = \Theta\ (n^{log_4^2} \log^{0+1} n)$

$= \boldsymbol{\Theta\ (n^{1/2} \log n)}$

c) $T(n) = 2T(n/4) + n$

$a = 2, b = 4, k =1, p =0$

we don't have anything in log so p =0

$b^k = 4$

So, $a < b^k$ which is case 3

Since p=0, we choose instruction (a)

$T(n) = \Theta\ (n \log^0 n)$

$= \boldsymbol{\Theta\ (n)}$

d) $T(n) = 2T(n/4) + n2$

$a = 2, b = 4, k = 2, p = 0$

we don't have anything in log so p =0

$b^k= 4^2 = 16$

So, $a < b^k$ which is case 3

Since p=0, we choose instruction (a)

$T(n) = \Theta\ (n^2 \log^0 n)$

$= \boldsymbol{\Theta\ (n^2)}$

Question 6: Use the master method to show that the solution to the binary-search recurrence $T(n)=T(n/2) +\Theta\ (1)$ is $T(n)=\Theta(\lg n)$. (See Exercise 2.3-5 for a description of binary search.)

Solution:

The master's theorem provides bounds for recurrences of the form

$T(n) = aT(n/b)+\Theta(n^k\log^p n)$ where $f(n) = \Theta(n^k\log^p n)$

a>=1, b>1, k>=0 and P is real number and

There are 3 cases to define the tight asymptotic bound for the recurrence

Case 1: If $a>b^k$, then $T(n) = \Theta(n^{\log_b a})$

Case 2: If $a=b^k$

d) If P>-1. Then $T(n) = \Theta(n^{\log_b a}\log^{p+1} n)$
e) If p = -1, then $T(n) = \Theta(n^{\log_b a}\log\log n)$
f) If p<-1, then $T(n) = \Theta(n^{\log_b a})$

Case 3: If $a<b^k$

e) If p>=0, then $T(n) = \Theta(n^k \log^p n)$
f) If p<0, then $T(n) = O(n^k)$

With the above set of rules, lets prove $T(n) = \Theta(\log n)$

Given equation is $T(n) = T(n/2) + \Theta(1)$
$a = 1$, $b = 2$, $k = 0$, $p=0$
since $n^0 = 1$, we consider $k = 0$ as it will be in n^k form inside Θ and we don't have anything in log so $p = 0$

$b^k = 2^0 = 1$
So, $a = b^k$ which is case 2
Since $p=0$ and $p>-1$, we choose instruction (a)
$T(n) = \Theta(n^{log_2^1} \log^{0+1} n)$
$= \Theta(n^0 \log n)$
$= \boldsymbol{\Theta(\log n)}$