

Assignment: Recursion, Recurrence Relations and Divide & Conquer

1. Solve recurrence relation using three methods:

Write recurrence relation of below pseudocode that calculates x^n , and solve the recurrence relation using three methods that we have seen in the explorations.

```
power2(x,n):  
if n==0:  
    return 1  
if n==1:  
    return x  
if (n%2)==0:  
    return power2(x, n//2) * power2(x,n//2)  
else:  
    return power2(x, n//2) * power2(x,n//2) * x
```

a) Using substitution method:

1) $\Rightarrow T(n) = 2T(n/2) + 2 \rightarrow \textcircled{1}$
Find $T(n/2)$ using eqⁿ $\textcircled{1}$
 $T(n/2) = 2T(n/2^2) + 2$
Put in eqⁿ $\textcircled{1}$
 $T(n) = 2[2T(n/2^2) + 2] + 2$
 $T(n) = 2^2 T(n/2^2) + 2^2 + 2 \rightarrow \textcircled{2}$
Find $T(n/2^2)$ using eqⁿ $\textcircled{1}$
 $T(n/2^2) = 2T(n/2^3) + 2$
Put in eqⁿ $\textcircled{2}$
 $T(n) = 2^2 [2T(n/2^3) + 2] + 2^2 + 2$
 $T(n) = 2^3 T(n/2^3) + 2^3 + 2^2 + 2$
 $T(n) = 2^k T(n/2^k) + \frac{2(2^k - 1)}{2 - 1}$
 $n/2^k = 1 \Rightarrow n = 2^k$
take log both sides
 $k = \log_2 n$
 $T(n) = 2^{\log_2 n} T(n/2^{\log_2 n}) + 2(2^{\log_2 n} - 1)$
 $= n \log_2 2 + (n/\log_2 2) + 2(n \log_2 2 - 1)$
 $= n T(1) + 2(n - 1)$
 $T(n) = \Theta(n)$

b) Using master method:

b) $T(n) = a T\left(\frac{n}{b}\right) + f(n)$
 $a = 2, b = 2, f(n) = 2$
 $n^{\log_b a}$
 $n^{\log_2 2}$
 $n^1 > 2$
 $T(n) = \Theta(n)$ based on Case 1

c) Recursion-tree method:

c)

level		cost
1	n	2
2	$n/2 \quad n/2$	2^2
3	$n/2^2 \quad n/2^2 \quad n/2^2 \quad n/2^2$	2^3
...
h	$n/2^h \quad n/2^h \quad \dots \quad n/2^h$	2^h

total cost = $2 + 2^2 + 2^3 + \dots + 2^h$
 $n/2^h = 1 \Rightarrow n = 2^h$
 $h = \log_2 n$
 $= \frac{2(2^h - 1)}{2 - 1} \Rightarrow 2(2^{\log_2 n} - 1)$
 $= 2(n^{\log_2 2} - 1)$
 $= 2n - 2$
 $T(n) = \Theta(n)$

2. **Solve recurrence relation using any one method:**

Find the time complexity of the recurrence relations given below using any one of the three methods discussed in the module. Assume base case $T(0)=1$ or/and $T(1) = 1$.

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

a.

$T(n) = 4T(n/2) + n$
 $a=4, b=2, f(n)=n$
 $\log_2 4 = 2$
 $n^2 > n$
 $T(n) = O(n^2)$ based on case 1

b) $T(n) = 2T(n/4) + n^2$

a.

$T(n) = 2T(n/4) + n^2$
 $a=2, b=4, f(n)=n^2$
 $\log_4 2 = 0.5$
 $n^{0.5} < n^2$
 $T(n) = O(n^2)$ based on case 3

3. **Implement an algorithm using divide and conquer technique:** Given two sorted arrays of size m and n respectively, find the element that would be at the k^{th} position in combined sorted array.

- a. Write a pseudocode/describe your strategy for a function $k\text{thElement}(\text{Arr1}, \text{Arr2}, k)$ that uses the concepts mentioned in the divide and conquer technique. The function would take two sorted arrays Arr1 , Arr2 and position k as input and returns the element at the k^{th} position in the combined sorted array.

i Function $k\text{thElement}(\text{Arr1}, \text{Arr2}, k)$:

if Arr1 is empty:

return $\text{Arr2}[k - 1]$

if Arr2 is empty:

return $\text{Arr1}[k - 1]$

if k is 1:

```
return min(Arr1[0], Arr2[0])
```

```
mid1 = min(length(Arr1), k // 2)
```

```
mid2 = min(length(Arr2), k // 2)
```

```
if Arr1[mid1 - 1] < Arr2[mid2 - 1]:
```

```
    return kthelement(Arr1[mid1:], Arr2, k - mid1)
```

```
else:
```

```
    return kthelement(Arr1, Arr2[mid2:], k - mid2)
```

- b. Implement the function `kthElement(Arr1, Arr2, k)` that was written in part a. Name your file **KthElement.py**

Examples:

`Arr1 = [1,2,3,5,6] ; Arr2= [3,4,5,6,7] ; k= 5`

Returns: 4

Explanation: 5th element in the combined sorted array [1,2,3,3,4,5,5,6,6,7] is 4