James Stallkamp

CS325 HW2

1) Recurrence: T(n) = T(n/2)+T(n/2)+1+1 = 2T(n/2)+2

Substitution method:

T(n) = 2T(n/2) + 2 -> 1

T(n) = 1 if n = 1 and 2T(n/2)+2 if n>1

T(n/2) = 2T(n/4) + 2 -> 2

eq2 in eq1

T(n) = 2[2T(n/4)+2]+2 ->3

T(n)4T(n/4)+6 ->3

T(n/4) = 2T(n/8) +2 -> 4

substitute equ in eq 3

T(n) = 4[2T(n/8)+2]+6

T(n) = 8T(n/8)+14

T(n) = 2kT(n/2k) + (2k-1)*2

assume n/2k=1 n=2k

T(n) = 2log(n)*T(n/2log(n))+(2log(n) - 1) * 2

T(n) = nT(n/n)+(n-1)*2

T(n) = nT(1) + 2n -2

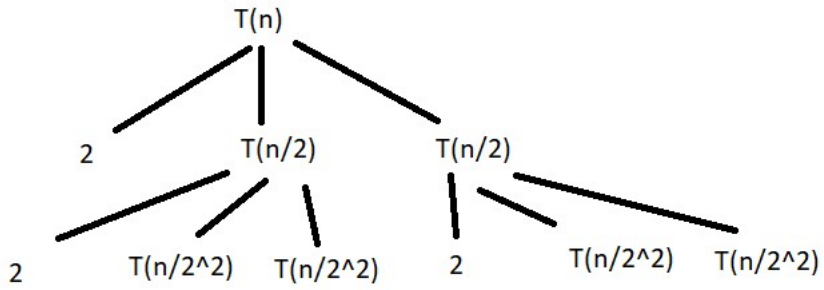
T(n)= n + 2n -2

T(n) = Θ(n)

Masters Method T(n) = 2T(n/2)+2 A=2 B=2 f(n)=2 k =0 2>2^0 so case 3

Θ(n^log2(2)) = Θ(n)

Recurrence Tree Method



2k T(n/2^k)+(2k-1)*2

T(n) = nT(n/n)+(n-1)*2 assume n/2k = 1 n = 2k

T(n) = nT(1)+2n-2 T(1) = 1

T(n) = n + 2n -2

T(n) = Θ(n)

2)

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

Using masters method A=4 B=2 f(n) = n k =1 a>b^k = 4>2^1 so case 3

T(n) = Θ(n^2)

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

Using masters method A=2 B=4f(n) = n^2 k =2 a>b^k = 2<4^2 so case 1

T(n) = Θ(n^2)

3) KthElement using a merge sort.

First combine the two arrays into the final array simply by appending. Then perform a merge sort on the newly combined array and print the kth element. A merge sort will split the array in half and recursively call merge sort on the first half, then the second, and then merge the two sorted halfs. This merge sort will continue splitting the array in smaller arrays until it comes to

the base case of a single element and then works its way back up building the array.