Lecture 26: Analyzing Divide and Conquer Algorithms

Date: November 4, 2019.

```
MergeSort(A[l .. u])
    if u - l ≥ 1
        m = ⌊ (u-l)/2 ⌋
        MergeSort(A[l .. m])
        MergeSort(A[m+1 .. u])
        Merge(A[l .. u], m)
```

```
Merge(A[l .. u], m)
    i = l; j = m+1
    for k = l to u
        if j > u
            B[k] = A[i]; i = i+1
        else if i > m
            B[k] = A[j]; j = j+1
        else if A[i] < A[j]
            B[k] = A[i]; i = i+1
        else
            B[k] = A[j]; j = j+1
    for k = l to u
        A[k] = B[k]
```

Problem 1. What is the running time of the above algorithm?

$T(n)$ – running time of algo on inputs of length n.

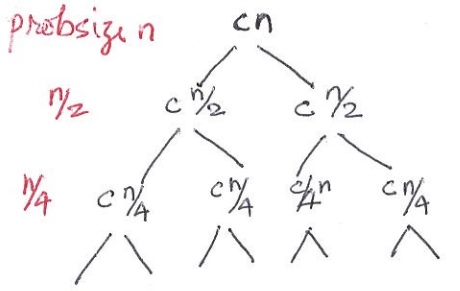
$T(n) \le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 2 + 2 + 5n + n$

$\le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn$

Assume that n is a power of 2.

$\rightarrow T(n) \le 2T(n/2) + cn$] $\boxed{T(n) = 2T(n/2) + \Theta(n)}$

$\le 2\left[2T(n/4) + c\frac{n}{2}\right] + cn$



Sum at level 0 $= cn$

Sum at level 1 $= cn/2 + cn/2 = cn$

Sum at level $i = cn$.

(1) element → 1 1 1 1

k s.t. $\frac{n}{2^k} = 1 \Rightarrow k = \log_2 n$

$T(n) \le cn\lceil \log n \rceil = O(n \log n)$

$T(n) \le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn$

$\le 2T\left(\frac{n}{2} + 1\right) + cn$ ←

Consider $S(n) = T(n+\alpha)$ Goal: Choose α so that $S(n) \le 2S(n/2) + dn$

$S(n) = T(n+\alpha)$

$\le 2T\left(\frac{n}{2} + \frac{\alpha}{2} + 1\right) + c(n+\alpha)$

$\le 2S\left[\frac{n}{2} + 1 - \frac{\alpha}{2}\right] + c(n+\alpha)$

Taking $\alpha = 2$, $S(n) \le 2S(n/2) + dn$; $S(n) = O(n \log n)$

$T(n) = S(n-2) = O((n-2)\log(n-2)) = O(n \log n)$

```
BinarySearch(A[l ..  u], x)
    if (u - l < 0) return NO
    mid = ⌊l+u/2⌋
    m = A[mid]
    if (x = m) return YES
    else if (x < m)
        return BinarySearch(A[l ..  mid-1],x)
    else
        return BinarySearch(A[mid+1 ..  u], x)
```

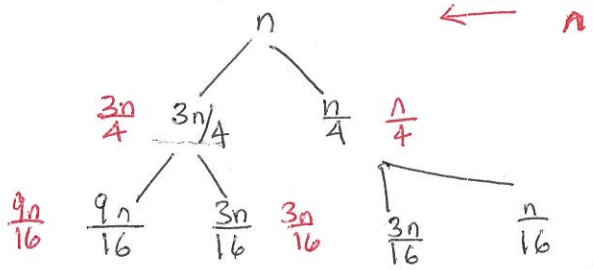
Prob size n → c
n/2 → c
n/4 → c
...

Problem 2. What is the running time of the above algorithm?

$T(n) = T(n/2) + c$.
Work at level $i = c$
levels $= \log n$
$\Rightarrow T(n) = c \log n = O(\log n)$

Problem 3. Suppose the running of an algorithm is given by $T(1) = 1$ and

$$T(n) = T(3n/4) + T(n/4) + n$$



Sum at level $i \leq n$
levels $= O(\log n)$
Total time $\leq O(n \log n)$.

$\log_a n = \frac{\log_b n}{\log_b a}$