Cheat Sheet - Exam 1

Thursday, October 10, 2019 10:28 PM

Master's Theorem

General Form: $T(n) = aT(\frac{n}{b}) + \delta(n)$
where $a \geq 1$, $b > 1$ $f(n) = \Theta(n^k \log^p n)$

Look for: ① $\log_b a$, ② k

Case 1: if $\log_b a > k$
$T(n) \in \Theta(n^{\log_b a})$

Case 2: if $\log_b a = k$
- Sub-Case 1: if $p > -1$, $T(n) \in \Theta(n^k \log^{p+1} n)$
- Sub-Case 2: if $p = -1$, $T(n) \in \Theta(n^k \log(\log(n)))$
- Sub-Case 3: if $p < -1$, $T(n) \in \Theta(n^k)$

Case 3: if $\log_b a < k$
- Sub-Case 1: if $p \geq 0$, $T(n) \in \Theta(n^k \log^p(n))$
- Sub-Case 2: if $p < 0$, $T(n) \in O(n^k)$

Limit Test

$L = \lim_{n \to \infty} \frac{f(n)}{g(n)} \Rightarrow$ if $0 < L < \infty$, $f(n) \in \Theta(g(n))$
if $L = 0$, $f(n) \in O(g(n))$
if $L = \infty$, $f(n) \in \Omega(g(n))$

Ratio Test

$\sum_{n=0}^{\infty} \frac{f(n)}{g(n)} \Rightarrow \sum_{n=0}^{\infty} a_n$

$L = \lim_{n \to \infty} \left| \frac{a_{n+1}}{a_n} \right|$ (factorial)

Root Test

$L = \lim_{n \to \infty} |a_n|^{1/n}$

if $L < 1$, Series Converges
$L > 1$, Series diverges
$L = 1$, Test Inconclusive

Big-O: Upper
Big-Ω: Lower
Big-Θ: Tight

Loop Invariants (See Ex.)

- Initialization, Maintinence, Termination

Recurrence Relations (See Ex.)

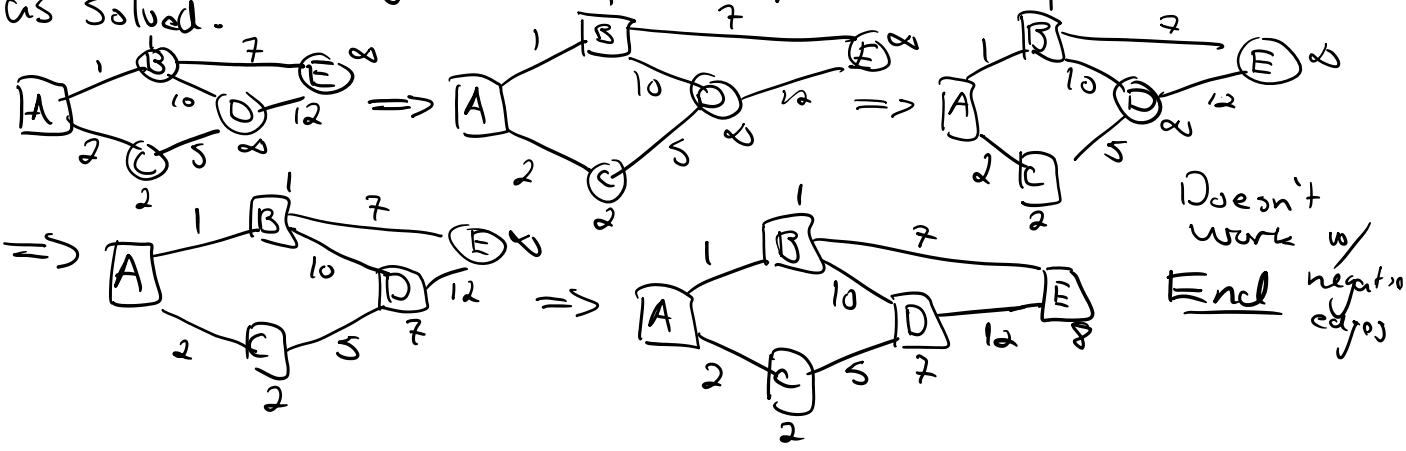
General Form: $T(n) = aT(g(n)) + f(n)$
a = # of sub-problems
$g(n)$ = size of sub-problems
$f(n)$ = Time to combine smaller problems

$T(n) = \begin{cases} \Theta(1), & n \leq c \\ aT(\frac{n}{b}) + f(n) & n > c \end{cases}$

- Recurrence Trees, Unrolling, Master's.

Interval Scheduling - Pick earliest finish time.

Dijkstra's Algo - Single Source Shortest path. Source node cants as Solved.



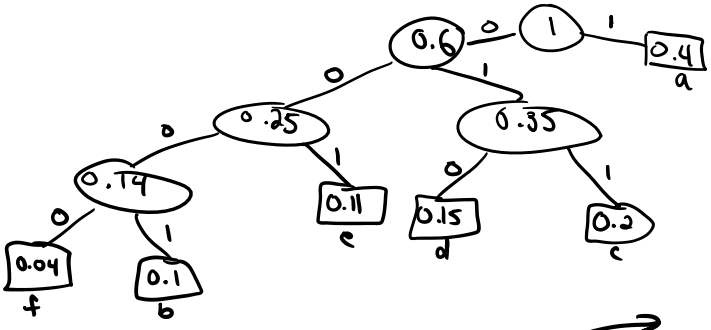
Kruskal's - Select shortest edge, as long as no cycle.

Prim's - Select shortest edge connected to existing MST that doesn't create cycle.

Huffman Coding - Used for data compression

Ex. a = 0.4, b = 0.1, c = 0.2, d = 0.15, e = 0.11, f = 0.04

Greedy Algos have greedy choice property and optimal substructure

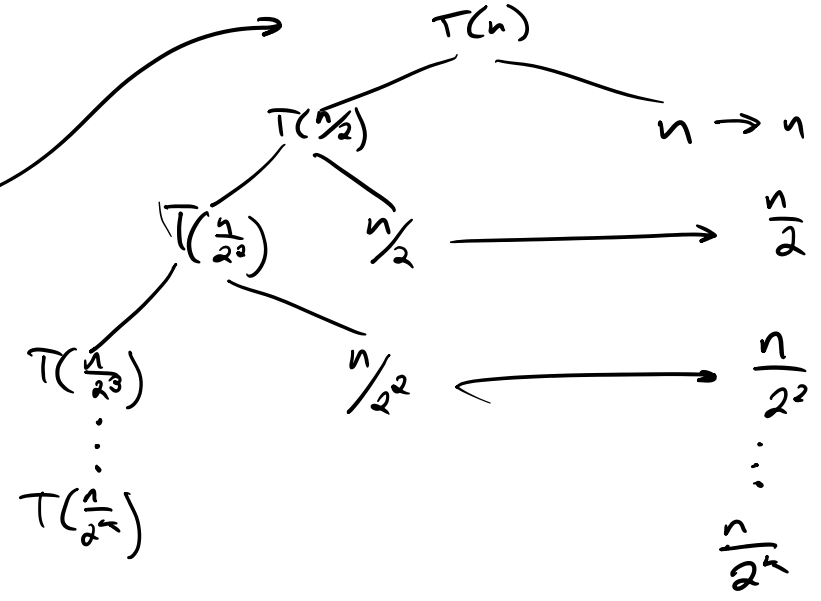


← Huffmen Tree

Examples:

Recurrence Trees -

$$T(n) = \begin{cases} 1 & n=1 \\ T(\frac{n}{2}) + n & n>1 \end{cases}$$



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

$$= n\left(\frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^k}\right)$$

$$= n\sum_{i=0}^{k} \frac{1}{2^i}$$

$T(n) = n \Rightarrow \boxed{\Theta(n)}$

Get T(n) from Code - $\Rightarrow T(n) = 2T(\frac{n}{2}) + \Theta(n), \quad n > 1$

```
def foo(n)
    if n>1
        for i = 0 to n
            print("hey")
        foo(n/2)
        foo(n/2)
```

$T(n) = \Theta(1), \quad n = 1$

Insertion Sort:
Worst Case: $\Theta(n^2)$
Best Case: $\Omega(n)$

Ex. $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$ - Use Recursion Tree

$T(n) \in \Omega(n \log(n))$ path $= \log_3 n$

$cn(\log_3 n + 1) \geq cn \log_3 n = \frac{c}{\log_3} n \log n$

$Cn(\log_3 n + 1) \geq Cn \log_3 n = \frac{C}{\log_3} n \log n$