CSE 102 11-18-21

- midterm 2 : 2PM – 10PM

  time limit : 120 min = 2 hours.

## Greedy Algorithms

- Read : 16.3 Huffman Codes
- Read : 16.4 Matroids and the greedy algorithm.

# Lower Bounds & Comp. complexity

Consider some problem, $\mathcal{P}$ in all of its instances. We have 2 goals

1.) determine an algorithm that solves $\mathcal{P}$, and find an asymptotic upper bound $O(f(n))$ on its runtime. We aim to reduce $f(n)$ as far as possible by discovering better algorithms for $\mathcal{P}$.

2.) Prove that any algorithm solving $\mathcal{P}$ must run in time $\Omega(g(n))$. We aim to increase $g(n)$ by finding finer proofs.

We're happy when $f(n) = \Theta(g(n))$.

(1) is called Algorithmics, or Algorithm design and analysis.

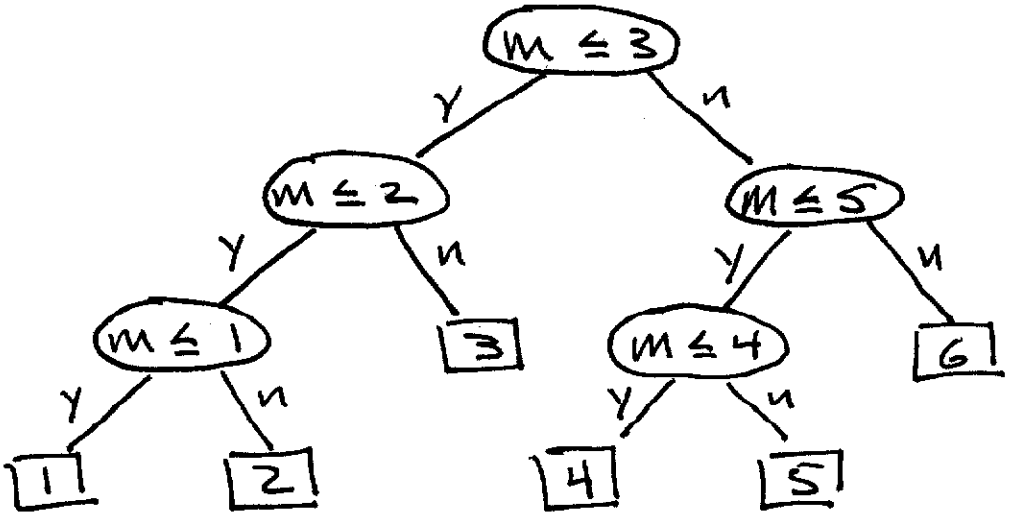
(2) is called Computational complexity.

## Decision Tree Lower Bounds

Ex. Let $m \in \{1, 2, 3, 4, 5, 6\}$, unknown.

Game: determine $m$ by asking a seq. of yes/no questions.

decision tree for binary search:



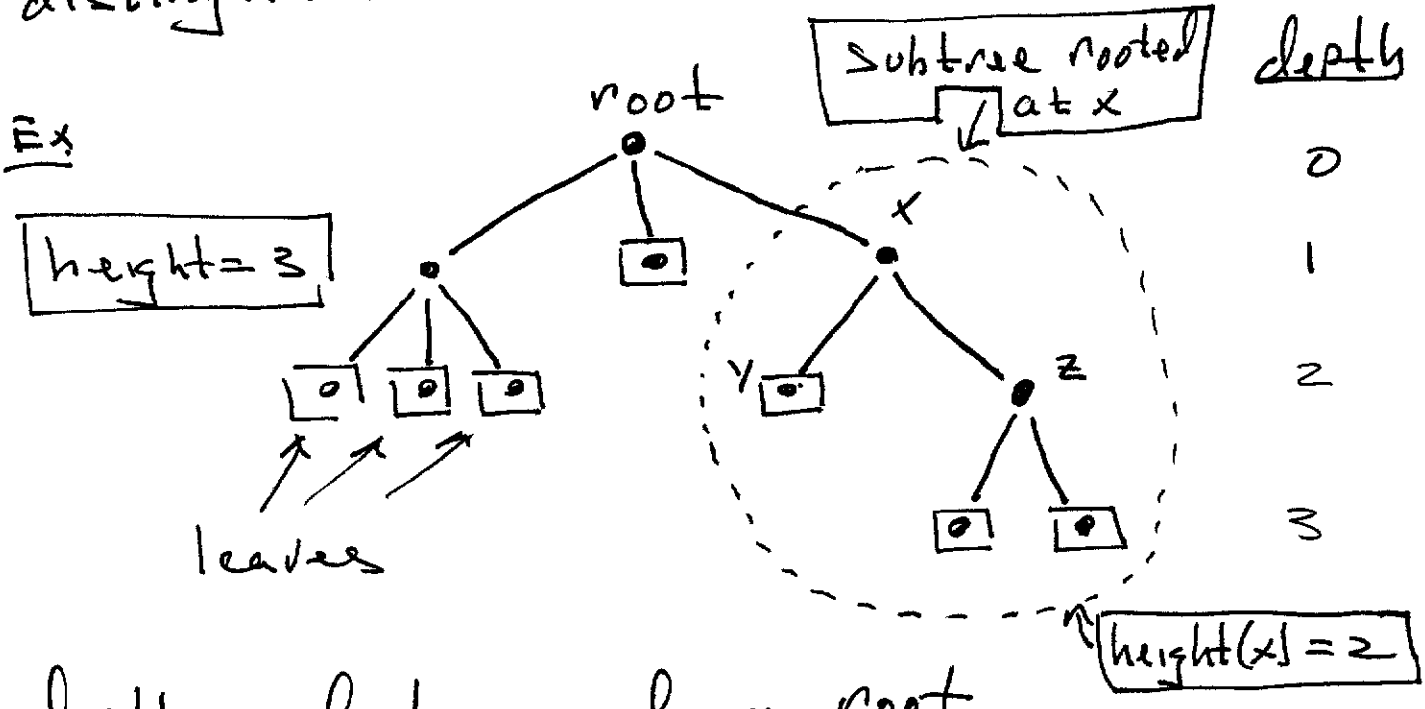
worst case # of comp. = 3

avg. case # of comp

$$= \frac{4 \cdot 3 + 2 \cdot 2}{6} = \frac{16}{6} = \frac{8}{3} = 2.66\ldots$$

## Trees:

A rooted tree is a tree with a distinguished vertex called root.




depth = distance from root.

Parent: unique vertex both adjacent and one closer to root

note: root has no Parent

The adjacent vertices to a node, other than Parent are children.

leaf: a node that has no children

internal node: a non-leaf.

height of a tree: depth of its deepest leaf.

height of a node: height of subtree rooted at that node.

note: height of a leaf is necessarily 0.

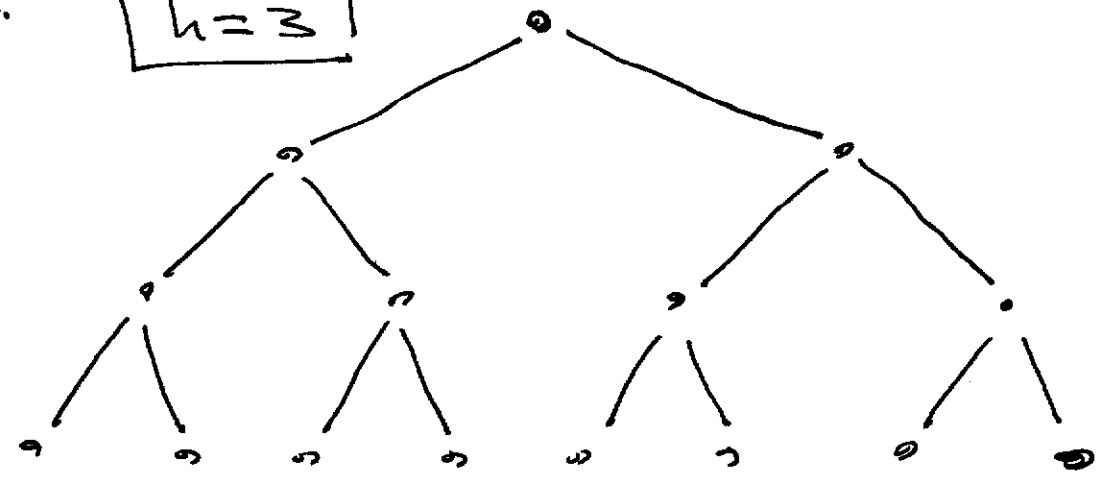
**Defn**

a *k-ary tree* is a rooted tree where each node has at most k children.

**Defn**

a *complete binary tree* (CBT) is a binary tree in which every internal node has exactly two children.

Ex. $h=3$



| depth | # nodes |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |

$(\#\text{ nodes at depth } d) = 2^d$

$\#\text{ leaves} = n = 2^h$

Observe: $\boxed{h = \lg(n)}$ $h$ = height, $n$ = # leaves

i.e. in a CRT

$$\text{height} = \lg(\#\text{leaves}).$$

**Theorem**

Let $T$ be a binary tree with $n$ leaves and height $h$. Then

$$h \geq \lceil \lg(n) \rceil.$$

**Proof.**

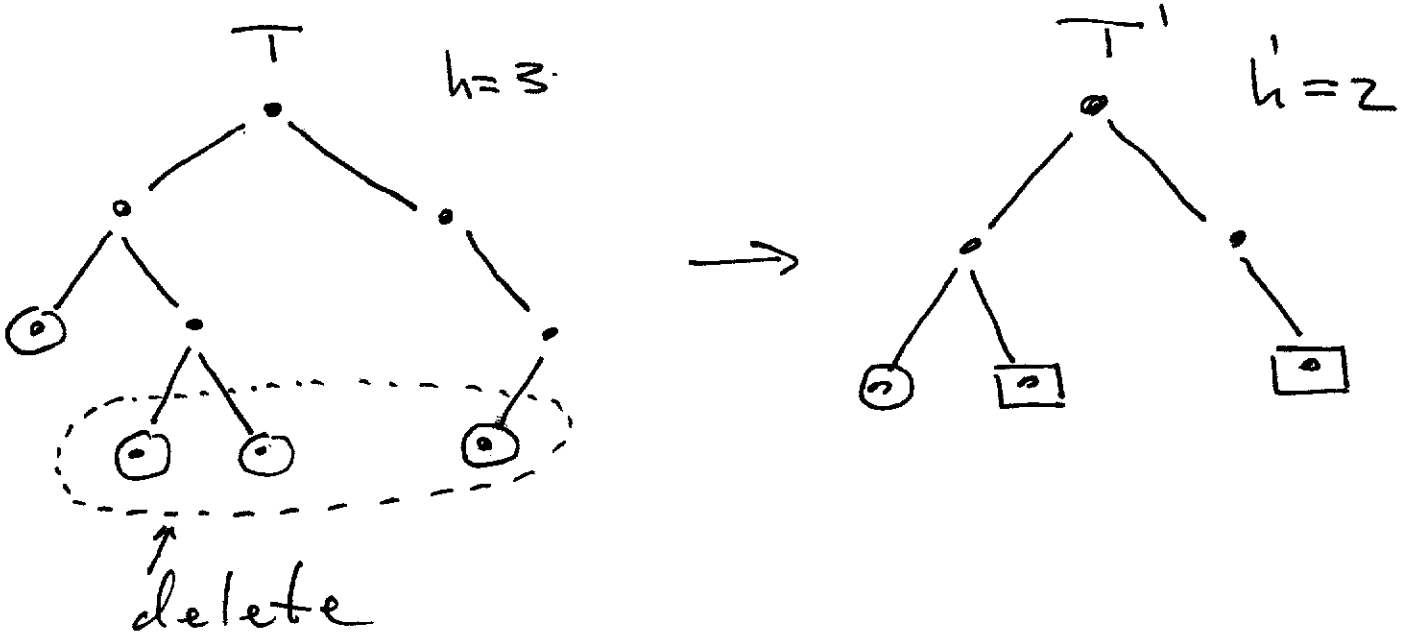
Let $L(T)$ and $H(T)$ denote the # of leaves and height (resp.) of a binary tree $T$.

we use induction on $h = H(T)$.

I. if $h = 0$, we have root with no children, so $n = L(T) = 1$. The inequality $h \geq \lceil \lg(n) \rceil$ is $0 \geq 0$, which is true.

IIb. Let $h > 0$. Assume for any binary tree $T'$ with $H(T') = h - 1$ that $H(T') \geq \lceil \lg(L(T')) \rceil$. we must show that $H(T) \geq \lceil \lg(L(T)) \rceil$, i.e. $h \geq \lceil \lg(n) \rceil$.

Let $T'$ be the binary tree obtained from $T$ by deleting all leaves at depth $h$ (with all incident edges.)




since each node in $T$ has at most 2 children, we have $L(T) \leq 2L(T')$.

Hence

$$L(T') \geq \frac{L(T)}{2}.$$

note also: $H(T') = H(T) - 1 = h - 1$

so

$$h - 1 = H(T')$$

$$\geq \lceil \lg(L(T')) \rceil \quad \text{(by ind. hyp.)}$$

$$\geq \left\lceil \lg\left(\frac{L(T)}{2}\right) \right\rceil$$

$$= \lceil \lg(L(T)) - 1 \rceil$$

$$= \lceil \lg(n) \rceil - 1$$

$\therefore \quad h \geq \lceil \lg(n) \rceil$, as required. ∎

Exercise: Let $T$ be a $k$-ary tree with $n$ leaves and height $h$. Prove that $h \geq \lceil \log_k(n) \rceil$.

How do we use this to reason about algorithms?

Given $\mathcal{P}$, consider all algorithms that solve $\mathcal{P}$ by doing a seq. of operations, each having one of $K$ possible outcomes. (We <u>$K$-ary Probes</u> or <u>Probes</u>.) Let $n$ denote the size of an instance. Let $f(n)$ be the # of possible algorithm outputs for instances of size $n$ (i.e. <u>verdicts</u>).

⋮

Continue P.3 of handout.