cse 102 3-3-20

Lower Bounds

Let P be a Problem, i.e. the set of all instances of a Problem.

Two goals:

1. find an algorithm that solves P, also find an upper bound $O(f(n))$ on its runtime. We wish to reduce $f(n)$ as much as possible.

2. Prove that any algorithm solving P runs in time $\Omega(g(n))$. We wish to increase $g(n)$ as much as possible.

Happy when $f(n) \sim g(n)$, or at least $f(n) = \Theta(g(n))$.

Two kinds of L.B.s

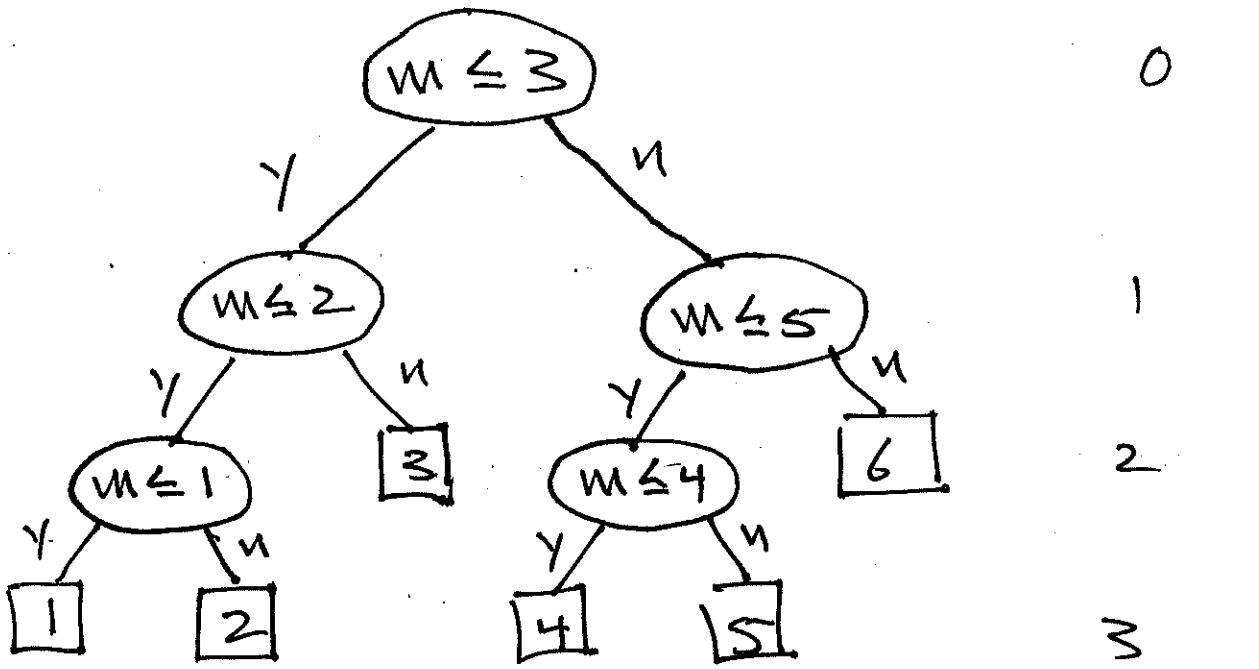
- Decision tree L.B.s (information theoretic.)
- Adversary arguments.

Ex. Game:

(1) I think of $m \in \{1, 2, 3, 4, 5, 6\}$

(2) you try to guess it, by asking only yes/no questions.

To play (2) use 'binary search'



Decision tree ↑

conclude: at most 3 questions are necessary. Also the average # of questions is

$$\frac{2+2+3+3+3+3}{6} = \boxed{2.66}$$

Definitions

rooted tree: tree with distinguished vertex called root.

node = vertex.

depth of a node: distance from root.

Parent of a node x: unique node on path from root to x, with depth 1 less.

a child of x: a node having x as parent.

a leaf: a node having no children. an internal node: non-leaf

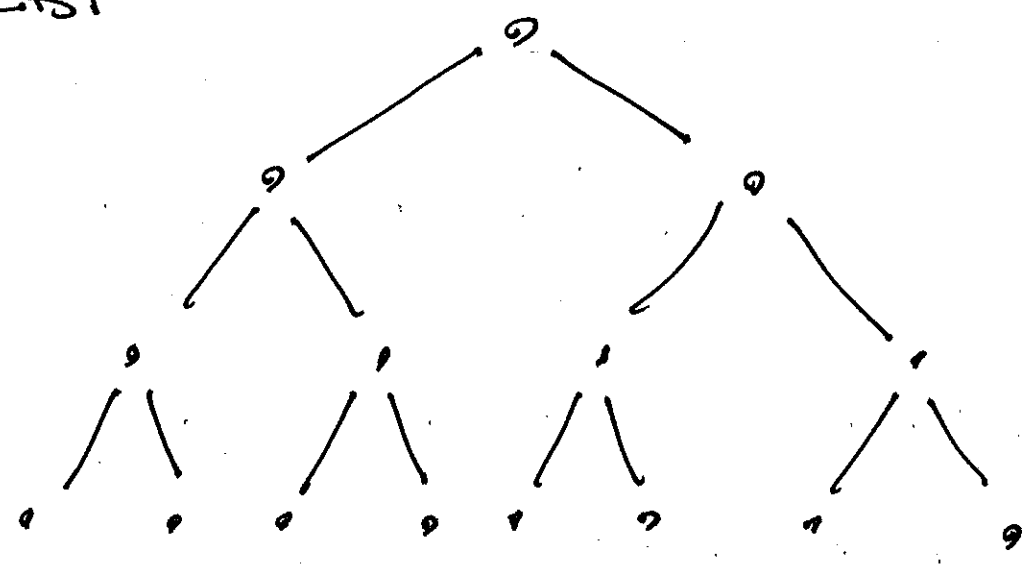
k-ary tree: all nodes have at most k children.

Complete Binary tree (CBT)

- all leaves at same depth
- every internal node has exactly 2 children

likewise for complete k-ary trees.

CBT



depth	#nodes
0	1
1	2
2	4
3	8

(# nodes at depth d) $= 2^d$

Height of a rooted tree: depth of deepest leaf.

If a CBT has height h, and n leaves,

Then $n = 2^h$, so

$$\boxed{h = \lg(n)}$$

Also CBT's have minimum possible height among all B.T.'s having same # of leaves.

Theorem

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

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

Proof.

let $L(T)$, $H(T)$ denote # leaves & height of T (resp).

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

B. if $h = 0$, then T has just 1 node, the root, which is a leaf. $\therefore n = L(T) = 1$ and

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

becomes

$$0 \geq 0$$

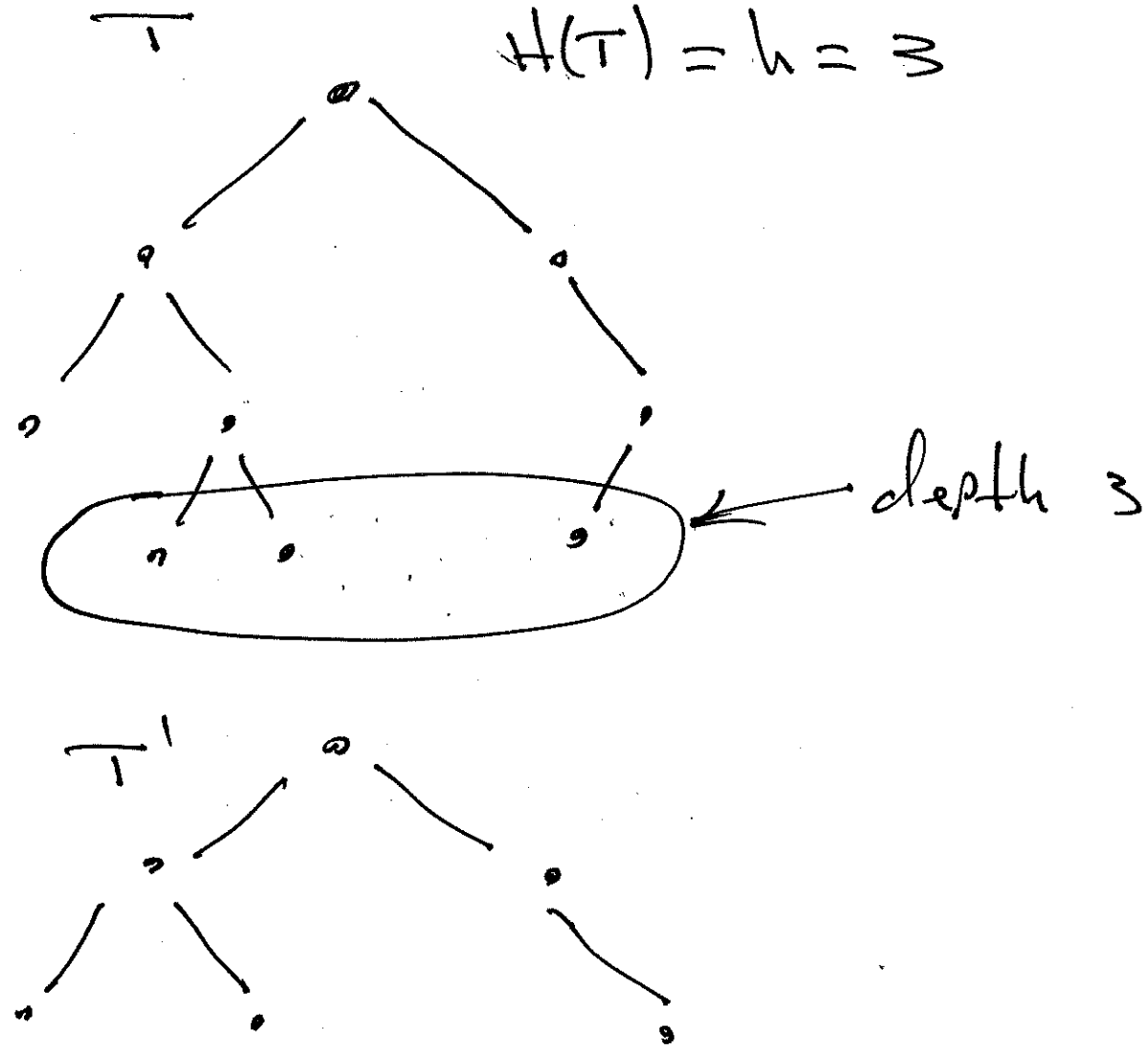
which is true.

II. Let $h > 0$. Assume for any B.T. T' with $H(T') = h-1$ that $H(T') \geq \lceil \lg(L(T')) \rceil$. We must show that if T is a B.T. with $H(T) = h$ and $L(T) = n$, then

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

Let T be a B.T. with height h and n leaves.

Let T' be the tree obtained by deleting all leaves in T at depth h.



Then $H(T') = h-1$. By ind. hyp.

$$H(T') \geq \lceil \lg(L(T')) \rceil$$

Also since T is a binary tree each node has at most 2 children. Hence

$$L(T) \leq 2L(T')$$

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

Thus

$$\begin{aligned} h-1 &= H(T') \\ &\geq \lceil \lg(L(T')) \rceil \quad \left\{ \begin{array}{l} \text{by ind.} \\ \text{hyp.} \end{array} \right. \end{aligned}$$

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

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

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

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

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

Exercise:

Let T be a K-ary tree with n leaves & height h.

Prove $h \geq \lceil \log_K(n) \rceil$

Given Problem Γ, consider all algorithms that solve Γ by asking a seq. of K-ary questions (Probes), i.e. questions having at most K answers.

Let n denote size of an instance of Γ, and let f(n) denote the number of poss. verdicts for that size.

Any algorithm for Γ can be represented by a K-ary decision tree.

The internal nodes are k-ary probes of input data, leaves are verdicts

every descending path from root to leaf is a poss. execution seq. of queries.

Note: cannot have fewer leaves than verdicts, i.e.

$$\#\text{leaves} \geq f(n)$$

So in this tree T:

$$f(n) \leq L(T).$$

By the exercise

$$h \geq \lceil \log_k(L(T)) \rceil \geq \lceil \log_k(f(n)) \rceil .$$

we've proved:

Theorem

If $\mathcal{P}$ has $f(n)$ pos. verdicts on input of size n, then no algorithm for $\mathcal{P}$ that performs only k-ary probes, can perform fewer than $\lceil \log_k(f(n)) \rceil$ such probes in worst case.

i.e. $\lceil \log_K(f(n)) \rceil$ is a lower bound on run time of such an algorithm.

$$\lceil \log_K(f(n)) \rceil = \Theta(\log(f(n)))$$

Important Note: all this does not imply that such an algorithm exists. It only says no algorithm doing fewer than $\lceil \log_K f(n) \rceil$ K-ary probes does exist.

Ex. guess $w \in \{1, 2, 3, 4, 5, 6\}$
using yes/no questions.

$K = 2$

$n = 6$

$f(n) = 6$

$\therefore$ # questions $\geq \lceil \log_2(6) \rceil = 3$

Ex guess $w \in \{1, 2, \ldots, 10^6\}$
using y/n questions

questions $\geq \lceil \log_2(10^6) \rceil = 20$

Ex guess $m \in \{1, \ldots, 10^6\}$ using 3-way questions.

$$\#\text{questions} \geq \lceil \log_3(10^6) \rceil = 13$$

Theorem

Any comparison based algorithm must do, in worst case, at least $\lceil \lg(n!) \rceil$ comparisons on input arrays of length n.

$$\#\text{comp} \geq \lceil \lg(n!) \rceil$$

asymp. lower bound $= \Theta(\log(n!))$

$= \Theta(n \log n)$,