CSE 102 5-23-24

Recorded Lecture

---

more greedy algorithms: Read

- (16.3) Huffman codes
- (16.4) Matroids, the greedy algorithm.

Handout: Lower bounds

Consider a problem P. Let n denote the size of an instance of P.

Goals:

1. determine an algorithm that solves P. find an asym. upper bound $O(f(n))$ on its runtime. We aim to reduce $f(n)$ by finding better algorithms

2. Prove that any algorithm solving P runs in time $\Omega(g(n))$ for some $g(n)$. We aim to increase $g(n)$ by finding better proofs.

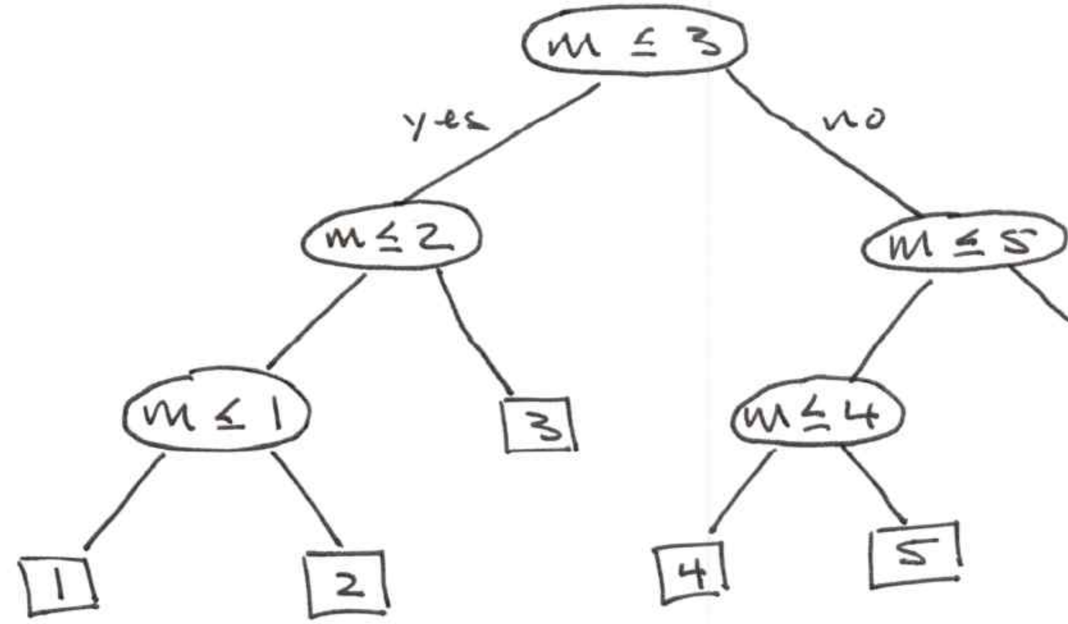
We're happy when $f(n) = \Theta(g(n))$. Then we have best possible algorithm.

(1) is algorithmics. (2) is called computational complexity.

Decision Tree Lower Bounds

Ex. let $m \in \{1, 2, 3, 4, 5, 6\}$. Problem: Determine m by asking a seq. of yes/no questions.

Variation on Binary Search.



avg. # Probes $= \frac{4 \cdot 3 + 2 \cdot 2}{6} = \boxed{2.66}$

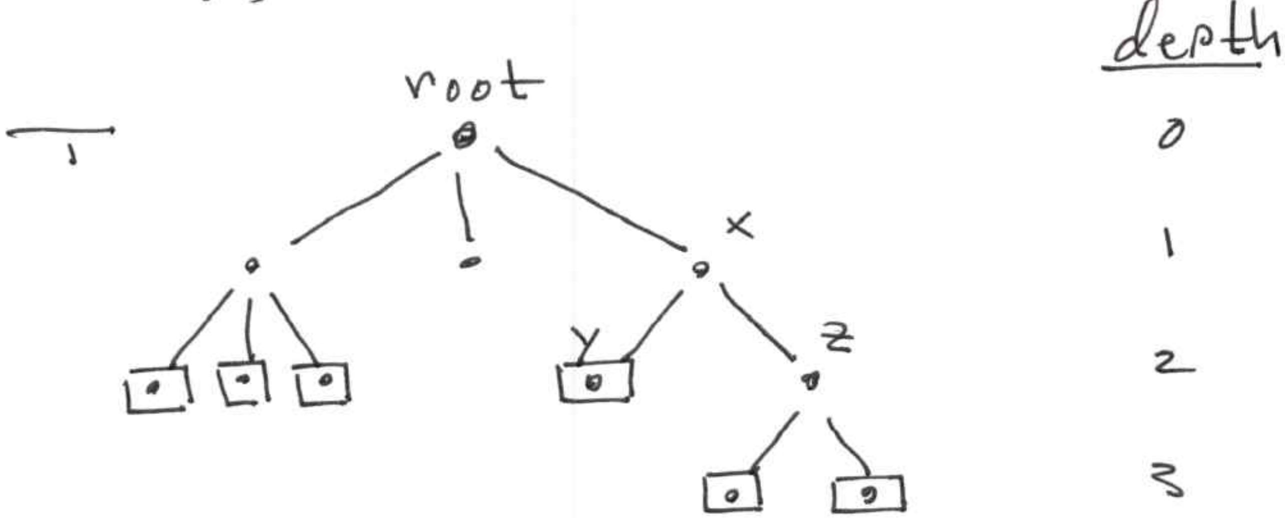
Note:

- 2 questions are not sufficient. There are only 4 combinations of 2 yes/no answers: (y, y), (y, n) (n, y), (n, n), not enough to distinguish all 6 possible verdicts
- 3 is a lower bound for runtime of any algorithm solving this problem.
- 3 is also an upper bound

Defn

A <u>rooted tree</u> is a tree with a distinguished vertex (node) called <u>root</u>.

- <u>depth</u> of a node is distance from root
- <u>Parent</u> of a node y is unique node x adj to y, one closer to root



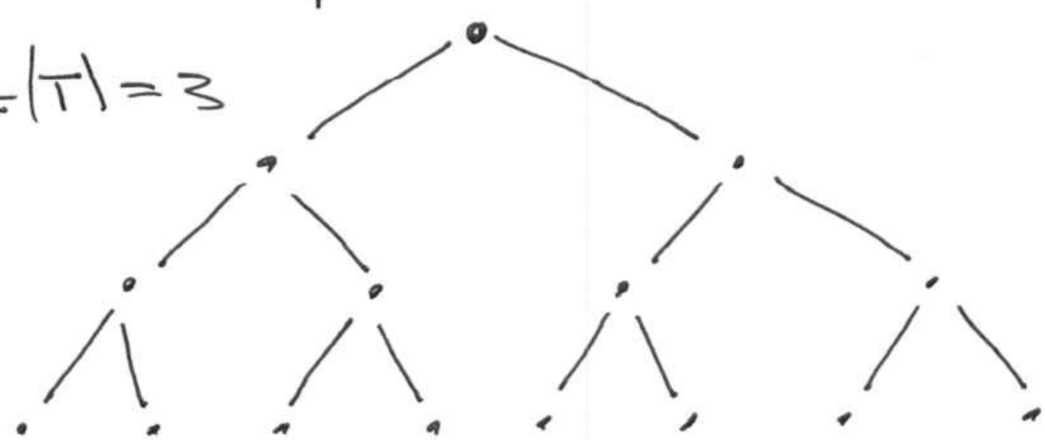
- <u>child</u> of x is any node having x as Parent
- <u>leaf</u> is any node having no children
- <u>internal node</u> is a non-leaf.
- height(T) is depth of deepest leaf.
- height(x) is height of subtree rooted at x.

- <u>K-ary tree</u> is a rooted tree in which each node has at most 2 children.

- <u>complete binary tree</u> (CBT)
  every internal node has exactly 2 children, all leaves at same depth.

T

height(T) = 3



depth	# nodes
0	1
1	2
2	4
3	8

<u>note</u> # node at depth d is 2^d.

$\therefore$ # leaves $= 2^h$ where $h = \text{height}(T)$.

<u>note</u>: if $\underline{\underline{n = \#\text{leaves}}} = 2^h$, then

$$h = \lg(n)$$

Theorem

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

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

Proof

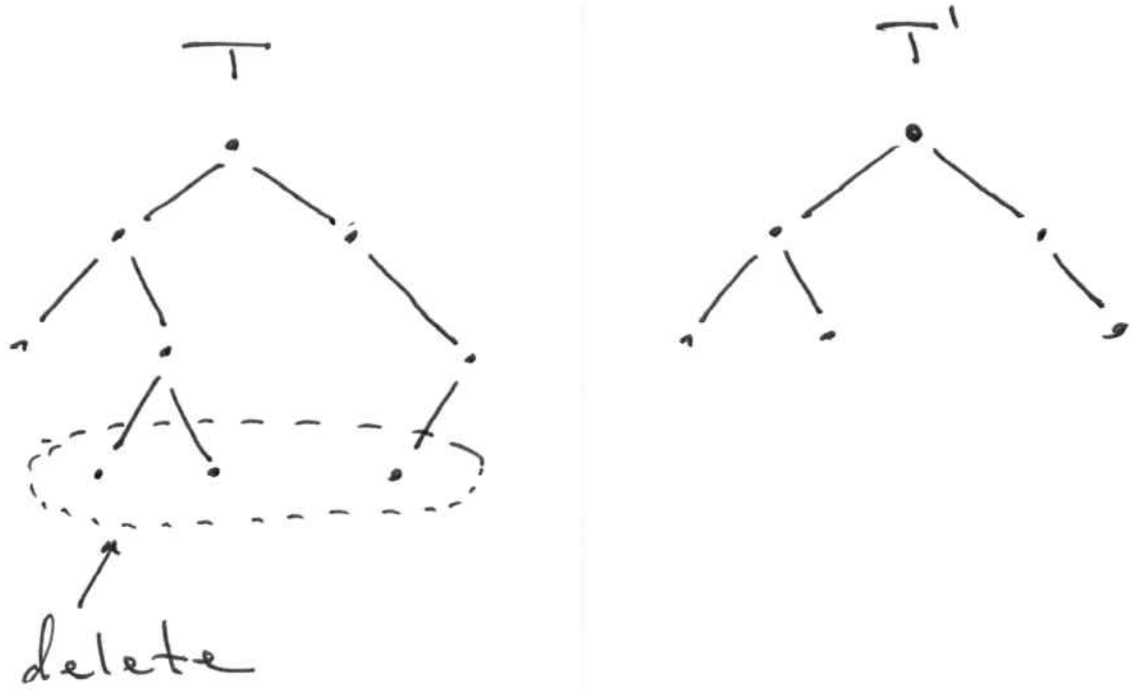
Let $L(T) = \#$ leaves in T, and $H(T) =$ height(T). Use induction on $h = H(T)$.

I. If $h = 0$, then T has only one node, the root, which is a leaf. $\therefore$ $n = L(T) = 1$. So $h \geq \lceil \lg(n) \rceil$ becomes $0 \geq 0$ which is true.

II b. 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 $H(T) \geq \lceil \lg(L(T)) \rceil$, i.e. $h \geq \lceil \lg(n) \rceil$.

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

<u>illustrate</u>



observe $H(T') = h-1$. By the induction hypothesis $H(T') \geq \lceil \lg(L(T')) \rceil$. Since each node in T has at most 2 children, we have $L(T) \leq 2L(T')$, Hence $L(T') \geq \frac{L(T)}{2}$. Now

$$
\begin{aligned}
h-1 &= H(T') \\
&\geq \lceil \lg(L(T')) \rceil \text{ by ind. hyp.} \\
&\geq \left\lceil \lg\left(\frac{L(T)}{2}\right) \right\rceil \\
&= \lceil \lg(L(T)) - 1 \rceil \\
&= \lceil \lg(L(T)) \rceil - 1 = \lceil \lg(n) \rceil - 1
\end{aligned}
$$

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

$\blacksquare$

Exercise

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

How to find a lower bound using k-ary trees. Given P, consider all algorithms that solve P by performing a sequence of basic operations, each with one of k possible outcomes: called k-ary probes. Let n denote the size of an instance, and $f(n)$ the the # of possible algorithm outputs (verdicts)

Any algorithm of this kind can be represented by a k-ary decision tree. Each internal node represents a probe of the input data, Each of its k children represents the outcome. Each leaf represents an algorithm output, i.e. a verdict. Each descending path from root to leaf represents a sequence of probes of the data, leading to an algorithm output.

Its possible that more than one path leads to same verdict. But there cannot more verdicts than leaves. Thus $f(n) \leq L(T)$

By the exercise

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

We have proved.

Theorem

No algorithm for P that uses only K-ary probes can perform fewer than $\lceil \log_K(f(n)) \rceil$ such probes on input of size n.

$\therefore$ $\lceil \log_K(f(n)) \rceil$ is a lower bound for the worst case runtime of such an algorithm, asymptotically the lower bound is $\Omega(\log(f(n)))$.

Ex. guessing game again. Let

$S = \{1, 2, 3, \dots, n\}$.

Problem: find $m \in S$ by asking only k-ary questions. we have $f(n) = n$ Possible verdicts. any valid algorithm must ask at least $\lceil \log_k(n) \rceil$ questions in worst case.

If $n = 6$, $k = 2$ then $\lceil \lg(6) \rceil = 3$. If $n = 1000000 = 10^6$, $k = 2$, then $\lceil \lg(10^6) \rceil = 20$

If $n = 10^6$ and $k = 3$, then

$$\lceil \log_3 10^6 \rceil = 13.$$

Theorem

Any comparison based sorting algorithm must do, in worst case, at least

$$\lceil \lg(n!) \rceil$$

comparisons on arrays of length n.

Proof

A verdict for an array $A[1...n]$ of length n is a re-arrangement of the array, of which there are $f(n) = n!$. Each comparison ($A_i \le A_j$ or $A_i < A_j$) has one of 2 possible outcomes: true or false. ∴ worst case # comparisons is $\ge \lceil \lg(n!) \rceil$. Asymptotically

worst case runtime $\Omega(n \log n)$, by stirling's formula. ∎