CSE 102 11-23-21

- Project: ext. 2 days → Wed. (last!)
- midz: solutions Posted
- final exam: Thur. 12-9-21
- hw7: Posted
- office hrs

Theorem

If T is a k-ary tree with n leaves and height h, then

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

Proof: exercise.

Theorem

Suppose a problem P has $f(n)$ possible outputs (verdicts) on input of size n.

Then no algorithm for P that performs only k-ary probes as basic operation (an op. having at most k outcomes) can perform fewer than $\lceil \log_k(f(n)) \rceil$ such probes on input of size n, in worst case. Thus $\lceil \log_k(f(n)) \rceil$ is a lower bound on the worst case runtime of such algorithms.

Note: This Thm does not assert existence of an algorithm solving P, only the non-existence of one that solves too efficiently.

Proof.

Given P, consider algorithms that solve P using only k-ary probes.

n : Problem size

$t(n)$: # of possible outputs (verdicts)

Represent operation of such an algorithm by a decision tree.

internal nodes : branch points at k-ary probes, so have at most k children

leaves : outputs, verdicts.

a downward path in this k-ary tree from root to a leaf represents a seq. of probes that lead to an output.

so max length of such a seq. is height of tree, h. let $L(T)$ = # leaves. Then

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

note: there could be multiple leaves representing the same verdict, i.e. diff. paths to same output. so we could have

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

For the algorithm to work, we can't have $L(T) < f(n)$.

Thus

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

$\therefore$ $\lceil \log_K(f(n)) \rceil$ is a lower bound on # of K-ary Probes, i.e the runtime (worst case.) ∎

Note:
changing K changes lower bound, but not the asymptotic. lower bound:

$$\Omega(\log(f(n)))$$

for any K.

Ex. recall guessing game:

guess a number $m \in S = \{1, 2, \ldots, n\}$

asking only 2-ary questions.

say $n=6$, $K=2$: so $f(n) = n$, so

lower bound $\lceil \log_2(n) \rceil = \lceil \lg(6) \rceil = 3$, so

2 questions will not suffice.

say $n = 1{,}000{,}000$ $K=2$: $f(n) = n$ so

L.B: $\lceil \lg(10^6) \rceil = \lceil 6 \cdot \lg(10) \rceil = 20$

say $n = 10^6$, $K=3$, $f(n)=n$:

L.B. $\lceil \log_3(10^6) \rceil = 13$, so 12 questions

do not suffice.

Theorem

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

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

comparisons on input of length n.

Comparison based means can only query the input array by comparing sizes of its elements.

Asymptotically, the lower bound

$$\lceil \lg(n!) \rceil = \Omega(n \log n)$$

Proof:

Note comparison $A_i < A_j$ has 2 outcomes, true or false. So there are 2-ary Probes. $\boxed{k=2}$

Also note: # of re-arrangements of n distinct array elements is $\boxed{f(n) = n!}$, each of which is a possible output. Thus the worst case # of comparisons is

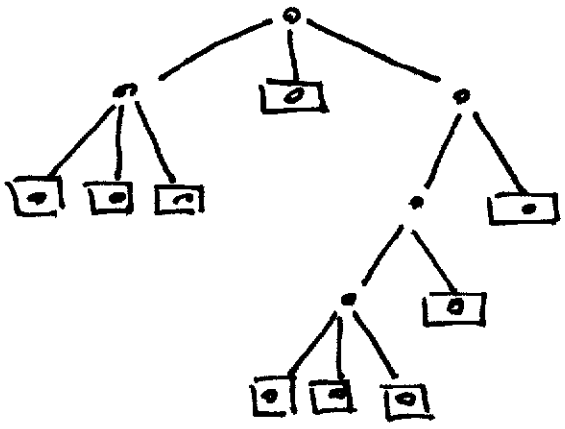
$$\lceil \lg(n!) \rceil = \Omega(n \log n).$$

Defn

The <u>Average height</u> of a k-ary tree is its average leaf depth. (We assume each leaf is equally weighted in this average.) So if T has n leaves at depths $d_1, d_2, \ldots, d_n$ then avg. height is

$$a = \frac{\sum_{i=1}^{n} d_i}{n}$$

Ex. $k=3$, $n=9$



depth

0

1

2

3

4

~ 0

$$a = \frac{1 \cdot 1 + 4 \cdot 2 + 1 \cdot 3 + 3 \cdot 4}{9}$$

$$= \frac{24}{9} = \frac{8}{3} = 2.66\ldots$$

Theorem

The average height a of a K-ary tree having n leaves satisfies

$$a \geq \log_K(n)$$

Note: See P. 416 of B.B.

Here a proof of case $K = 2$

Theorem

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

$$\lg(n!) = \Omega(n \log n)$$

comparisons on input arrays of length n.

Summary: (decision tree lower bounds)

(1) determine K, the maximum # of outcomes to each Probe of data.

(2) determine $f(n)$, the # of Possible algorithm outputs (verdicts)

(3) Conclude that any algorithm using only Probes of Kind in (1) must do at least

- worst case: $\lceil \log_k(t(n)) \rceil = \Omega(\log(t(n)))$
- avg case: $\log_k(t(n)) \rceil = \Omega(\log(t(n)))$

Probes on input of size n

Warning: nothing here asserts the existence of algorithms that achieve the lower bound, only the non-existence of algorithms that do better.

Adversary Arguments

Ex. Guessing game again

guess $x \in S = \{1, 2, \ldots, n\}$

Players

A: randomly choose $x \in S$, answer queries from

B: asks a sequence $Q_1, Q_2, \ldots$, of yes/no questions.

Suppose A cheats: doesn't really have a number $x \in S$. instead A gives a sequence of answers, each consistent with prior answers and

each implying that the search space is as large as possible, and thereby prolonging the game.

At some point the candidates for x reduce to one element. A commits to that # and ends the game. It just appears that B was unlucky, since all A's answers are consistent with x.

Notation

Let S_i = {remaining candidates after Q_i is asked and answered.}

$S_0 = S$.