# Lower Bounds & Computational Complexity

Consider the set of all algorithms (known & unknown) which solve some problem $P$ in all its instances.

Our goals are twofold:

1.) Find an algorithm which solves $P$ in (worst case) time $O(f(n))$ for some function $f(n)$ which we aim to reduce as far as possible.

2.) Prove that any algorithm which solves $P$ must run in (worst case) time $\Omega(g(n))$ for some function $g(n)$ which we aim to increase as far as possible.

Here $n$ denotes the 'size' of an instance of problem $P$.

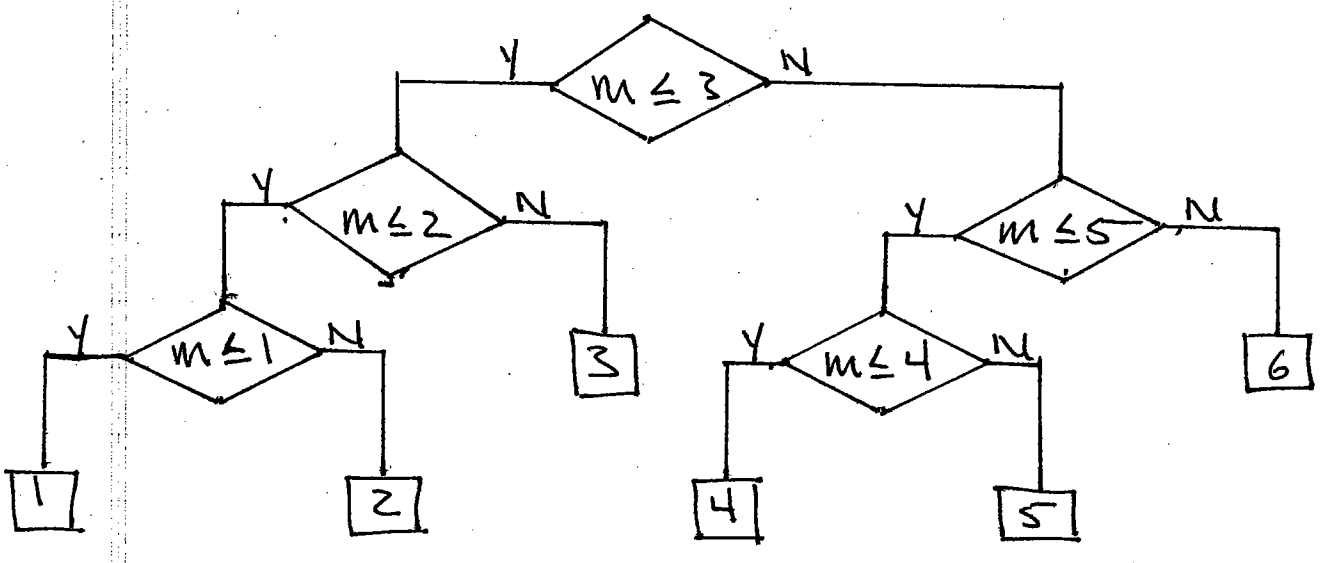
We are happy when $f(n) = \Theta(g(n))$ for then we know we have the best possible algorithm to solve $P$ (apart from improvements in hidden constants.)

(1) is called <u>Algorithmics</u> by some authors, while (2) is the Theory of <u>Computational Complexity</u>. The function $g(n)$ in (2) is called a <u>lower bound</u> on the complexity of Problem P.

## <u>Decision Trees / Information Theoretic Lower Bounds</u>

<u>Ex.</u> Let $m$ be an integer in the range $1 \le m \le 6$. Problem: Determine the value of $m$ by asking a sequence of Yes/No questions.

This problem is similar to binary search!



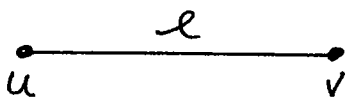

Worst case # comp. = 3

Avg. case # comp. $= \frac{3+3+2+3+3+2}{6} = \frac{16}{6} = 2.66$

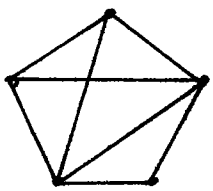
APPARENTLY THE ANSWER CAN BE OBTAINED BY ASKING NO MORE THAN 3 QUESTIONS. (will 2 SUFFICE ?)

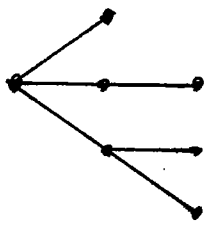
REVIEW: GRAPHS - TREES - ROOTED TREES.

A GRAPH $G = (V, E)$ is A PAIR OF SETS CALLED VERTICES ($V$) AND EDGES ($E$). EACH EDGE JOINS A (UNIQUE) PAIR OF (DISTINCT) VERTICES. TWO VERTICES WHICH ARE JOINED BY AN EDGE ARE CALLED ADJACENT

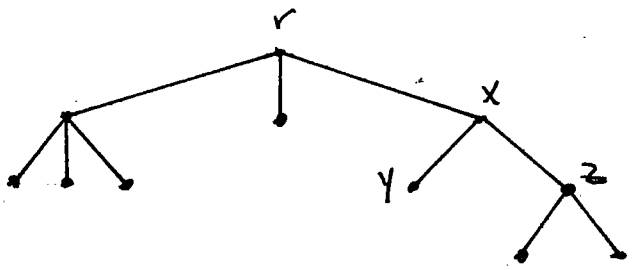



A PATH IN G IS A SEQUENCE OF CONSECUTIVELY ADJACENT VERTICES. G IS CALLED CONNECTED IF EVERY PAIR OF VERTICES IN G ARE JOINED BY A PATH. A CYCLE IS A CLOSED PATH, i.e. A PATH IN WHICH THE INITIAL AND TERMINAL VERTICES ARE IDENTICAL. G IS CALLED ACYCLIC IF IT CONTAINS NO CYCLES. A TREE IS A CONNECTED ACYCLIC GRAPH.

GRAPH: 

TREE: 

A <u>ROOTED TREE</u> IS A TREE IN WHICH ONE VERTEX HAS BEEN DISTINGUISHED AS THE <u>ROOT</u>. THE VERTICES IN SUCH A TREE ARE OFTEN CALLED <u>NODES</u>. THE <u>DEPTH</u> OF A NODE IS ITS DISTANCE FROM THE ROOT. (DISTANCE MEANS SHORTEST PATH LENGTH.)




DEPTH

0

1

2

3

THE <u>CHILDREN</u> OF A NODE x ARE THOSE NODES ADJACENT TO x WHOSE DEPTH IS ONE MORE THAN THAT OF x. THE <u>PARENT</u> OF x IS THE (UNIQUE) NODE WHICH IS ADJACENT TO x AND HAS DEPTH ONE LESS THAN THAT OF x.

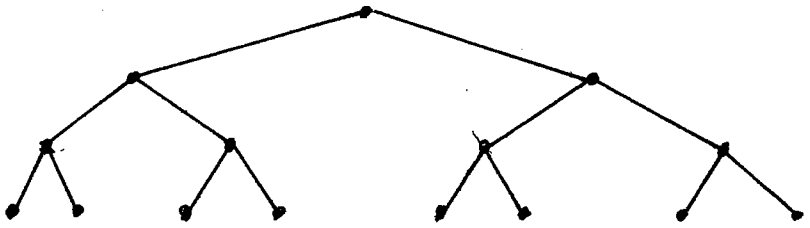
THE ROOT IS THE ONLY NODE WHICH HAS NO PARENT (SINCE THE ROOT HAS DEPTH 0.) IF A NODE y HAS NO CHILDREN, IT IS CALLED A <u>LEAF</u>. A NON-LEAF NODE IS CALLED AN <u>INTERNAL NODE</u>.

THE HEIGHT OF A ROOTED TREE IS ITS MAXIMUM NODE DEPTH, I.E. THE LENGTH OF A LONGEST DOWNWARD PATH FROM THE ROOT TO A LEAF. THE HEIGHT OF A NODE IS THE HEIGHT OF THE SUBTREE ROOTED AT THAT NODE.

A BINARY TREE IS A ROOTED TREE IN WHICH EACH NODE HAS AT MOST 2 CHILDREN. MORE GENERALLY A K-ARY TREE IS A ROOTED TREE IN WHICH EACH NODE HAS AT MOST K CHILDREN.

A COMPLETE BINARY TREE (CBT) IS A BINARY TREE IN WHICH ALL LEAVES ARE AT THE SAME DEPTH, AND EACH INTERNAL NODE HAS EXACTLY 2 CHILDREN

$h=3$



| DEPTH | # NODES |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |

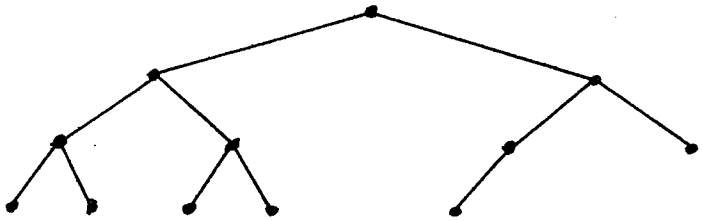
# NODES AT DEPTH $d = 2^d$

# LEAVES = # NODES AT DEPTH $h = 2^h$

∴ THE HEIGHT OF A CBT WITH $n$ LEAVES IS $h = \lg(n)$.

AN <u>ALMOST COMPLETE BINARY TREE</u> (ACBT) IS A BINARY TREE WHICH HAS THE MAXIMUM POSSIBLE NUMBER OF NODES AT EACH DEPTH, EXCEPT POSSIBLY THE LAST, WHICH IS FILLED FROM LEFT TO RIGHT.

$h = 3$



(AN ACBT IS THE BASIS OF THE HEAP DATA STRUCTURE.)

<u>EXERCISE</u>

PROVE THAT AN ACBT WITH $n$ LEAVES AND HEIGHT $h$ SATISFIES $\lceil \lg n \rceil \le h \le \lceil \lg n \rceil + 1$

<u>THEOREM</u>

THE HEIGHT $h$ OF ANY BINARY TREE WITH $n$ LEAVES SATISFIES

$$h \ge \lceil \lg n \rceil .$$

<u>NOTATION:</u>

LET $L(T)$ AND $H(T)$ DENOTE THE NUMBER OF LEAVES AND THE HEIGHT OF A

BINARY TREES T.

PROOF.

LET T BE A BINARY TREE WITH $h = H(T)$ AND $n = L(T)$. WE SHOW BY INDUCTION ON $h$ THAT $h \geq \lceil \lg n \rceil$.

I. IF $h = 0$ THEN T CONTAINS JUST ONE NODE, AND $n = 0$. THUS IN THIS CASE $h \geq \lceil \lg n \rceil$.

II. LET $h > 0$ AND ASSUME THE RESULT HOLDS FOR ANY BINARY TREE OF HEIGHT $h-1$. LET T' BE THE BINARY TREE OBTAINED BY DELETING ALL LEAVES AT DEPTH $h$ FROM T (ALONG WITH ALL INCIDENT EDGES.)

OBSERVE THAT $H(T') = h-1$, AND BY THE INDUCTION HYPOTHESIS,

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

SINCE EACH NODE IN T CAN HAVE AT MOST 2 CHILDREN, WE ALSO HAVE $L(T) \leq 2 L(T')$, WHENCE

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

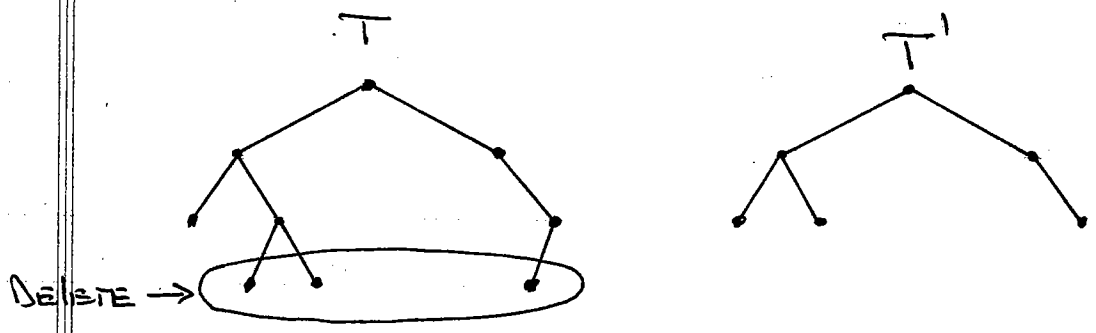
PUTTING THESE INEQUALITIES TOGETHER GIVES

$$\begin{aligned} h-1 &= H(T') \\ &\geq \lceil \lg L(T') \rceil \\ &\geq \lceil \lg \tfrac{L(T)}{2} \rceil \\ &= \lceil \lg (\tfrac{n}{2}) \rceil \\ &= \lceil \lg n - 1 \rceil \\ &= \lceil \lg n \rceil - 1 , \end{aligned}$$

AND THEREFORE $h \geq \lceil \lg n \rceil$ AS REQUIRED.

///.

ILLUSTRATION:




$h = H(T) = 3$
$n = L(T) = 4$

$H(T') = 2$
$L(T') = 3$

NOTE ONE CAN DEFINE IN A SIMILAR MANNER COMPLETE AND ALMOST COMPLETE K-ARY TREES, AND PROVE FORMULAS $h = \log_k n$ AND $\lceil \log_k n \rceil \leq h \leq \lceil \log_k n \rceil + 1$ RESPECTIVELY.

EXCERCISE: CARRY OUT THE ABOVE DEFINITIONS AND PROOFS.

THEOREM:
THE HEIGHT $h$ OF ANY K-ARY TREE WITH $n$ LEAVES SATISFIES

$$h \geq \lceil \log_k n \rceil$$

EXCERCISE: PROVE THIS BY INDUCTION ON THE HEIGHT $h$.

A DECISION TREE IS A WAY TO REPRESENT THE WORKING OF AN ALGORITHM ON ALL POSSIBLE INPUTS OF A GIVEN SIZE, USING A K-ARY TREE.

EACH INTERNAL NODE REPRESENTS AN OPERATION OR TEST OF SOME KIND ON THE INPUT DATA. EACH LEAF REPRESENTS AN OUTPUT OR VERDICT.

EACH DOWNWARD PATH FROM THE ROOT TO A LEAF REPRESENTS A PARTICULAR SEQUENCE OF TESTS LEADING TO A CONCLUSION, i.e. A PARTICULAR LOGICAL PATHWAY TAKEN BY THE ALGORITHM.

$K$ = MAXIMUM NUMBER OF POSSIBLE OUTCOMES TO EACH TEST

$n$ = NUMBER OF LEAVES
$\geq$ NUMBER OF POSSIBLE VERDICTS

$h$ = HEIGHT
= MAXIMUM NUMBER OF TESTS NECESSARY TO REACH A VERDICT.

Ex

WE RETURN TO THE EXAMPLE OF FINDING $m \in \{1, \ldots, 6\}$ BY ASKING ONLY YES/NO QUESTIONS. IS THERE AN ALGORITHM WHICH DETERMINES (ANY) $m$ USING AT MOST 2 QUESTIONS?

# VERDICTS $= n = 6$
# OUTCOMES ON EACH TEST $= K = 2$
MAX # OF QUESTIONS $= h$.

BY THE PREVIOUS THEOREM $h \geq \lceil \lg 6 \rceil = 3$.

OBSERVE THAT THE OPERATION OF ANY ALGORITHM WHICH SOLVES THIS PROBLEM CAN BE REPRESENTED BY A DECISION TREE WITH $K=2$ AND $n=6$ LEAVES.

SINCE $h \geq 3$, WE KNOW 3 IS A LOWER BOUND ON THE WORST CASE NUMBER OF QUESTIONS NECESSARY IN ANY ALGORITHM WHICH SOLVES THIS PROBLEM.

$\therefore$ 2 QUESTIONS DO NOT SUFFICE.

EX

GIVEN $1 \leq m \leq 10^6$, FIND A LOWER BOUND ON THE NUMBER OF YES/NO QUESTIONS NECESSARY TO DETERMINE $m$.

ANS. $K=2$, $n=10^6$, $h \geq \lceil \lg(10^6) \rceil = 20$

$\therefore$ THIS PROBLEM CANNOT BE SOLVED (IN GENERAL) WITH 19 OR FEWER QUESTIONS.

NOTE: THIS DOES NOT SHOW THAT 20 QUESTIONS WILL SUFFICE, ONLY THAT 19 WILL NOT.

EXERCISE

SHOW THAT BINARY SEARCH (WITH TARGET = m) SOLVES THIS PROBLEM IN NO MORE THAN 20 QUESTIONS (i.e. COMPARISONS.)

EX.

SAME PROBLEM, BUT NOW WE ARE ALLOWED TO ASK QUESTIONS WITH 3 ALTERNATIVES.

ANS. $k=3$, $n=10^6$, $h \geq \lceil \log_3(10^6) \rceil = 13$.

THUS ANY ALGORITHM WHICH SOLVES THIS PROBLEM MUST ASK AT LEAST 13 QUESTIONS IN WORST CASE.

AGAIN, THIS DOES NOT SHOW THAT 13 QUESTIONS WILL SUFFICE, ONLY THAT 12 WILL NOT.

THEOREM

ANY COMPARISON BASED SORTING ALGORITHM MUST DO, IN WORST CASE, AT LEAST $\lceil \lg(n!) \rceil$ COMPARISONS ON INPUT ARRAYS OF LENGTH $n$.

PROOF:

THE DECISION TREE CORRESPONDING TO ANY SUCH ALGORITHM HAS $K=2$ OUTCOMES FOR EACH COMPARISON AND $n!$ POSSIBLE VERDICTS (SINCE EACH OUTPUT IS A PERMUTATION OF THE INPUT ARRAY.) THE HEIGHT OF SUCH A TREE MUST SATISFY

$$h \geq \lceil \lg n! \rceil .$$

THEREFORE, ANY ALGORITHM WHICH SOLVES THIS PROBLEM MUST DO AT LEAST $\lceil \lg n! \rceil$ COMPARISONS, IN WORST CASE.

///

COROLLARY

ANY SUCH ALGORITHM RUNS IN (WORST CASE) TIME $\Omega(n \lg n)$.

PROOF:

BY STIRLING'S FORMULA $\lceil \lg n! \rceil = \Omega(n \lg n)$.

///

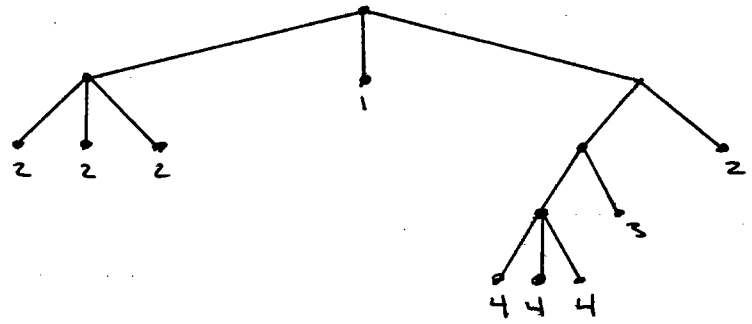
DEFN

THE AVERAGE HEIGHT OF A K-ARY TREE IS THE AVERAGE DEPTH OF EACH OF ITS LEAVES.

i.e. IF $T$ HAS $n$ LEAVES AT DEPTHS $d_1, \ldots, d_n$, THEN THE AVERAGE HEIGHT OF $T$ IS

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

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




$h = 4$

$a = 2.67$

$$a = \frac{2+2+2+1+2+3+4+4+4}{9} = \frac{24}{9} = \frac{8}{3} = 2.67$$

JUST AS THE HEIGHT OF A DECISION TREE GIVES THE WORST CASE NUMBER OF TESTS PERFORMED BY AN ALGORITHM, THE AVERAGE HEIGHT GIVES THE AVERAGE NUMBER OF TESTS PERFORMED, ASSUMING THAT EACH VERDICT (i.e. LEAF) IS EQUALLY LIKELY.

THEOREM

THE AVERAGE HEIGHT $a$ OF A $K$-ARY TREE HAVING $n$ LEAVES SATISFIES

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

COROLLARY

THE AVERAGE HEIGHT $a$ OF A BINARY TREE HAVING $n$ LEAVES SATISFIES

$$a \geq \lg(n).$$

SEE P. 416 OF BRASSARD & BRATLEY FOR PROOF OF COROLLARY ($K=2$).

THEOREM.

ANY COMPARISON SORT MUST, IN AVERAGE CASE, AT LEAST $\lg(n!)$ COMPARISONS ON INPUT ARRAYS OF LENGTH $n$.

COROLLARY

ANY COMPARISON BASED SORTING ALGORITHM RUNS IN (AVERAGE CASE) TIME $\Omega(n \lg n)$.

DECISION TREE ARGUMENTS FOR LOWER BOUNDS ARE VERY EASY TO USE:

(1) DETERMINE THE MAXIMUM NUMBER OF OUTCOMES TO EACH TEST OF THE INPUT DATA, CALL THAT NUMBER K.

(2) DETERMINE THE NUMBER OF VERDICTS $f(n)$ (i.e. POSSIBLE ALGORITHM OUTPUTS) AS A FUNCTION OF THE INPUT SIZE n.
e.g. SEARCHING: $f(n) = n$
SORTING: $f(n) = n!$

(3) CONCLUDE THAT ANY ALGORITHM WHICH SOLVES THE PROBLEM USING ONLY TESTS OF THE KIND ANALYSED IN (1) MUST DO AT LEAST

$$\text{WORST CASE: } \lceil \log_K f(n) \rceil$$
$$\text{AVERAGE CASE: } \log_K f(n)$$

TESTS ON INPUT OF SIZE n.

NOTE: THESE ARGUMENTS DO NOT ASSERT THE EXISTENCE OF ALGORITHMS WHICH PERFORM THE NUMBER OF TESTS CITED ABOVE. INSTEAD THEY ASSERT THE NON-EXISTENCE OF ALGORITHMS WHICH DO FEWER TESTS.