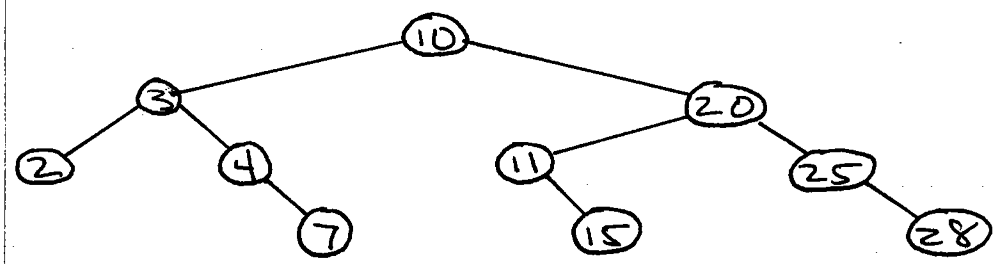
Ex.

2 3 4 7 10 11 15 20 25 28



#
1
2
3
4

AT A GLANCE WE SEE THAT AT WORST 4 COMPARISONS ARE PERFORMED, WHILE ON AVERAGE THERE ARE

$$\frac{1\cdot 1 + 2\cdot 2 + 4\cdot 3 + 3\cdot 4}{10} = 2,9$$

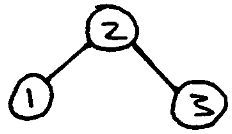
COMPARISONS. CLEARLY THERE IS AT BEST 1 COMPARISON FOR ANY LIST.

LET $W(n)$ DENOTE THE WORST CASE NUMBER OF COMPARISONS PERFORMED ON A LIST OF LENGTH n. THE PREVIOUS EXAMPLE SHOWS THAT $W(10) = 4$.

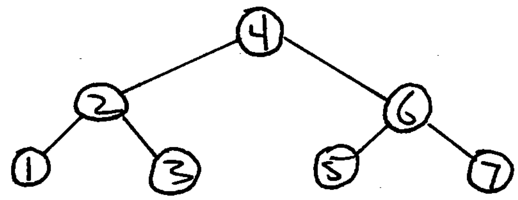
WE WILL FIND A GENERAL FORMULA FOR $W(n)$. NOTE HOWEVER THAT $W(n)$ DEPENDS ONLY ON n, AND NOT ON THE PARTICULAR NUMBERS IN THE LIST. WE THEREFORE ALWAYS TAKE THE LIST TO BE $1, 2, 3, \ldots, n$.

OBSERVE

$n=3$: 1 2 3 $\qquad W(3)=2$

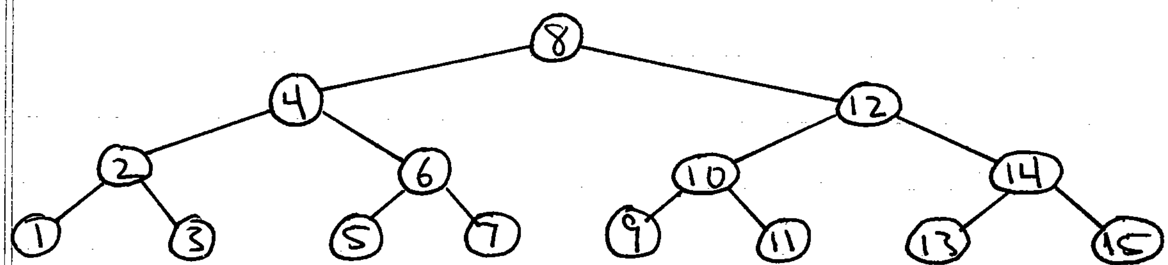


$n=7$: 1 2 3 4 5 6 7 $\qquad W(7)=3$



$n=15$: $W(15)=4$

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



OBSERVE THAT IF $n = 2^K - 1$ THEN THE CORRESPONDING TREE IS COMPLETE (i.e. EACH NODE HAS DEGREE 0 OR 2), AND HAS DEPTH K, WHENCE $W(n) = K$.

THEOREM

IF $2^{K-1} - 1 < n \le 2^K - 1$, THEN THE NUMBER OF COMPARISONS PERFORMED BY BINARY SEARCH ON ANY LIST OF LENGTH n IS AT WORST:

$$W(n) = K$$

WE WOULD LIKE TO EXPRESS $W(n)$ AS A FORMULA INVOLVING n ONLY, TO THIS END WE INTRODUCE <u>LOGARITHMS</u>

<u>DEFN:</u>

LET $b>1$, $x>0$; THEN $\log_b(x)$ IS DEFINED TO BE THE POWER YOU MUST RAISE b TO, TO GET x. i.e.

$$y = \log_b(x) \iff x = b^y$$

<u>EX.</u>

$\log_3 9 = 2$	SINCE	$3^2 = 9$
$\log_5 125 = 3$	"	$5^3 = 125$
$\log_{10}(10000) = 4$	"	$10^4 = 10000$
$\log_2(32) = 5$	"	$2^5 = 32$

THE NUMBER b IS CALLED THE <u>BASE</u>. CERTAIN BASES HAVE A SPECIAL NOTATION.

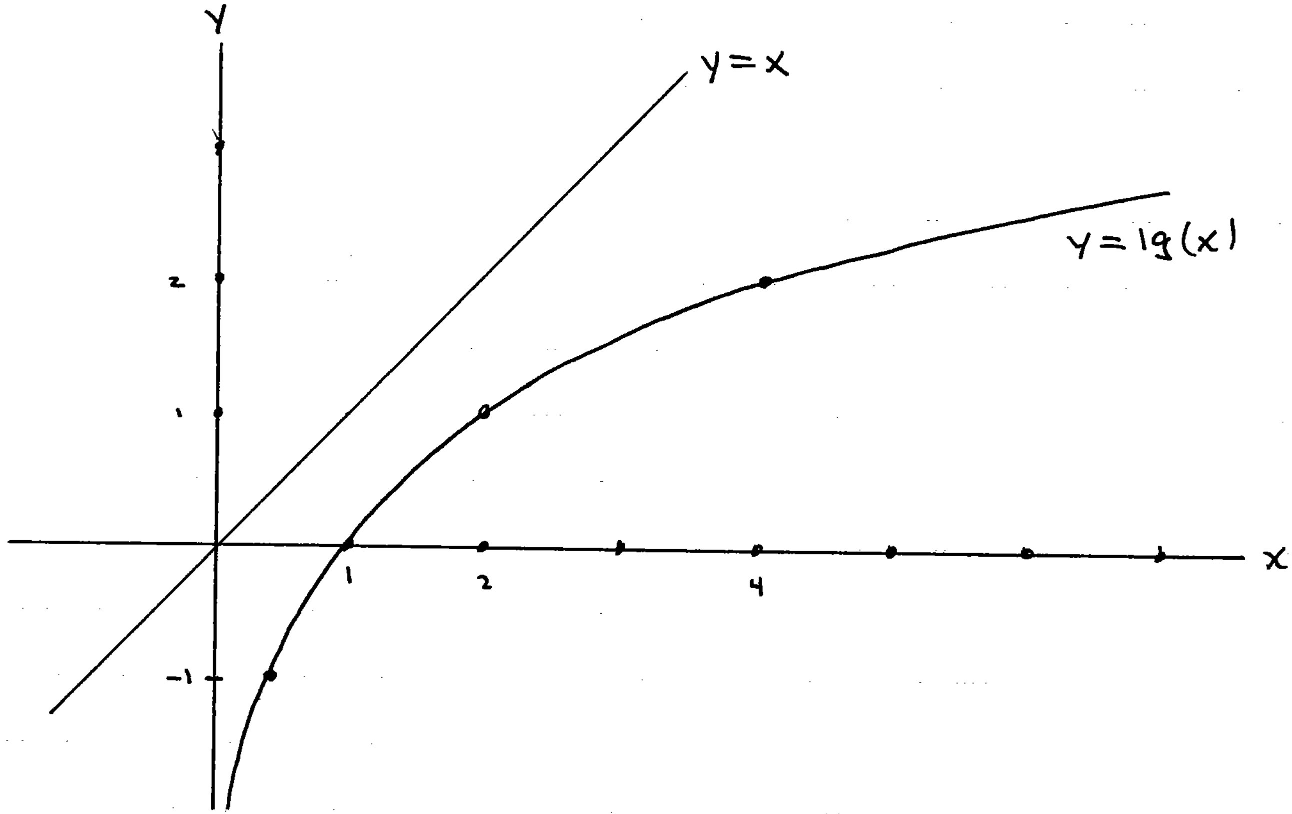
<u>COMMON log</u> OR <u>NAPERIAN log</u> : $\log_{10}$ (WRITE $\log$)

<u>NATURAL log</u> : $\log_e$ (WRITE $\ln$) WHERE $e = 2.71828\ldots$

<u>BINARY log</u> : $\log_2$ (WRITE $\lg$)

$\lg = \log_2$ IS MOST OFTEN USED IN COMPUTER SCIENCE.

Ex. $\lg(1)=0$, $\lg(2)=1$, $\lg(4)=2$, .., $\lg(2^K)=K$



OBSERVE THAT $\lg(n) < n$ FOR ALL $n > 0$.
$\lg(n)$ GROWS VERY SLOWLY WITH n, AND ANY ALGORITHM WHOSE RUNNING TIME IS $\Theta(\lg(n))$ IS CONSIDERED VERY EFFICIENT.

RECALL THE PREVIOUS THEOREM:

$$2^{K-1}-1 < n \leq 2^K - 1 \iff W(n) = K.$$

TO FIND K AS A FUNCTION OF n WE MANIPULATE THE INEQUALITY

$$2^{K-1} < n+1 \leq 2^K,$$

THEN APPLY lg TO EACH TERM:

$$\lg(2^{k-1}) < \lg(n+1) \leq \lg(2^k)$$

$$\therefore \quad k-1 < \lg(n+1) \leq k$$

THUS $k = \lceil \lg(n+1) \rceil$. AN EASY EXERCISE SHOWS THAT

$$\lceil \lg(n+1) \rceil = \lfloor \lg(n) \rfloor + 1 .$$

THUS THE WORST CASE NUMBER OF COMPARISONS FOR BINARY SEARCH IS

$$W(n) = \lfloor \lg(n) \rfloor + 1 .$$

A MOMENT'S THOUGHT SHOULD CONVINCE YOU THAT $\lfloor \lg(n) \rfloor + 1 = \Theta(\lg(n))$

THUS BINARY SEARCH HAS RUNNING TIME $\Theta(\lg(n))$, WHICH IS SUPERIOR TO THE TIME $\Theta(n)$ FOR SEQUENTIAL SEARCH.

<u>EXERCISE</u>

LET $n = 2^k - 1$. FIND THE AVERAGE NUMBER OF COMPARISONS DONE BY BINARY SEARCH UNDER THE ASSUMPTION THAT TARGET IS IN THE LIST.

ANSWER:

$$A(n) = \frac{\sum_{i=1}^{k} i \cdot 2^{i-1}}{n} = \lg(n+1) - 1 + \frac{\lg(n+1)}{n}$$

$\uparrow$ EASY $\qquad$ $\uparrow$ HARD

THUS THE AVERAGE RUNNING TIME FOR BINARY SEARCH IS ALSO $\Theta(\lg(n))$.

SUPPOSE WE HAVE AN UNSORTED LIST WE WISH TO SEARCH. WE COULD USE SEQUENTIAL SEARCH IN $\Theta(n)$ TIME, OR WE COULD DO SELECTION SORT FOLLOWED BY BINARY SEARCH IN

$$\Theta(n^2 + \lg(n)) = \Theta(n^2)$$

TIME, WHICH IS APPARENTLY WORSE.

BUT OBSERVE THAT FOR MANY APPLICATIONS WE ONLY SORT ONCE, WHILE WE SEARCH MANY TIMES.

WHICH COMBINATION WE PREFER DEPENDS ON HOW MANY TIMES WE INTEND TO SEARCH.

READ THE BOOK'S ANALYSIS OF THE PATTERN MATCHING ALGORITHM ("FORWARD MARCH").

HW3, CHAPTER 3 P. 109!

1, 2, 4, 5, 9bc, 10, 12, 14, 15, 18, 22, 27

HW3

HW4 & ? MORE

AN ALGORITHM IS SAID TO BE OF <u>EXPONENTIAL</u> ORDER IF ITS RUNNING TIME IS $\Theta(2^n)$ (OR b^n FOR ANY $b>1$). (OR WORSE)

<u>EX.</u> BRUTE FORCE CHESS, TRAVELING SALESMAN, HAMILTONIAN CIRCUITS (SEE BOOK.)

EXPONENTIAL FUNCTIONS LIKE 2^n GROW FASTER THAN ANY POLYNOMIAL n^k, WHICH IN TURN GROW FASTER THAN $lg(n)$.

EXPONENTIAL ALGORITHMS ARE PRACTICAL ONLY FOR VERY SMALL VALUES OF n.