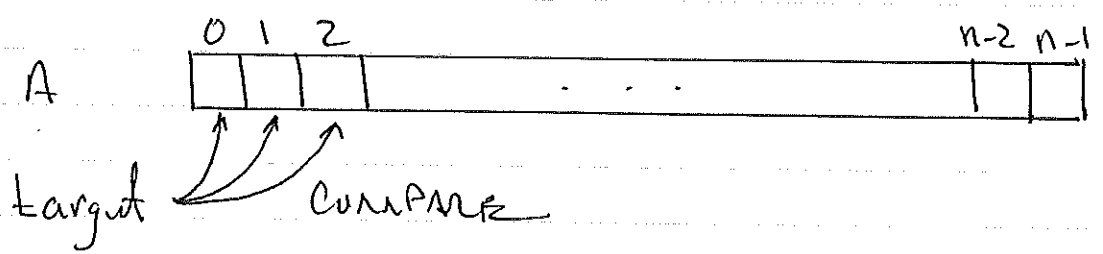


CHAPTER 10 EFFICIENCY OF ALGORITHMS

PROBLEM: SEARCH AN INT ARRAY FOR A SPECIFIED TARGET t , RETURN ITS INDEX IF FOUND, -1 IF NOT FOUND.

EX. SEQUENTIAL SEARCH



In C this looks like:

```

int SequentialSearch (int *A, int n, int t) {
  int i=0, found=0;
  while (i < n && !found) {
    if (A[i] == t)
      found = 1;
    else
      i++;
  }
  if (!found) i = -1;
  return i;
}

```

IT'S POSSIBLE TO WRITE THIS MORE SUCCINCTLY. (EXERCISE).

To analyse the efficiency of an algorithm we choose a representative basic operation (also known as operation) and count it in best, worst, and average cases.

For the searching problem the basic operation is comparison of t to an array element.

$$\begin{aligned} \text{BEST CASE \# COMP} &= 1 \\ \text{WORST CASE \# COMP} &= n \end{aligned}$$

For average case assume t is in list, and equally likely to be in any position.

$$\text{AVG. CASE \# COMP} = \frac{1+2+\dots+n}{n} = \frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2}$$

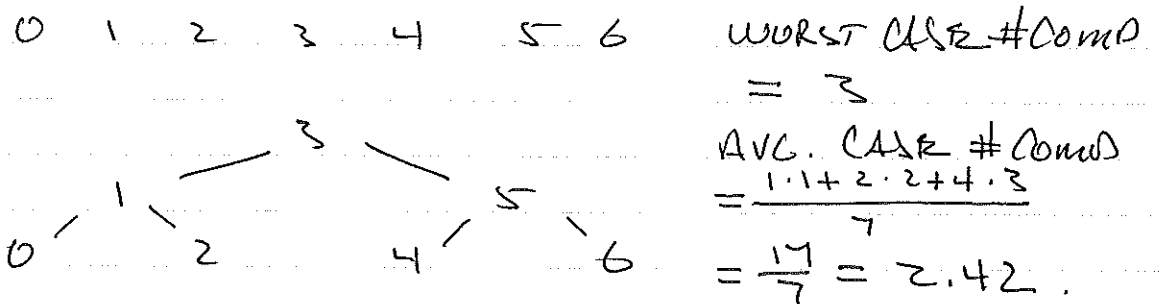
Ex Binary Search

Same problem except now we require that A be sorted. For the sake of simplicity assume $A[i] = i$ ($0 \leq i \leq n-1$) i.e.

	0	1	2						$n-2$	$n-1$
A	0	1	2						$n-2$	$n-1$

THE OPERATION OF BINARY SEARCH CAN BE
REPICTED BY A BINARY SEARCH TREE

e.g. $n=7$



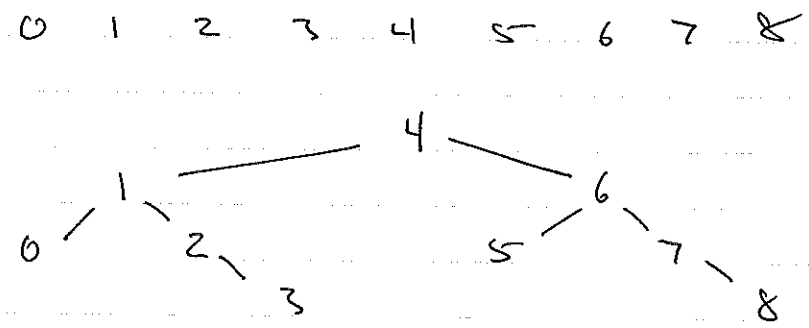
IN C WE HAVE

```

int BinarySearch (int *A, int n, int t) {
    int i = 0, j = n - 1, found = 0, m;
    while (i <= j && !found) {
        m = (i + j) / 2;
        if (t == A[m])
            found = 1;
        else if (t < A[m])
            j = m - 1;
        else
            i = m + 1;
    }
    if (!found) m = -1;
    return m;
}

```

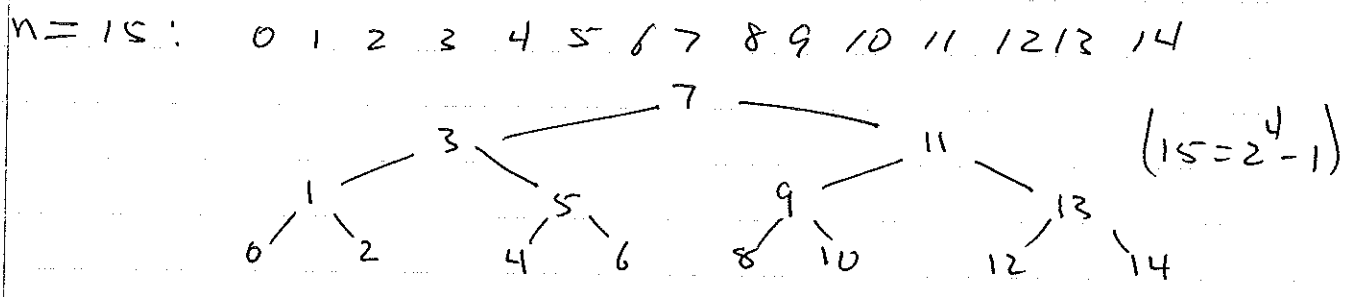
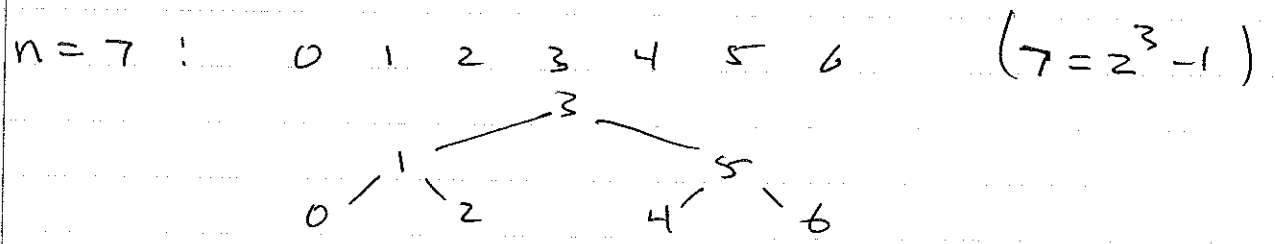
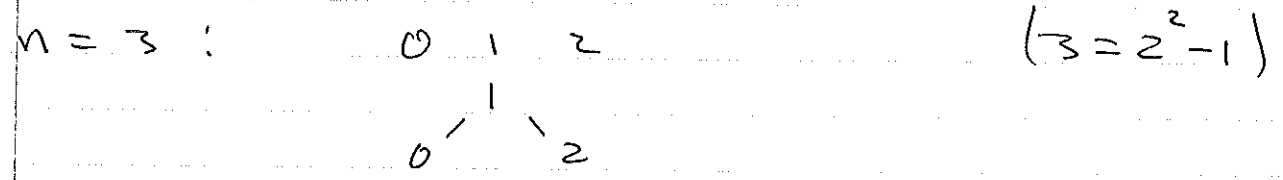
e.g. $n = 9$



worst case # comp = 4

$$\text{Avg. case \# comp} = \frac{1 \cdot 1 + 2 \cdot 2 + 4 \cdot 3 + 2 \cdot 4}{9} = \frac{25}{9} = 2.77$$

Notice there are certain magic numbers in these binary search trees:



LET $w(n)$ DENOTE THE WORST CASE NUMBER OF COMPARISONS PERFORMED BY BINARY SEARCH ON ARRAYS OF LENGTH n .

WE SEE THAT IF n SATISFIES

$$2^{k-1} - 1 < n \leq 2^k - 1$$

FOR SOME INTEGER k , THEN $w(n) = k$.

THE ABOVE INEQUALITY IS EQUIVALENT TO

$$2^{k-1} \leq n < 2^k$$

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

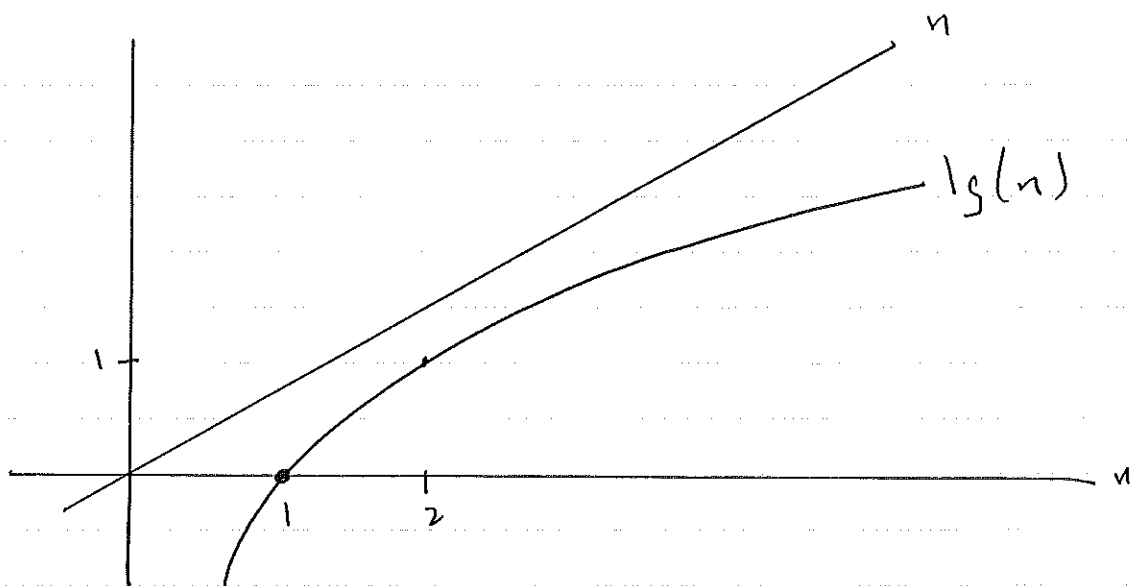
$$\therefore k-1 = \lfloor \lg(n) \rfloor$$

$$\therefore k = \lfloor \lg(n) \rfloor + 1$$

$$\therefore w(n) = \lfloor \lg(n) \rfloor + 1$$

FOR LARGE n , $w(n) \approx \lg(n)$. COMPARING BINARY SEARCH TO SEQUENTIAL SEARCH WHICH IS BETTER?

i.e. WHICH FUNCTION GROWS FASTER n OR $\lg n$?



ANSWER n GROW MUCH FASTER THAN $\lg n$. IN FACT

$$\lim_{n \rightarrow \infty} \frac{\lg(n)}{n} = 0$$

THIS MEANS THAT THE RATIO $\frac{\lg(n)}{n}$ CAN BE MADE ARBITRARILY SMALL BY TAKING n SUFFICIENTLY LARGE,

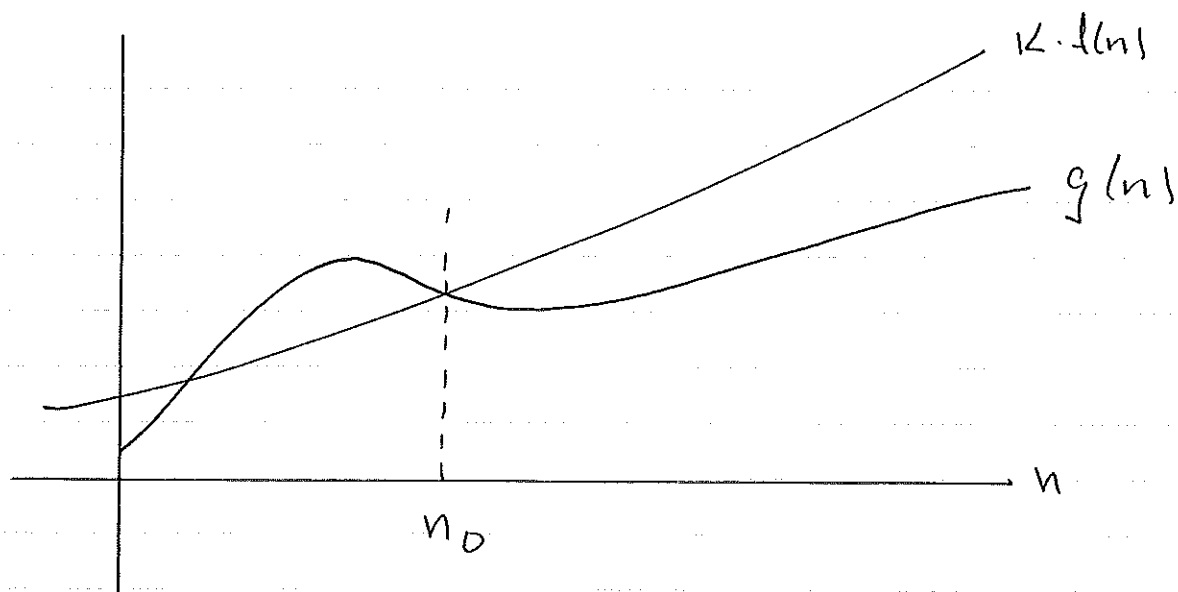
DEFN.

A FUNCTION $g(n)$ IS SAID TO BE BIG O OF A FUNCTION $f(n)$, I.E. IN THE CLASS $O(f(n))$ IFF THERE EXIST POSITIVE K, n_0 SUCH THAT FOR ALL $n \geq n_0$:

$$0 \leq g(n) \leq K \cdot f(n)$$

WE WRITE, WITH SOME ABUSE OF NOTATION,

$$g(n) = O(f(n))$$



Ex. $n^2 + 1 = O(n^3)$ since for all $n \geq 1$, $n^2 + 1 \leq 2n^3$. i.e. $n_0 = 1$, $k = 2$.

OBSERVE THAT IF ONE PAIR n_0, k EXISTS MAKING THE REQUIRED INEQUALITY TRUE THEN INFINITELY MANY SUCH PAIRS EXIST.

NOTE THE ABOVE EXAMPLE GENERALIZED TO ANY POLYNOMIALS $P(n), Q(n)$ WITH $\deg(P) \leq \deg(Q)$; IN SUCH A CASE

$$P(n) = O(Q(n)).$$

e.g. $2n^3 + 4n - 5 = O(n^4)$

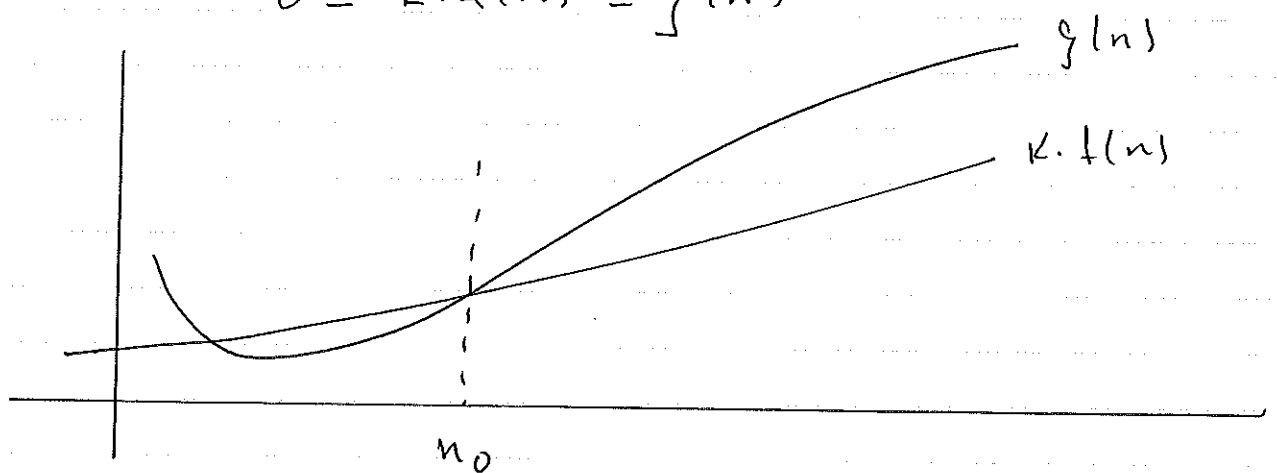
DEFN:

$g(n)$ is in the class $\Omega(f(n))$ IFF $f(n)$ is in $O(g(n))$. i.e.

$$g(n) = \Omega(f(n)) \text{ IFF } f(n) = O(g(n)).$$

i.e. THERE EXIST POSITIVE k, n_0 SUCH THAT FOR ALL $n \geq n_0$:

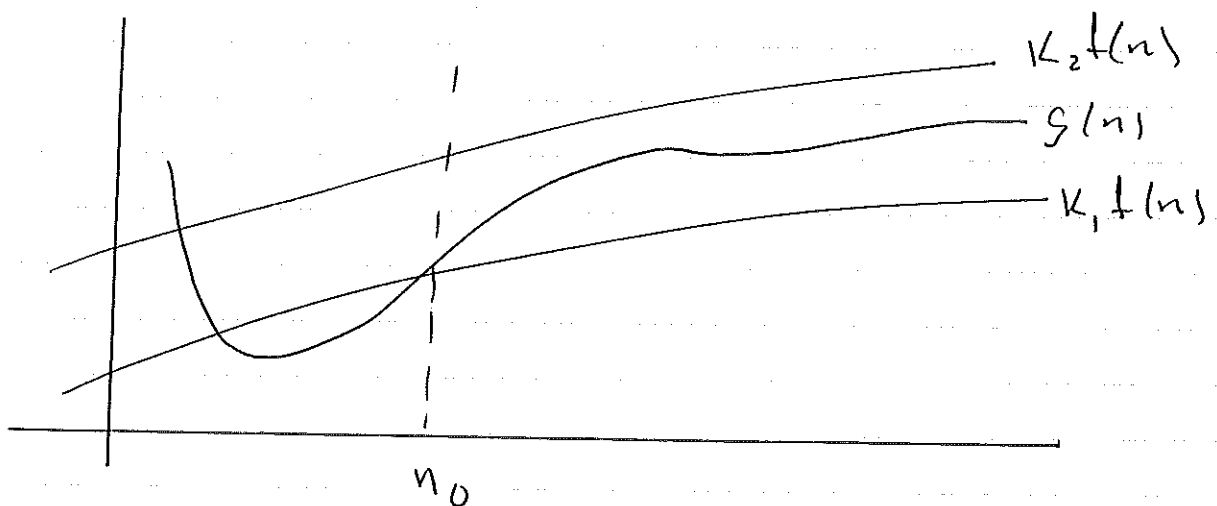
$$0 \leq k \cdot f(n) \leq g(n)$$



DEFN: $\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$

i.e. $g(n) = \Theta(f(n))$ IFF THERE EXIST POSITIVE n_0, k_1, k_2 SUCH THAT FOR ALL $n \geq n_0$:

$$0 \leq k_1 f(n) \leq g(n) \leq k_2 f(n).$$



THEOREM

IF $P(n)$ AND $Q(n)$ ARE POLYNOMIALS, THEN

$$P(n) = \Theta(Q(n)) \text{ IFF } \deg(P(n)) = \deg(Q(n))$$

e.g. $6n^5 + 2n^4 + 4n^3 + 17n^2 - 12n + 1 = \Theta(n^5)$

THEOREM

IF $g(n) = \text{CONST} \cdot f(n)$ THEN $g(n) = \Theta(f(n))$

e.g. $10n \lg(n) = \Theta(n \lg n)$

THESE AND MANY OTHER THEOREMS CONCERNING ASYMPTOTIC GROWTH OF FUNCTIONS WILL BE PROVED RIGOROUSLY IN CMPS 101.

INFORMALLY, TO CLASSIFY A FUNCTION BY ASYMPTOTIC GROWTH RATE MEANS

- IDENTIFY THE HIGHEST ORDER (I.E. FASTEST GROWING) TERM(S) AND DISCARD ANY LOWER ORDER TERMS.
- DISCARD THE COEFFICIENT OF THE HIGHEST ORDER TERM, I.E. REPLACE IT BY 1.

e.g. $12n^2 + 4n \lg n + 5n - 3 = \Theta(n^2)$