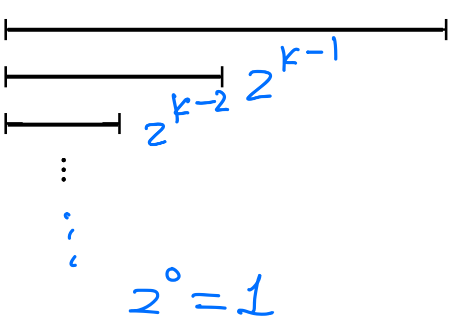
MAT2440, Classwork35, Spring2025

ID:________________________ Name:________________________

1. The time complexity analysis of the **binary search** algorithm

procedure *binary_search*(x: integer, $a_1, a_2, \cdots, a_n$: distinct integers)
$n :=$ the length of $\{a_i\}$
$i := 1$ (which is left end location)
$j := n$ (which is right end location)
    while ($i \le j$) ← 1 comparison
        m:= $\left\lfloor \frac{i+j}{2} \right\rfloor$ ← 2 operations (add, div.)
        if $x > a_m$ **then** $i := m + 1$ ← 2 operations (1 comp. 1 add)
            else $j := m$
    if $x = a_i$ **then** *location* $:= i$
        else *location* $:= 0$
return *location* {either the subscript of the term that equals x, or 0 if x is not found.}

For simplicity assuming $n =$ 2^k.



Inside the loop (while)		
# of elements	# of operations	
2^k	5	
2^{k-1}	5	
2^{k-2}	5	k times
⋮	⋮	
2^1	5	
2^0=1	stop	
(if) Outside the loop	1	

$n = 2^k \Rightarrow k = \log_2(n)$

Therefore, the time complexity is

$f(n) =$ $5k+1$ $=$ $5 \cdot \log_2(n) + 1$ $\sim$ $5 \cdot \log_2(n)$ $=$ $5 \cdot \frac{\log(n)}{\log(2)}$ $= O($ $\log(n)$ $)$

↑ change base property

which is logarithmic complexity.

2. Binary search ($O(\log n)$) is more efficient than linear search ($O(n)$).

3. The worse-case time complexity analysis of the **sorting** algorithm: **Bubble Sort**.

> **procedure** *bubblesort*($a_1, a_2, \cdots, a_n$: real numbers with $n \geq 2$)
> $n :=$ the length of $\{a_i\}$
> **for** $i := 1$ **to** $n - 1$
>   **for** $j := 1$ **to** $n - i$
>   **if** $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}
> $\{a_1, a_2, \cdots, a_n$ is in increasing order$\}$

$i = 1$, $j = 1$ to $n-1$ ← $n-1$ operations (comparisons)
$i = 2$, $j = 1$ to $n-2$ ← $n-2$ operations
⋮
$i = n-1$ $j = 1$ to 1 ← 1 operations

Total operations:
$(n-1)+(n-2)+(n-3)+\cdots+2+1$
$= \frac{(1+(n-1))(n-1)}{2} = \frac{n(n-1)}{2}$

The time complexity is $f(n) = \frac{n^2-n}{2} = O(n^2)$ which is polynomial complexity.

4. The worse-case time complexity analysis of the **sorting** algorithm: **Insertion Sort**.

> **procedure** *insertionsort*($a_1, a_2, \cdots, a_n$: real numbers with $n \geq 2$)
> $n :=$ the length of $\{a_i\}$
> **for** $i := 2$ **to** n
> $j := 1$
>   **while** ($a_i > a_j$ and $i > j$)
> $j := j + 1$
> $m := a_i$
>   **for** $k := 0$ **to** $i - j - 1$
> $a_{i-k} := a_{i-k-1}$
> $a_j := m$
> $\{a_1, a_2, \cdots, a_n$ is in increasing order$\}$

$i = 2$,	$a_2 > a_1, 2 > 1, j = 2$	← 3 operations
$i = 3$,	$a_3 > a_1, 3 > 1, j = 2$	← 3×2 operations
	$a_3 > a_2, 3 > 2, j = 3$	
$i = 4$,	$a_4 > a_1, 4 > 1, j = 2$	← 3×3 operations
⋮	$a_4 > a_2, 4 > 2, j = 3$	
	$a_4 > a_3, 4 > 3, j = 4$	
$i = n$,	$a_n > a_1, n > 1, j = 2$	← $3 \times (n-1)$ operations
	$a_n > a_2, n > 2, j = 3$	

$a_n > a_3$, $n > 3$, $j = 4$
$a_n > a_{n-1}$

Total operations:
$3 + 3\times 2 + 3\times 3 + 3\times 4 + \cdots + 3\times(n-1)$
$= 3(1+2+3+4+\cdots+(n-1))$
$= 3 \cdot \frac{n(n-1)}{2} = 3\left(\frac{n^2-n}{2}\right)$

The time complexity is $f(n) = 3\left(\frac{n^2-n}{2}\right) = O(n^2)$ which is polynomial complexity.

Both the bubble sort and the insertion sort have the worse-case complexity $O(n^2)$