3.1-2) By O definition, there exists it contents such that

Ci. Up = (Uta) = C. Up Au> Uo

It is known that:

Norefore,

$$\frac{1}{2}$$
 n \leq n+a \leq 2n, when n \geq 2 lal

Since b > 0, the inequality also holds when all ports are raised to the set bin power:

4) By definition we have, for all n = no:

Divide by n2

$$C_4 \le 2 + \frac{3}{0} + \frac{1}{0^2} \le C_2$$

Lets arbitrarily dalls no = 03, then $2+\frac{3}{3}+\frac{1}{\alpha}=\frac{28}{9}$

Hence, choosing the $0 < c_1 \le \frac{0.8}{9}$ and that $\frac{28}{9} \le c_2 \le +\infty$ for 0 = 3.

By definition, 200+20+1 = O(00)

Out of these two, the infinior one is a tighter constraint. If it complies with the inferior constraint, it will also comply with the appear constraint

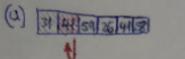


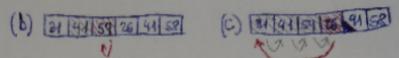
Algerithms

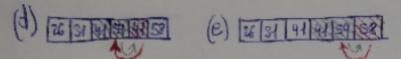
204280366

Albert Sobatta Millan Trayols

2.1) A = < 31, 41, 59, 26, 44, 58>







2-2) Insortion-Sort (Non-increasing)

for j=2 to A. length:

Key = ALIJ

C=j-1

while i > 0 and A[i] < beg

A[i]A = [i]A

A[i+1] = Key

2.3) Linear Search (A.V) white 12 (0+1) and res

teturn re

Maintenance: res will hold WID

Endidization: case where A=XX=X empty hence it returns NIS.

case where true before 4st item Because we haven't storted the ite theorning is no evidence that v hence res = NIL.

- 3.1-3) The statement user big O notation, which essentially defines the upper bound. Here's signing at least is meaningless since the Mower bound is also the upper bounds and thus no ranging time can be greater than the upper bounds.
- 3.1-1) According to the definition of O(1), there must exist positive constants
 (4, C2 and no such that

C1. (tcu)+a(u)) ≤ mox(t(u),a(u)) ≤ C2. (t(u)+a(u)), y n>n0

Since f(n) and g(n), there exist my are asymptotically non-negative, there exist an no such that $f(n) \ge 0$ and $g(n) \ge 0$ for all $n \ge n_0$. It thereof thus follows that the following stalements of true:

· f(n)+g(n) ≥ f(n) ≥ 0 } Vn≥no

Since max (for) g(n) can either be f(n) or g(n), the above statements dready proove the right-most inequality for constant $c_2 = 1$. That is:

 $\max(f(n),g(n)) \leq 1 \cdot \left[f(n) + g(n)\right] \Rightarrow \begin{cases} f(n) \leq f(n) + g(n) \\ g(n) \leq f(n) + g(n) \end{cases}$

Similarly the left-next inequality we know that:

- · 0 < f(n) < max[f(n), g(n)]
- · 0 < g(n) < max[fm), g(n)] +

 $0 \le f(n) + g(n) \le 2 \cdot \max[f(n), g(n)] \Rightarrow \frac{1}{2} \cdot [f(n) + g(n)] \le \max[f(n), g(n)]$ where $c_1 = \frac{1}{2}$

Thus, we have found values for $c_1=\frac{1}{2}$ and $c_2=1$ so that to earply with eq. 1. Hence, by definition, $\max(\text{Fon}),8(n))=\Theta(\text{finitg(n)})$

BEq

2-2-2)

for i=1 to (A-longth-1):

for j=i+1 to A. length:

if A[small] > A[j]:

small = j

if small != i: swap (AEsmall], A[i])

It only iterates up to n-1 because it is sorting in increasing orders the elements. From A Larger of When i = n-1, all the elements up to n-1 are sorted that increased order, taking the minimum value from A [n-1, n], hence, the remaining value at position n must be greater than that at n-1, and thus all that preced it.

Loop Invariants
There are toubloops so there are invariants
for each:

- Alon, i-1] is in sorted according

- All entries from AIO,..., i-II must be smaller or equal to the entries from AII, nI

· Inner loop

- All entries from Alin..., i-1] should

be larger or equal than Alismall.

talking the minimum value from Alix..., 10

Performance
Best Case: O(n2)
West Case: O(n2)

The dominant terms are the two for loops. It performs the same number of operations in both the best and watth cases as it compares element-by-element

2.2-3) On the average case, there need to be (Nz) elements checked.

It is the point in-between the apper allowed and the leaver(1) bound.

In the worst case, the method iterates

over all the elements in the array horse in times

What Case: $\Theta(n)$

This because the leading coefficient is ignored in theta (O) notation.

Ag 22

2.3) Continued

Loop Invariant: V K ∈ [4, i), A [K] ≠ V

Initialisation:

Maintenance: Lets assume true for the first int iterations, of the for loop.

At the start of the at ith iteration, if ACIJ=V, the current iteration is the final one, (see termination). Otherwise, if ACIJ \$1, we have:

 $\forall u \in [h, i)$, $A[u] \neq v$ and $A[i] \neq v \iff \forall u \in [h, i+1)$, $A[u] \neq v$, which means the huariant loop will also be true at the start of the next iteration $(i+1)^{hh}$).

Termination: the for loop may end for two reasons.

- 1. If A[i] == v, it will return the value of i. The this case, satisfies the loop invariant burially previous iterations of a such that Orden(i-1) the aire not equal to M

 A[i]==v > V M G [1,i), A[M] \pm v
- 2. If i = A.length + 1, in which case we are at the begginning of the $(A.length + 1)^{th}$ items $V \times E [1, A.length + 1)$. A $[N] \neq V + 4 \Rightarrow V \times E [1, A.length)$, $A[N] \neq V$ and NIL is the value retained. Hence, the conditions remain true.

Code

for i=1 to A. length

if A[i] == V

return i

return NIL