Selection Sort

arr [ ] = | 1 | 7 | 9 | 2 | 3 | 0 | sorted → | 0 | 1 | 2 | 3 | 7 | 9 |

Round

i. Select the SMALLEST element and place it to the right place.

Initial: arr[ ] =

0	1	2	3	4
64	25	12	22	11

Round 1:

place i = 0 (64)

element 1st smallest element (11)

Swap

11	25	12	22	64

(after Round 1)

Round 2:

place i = 1 (25)

element 2nd smallest element (12)

Swap

0	1	2	3	4
11	12	25	22	64

(after Round 2)

Round 3:

0	1	2	3	4
11	12	22	25	64

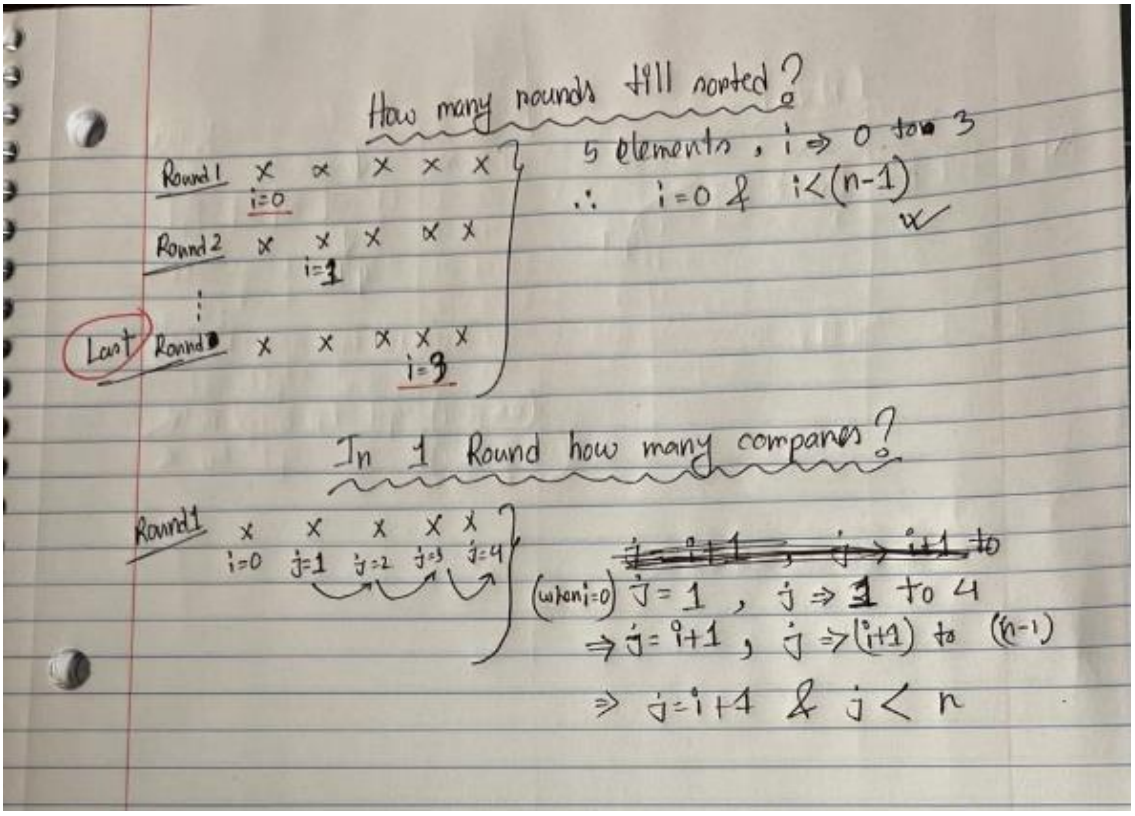
(after Round 3)

Round 4:

0	1	2	3	4
11	12	22	25	64

(last element in the perfect place)

∴ Total Round = 4
= (total element − 1)
= (n − 1)



Time Efficiency Analysis of Selection Sort

Unsorted elements	Number of comparisons required to select either the smallest or the largest	Number of swapping
n	n-1	1
n-1	n-2	1
…	…	…
3	2	1
2	1	1
1	0	0
	$n(n-1)/2$	n-1

Time Efficiency Analysis of Selection Sort

Note that comparing $n-1$ times and iterating $n-1$ time is the most amount of work selection sort does when sorting data

- In selection sort …
  1. … makes $n-1$ comparisons
  2. … performs 1 swap (i.e., 3 assignments)

  Done in sequence

  -> **max[** $O(n-1)$, $O(1)$ **]** = **max[** $O(n)$, $O(1)$ **]** = $O(n)$
- Then 1. and 2. are done in sequence $n-1$ times

  -> $O(n-1) * O(n) = O(n) * O(n) = O(n * n) = O(n^2)$

19

Time Efficiency Analysis of Selection Sort

- Is $O(n^2)$ the time efficiency of the best, worst or average c scenario?
- Would the way the data is organized affect the number o operations selection sort perform (affect its time efficiency
  - For example:
    - If the data was already sorted (in the desired sort order, e.g., ascending)?
    - If the data was sorted but in the other sort order (e.g., descending)?
    - If the data was unsorted?
    - Let's check it out! ☺ https://www.toptal.com/developers/sortina-algor

Summary – Selection Sort Algorithm

- The way the data is organized **does not** affect the number of operations selection sort perform, i.e., does not affect its time efficiency
- Time efficiency
  - Best case scenario: $O(n^2)$
  - Worst case scenario: $O(n^2)$
  - Average case scenario: $O(n^2)$
- Space efficiency
  - **in-place** sorting algorithm => $O(1)$

21