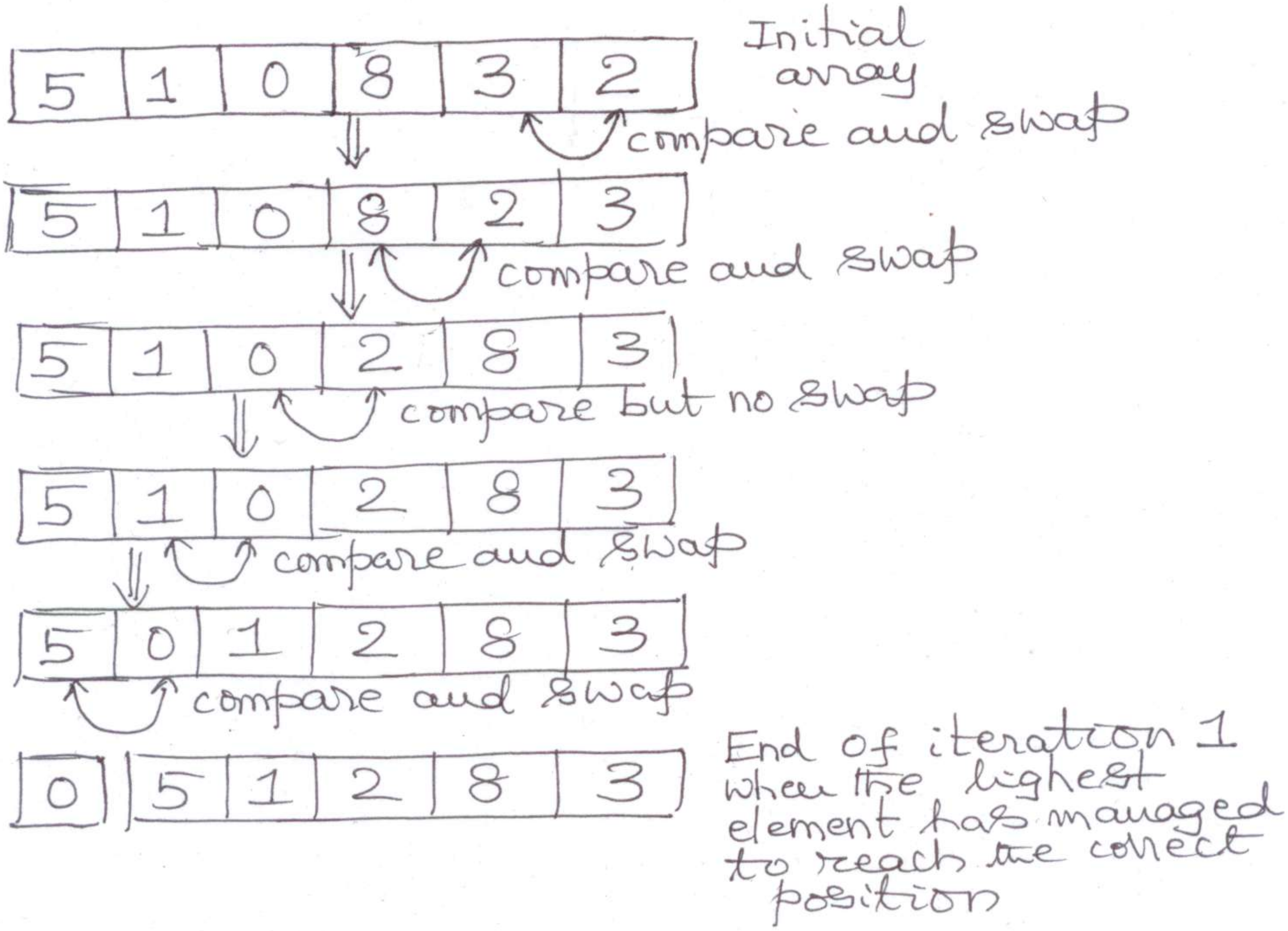
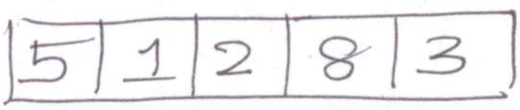
BUBBLE SORT

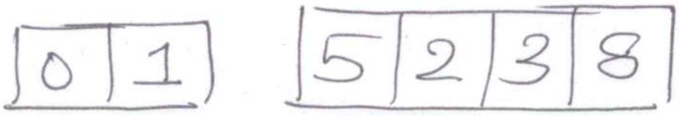
This method proceeds in (n−1) iterations. In each iteration, the lightest (smallest) element of the unsorted portion of the array "bubbles" up to its "correct" position, through a series of comparisons and exchanges. An example follows.



In iteration 2, we repeat the above procedure with the unsorted portion



to obtain



and this repeats.

Summary of Bubble Sort

State of the array at the end of Iteration "i"

A[0]	A[1]	...	A[i-1]		A[i]	...	A[n-1]

sorted as in final order — unsorted portion

⇓ Iteration (i+1)
The next lowest key from unsorted portion bubbles up and becomes A[i]

A[0]	A[1]	...	A[i-1]	A[i]		A[i+1]	...	A[n-1]

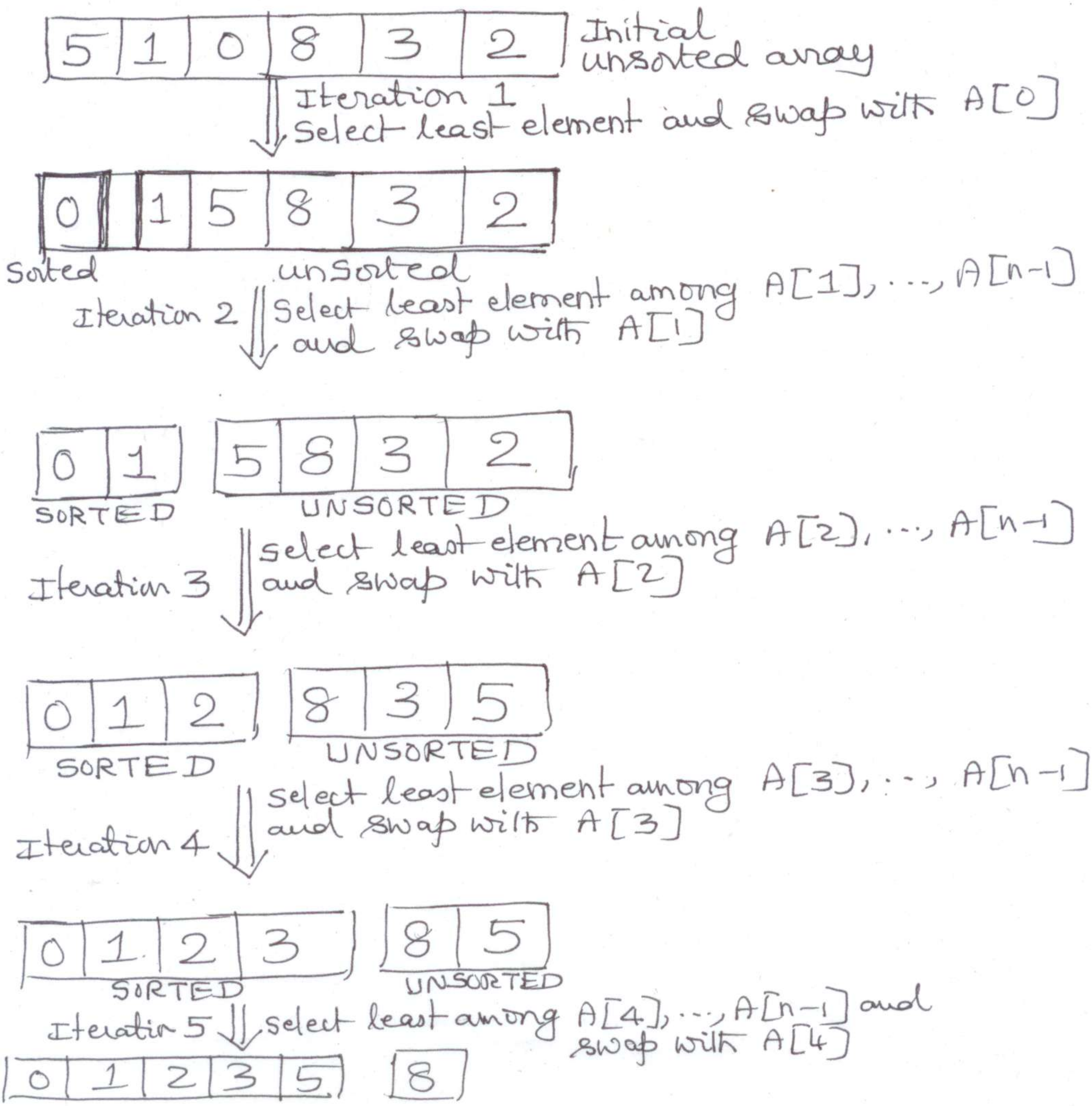
sorted as in final order — unsorted portion

- Number of Iterations = $n-1$

Number of	Minimum	Maximum
Comparisons	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Exchanges	0 (happens for sorted input array)	$\frac{n(n-1)}{2}$

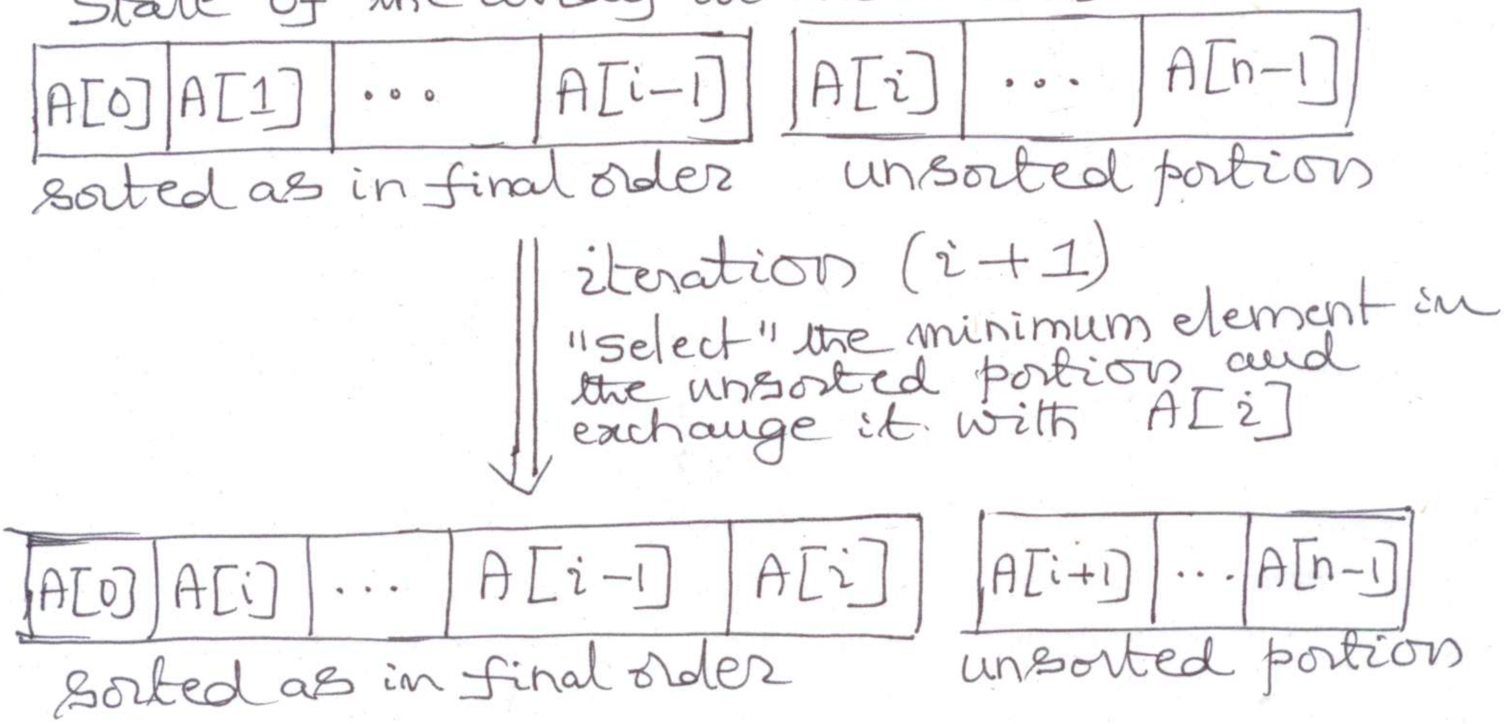
SELECTION SORT

In each iteration, the smallest element in the unsorted portion is "selected" and exchanged with the appropriate element. An example follows.



Summary of Selection Sort

State of the array at the end of iteration "i"



Number of iterations = $n-1$

Number of	Minimum	Maximum
Comparisons	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Exchanges	$(n-1)$ (can be made zero at the cost of an additional $(n-1)$ comparisons	$(n-1)$

INSERTION SORT

In this method, there are (n−1) iterations. At the end of iteration (i−1), the elements A[0], ..., A[i−1] would be in ascending order. During iteration "i", the element A[i] is inserted into its correct position among A[0], ..., A[i] through a series of comparisons and exchanges. An example:

5	1	0	8	3	2

Initial unsorted array

compare and swap

Iteration 1
Insert A[1] into its correct position among A[0], A[1]

1	5	0	8	3	2

compare and swap

Iteration 2 (step 1)
First compare and swap A[2] and A[1]

1	0	5	8	3	2

compare and swap

Iteration 2 (step 2)
compare and swap A[1] and A[0]

0	1	5	8	3	2

Array at the end of iteration 2

0 1 5: sorted, but may not be as in final order
8 3 2: unsorted

During iteration 3, A[3] would be inserted into its proper position and we obtain

0	1	5	8	3	2

and so on.

Summary of Insertion Sort

State of the array at the end of iteration "i-1"

A[0]	A[1]	. . .	A[i-1]	A[i]	. . .	A[n-1]

sorted, but not necessarily as in final order — unsorted portion

⇓ iteration (i)
Insert A[i] in the correct position among A[0], .., A[i-1] through a series of exchanges

A[0]	A[1]	. . .	A[i-1]	A[i]	A[i+1]	. . .	A[n-1]

sorted, but not necessarily as in final order — unsorted portion

Number of iterations = $n-1$

Number of	minimum	maximum
comparisons	$n-1$ (if the input array is already sorted)	$\frac{n(n-1)}{2}$
exchanges	0 (if the input array is already sorted)	$\frac{n(n-1)}{2}$