Selection Sort Algorithm

CS2002D Program Design

Selection Sort - Visualization

Which among the two takes more swaps?

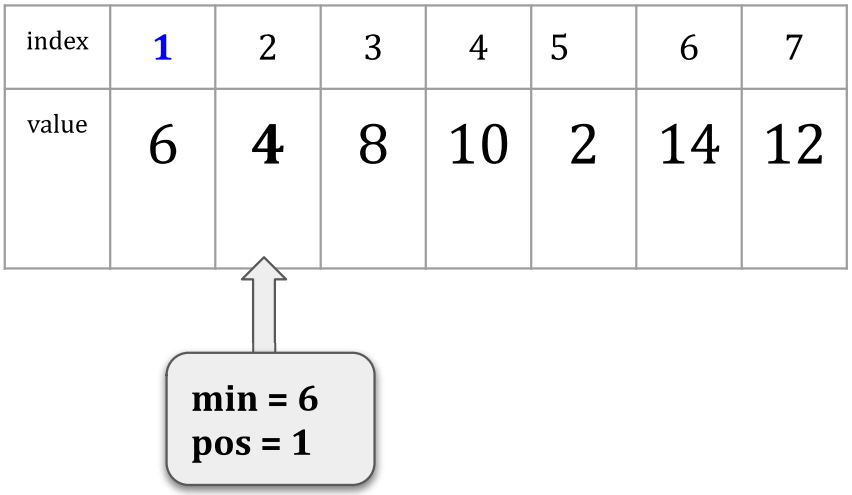
Selection Sort

- One among the simple comparison based sorting technique.
- In each iteration, the array is divided into two parts such that the first part of the array is sorted and the second part is unsorted.
- The algorithm begins with an empty first part and the entire array in the second part.
- The minimum element from the unsorted part is appended to the sorted first part in each iterations.
- At the end, the second part vanishes and the whole array constitute the first part - the sorted array.

Selection Sort

SELECTION_SORT(A)

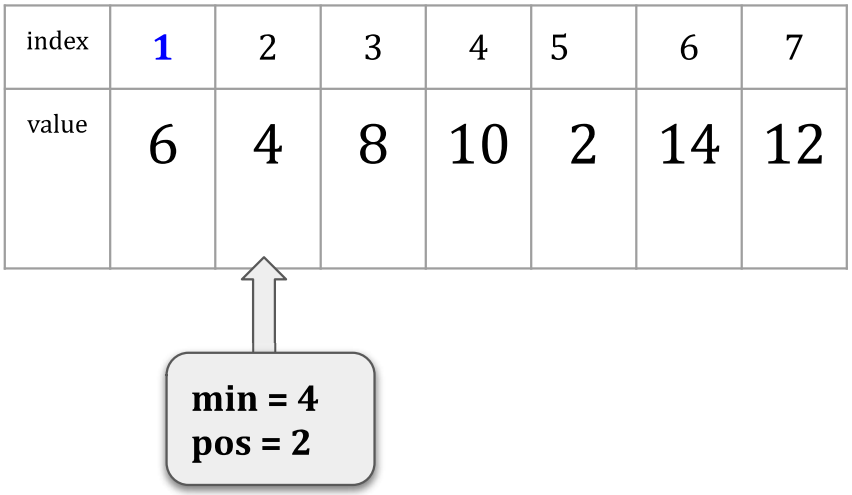
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

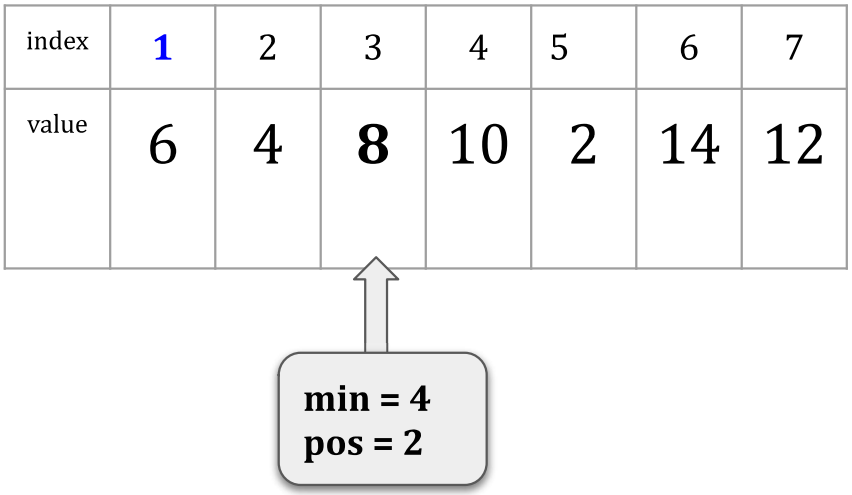
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

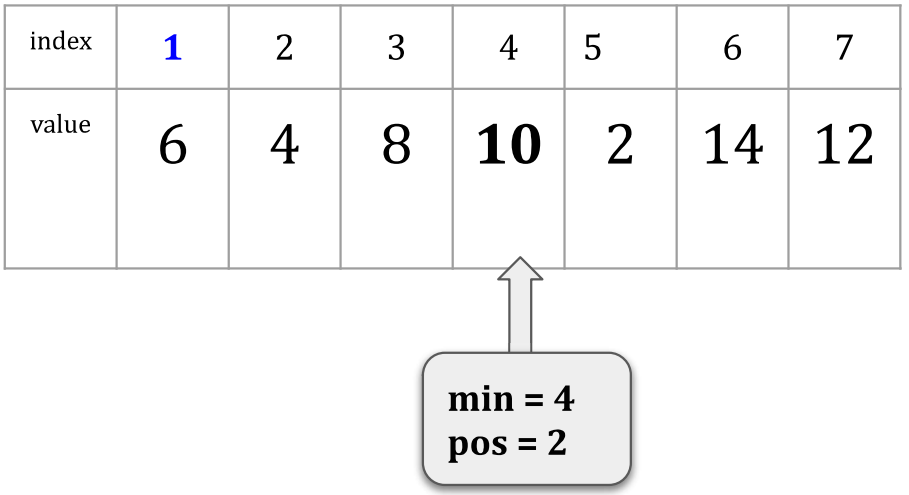
1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	6	4	8	10	**2**	14	12

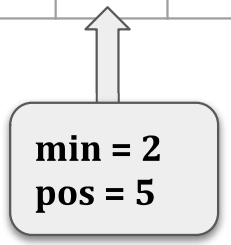
min = 2
pos = 5

Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	6	4	8	10	2	**14**	12



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

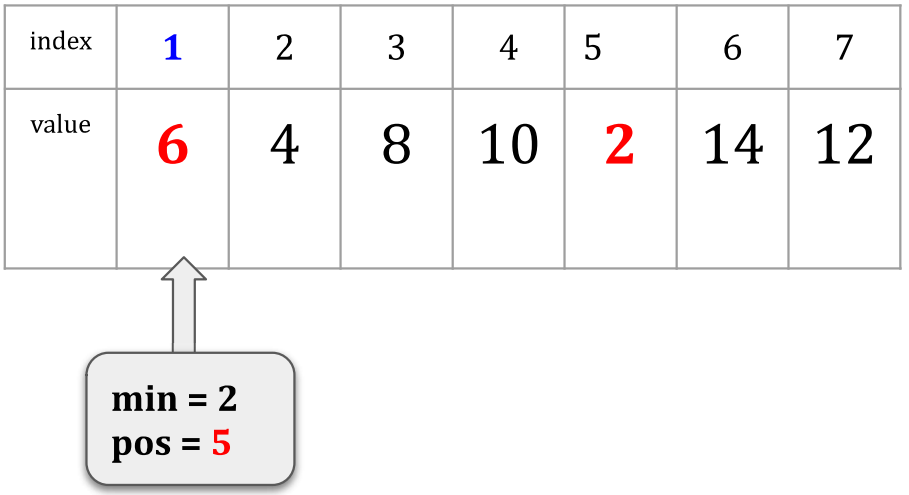
index	1	2	3	4	5	6	7
value	6	4	8	10	2	14	**12**

min = 2
pos = 5

Selection Sort

SELECTION_SORT(A)

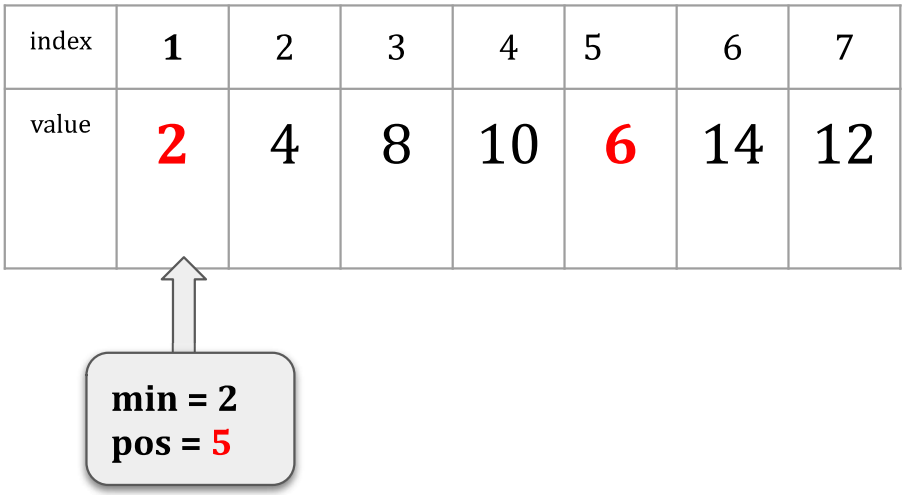
1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

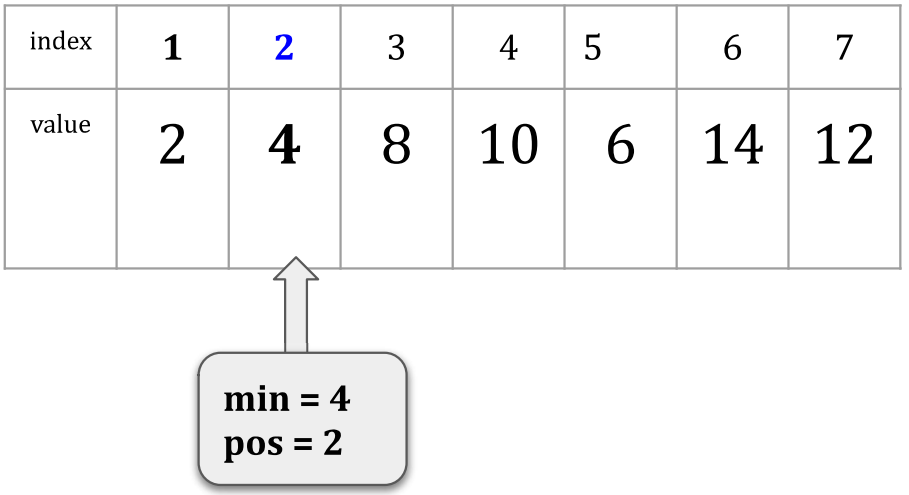
1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

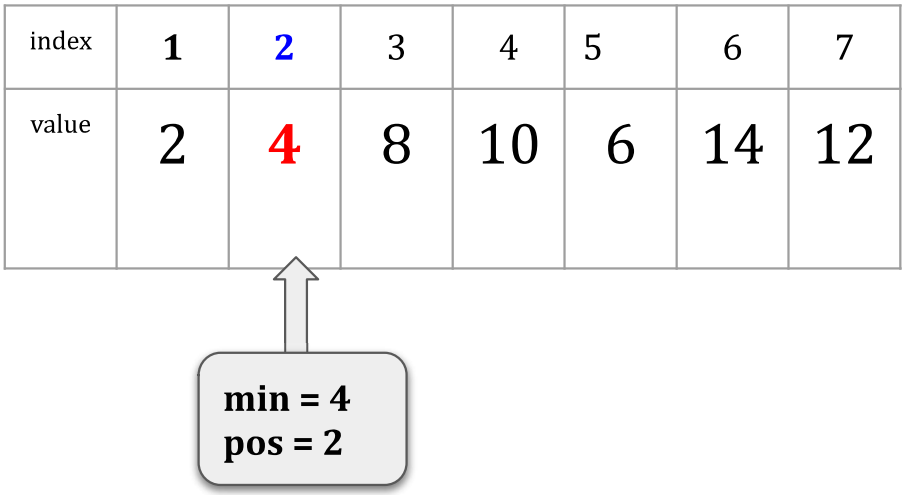
1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

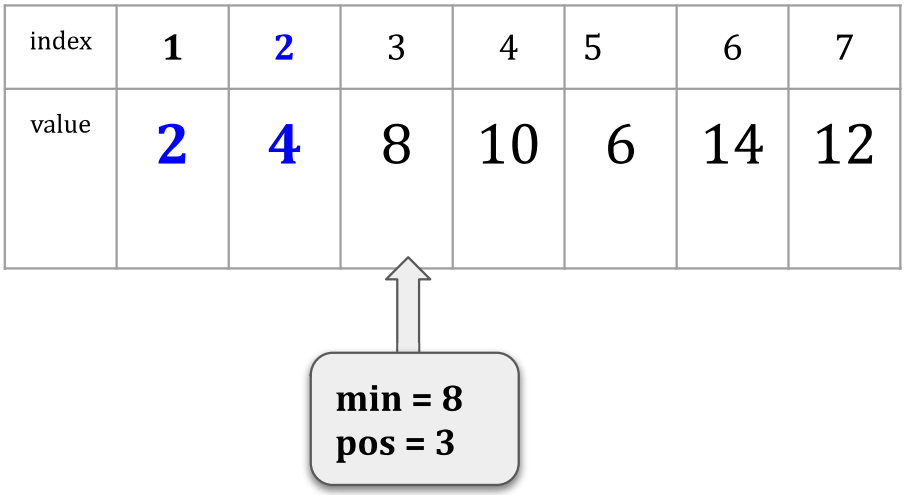
1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

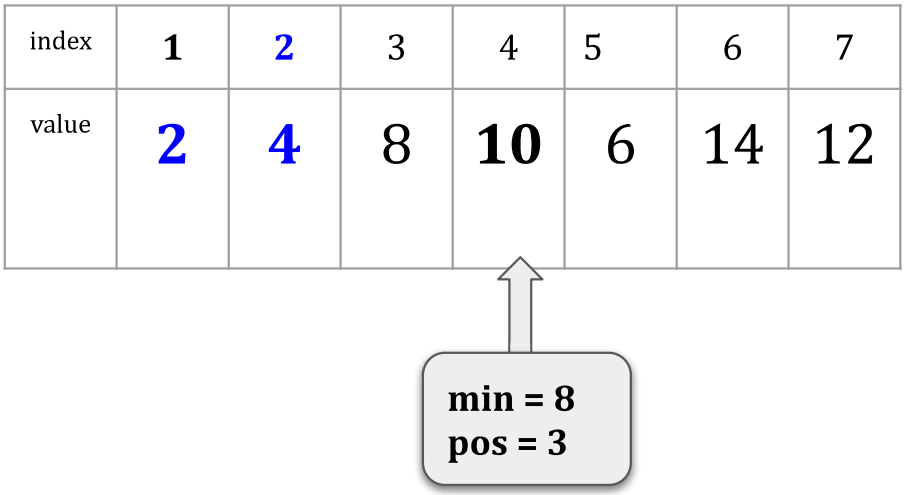
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

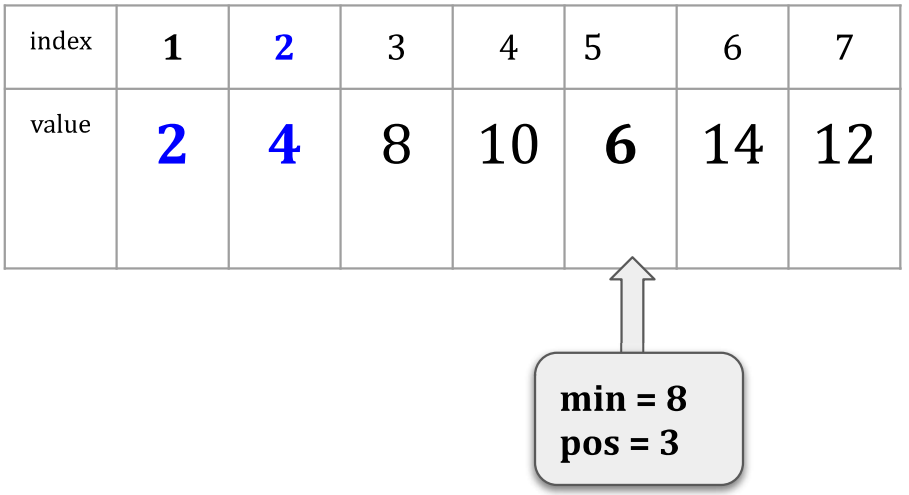
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

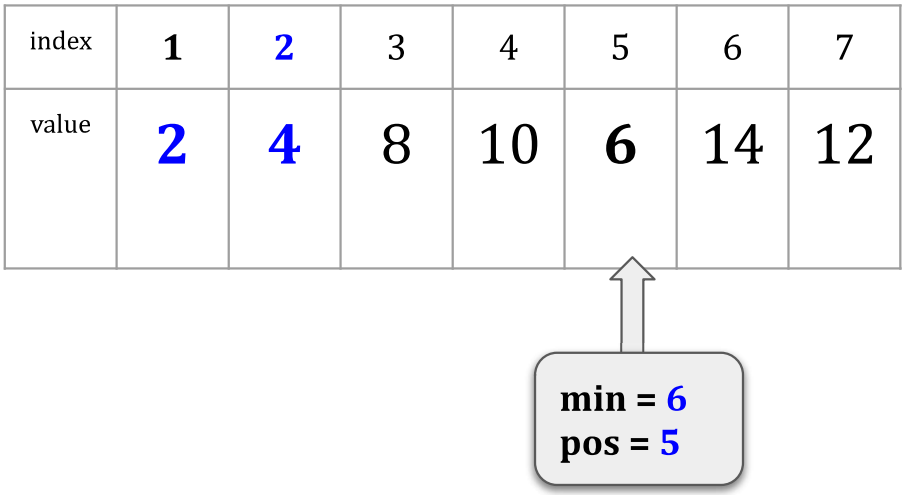
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	8	10	6	14	12

min = 6
pos = 5

Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]

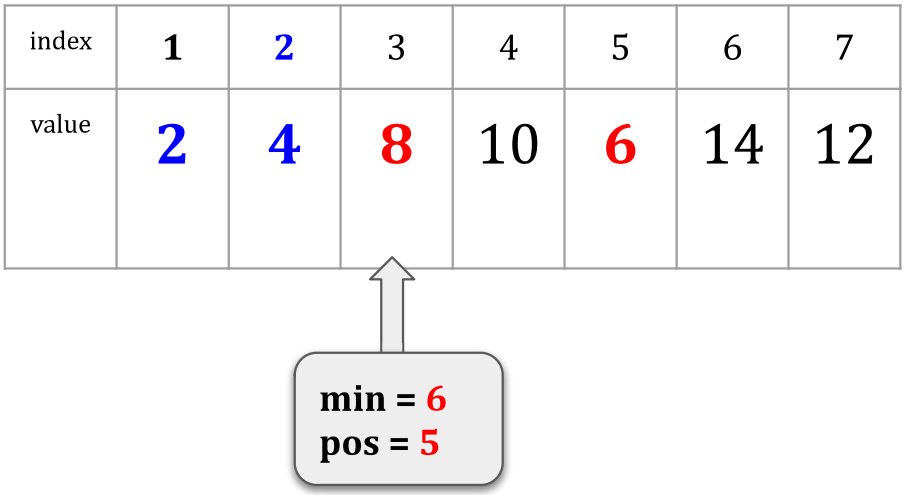
index	1	2	3	4	5	6	7
value	2	4	8	10	6	14	12

min = 6
pos = 5

Selection Sort

SELECTION_SORT(A)

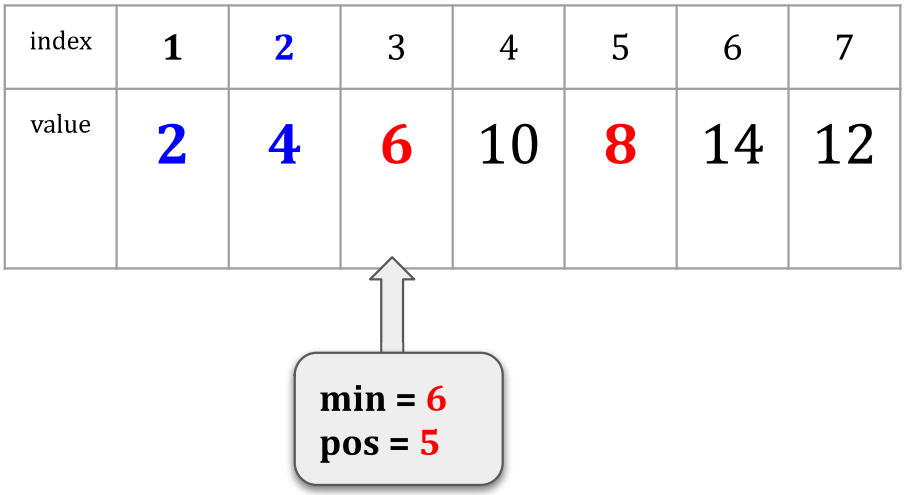
1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

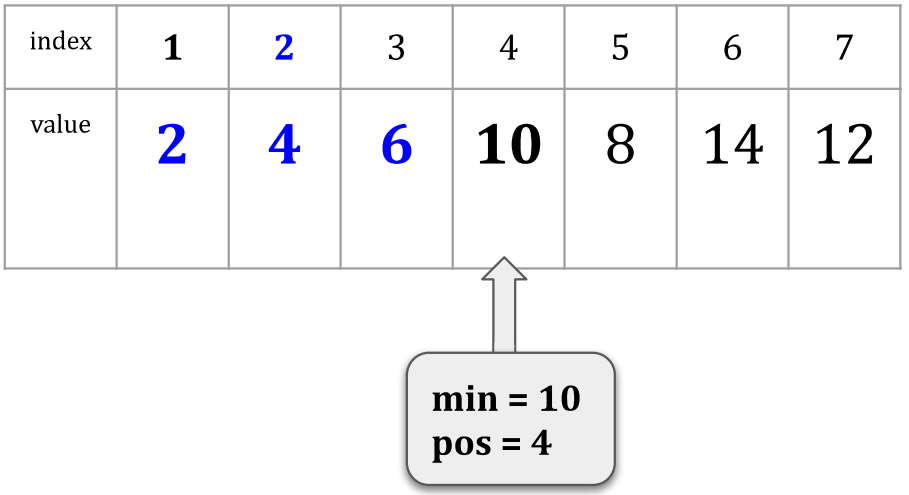
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

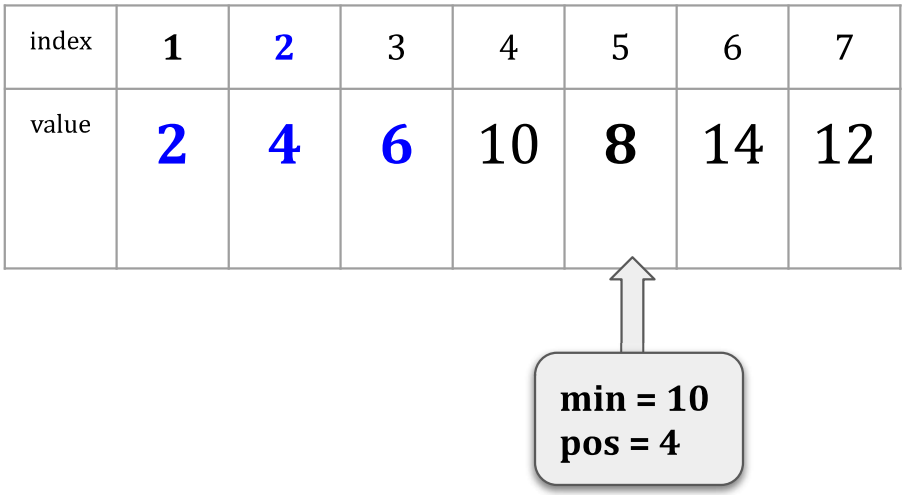
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

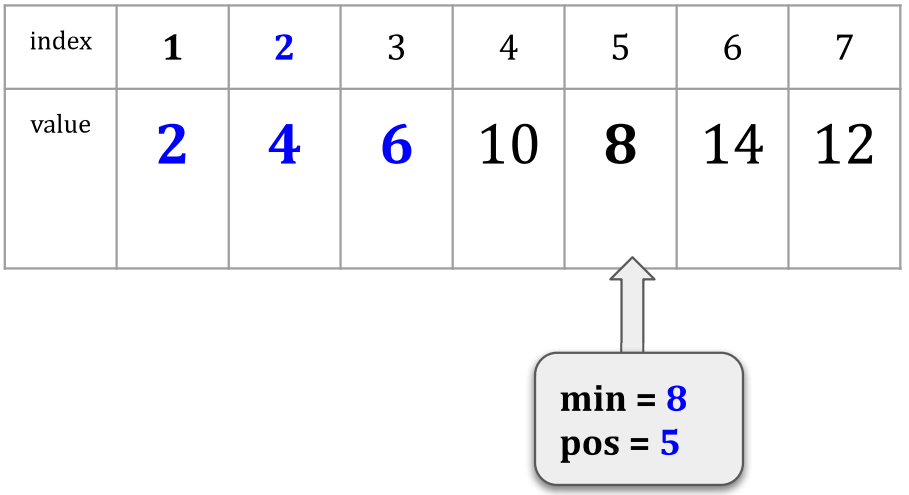
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

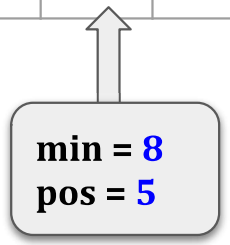


Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	6	10	8	14	12



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	6	10	8	14	12

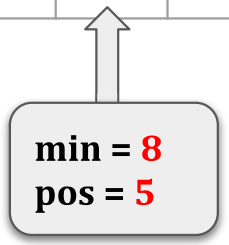
min = 8
pos = 5

Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

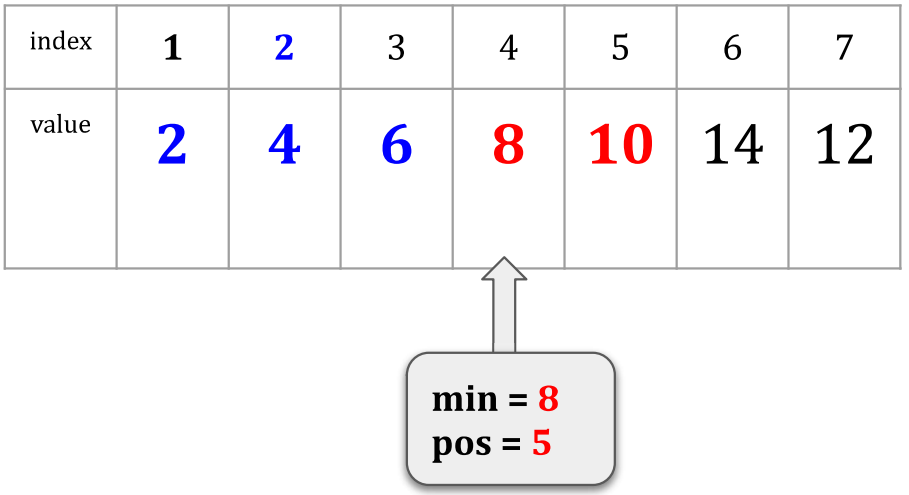
index	1	2	3	4	5	6	7
value	2	4	6	10	8	14	12



Selection Sort

SELECTION_SORT(A)

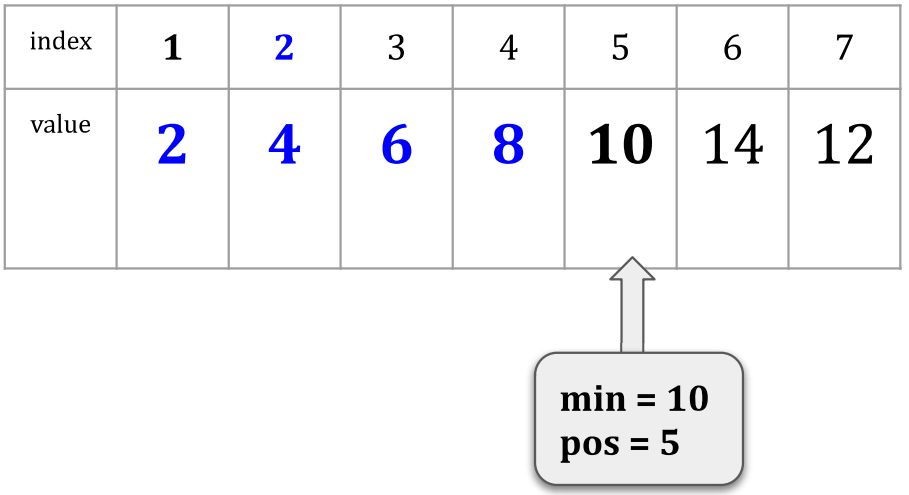
1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

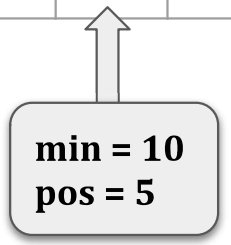


Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	6	8	10	14	12

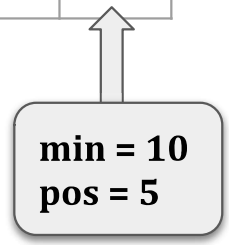


Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]

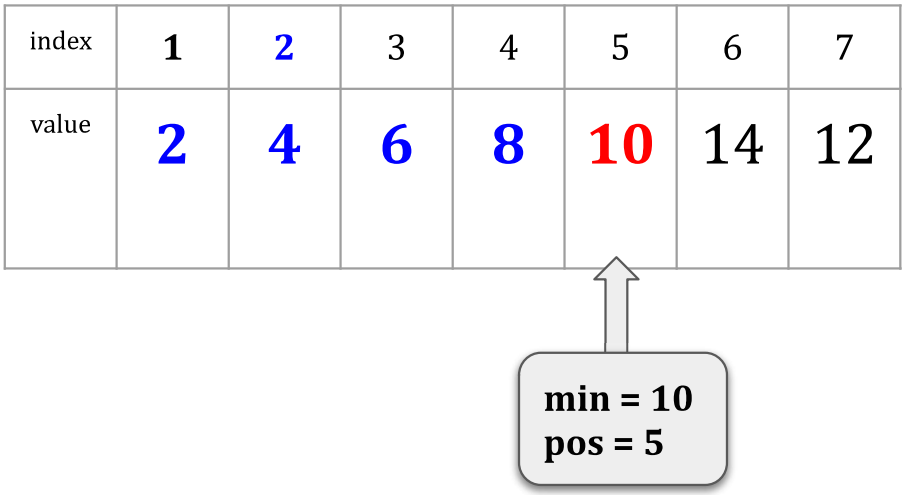
index	1	2	3	4	5	6	7
value	2	4	6	8	10	14	12



Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

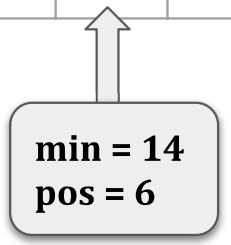


Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	6	8	10	14	12

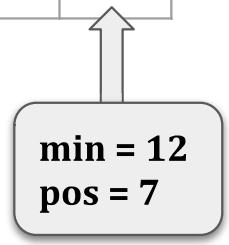


Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
   a. Initialize min=A[i], pos=i
   b. for j=i+1 to A.length
      if (min>A[j])
         Update min=A[j]
         and pos=j
   c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	6	8	10	14	12

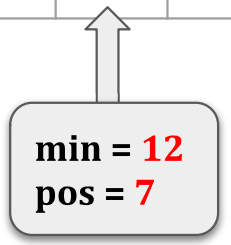


Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	6	8	10	14	12

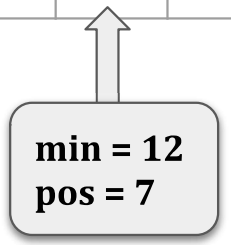


Selection Sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

index	1	2	3	4	5	6	7
value	2	4	6	8	10	12	14



Selection Sort

1. Loop invariant: (ascending order)
   a. At the start of i^{th} iteration of the for loop, the subarray A[1… i-1] is sorted, which contains the least 'i-1' elements.
2. # comparisons (step 1.b) to sort the array in asce. order,
   a. If the input is in asce. order $(n(n-1)/2)$
   b. If the input is in desc. order $(n(n-1)/2)$
3. Minimum and maximum number of comparisons and swaps?

Questions

1. At the start of i^{th} iteration of the for loop, the subarray A[1… i-1] contains the **first** i-1 elements of the original array A, in sorted order.
   a. True for insertion sort
   b. False for selection sort
2. To sort 'n' elements,
   a. Insertion sort requires _____ copy operations (steps 6 and 8) ? $nC_2 + n-1$
   b. Selection sort requires _____ swaps (step 1.c)? $n-1$
   c. Selection sort requires _____ comparisons (step 1.b)? nC_2
3. Is selection sort Stable? What about the other sorts?

Proof of correctness - Selection sort

Loop invariant: (ascending order)

At the start of i^{th} iteration of the for loop, the subarray A[1… i-1] is sorted, which contains the least 'i-1' elements of Array A.

- **Initialization:** Loop invariant trivially holds before the first loop iteration. A[1] contains the minimum element, which is sorted by itself.
- **Maintenance:** At the end of i^{th} iteration, the minimum of A[i..A.length] is selected and placed at i^{th} position
- A[1..i-1] consists of the least i-1 elements in A, in sorted order.
- **Termination**: When the loop terminates, i = A.length and the subarray A[1… A.length-1] is sorted and A[A.length] has the max element.

Selection Sort and Insertion sort

SELECTION_SORT(A)

1. for i=1 to A.length - 1
    a. Initialize min=A[i], pos=i
    b. for j=i+1 to A.length
        if (min>A[j])
            Update min=A[j]
            and pos=j
    c. Swap A[i] with A[pos]

INSERTION_SORT(A)

1. for j=2 to A.length
2. key = A[j];
4. i = j-1
5. while i > 0 and A[i] > key
6. A[i+1]=A[i]
7. i=i-1
8. A[i+1]= key

Thank You !!!