14.2 Selection sort

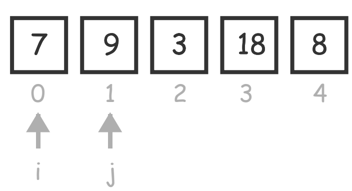
Selection sort

Selection sort is a sorting algorithm that treats the input as two parts, a sorted part and an unsorted part, and repeatedly selects the proper next value to move from the unsorted part to the end of the sorted part.

PARTICIPATION ACTIVITY | 14.2.1: Selection sort.

1 2 3 4 5 6 2x speed

Unsorted Sorted




```
for (i = 0; i < numbersSize - 1; ++i) {

   // Find index of smallest remaining element
   indexSmallest = i
   for (j = i + 1; j < numbersSize; ++j) {

      if (numbers[j] < numbers[indexSmallest]) {
         indexSmallest = j
      }
   }

   // Swap numbers[i] and numbers[indexSmallest]
   temp = numbers[i]
   numbers[i] = numbers[indexSmallest]
   numbers[indexSmallest] = temp
}
```

Selection sort treats the input as two parts, a sorted and unsorted part. Variables i and j keep track of the two parts.

Captions

1. Selection sort treats the input as two parts, a sorted and unsorted part. Variables i and j keep track of the two parts.
2. The selection sort algorithm searches the unsorted part of the array for the smallest element; indexSmallest stores the index of the smallest element found.
3. Elements at i and indexSmallest are swapped.
4. Indices for the sorted and unsorted parts are updated.
5. The unsorted part is searched again, swapping the smallest element with the element at i.
6. The process repeats until all elements are sorted.

Feedback?

The index variable i denotes the dividing point. Elements to the left of i are sorted, and elements including and to the right of i are unsorted. All elements in the unsorted part are searched to find the index of the element with the smallest value. The variable indexSmallest stores the index of the smallest element in the unsorted part. Once the element with the smallest value is found, that element is swapped with the element at location i. Then, the index i is advanced one place to the right, and the process repeats.

The term "selection" comes from the fact that for each iteration of the outer loop, a value is selected for position i.

PARTICIPATION ACTIVITY | 14.2.2: Selection sort algorithm execution.

Assume selection sort's goal is to sort in ascending order.

1) Given list (9, 8, 7, 6, 5), what value will be in the 0^{th} element after the first pass over the outer loop (i = 0)?

Check Show answer

Answer

5

Each outer loop iteration moves the smallest element in the unsorted part, in this case 5, to the i^{th} position. In other words, the correct element for the i^{th} position is "selected", hence the algorithm's name.

2) Given list (9, 8, 7, 6, 5), how many swaps will occur during the first pass of the outer loop (i = 0)?

Check Show answer

Answer

1

One swap occurs during each pass of the outer loop. The swap occurs at the end of the outer loop, once the search for the smallest element (inner loop) has completed.

3) Given list (5, 9, 8, 7, 6) and i = 1, what will be the list after completing the second outer loop iteration? Type answer as: 1, 2, 3

Check | Show answer

Answer

5, 6, 8, 7, 9

The sorted part is just (5) and the unsorted part is (9, 8, 7, 6). The second outer loop iteration will swap the first element in the unsorted part (9) with the smallest element in the unsorted part (6).

Feedback?

Selection sort runtime

Selection sort has the advantage of being easy to code, involving one loop nested within another loop, as shown below.

Figure 14.2.1: Selection sort algorithm.

```
SelectionSort(numbers, numbersSize) {
   i = 0
   j = 0
   indexSmallest = 0
   temp = 0  // Temporary variable for swap

   for (i = 0; i < numbersSize - 1; ++i) {

      // Find index of smallest remaining element
      indexSmallest = i
      for (j = i + 1; j < numbersSize; ++j) {

         if ( numbers[j] < numbers[indexSmallest] ) {
            indexSmallest = j
         }
      }

      // Swap numbers[i] and numbers[indexSmallest]
      temp = numbers[i]
      numbers[i] = numbers[indexSmallest]
      numbers[indexSmallest] = temp
   }
}

main() {
   numbers[] = { 10, 2, 78, 4, 45, 32, 7, 11 }
   NUMBERS_SIZE = 8
   i = 0

   print("UNSORTED: ")
   for (i = 0; i < NUMBERS_SIZE; ++i) {
      print(numbers[i] + " ")
   }
   printLine()

   SelectionSort(numbers, NUMBERS_SIZE)

   print("SORTED: ")
   for (i = 0; i < NUMBERS_SIZE; ++i) {
      print(numbers[i] + " ")
   }
   printLine()
}
```

```
UNSORTED: 10 2 78 4 45 32 7 11
SORTED: 2 4 7 10 11 32 45 78
```

Feedback?

Selection sort may require a large number of comparisons. The selection sort algorithm runtime is $O(N^2)$. If a list has N elements, the outer loop executes N - 1 times. For each of those N - 1 outer loop executions, the inner loop executes an average of $\frac{N}{2}$ times. So the total number of comparisons is proportional to $(N-1) \cdot \frac{N}{2}$, or O(N^2). Other sorting algorithms involve more complex algorithms but have faster execution times.

PARTICIPATION ACTIVITY 14.2.3: Selection sort runtime.

Enter an integer value for each answer.

1) When sorting a list with 50 elements, `indexSmallest` will be assigned to a minimum of ____ times.

Check Show answer

Answer

50

One assignment to `indexSmallest` occurs before the outer loop starts. A list of 50 elements causes 49 outer loop iterations, each of which assigns to `indexSmallest`. The inner loop never assigns to `indexSmallest` if the list is already sorted. So a minimum of 1 + 49 + 0 = 50 assignments occur.

2) About how many times longer will sorting a list of 2X elements take compared to sorting a list of X elements?

Check Show answer

Answer

4

Selection sort is $O(N^2)$, so doubling the number of elements increases runtime by about 4 times. $(2X)^2$ / X^2 = $4X^2$ / X^2 = 4.

3) About how many times longer will sorting a list of 10X elements take compared to sorting a list of X elements?

Check Show answer

Answer

100

Selection sort is $O(N^2)$, so increasing the number of elements by a factor of 10 increases runtime by a factor of about 100. $(10X)^2$ / X^2 = $100X^2$ / X^2 = 100.

Feedback?

CHALLENGE ACTIVITY 14.2.1: Selection sort.

437612.2739130.qx3zqy7

Start

When using selection sort to sort a list with 26 elements, what is the minimum number of assignments to `indexSmallest` once the outer loop starts?

Ex: 4

1 2 3 4

Check Next

Feedback?

How was this section?

Provide section feedback