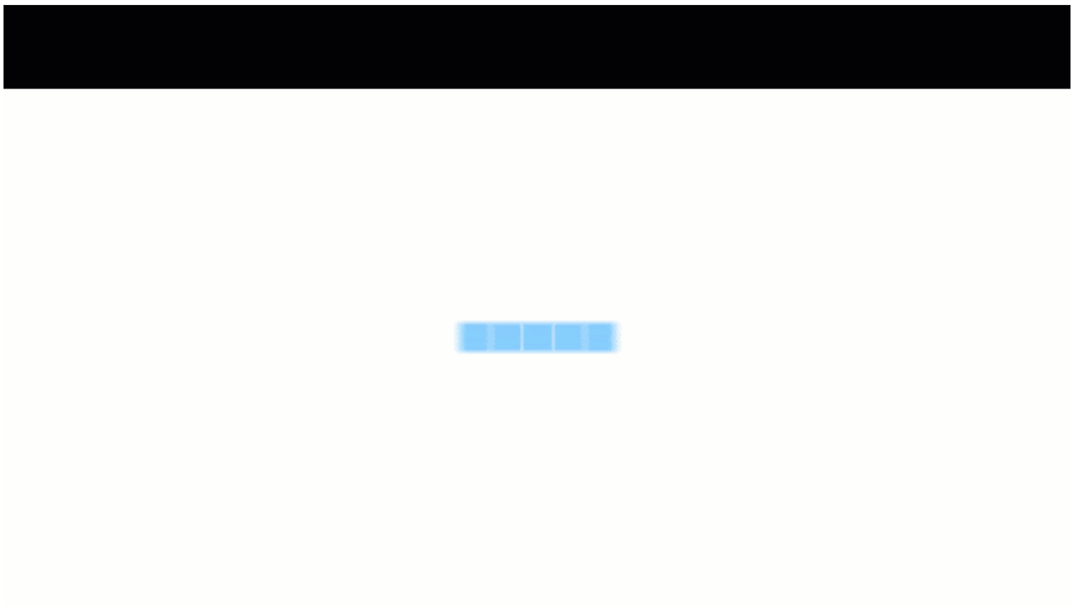
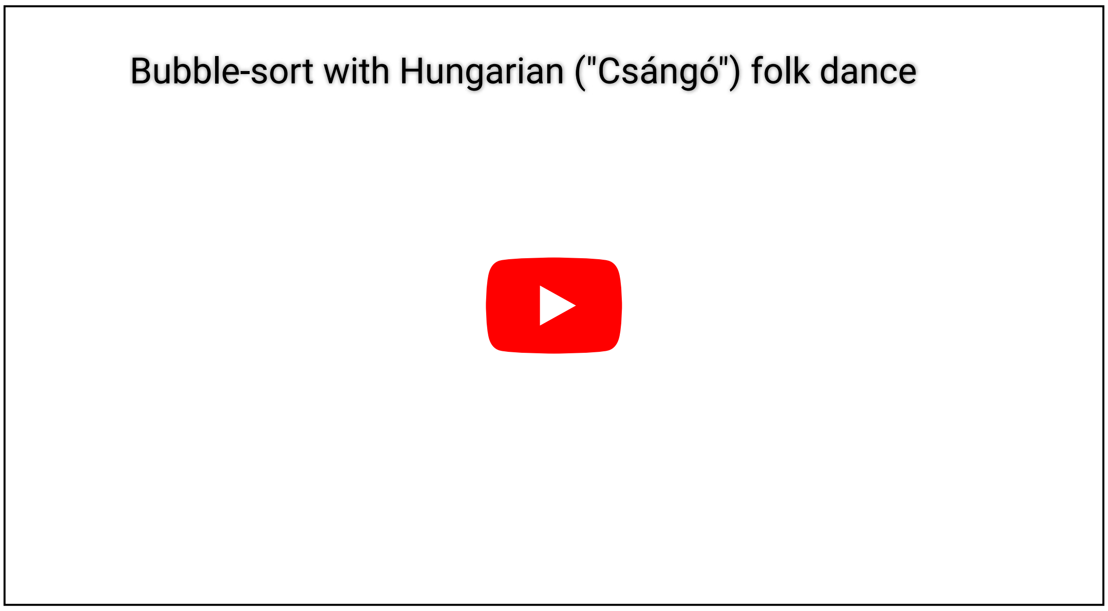
Simple Sorting

Simple Sort, also known as Bubble Sort, is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. The swapping is repeatedly passing through the list until it is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list.



Recommended: Please take 20 minutes to solve it on your own first, write the algorithm before moving on to the solution at the end of this skill.



Simple Sorting

Selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1. The subarray which is already sorted.
2. Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element from the unsorted subarray is picked up and moved to the sorted subarray.

Simple Sort Algorithms

Let's see the example below.

We are going to implement the algorithm of the simple sort and bubble sort.

```
PROCEDURE swap(VAR xp, VAR yp : INTEGER)
VAR
   tmp : INTEGER;
BEGIN
  tmp := xp;
  xp := yp;
  yp := tmp;
END
/* *** Bubble sort *** */
PROCEDURE bubble_sort(VAR tab : ARRAY_OF INTEGER)
VAR
   i,j,n : INTEGER;
BEGIN
   n := tab.length;
   FOR i FROM 0 TO n- 1 STEP 1  DO
       // Last i elements are already in place
       FOR j  FROM 0 TO n-i-1 STEP 1  DO
           IF (tab[j] > tab[j+1]) THEN
               swap(tab[j], tab[j+1])
           END_IF
       END_FOR
   END_FOR
END
```

And this is the implementation of the selection sort algorithm.

```
PROCEDURE swap(VAR xp, VAR yp : INTEGER)
VAR
   tmp : INTEGER;
BEGIN
  tmp := xp;
  xp := yp;
  yp := tmp;
END
```

```
/* *** selection sort *** */
PROCEDURE selection_sort(VAR tab : ARRAY_OF INTEGER)
VAR
   i,j,min_idx,n : INTEGER;
BEGIN
   n := tab.length;
   // one by one move boundary of sub-array
   FOR i FROM 0 TO n-2 STEP 1  DO
       min_idx := i;
       FOR j  FROM i  TO n-i-2 STEP 1  DO
           IF (tab[j]<tab[min_idx]) THEN
               min_idx := j;
           END_IF
       END_FOR
   // swap the found minimum element with the first element
   swap(arr[min_idx],arr[i])
   END_FOR
END
```

‹ Previous   next ›