Sorting Algorithms

CSC140 Foundations of Computer Science

24 February 2020

Please take your time on this exercise. Study the code as you go. Learn what each piece does. Ask classmates, read on the Internet, or ask your instructor to answer any questions you might have about the use of the Python language or the logic of this program.

You will see this again. You must understand the whole program completely.

1. Create a copy of this program:

```
import random

# specify the size of the lists that
# this program will create and sort
LENGTH = 8

def make_unsorted_list( size ):
    # make a list that contains random integers in the
    # interval [10, 99]----these are 2 digit numbers
    # size is the number of random numbers to put in the
    # list
    return [ random.randint(10, 100) for i in range(size) ]

if __name__ == "__main__":
    # test the make_unsorted_list() function
    data = make_unsorted_list( LENGTH )
    print( data )
```

2. Add to the previous program to make a copy of this program:

```
import random

# specify the size of the lists that
# this program will create and sort
LENGTH = 8
```

```
def make_unsorted_list( size ):
    # make a list that contains random integers in the
    # interval [10, 99]----these are 2 digit numbers
    #
    # size is the number of random numbers to put in the
    # list
    return [ random.randint(10, 100) for i in range(size) ]

def position_of_minimum( values, index ):
    # find the index (position) of the smallest number
    # in that part of a list of numbers that begins
    # at a specified index (position)
    #
    # values is a list of numbers
    #
    # index is the position of a number within values
    # at which the search for the smallest value will
    # begin
    best_guess_so_far = index
    for i in range(index + 1, len(values) ):
        if values[i] < values[best_guess_so_far]:
            best_guess_so_far = i
    return best_guess_so_far

if __name__ == "__main__":
    # test the make_unsorted_list() function
    data = make_unsorted_list( LENGTH )
    print( data )

    print( "\n" )

    start = 4
    pos = position_of_minimum( data, start )
    print( f"search for smallest value beginning at index = {start:2d
    print( f"position = {pos:2d}" )
    print( f"value = {data[pos]:2d}" )
```

3. Add to the previous program to make a copy of this program:

```
import random

# specify the size of the lists that
# this program will create and sort
LENGTH = 8

def make_unsorted_list( size ):
```

```
    # make a list that contains random integers in the
    # interval [10, 99]----these are 2 digit numbers
    #
    # size is the number of random numbers to put in the
    # list
    return [ random.randint(10, 100) for i in range(size) ]

    # end of make_unsorted_list()

def position_of_minimum( values, index ):
    # find the index (position) of the smallest number
    # in that part of a list of numbers that begins
    # at a specified index (position)
    #
    # values is a list of numbers
    #
    # index is the position of a number within values
    # at which the search for the smallest value will
    # begin
    best_guess_so_far = index
    for i in range(index + 1, len(values) ):
        if values[i] < values[best_guess_so_far]:
            best_guess_so_far = i
    return best_guess_so_far

    # end of position_of_swap()

def swap( values, i, j ):
    # exchange the values at two specified positions
    # in a list of numbers
    #
    # values ia a list of numbers
    # i is the index (position) an element of values
    # j is the index (position) of another element of values
    temporary = values[i]
    values[i] = values[j]
    values[j] = temporary

    # end of swap()

if __name__ == "__main__":
    # test the make_unsorted_list() function
    data = make_unsorted_list( LENGTH )
    print( data )

    print( "\n" )
```

```
start = 4
pos = position_of_minimum( data, start )
print( f"search for smallest value beginning at index = {start:2d
print( f"position = {pos:2d}" )
print( f"value = {data[pos]:2d}" )

print( "\n" )

i = 4
j = 6
print( f"swap elements at positions {i:2d} and {j:2d}" )
print( "before swap = ", data )
swap( data, i, j )
print( " after swap = ", data )
```

4. Add to the previous program to make a copy of this program:

```
import random

# specify the size of the lists that
# this program will create and sort
LENGTH = 8

def make_unsorted_list( size ):
    # make a list that contains random integers in the
    # interval [10, 99]----these are 2 digit numbers
    #
    # size is the number of random numbers to put in the
    # list
    return [ random.randint(10, 100) for i in range(size) ]

    # end of make_unsorted_list()

def position_of_minimum( values, index ):
    # find the index (position) of the smallest number
    # in that part of a list of numbers that begins
    # at a specified index (position)
    #
    # values is a list of numbers
    #
    # index is the position of a number within values
    # at which the search for the smallest value will
    # begin
    best_guess_so_far = index
    for i in range(index + 1, len(values) ):
```

```
            if values[i] < values[best_guess_so_far]:
                best_guess_so_far = i
        return best_guess_so_far

        # end of position_of_swap()

    def swap( values, i, j ):
        # exchange the values at two specified positions
        # in a list of numbers
        #
        # values ia a list of numbers
        # i is the index (position) an element of values
        # j is the index (position) of another element of values
        temporary = values[i]
        values[i] = values[j]
        values[j] = temporary

        # end of swap()

    def selection_sort( values ):
        # sort a list of numbers using the
        # selection sort algorithm
        #
        # values is the list of numbers to sort
        for i in range( len(values) ):
            j = position_of_minimum( values, i )
            swap( values, i, j )

        # end of selection_sort()

    if __name__ == "__main__":
        # test the make_unsorted_list() function
        data = make_unsorted_list( LENGTH )
        print( data )

        print( "\n" )

        start = 4
        pos = position_of_minimum( data, start )
        print( f"search for smallest value beginning at index = {start:2d}" )
        print( f"position = {pos:2d}" )
        print( f"value = {data[pos]:2d}" )

        print( "\n" )

        i = 4
```

```
    j = 6
    print( f"swap elements at positions {i:2d} and {j:2d}" )
    print( "before swap = ", data )
    swap( data, i, j )
    print( " after swap = ", data )

    print( "\n" )

    print( "sort list with selection sort" )
    selection_sort( data )
    print( "sorted list = ", data )
```

5. Add to the previous program to make a copy of this program:

```
import random

# specify the size of the lists that
# this program will create and sort
LENGTH = 8

def make_unsorted_list( size ):
    # make a list that contains random integers in the
    # interval [10, 99]----these are 2 digit numbers
    #
    # size is the number of random numbers to put in the
    # list
    return [ random.randint(10, 100) for i in range(size) ]

    # end of make_unsorted_list()

def position_of_minimum( values, index ):
    # find the index (position) of the smallest number
    # in that part of a list of numbers that begins
    # at a specified index (position)
    #
    # values is a list of numbers
    #
    # index is the position of a number within values
    # at which the search for the smallest value will
    # begin
    best_guess_so_far = index
    for i in range(index + 1, len(values) ):
        if values[i] < values[best_guess_so_far]:
            best_guess_so_far = i
    return best_guess_so_far
```

```
    # end of position_of_swap()

def swap( values, i, j ):
    # exchange the values at two specified positions
    # in a list of numbers
    #
    # values ia a list of numbers
    # i is the index (position) an element of values
    # j is the index (position) of another element of values
    temporary = values[i]
    values[i] = values[j]
    values[j] = temporary

    # end of swap()

def selection_sort( values ):
    # sort a list of numbers using the
    # selection sort algorithm
    #
    # values is the list of numbers to sort
    for i in range( len(values) ):
        j = position_of_minimum( values, i )
        swap( values, i, j )

    # end of selection_sort()

def insert( values, i ):
    # given a list of numbers whose first i elements are in order,
    # insert the i-th element into the proper place so that the
    # first i + 1 elements are then in order
    #
    # values is a list of numbers
    # i is index of the first element that follows
    # part of the list that is already known to be
    # sorted
    j = i - 1
    while j >= 0 and values[j] > values[i]:
        swap( values, i, j )
        i = i - 1
        j = j - 1

    # end of insert()

if __name__ == "__main__":
    # test the make_unsorted_list() function
    data = make_unsorted_list( LENGTH )
```

```
    print( data )

    print( "\n" )

    start = 4
    pos = position_of_minimum( data, start )
    print( f"search for smallest value beginning at index = {start:2d
    print( f"position = {pos:2d}" )
    print( f"value = {data[pos]:2d}" )

    print( "\n" )

    i = 4
    j = 6
    print( f"swap elements at positions {i:2d} and {j:2d}" )
    print( "before swap = ", data )
    swap( data, i, j )
    print( " after swap = ", data )

    print( "\n" )

    print( "sort list with selection sort" )
    selection_sort( data )
    print( "sorted list = ", data )

    prefix = [ 2, 5, 9, 14 ]
    suffix = [ 3, 7, 8, 11 ]
    samples = prefix + suffix
    print( "begin to sort list with insertion sort" )
    print( "   unsorted list = ", samples )
    insert( samples, 4 )
    print( "more sorted list = ", samples )

    # end of main
```

6. Add to the previous program to make a copy of this program:

```
import random

# specify the size of the lists that
# this program will create and sort
LENGTH = 8

def make_unsorted_list( size ):
    # make a list that contains random integers in the
    # interval [10, 99]----these are 2 digit numbers
```

```
    #
    # size is the number of random numbers to put in the
    # list
    return [ random.randint(10, 100) for i in range(size) ]

    # end of make_unsorted_list()

def position_of_minimum( values, index ):
    # find the index (position) of the smallest number
    # in that part of a list of numbers that begins
    # at a specified index (position)
    #
    # values is a list of numbers
    #
    # index is the position of a number within values
    # at which the search for the smallest value will
    # begin
    best_guess_so_far = index
    for i in range(index + 1, len(values) ):
        if values[i] < values[best_guess_so_far]:
            best_guess_so_far = i
    return best_guess_so_far

    # end of position_of_swap()

def swap( values, i, j ):
    # exchange the values at two specified positions
    # in a list of numbers
    #
    # values ia a list of numbers
    # i is the index (position) an element of values
    # j is the index (position) of another element of values
    temporary = values[i]
    values[i] = values[j]
    values[j] = temporary

    # end of swap()

def selection_sort( values ):
    # sort a list of numbers using the
    # selection sort algorithm
    #
    # values is the list of numbers to sort
    for i in range( len(values) ):
        j = position_of_minimum( values, i )
        swap( values, i, j )
```

```
    # end of selection_sort()

def insert( values, i ):
    # given a list of numbers whose first i elements are in order,
    # insert the i-th element into the proper place so that the
    # first i + 1 elements are then in order
    #
    # values is a list of numbers
    # i is index of the first element that follows
    # part of the list that is already known to be
    # sorted
    j = i - 1
    while j >= 0 and values[j] > values[i]:
        swap( values, i, j )
        i = i - 1
        j = j - 1

    # end of insert()

def insertion_sort( values ):
    for i in range( len(values) ):
        insert( values, i )

if __name__ == "__main__":
    # test the make_unsorted_list() function
    data = make_unsorted_list( LENGTH )
    print( data )

    print( "\n" )

    start = 4
    pos = position_of_minimum( data, start )
    print( f"search for smallest value beginning at index = {start:2d
    print( f"position = {pos:2d}" )
    print( f"value = {data[pos]:2d}" )

    print( "\n" )

    i = 4
    j = 6
    print( f"swap elements at positions {i:2d} and {j:2d}" )
    print( "before swap = ", data )
    swap( data, i, j )
    print( " after swap = ", data )
```

```
print( "\n" )

print( "sort list with selection sort" )
selection_sort( data )
print( "sorted list = ", data )

prefix = [ 2, 5, 9, 14 ]
suffix = [ 3, 7, 8, 11 ]
samples = prefix + suffix
print( "begin to sort list with insertion sort" )
print( "   unsorted list = ", samples )
insert( samples, 4 )
print( "more sorted list = ", samples )

print( "\n" )

print( "sort list with insertion sort" )
measurements = make_unsorted_list( LENGTH )
print( "unsorted list = ", measurements )
insertion_sort( measurements )
print( "  sorted list = ", measurements )

# end of main
```

7. How many pairs of elements of values does a call to position_of_minimum() examine?

8. Modify the definition of selection_sort () so that it returns to its caller a count of the number of times that it compares the values of two elements in a list.

9. Modify the definition of insert () so that it returns to its caller a count of the number of times that it compares the values of two elements of a list.

10. Modify the definition of insertion_sort () so that it returns to its caller a count of the number of times that it compares the values of two elements of a list.

11. Modify this program so that it compares the amount of work that the two sort algorithms do in sorting the same list. You will need to learn how to make a copy of a list.

12. Conduct an experiment. How does the performance of the selection sort compare to the performance of the insertion sort?