ICS4U Unit 1 Activity 6 - Array Reading and Worksheet

Phone Pyae Thet Khine
Teacher NEDA
ICS4U
8th Feb 2022

1. Write an Algorithm to insert and delete array elements.

```
def insert_integers(arr):
    """
    The function `insert_integers` takes an array as an argument and
appends integers to the array until
    the user enters 'q' to quit

    :param arr: The array to insert integers into
    """
    while True:
        x = input("Enter an integer (r to remove): ")
        if x == 'r':
            break
        try:
            arr.append(int(x))
        except ValueError:
            print("ValueError. Enter an integer (r to remove): ")

    print(f"{arr} <== Inserted array elements")


def delete_integers(arr):
    """
    The function takes in an array and allows the user to remove elements
from the array until the array
    is empty

    :param arr: an array of integers
    """

    while len(arr) != 0:
        y = input("Enter the index you want to remove (q to quit): ")
        if y == 'q':
            break
        try:
            arr.pop(int(y))
        except IndexError:
```

```
            print("IndexError. Enter a valid index number.")
        except ValueError:
            print("ValueError. Enter the index you want to remove (q to
quit): ")
        else:
            print(arr)
    print("Array is empty. No more elements to remove.")


def insert_delete_integers():
    """
    This function will insert integers into an array and then delete
integers from the array.
    """
    arr = []  # Declaring a blank array

    insert_integers(arr)
    delete_integers(arr)

insert_delete_integers()
```

Output

```
Enter an integer (r to remove): 1
Enter an integer (r to remove): 2
Enter an integer (r to remove): 3
Enter an integer (r to remove): 4
Enter an integer (r to remove): 5
Enter an integer (r to remove): r
[1, 2, 3, 4, 5] <== Inserted array elements
Enter the index you want to remove (q to quit): 0
[2, 3, 4, 5]
Enter the index you want to remove (q to quit): 0
[3, 4, 5]
Enter the index you want to remove (q to quit): 0
[4, 5]
Enter the index you want to remove (q to quit): 0
[5]
Enter the index you want to remove (q to quit): 0
[]
Array is empty. No more elements to remove.
```

2.Create linear and binary search algorithm to find data in an array.

```
def make_unique_list():
    """
    It creates a list of 30 random integers between 0 and 99, then it
removes all duplicates from the
```

```
    list
    :return: A list of unique integers.
    """
    import random

    highest_integer = 99
    lowest_integer = 0

    non_unique_list = []

    for i in range(30):
        non_unique_list.append(random.randint(lowest_integer,
highest_integer))

    unique_list = list((set(non_unique_list)))

    print(f"\n{non_unique_list} <== list, probably not unique.\n")
    print(f"{unique_list} <== truly unique list.\n")

    return unique_list


def get_search_value():
    while True:
        try:
            search_value = int(
                input(f"Which integer do you want to search? (0 to 99): "))
        except ValueError:
            print("ValueError. Please enter an integer.")
        else:
            break

    return search_value


# ============= #
# Binary Search #
# ============= #


def binary_search():
    """
    The function takes a list of unique numbers and a search value, and
returns the index of the search
    value in the list
    """

    # Declaring variables that will be used in the binary_search()
function.
    unique_list = make_unique_list()
    search_value = get_search_value()
```

```
    start_index = 0
    end_index = len(unique_list) - 1

    found = False

    # A binary search algorithm. It is searching for a value in a list.
    # Finding the middle index of the list.
    while start_index <= end_index:
        middle_index = start_index + (end_index - start_index) // 2
        middle_value = unique_list[middle_index]

        # Checking if the middle value is equal to the search value. If it
is, it prints the index of the
        # middle value.
        if middle_value == search_value:
            print(f"The value that you are searching for is at index
{middle_index} (Binary Search).")
            found = True
            break
        # Checking if the middle value is less than the search value. If it
is, it adds 1 to the
        # start index and adds 1 to the count.
        elif middle_value < search_value:
            start_index = middle_index + 1
        # Checking if the middle value is greater than the search value. If
it is, it subtracts 1
        # from the end index and adds 1 to the count.
        else:
            end_index = middle_index - 1

    # Checking if the search value is not in the list. If it is not in the
list, it prints
    # a message.
    if found is False:
        print(f"The value that you are searching for is not in the list:
\n{unique_list}")


# ============= #
# Linear Search #
# ============= #


def linear_search():
    """
    The function linear_search() takes a list of unique numbers and a
search value, and returns the index
    of the search value if it is in the list, and returns a message if the
search value is not in the
    list.
    """
```

```
    unique_list = make_unique_list()
    search_value = get_search_value()

    # Iterating through the list and checking if the search value is equal
to the value in the list.
    # If it is, it prints the index of the value in the list.
    for i in range(len(unique_list)):
        if search_value == unique_list[i]:
            print(f"The value that you are searching for is at index {i}
(Linear Search).")
            break
    # Checking if the search value is not in the list.
    else:
        print(
            f"The value that you are searching for is not in the list:
\n{unique_list}")
```

Output

```
[94, 97, 94, 66, 55, 30, 57, 2, 71, 47, 57, 83, 61, 32, 14, 2, 60, 77, 10,
88, 86, 25, 84, 39, 71, 86, 85, 5, 55, 77] <== list, probably not unique.

[2, 5, 10, 14, 25, 30, 32, 39, 47, 55, 57, 60, 61, 66, 71, 77, 83, 84, 85,
86, 88, 94, 97] <== truly unique list.

Which integer do you want to search? (0 to 99): 94
The value that you are searching for is at index 21 (Binary Search).

[77, 24, 35, 70, 30, 20, 86, 82, 15, 14, 82, 74, 25, 33, 72, 86, 46, 40,
37, 61, 69, 44, 84, 30, 71, 35, 75, 72, 43, 52] <== list, probably not
unique.

[14, 15, 20, 24, 25, 30, 33, 35, 37, 40, 43, 44, 46, 52, 61, 69, 70, 71,
72, 74, 75, 77, 82, 84, 86] <== truly unique list.

Which integer do you want to search? (0 to 99): 77
The value that you are searching for is at index 21 (Linear Search).
```

3. Ceate sorting algorithm using bubble, insertion, selection to sort data in an array.

```
def make_random_list():
    """
    It creates a list of 30 random integers between 0 and 99
    :return: A list of 30 random integers between 0 and 99.
    """
    import random
    highest_integer = 99
```

```
    lowest_integer = 0

    random_list = []

    for i in range(30):
        random_list.append(random.randint(lowest_integer, highest_integer))

    print(f"\n{random_list} <== Random list.\n")

    return random_list


# Bubble Sort algorithm
def bubble_sort():
    """
    The function makes a random list, then iterates through the list,
comparing each element to the next
    element, and if the first element is greater than the second element,
the two elements are swapped
    """
    import time

    random_list = make_random_list()
    time.sleep(2)
    list_len = len(random_list)

    # Iterating through the list and comparing each element to the next
element, and if the first element
    # is greater than the second element, the two elements are swapped.
    print("Starting Bubble Sort")
    for i in range(list_len):
        for j in range(list_len):
            k = j + 1
            try:
                if random_list[j] > random_list[k]:
                    random_list[j], random_list[k] = random_list[k],
random_list[j]
                    print(random_list)
                    time.sleep(0.05)
            except IndexError:
                break
    # Printing the sorted array.
    print(f"\n{random_list} <== Sorted array")


# Insertion Sort algorithm
def insertion_sort():
    """
    For each element in the list, if the element is less than the element
before it, swap the two
    elements
    """
```

```
    import time

    random_list = make_random_list()
    time.sleep(2)
    list_len = len(random_list)

    # Iterating through the list and comparing each element to the element
before it, and if the
    # element is less than the element before it, the two elements are
swapped.
    print("Starting Insertion Sort")
    for h in range(list_len):
        for i in range(1, list_len):
            j = i - 1

            if random_list[i] < random_list[j]:
                random_list[i], random_list[j] = random_list[j],
random_list[i]
                print(random_list)
                time.sleep(0.05)

    print(f"\n{random_list} <== Sorted array")


# Selection Sort algorithm

def selection_sort():
    """
    The selection sort algorithm sorts an array by repeatedly finding the
minimum element (considering ascending order) from
    unsorted part and putting it at the beginning
    """

    import time

    random_list = make_random_list()
    time.sleep(2)
    list_len = len(random_list)

    print("Starting Selection Sort")
    # Iterating through the list and comparing each element to the next
element, and if the first element
    # is greater than the second element, the two elements are swapped.
    for i in range(list_len):
        start_index = i
        for j in range(start_index + 1, list_len):
            if random_list[start_index] > random_list[j]:
                start_index = j
        random_list[i], random_list[start_index] =
random_list[start_index], random_list[i]
        print(random_list)
        time.sleep(0.2)
```

```
    print(f"\n{random_list} <== Sorted array")
```

Output

```
[28, 67, 89, 3, 65, 15, 38, 12, 23, 95, 61, 12, 37, 16, 23, 8, 12, 48, 27,
80, 5, 49, 20, 87, 99, 23, 93, 84, 25, 15] <== Random list.

Starting Bubble Sort

[3, 5, 8, 12, 12, 12, 15, 15, 16, 20, 23, 23, 23, 25, 27, 28, 37, 38, 48,
49, 61, 65, 67, 80, 84, 87, 89, 93, 95, 99] <== Sorted array

[18, 60, 79, 23, 21, 59, 47, 6, 15, 82, 37, 67, 66, 68, 31, 63, 36, 89, 73,
82, 55, 49, 56, 79, 39, 25, 80, 86, 4, 11] <== Random list.

Starting Insertion Sort

[4, 6, 11, 15, 18, 21, 23, 25, 31, 36, 37, 39, 47, 49, 55, 56, 59, 60, 63,
66, 67, 68, 73, 79, 79, 80, 82, 82, 86, 89] <== Sorted array

[35, 28, 14, 25, 15, 31, 98, 29, 25, 53, 40, 58, 7, 38, 26, 48, 24, 91, 71,
9, 24, 98, 17, 6, 14, 68, 73, 21, 41, 46] <== Random list.

Starting Selection Sort

[6, 7, 9, 14, 14, 15, 17, 21, 24, 24, 25, 25, 26, 28, 29, 31, 35, 38, 40,
41, 46, 48, 53, 58, 68, 71, 73, 91, 98, 98] <== Sorted array
```

The Output is simplified for the PDF format. Please run the code to see the full animation of the sorting process.

4. Create a recursive algorithm to calculate a factorial.

```
def factorial():
    """
    This function takes an integer from the user and prints the factorial
of that integer
    """
    # Trying to get an integer from the user and if the user enters a non-
integer, it will print an error
    # message.
    try:
        x = int(input("Enter an integer for factorial: "))
    except ValueError:
        print("ValueError. Enter an integer for factorial")

    # A while loop that is multiplying the factorial by the integer and
```

```
then subtracting 1 from the
    # integer until the integer is 0.
    factorial_result = 1

    while x != 0:
        factorial_result *= x
        x -= 1
    print(f"Factorial of your integer is: {factorial_result}")
```

Output

```
Enter an integer for factorial: 5
Factorial of your integer is: 120
```