ICS4U Unit 3 Activity 8 - Unit Test Part 2

Phone Pyae Thet Khine
Teacher NEDA
ICS4U
14th May 2022

Knowledge/Understanding/Communication

Q1) Define, using examples where appropriate, the following computer terms:

1) Algorithm
Algorithm is a set of rules and or instructions which is needed to be followed to accomplish something especially in a computer program. Even a recipe can be said to be an algorithm but the term "Algorithm" is mostly used in Computer Science. Algorithms do not necessarily have to be codes, they can also be written in simple plain English or even Pseudo code.

Algorithm for Binary Search (Array needs to be a sorted array)

```
1. Ask for the value that the user wants to search.
2. Access the value in the middle index of the array.
3. If the middle value matches the search value, return the middle index. If the values do not match, continue.
4. If the search value is less than the middle value, discard the right side of the array. If the value is greater than the middle value, discard the left side of the array.
5. Repeat step number 2 till the search value is found. If the value is not found till the last element to be searched, return "not found".
```

2) Computational complexity analysis
Computational Complexity Analysis measures how wfficient an algorithm is and what resources are required to run the algorithm. It measures the time needed to run all the nested loops along with the whole program.
3) Recursive function
Recursive Functions are functions that has a recursive loop inside it and in that loop calls the function that initiates the loop itself sometimes with a different parameter or argument.

```python
def recursiveFunction(num):
    if num != 0:
```

```
        print(num)
        recursiveFunction(num - 1)

recursiveFunction(5)
```

Output

```
5
4
3
2
1
```

4) Loop
Loops are like recursive functions but without the function. This loops a section or a block of code until a condition is met.

```
x = 10

while x != 0:
        print(x)
        x -= 1
```

Output

```
10
9
8
7
6
5
4
3
2
1
```

Thinking/Inquiry

Q2) Compute a program to compare linear search and binary search.

```
def getSearchValue():

    while True:
        try:
```

```
            searchValue = int(
                input(f"Which integer do you want to search? (0 to 99): "))
        except ValueError:
            print("ValueError. Please enter an integer.")
        else:
            break

    return(searchValue)

def makeUniqueList():
    import random

    highestInteger = 99
    lowestInteger = 0

    nonUniqueList = []

    for i in range(30):
        nonUniqueList.append(random.randint(lowestInteger, highestInteger))

    uniqueList = list((set(nonUniqueList)))

    print(f"\n{nonUniqueList} <== list, probably not unique.\n")
    print(f"{uniqueList} <== truely unique list.\n")

    return(uniqueList)


def linearSearch(uniqueList, searchValue):
    count = 0
    for i in range(len(uniqueList) - 1):
        if searchValue == uniqueList[i]:
            print(f"The value that you are searching for is at index {i}
(Linear Search).")
            count += 1
            break

    else:
        print(
            f"The value that you are searching for is not in the list:
\n{uniqueList}")

    return(count)

def binarySearch(uniqueList, searchValue):

    start_index = 0
    end_index = len(uniqueList) - 1

    count = 0

    found = False
```

```
    while start_index <= end_index:
        middle_index = start_index + (end_index - start_index) // 2
        middle_value = uniqueList[middle_index]

        if middle_value == searchValue:
            print(f"The value that you are searching for is at index
{middle_index} (Binary Search).")
            found = True
            count += 1
            break
        elif middle_value < searchValue:
            start_index = middle_index + 1
            count += 1
        else:
            end_index = middle_index - 1
            count += 1

    if found is False:
        print(f"The value that you are searching for is not in the list:
\n{uniqueList}")

    return(count)


import time

uniqueList = makeUniqueList()
searchValue = int(input("What number are you searching for? (1 - 100): "))

start = time.time()
linearCount = linearSearch(uniqueList, searchValue)
end = time.time()
linearTime = end - start

start = time.time()
binaryCount = binarySearch(uniqueList, searchValue)
end = time.time()
binaryTime = (end - start)

print(f"Linear Search makes {linearCount} operation(s) and took
{linearTime}")
print(f"Binary Search makes {binaryCount} operation(s) and took
{binaryTime}")
```

Output

```
[57, 38, 72, 41, 97, 3, 35, 81, 16, 43, 50, 92, 70, 9, 61, 34, 40, 17, 58,
7, 66, 98, 73, 82, 68, 81, 0, 13, 44, 70] <== list, probably not unique.
```

```
[0, 3, 7, 9, 13, 16, 17, 34, 35, 38, 40, 41, 43, 44, 50, 57, 58, 61, 66,
68, 70, 72, 73, 81, 82, 92, 97, 98] <== truely unique list.

What number are you searching for? (1 - 100): 66
The value that you are searching for is at index 18 (Linear Search).
The value that you are searching for is at index 18 (Binary Search).
Linear Search makes 19 operation(s) and took 7.796287536621094e-05
Binary Search makes 4 operation(s) and took 1.9311904907226562e-05
```

This program counts the number of operations that each searching algorithm does and also measures the time taken for each algorithm to perform its task. These are then displayed in the output. As we can see in the output, Binary search is superior than Linear Search. It takes less time and operations for the Binary Search Algorithm to successfully search for the asked value.

Application

Q3) Create an algorithm and the program for Fibonacci numbers.

```
1. Get Numbber of sequences the user wants and subtract two.
2. Make a list containing these two elements [1, 2]
3. Sum the last two indexes and append them at the end of the list above.
4. Repeat recursively for the number calculated in Step 1.
```

```
def fibonacci():
    n = int(input("How many sequences of the fibonacci do you want?: "))

    if n == 1:
        print([1])
    elif n == 2:
        print([1, 2])
    else:
        n -= 2

        fibonacci = [1, 2]
        for i in range(n):
            fibonacci.append(fibonacci[-1] + fibonacci[-2])
        print(fibonacci)

fibonacci()
```

Output

```
How many sequences of the fibonacci do you want?: 10
[1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Communication

Q4) Write a program to make circle and then color it.

```
import turtle as t

t.speed(1)
t.fillcolor("green")
t.begin_fill()
t.circle(100)
t.end_fill()
```

Output