ENSF338 W23 – Assignment 2

Due by February 10 2023, end of day (Mountain Time)

Instructions

In order to complete this assignment, you must prepare a document in PDF format with the answer to each question. Several questions will also ask you to provide the code you implemented, specifying the name the file must have. You must also prepare these source code files and upload them to a GitHub repository. Once you completed the assignment, you must add a link to your repository prior to the answers. Then, you must upload **only the PDF document** to the "Assignment 2" D2L drop box for your group. You must do so by the assignment deadline. Finally, you must refrain from updating the files after the

Work distribution among team members

Each group member should strive to contribute approximately equally to the solution. When you prepare your answers document, you must create a section called "Work performed by each member" **at the beginning of the document**. In this section, you must specify clearly (1) what each member contributed to the solution, and (2) a rough estimate of the amount of work (as a percentage) performed by each member. **We will not discuss or address any complaint about group members if based on something different than what presented in this section**.

Answering the questions

Answers must consist of your own work based on the content discussed in lectures and labs. You may discuss assignments with your classmates, but each group must provide their own solutions to the questions. It is permitted to use AI/Q&A websites as an aid to produce code. **Any other use of external help will constitute academic misconduct and sanctioned as such.**

Late submission policy

- No late penalty for submissions up to 1H after the deadline
- Late submissions delivered between 1H and 24H after the deadline: 20% penalty
- Late submissions delivered more than 24H after the deadline will not be graded (i.e., they will receive 0 points)
- Deadlines are **strict**

Exercise 1

In the lab, we have discussed how **memoization**[1] can be used to improve the performance of an algorithm.

1. Explain, in general terms and your own words, what memoization is **(0.5 pts)**
2. Consider the following code:

```
def func(n):
    if n == 0 or n == 1:
        return n
    else:
        return func(n-1) + func(n-2)
```

3. What does it do? **(0.5 pts)**
4. Is this an example of a divide-and-conquer algorithm? Explain. **(0.5 pts)**
5. What is its time complexity? **(0.5 pts)**
6. Implement a version of the code above that use memoization to improve performance. Provide this as ex1.3.py. **(2 pts)**
7. Perform an analysis of your optimized code: what is its computational complexity? **(3 pts)**
8. Time the original code and your improved version, for all integers between 0 and 35, and plot the results. Provide the code you used for this as ex1.5.py. **(2 pt)**
9. Discuss the plot and compare them to your complexity analysis. **(1 pt)**

[1] Yep, not a typo, we are not talking about "memorization".

Exercise 2

Look at the following code:

```
import sys
sys.setrecursionlimit(20000)
def func1(arr, low, high):
    if low < high:
        pi = func2(arr, low, high)
        func1(arr, low, pi-1)
        func1(arr, pi + 1, high)
def func2(array, start, end):
    p = array[start]
    low = start + 1
    high = end
    while True:
        while low <= high and array[high] >= p:
            high = high - 1
        while low <= high and array[low] <= p:
            low = low + 1
        if low <= high:
            array[low], array[high] = array[high], array[low]
        else:
            break
    array[start], array[high] = array[high], array[start]
    return high
```

1. Explain what the code does and perform an average-case complexity analysis. Describe the process, not just the result. **(2 pts)**
2. Test the code on all the inputs at: https://raw.githubusercontent.com/ldklab/ensf338w23/main/assignments/assignment2/ex2.json
   Plot timing results. Provide your timing/plotting code as ex2.2.py. **(2 pts)**
3. Compare the timing results with the result of the complexity analysis. Is the result consistent? Why? **(2 pts)**
4. Change the code – if possible – to improve its performance on the input given in point 2. If possible, provide your code as ex2.4.py and plot the improved results. If not possible, explain why. **(2 pts)**
5. Alter the inputs given in point 2 – if possible - to improve the performance of the code given in the text of the question. The new inputs should contain all the elements of the old inputs, and nothing more. Plot the results and provide the new inputs as ex2.5.json). If not possible, explain why. **(2 pts)**

Exercise 3

In the Lecture, we have discussed multiple search algorithms, building up to what is known as the **Interpolation Search**

Interpolation search is a variant/enhancement on the Binary search for multiple reasons, and is implemented usually as:

```
def interpolation_search(arr, x):
    low = 0
    high = len(arr) - 1
    while low <= high and x >= arr[low] and x <= arr[high]:
        pos = low + int(((float(high - low) / ( arr[high] - arr[low])) * ( x - arr[low])))
        if arr[pos] == x:
            return pos
        if arr[pos] < x:
            low = pos + 1
        else:
            high = pos - 1
    return -1
```

1. What are some of the key aspects that makes Interpolation search better than Binary search (mention at least 2) **(2 pts)**
2. An underlying assumption of the interpolation search, is that sorted data are uniformly distributed. What happens if the data follows a different distribution (something like normal? Will the performance be affected? Discuss why (whether yes or no) **(3 pts)**
3. If we want to modify the Interpolation Search to follow a different distribution, Which part of the code will be affected? **(2 pts)**
4. When comparing: linear, Binary, and Interpolation Sort
    a. When is Linear Search your only option for searching your data as Binary and Interpolation might fail? **(1 pt)**
    b. What is a case that Linear search will outperform both Interpolation and Binary search, and Why? Is there a way to improve Binary and Interpolation Sort to resolve this issue? **(2 pts)**

Exercise 4

In the lecture recordings, we discussed some of the main differences between arrays (or lists in python) and Linked Lists.

1. Compare advantages and disadvantages of arrays vs linked list (complexity of task completion) **(2 pts)**
2. For arrays, we are interested in implementing a replace function that acts as a deletion followed by insertion. How can this function be implemented to minimize the impact of each of the standalone tasks? **(3 pts)**
3. Assuming you are tasked to implement a Singly Linked List with a sort function, given the list of sort functions below, state the feasibility of using each one of them and elaborate why is it possible or not to use them. Also show the expected complexity for each and how it differs from applying it to a regular array: **(1 pt each)**
    a. Selection sort
    b. Insertion Sort
    c. Merge Sort
    d. Bubble sort
    e. Quick Sort

Exercise 5

Stacks and Queues are a special form of linked lists with some modifications that makes operation better. For parts 1 and 2 assume we are using a ***singly linked list***

1. In stacks, insertion (push) adds the newly inserted data at head
    a. why? **(0.5 pts)**
    b. Can we insert data at the end of the linked list? **(0.5 pts)**
    c. If yes, then what is the difference in operation time (if any) for pushing and popping data from the stack? **(1 pt)**
2. In Queues, we added a new pointer that points to the tail of the linked list
    a. why? **(0.5 pts)**
    b. Can we implement the Queue without the tail? **(0.5 pts)**
    c. If yes, then what is the difference in operation time (if any) for enqueuing and dequeuing data from the stack? **(1 pt)**
    d. Can we change the behavior of the enqueue and dequeue where we enqueue at head and dequeue at tai? Do you think it is a good idea? **(1 pt)**
3. Revisit your answers for part 1 and 2 but now with the assumption that we are using ***circular doubly linked list*** **(5 pts)**