Homework 4

Due Wednesday, April 13, 2022 11:59 PM EST

Tomas Hornicek, no collaborators, no sources, no extension, 0 late days

1 Without Coding

Problem 1 (15 points)

Give the Big O run-time of the following algorithms. Explain/show your work.

Binary Search:

```
def binary_search(arr, low, high, x):
    # Check base case if low >
    high:
        return None
    else:
        mid = (high + low) // 2 element = arr[mid]

        if element == x: return mid
        elif element > x:
            return binary_search(arr, low, mid - 1, x) else: return
binary_search(arr, mid + 1, high, x)
```

Answer:

Time complexity: $O(\log_2(n)*1*1) = : O(\log_2(n))$

Reasoning: The Binary search is recursively calling itself on one half of the input. Therefore it is decreasing exponentially, which means the run time is $\log_2(n)$.

Selection Sort:

```
def selection_sort(arr):
    for i in range(len(arr)): smallest_index = i
        smallest_value = arr[i]

        # Find smallest element for j in range(i,
        len(arr)):
            candidate = arr[j] if candidate <
            smallest_value: smallest_index = j
            smallest_value = candidate

        # Swap front with smallest value temp = arr[i]
    arr[i] = smallest_value arr[smallest_index] = temp
    return arr
```

Answer:

Time complexity: $O(n*n*1) = O(n^2)$

Reasoning: The Selection sort iterates over the first for loop, giving us n. Then it iterates over the second nested for loop which gives us the run time of n. Because the for loop is nested, we multiply the two n's to give us $O(n^2)$.

Insertion Sort:

```
def insertion_sort(arr):

    for i in range(len(arr)): current_element = arr[i] # Find correct place in
        list j = 0 while j < i and arr[j] <= current_element: j += 1

        # Move all larger elements over 1 for x in range(i, j-
        1, -1): arr[x] = arr[x-1]

        # Insert current element arr[j] =
        current_element
```

Answer:

Time complexity: $O(n*(2n)) = O(n^2)$

Reasoning: The Insertion sort first iterates though a for loop, giving us the runtime of n. The first for loop has nested for and while loops, which give us together the runtime of 2n. Because the second while and second for loop is nested it multiplies with the initial for loop, giving us a total running time of $O(n^2)$

Problem 2 (10 points)

Simplify the following Big O expressions. Show work for partial credit.

1. $O(2*N + Log(N) + N^3 + 10) = O(n + \log(n) + n^3) = O(n^3)$
2. $O(1000) = O(1)$
3. $O(10 + 20*N + 10*N*Log(N) + 10*N^2) = O(n + n\log(n) + n^2) = O(n^2)$
4. $O(N*Log(N) + 20*N + N*Log(N)^2) = O(n\log(n) + n + 2n\log(n) = O(n\log(n))$
5. $O(2*N^2*Log(N)) = O(n^2\log(n))$

Problem 3 (15 points)

What are the 4 tenants of Object-Oriented Programming with brief explanations (<2 sentences) of each?

1. Abstraction: Using an interface or class, without knowing the implementation of the interface or class. This leads to reusability in code and for the developer to change code without affecting the client.
2. Data Encapsulation: Encapsulating both the data and the operations on the data into a single unit(class). This leads to code reusability and less errors in code.
3. Inheritance: Inheriting methods and variables from a parent class. This leads to reuse the methods in the base class as well as the capability to add additional functionality in the derived class.
4. Polymorphism: Inheriting methods and variables from a parent class and overwriting/overriding them. This means that you can use one method for different instances, which means we can use the same interface to perform variants of operations in sub classes.