CS566HW2-2024

September 20, 2024

0.1 Lab 2. Introduction to algorithms

This is second homework for CS 566.

0.2 Task 1. Solve the problem "Fibbonaci Number" from https://leetcode.com/problems/fibonacci-number/description/ using Python3

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

[1]:
```
class Solution:
    def fib(self, n: int) -> int:
        # implement Working Algorithm
        if n == 0:
          return 0
        elif n == 1:
          return 1
        return self.fib(n-2) + self.fib(n-1)
```

0.2.1 Do not modify the testing code below. If you get message "Mistake in test case #", it means that you algorithm is incorrect.

[2]:
```
#test_case_1
expected, nums = 1, 2
actual = Solution().fib(nums)
assert expected==actual, "Mistake in test case 1"

#test_case_2
expected, nums = 2, 3
actual = Solution().fib(nums)
assert expected==actual, "Mistake in test case 2"

#test_case_3
expected, nums = 3, 4
actual = Solution().fib(nums)
assert expected==actual, "Mistake in test case 2"
```

0.2.2 Write analysis of the Memory Complexity and Time Complexity using Aymptotic Notation O. (1 point)

Memory Analysis: O - O(n)

Time Analysis: O - O(2^n)

0.3 Task 2. Implement Selection Sort (4 points)

[3]:
```
# Selection Sort
# Please implement the function which takes as an input list of numbers and␣
 ↪returns the sorted list
# Use Selection Sort algorithm
def selection_sort(nums):
    # implement Working Algorithm
    n = len(nums)
    for i in range(n):
      min_idx = i
      # find smallest item from unsorted array
      for j in range(i+1,n):
        if nums[j] < nums[min_idx]:
          min_idx = j
      # move to sorted array
      if min_idx != i:
        # swap
        nums[i], nums[min_idx] = nums[min_idx], nums[i]
    return nums
```

[4]:
```
# testing code
# do NOT modify testing code below
def test_sorting(sorting_algorithm):
    nums = [5,4,3,2,10]
    expected = [2,3,4,5,10]
    actual = sorting_algorithm(nums)
    assert expected == actual, "Mistake in implementation"
```

[5]:
```
# if your implementation is correct, you should see "OK"
# if your implementation is not correct, you will see "Mistake in␣
 ↪implementation"
test_sorting(selection_sort)
print('OK')
```

```
OK
```

0.3.1 Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O - O(1)

Time Analysis: O - O(n^2)

Invariant:

The invariants are that after each pass through the array, the left portion of the array contains the smallest elements sorted in non-decreasing order. At the end of each iteration, the next smallest element is placed in its correct position in the sorted portion, ensuring that the sorted section grows while the unsorted section shrinks until the entire array is sorted.

0.3.2 Task 3. Implement Bubble Sort (4 points)

[6]:
```
# Bubble Sort
# Please implement the function which takes as an input list of numbers and␣
↪returns the sorted list
# Use Bubble Sort algorithm
def bubble_sort(nums):
    # implement Working Algorithm
    n = len(nums)
    for i in range(n):
      # last i elements are in place so skip those
      for j in range(0, n-i-1):
        # compares adjacent elements, swap if out of order
        if nums[j]>nums[j+1]:
          nums[j], nums[j+1] = nums[j+1], nums[j]

    return nums
```

[7]:
```
# if your implementation is correct, you should see "OK"
# if your implementation is not correct, you will see "Mistake in␣
↪implementation"
test_sorting(bubble_sort)
print('OK')
```

```
OK
```

0.3.3 Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O - O(1)

Time Analysis: O - O(n^2)

Invariant:

After each complete pass through the array, the largest unsorted element is placed in its correct position at the end of the array. The elements in the sorted portion (the rightmost part of the array) maintain their relative order. This ensures that the algorithm progressively moves unsorted elements into their correct positions