Name: M.Rohith
3122 21 5001 085

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam - 603 110
(An Autonomous Institution, Affiliated to Anna University, Chennai)

UCS2403: DESIGN & ANALYSIS OF ALGORITHMS

Assignment 4

1. Use the tool Hypothesis to generate counterexamples to show that the output of the buggy code is indeed wrong, by comparing it against your correct code.

   (a) Counting Inversions

      i. Consider the Python codes given in (1) and (2) below for finding the count of inversions in a list.

```
(1) def count_inversions1(nums):
        count = 0
        for i in range(1, len(nums)):
            if nums[i] < nums[i - 1]:
                count += 1
        return count


(2) def count_inversions2(nums):
        nums.sort()
        count = 0
        for i in range(1, len(nums)):
            if nums[i] < nums[i - 1]:
                count += 1
        return count
```

   (b) Comparison count sort

```
def comparison_count_sort(nums):
    count = [0] * len(nums)
    nums_sorted = [0] * len(nums)
    for i in range(len(nums) - 1):
        for j in range(i + 1, len(nums)):
            if nums[i] > nums[j]:
                count[i] += 1
            elif nums[i] < nums[j]:
                count[j] += 1
    for i in range(len(nums)):
```

Name: M.Rohith
3122 21 5001 085

```
            nums_sorted[count[i]] = nums[i]
    return nums_sorted
```

2. (a) Using the technique of divide-and-conquer, write a recursive program to find the maximum value in a given (unsorted) list of numbers.

   (b) Write the recurrence relation to find the time complexity of the algorithm. Find a closed form expression for the time complexity.

Ex no: 4
Date: 23-03-2023

Name: M.Rohith
3122 21 5001 085

1. a) Program code:

```
from hypothesis import given
from hypothesis.strategies import lists, integers

# correctcode
def count_inversions(nums):
    count = 0
    n = len(nums)
    for i in range(n):
        for j in range(i+1, n):
            if nums[i] > nums[j]:
                count += 1
    return count

# incorrectcode
def count_inversions1(nums):
    count = 0
    for i in range(1, len(nums)):
        if nums[i] < nums[i - 1]:
            count += 1
    return count

# hypo test
@given(lists(integers()))
def test_inversions(nums):
    assert count_inversions(nums) == count_inversions1(nums)

test_inversions()
```

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4> python 1a.py
Traceback (most recent call last):
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1a.py", line 30, in <module>
    test_inversions()
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1a.py", line 27, in test_inver
sions
    def test_inversions(nums):
  File "C:\Python310\lib\site-packages\hypothesis\core.py", line 1396, in wrapped_test
    raise the_error_hypothesis_found
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1a.py", line 28, in test_inver
sions
    assert count_inversions(nums) == count_inversions1(nums)
AssertionError
Falsifying example: test_inversions(
    nums=[0, 0, -1],
)
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4>
```

Ex no: 4
Date: 23-03-2023

Name: M.Rohith
3122 21 5001 085

Program code:

```
from hypothesis import given
from hypothesis.strategies import lists, integers

# correctcode

def count_inversions(nums):
    count = 0
    n = len(nums)
    for i in range(n):
        for j in range(i+1, n):
            if nums[i] > nums[j]:
                count += 1
    return count

#incorrect code

def count_inversions2(nums):
    nums.sort()
    count = 0
    for i in range(1, len(nums)):
        if nums[i] < nums[i - 1]:
            count += 1

    return count

# hypo test

@given(lists(integers()))
def test_inversions(nums):
    assert count_inversions(nums) == count_inversions2(nums)

test_inversions()
```

Ex no: 4
Date: 23-03-2023

Name: M.Rohith
3122 21 5001 085

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4> python 1b.py
Traceback (most recent call last):
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1b.py", line 32, in <module>
    test_inversions()
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1b.py", line 29, in test_inver
sions
    def test_inversions(nums):
  File "C:\Python310\lib\site-packages\hypothesis\core.py", line 1396, in wrapped_test
    raise the_error_hypothesis_found
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1b.py", line 30, in test_inver
sions
    assert count_inversions(nums) == count_inversions2(nums)
AssertionError
Falsifying example: test_inversions(
    nums=[1, 0],
)
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4>
```

b) Program code:

```python
from hypothesis import given
from hypothesis.strategies import lists, integers

#incorrect code

def comparison_count_sort_inc(nums):
    count = [0] * len(nums)
    nums_sorted = [0] * len(nums)
    for i in range(len(nums) - 1):
        for j in range(i + 1, len(nums)):
            if nums[i] > nums[j]:
                count[i] += 1
            elif nums[i] < nums[j]:
                count[j] += 1
    for i in range(len(nums)):
        nums_sorted[count[i]] = nums[i]
    return nums_sorted

#correct code

def comparison_count_sort_c(nums):
    count = [0] * len(nums)
    nums_sorted = [0] * len(nums)
    for i in range(len(nums) - 1):
```

Name: M.Rohith
3122 21 5001 085

```
        for j in range(i + 1, len(nums)):
            if nums[i] >= nums[j]:
                count[i] += 1
            elif nums[i] < nums[j]:
                count[j] += 1
    for i in range(len(nums)):
        nums_sorted[count[i]] = nums[i]
    return nums_sorted


@given(lists(integers()))
def countsort_tester(nums):
    assert comparison_count_sort_inc(nums) ==
comparison_count_sort_c(nums)

countsort_tester()
```

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4> python 1c.py
Traceback (most recent call last):
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1c.py", line 39, in <module>
    countsort_tester()
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1c.py", line 36, in countsort_
tester
    def countsort_tester(nums):
  File "C:\Python310\lib\site-packages\hypothesis\core.py", line 1396, in wrapped_test
    raise the_error_hypothesis_found
  File "C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4\1c.py", line 37, in countsort_
tester
    assert comparison_count_sort_inc(nums) == comparison_count_sort_c(nums)
AssertionError
Falsifying example: countsort_tester(
    nums=[1, 1],
)
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4>
```

Name: M.Rohith
3122 21 5001 085

2. a) Program code:

```
def maximum(low,high,arr):
    if low==high:
        return arr[low]
    else:
        mid=(low+high)//2
        leftmax=maximum(low,mid,arr)
        rightmax=maximum(mid+1,high,arr)
        return max(leftmax,rightmax)

l=[39,38,69,72,67,70]
max_val=maximum(0,len(l)-1,l)
print("List: ",l)
print("Maximum value in the list: ",max_val)
```

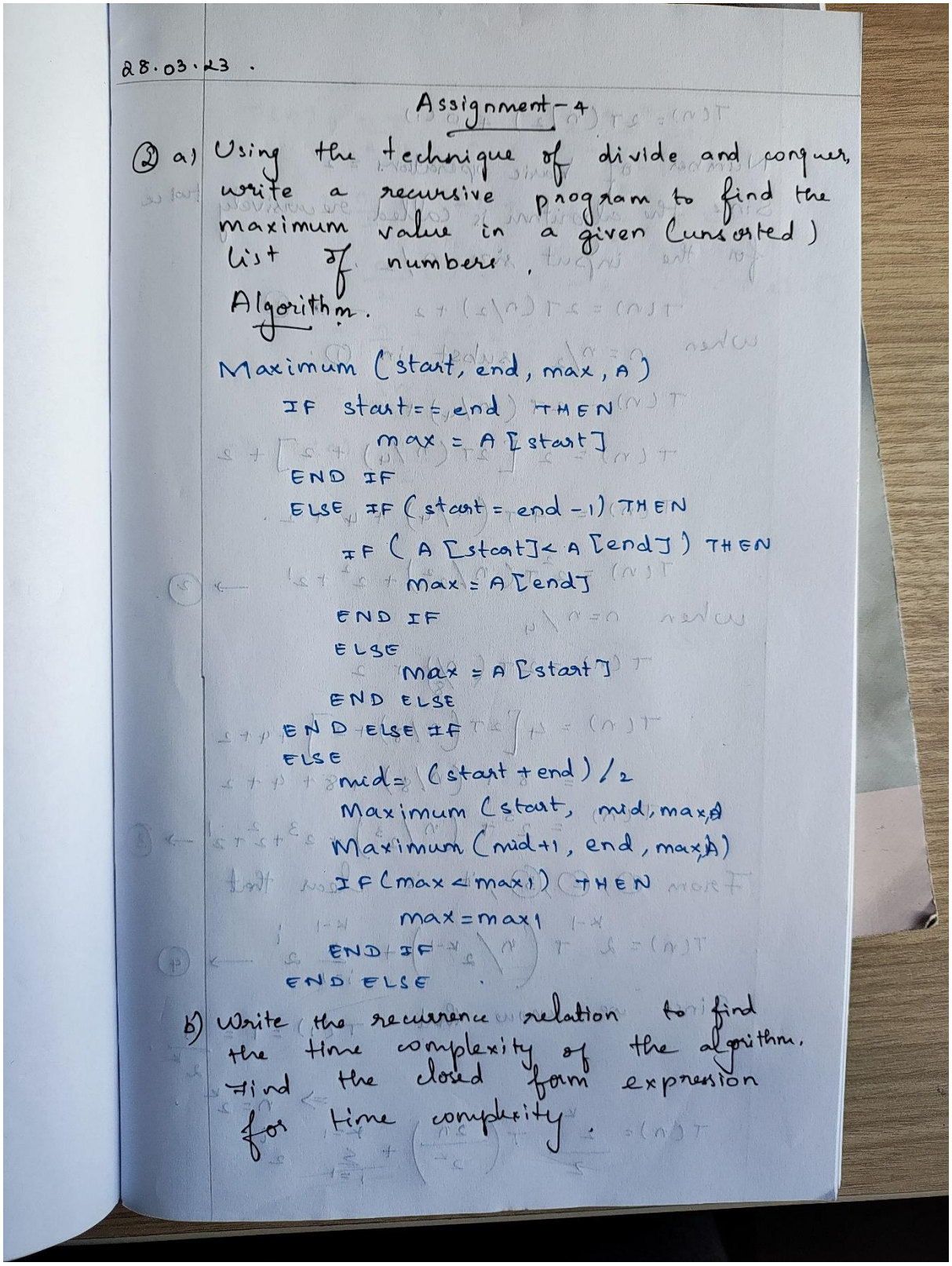
Output:

```
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4> python 2a.py
List:  [39, 38, 69, 72, 67, 70]
Maximum value in the list:  72
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-4>
```

Ex no: 4
Date: 23-03-2023

Name: M.Rohith
3122 21 5001 085

Algorithm:



28.03.23.

Assignment-4

(2) a) Using the technique of divide and conquer, write a recursive program to find the maximum value in a given (unsorted) list of numbers.

Algorithm.

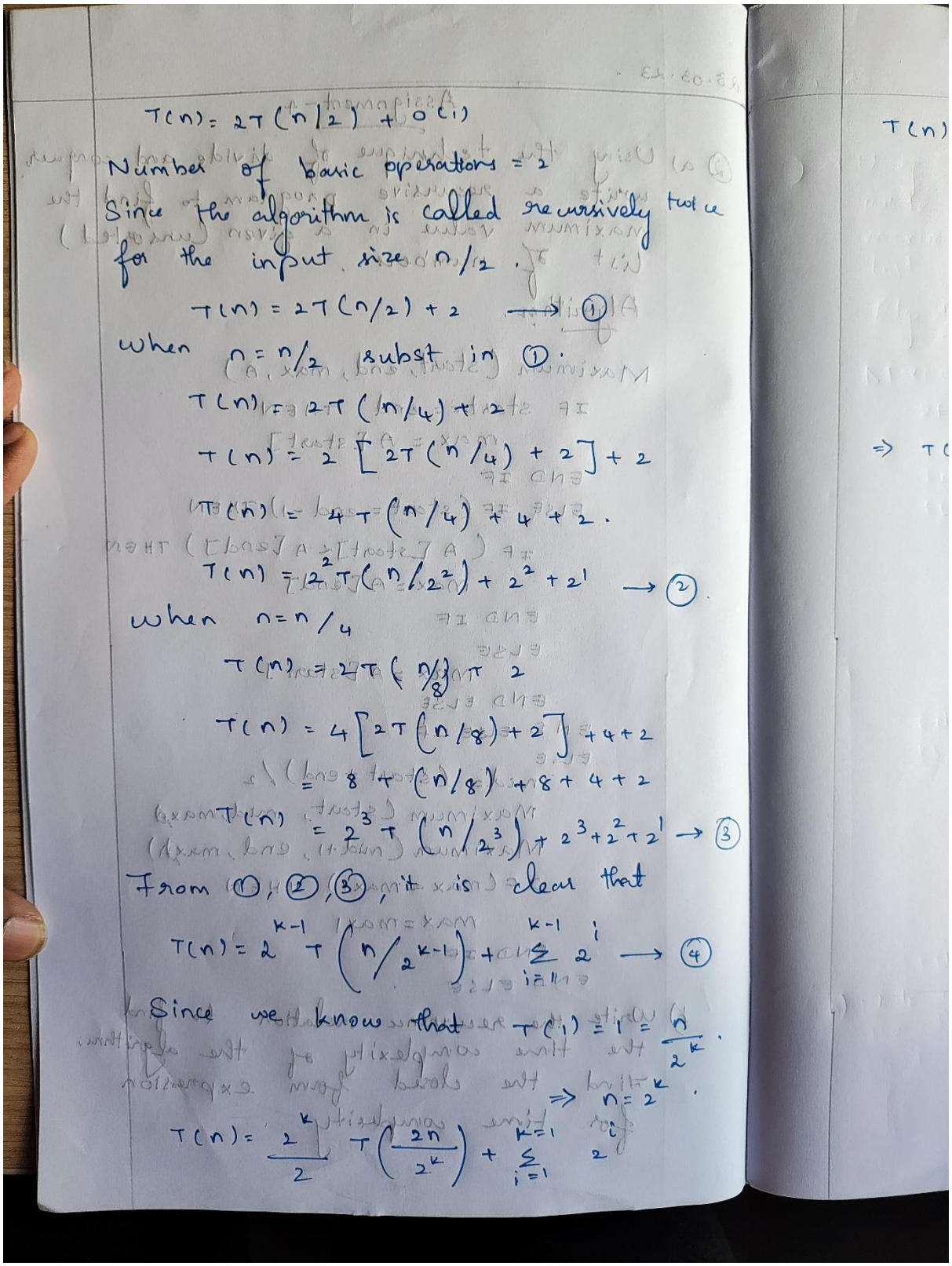
```
Maximum (start, end, max, A)
    IF start = = end THEN
        max = A [start]
    END IF
    ELSE IF (start = end - 1) THEN
        IF ( A [start] < A [end] ) THEN
            max = A [end]
        END IF
        ELSE
            Max = A [start]
        END ELSE
    END ELSE IF
    ELSE
        mid = (start + end) / 2
        Maximum (start, mid, max, A)
        Maximum (mid+1, end, max1, A)
        IF (max < max1) THEN
            max = max1
        END IF
    END ELSE
```

b) Write the recurrence relation to find the time complexity of the algorithm. Find the closed form expression for time complexity.

Ex no: 4
Date: 23-03-2023

Name: M.Rohith
3122 21 5001 085

b) Time complexity:



$T(n) = 2T(n/2) + O(1)$

Number of basic operations = 2

Since the algorithm is called recursively twice for the input size $n/2$.

$T(n) = 2T(n/2) + 2 \longrightarrow (1)$

when $n = n/2$, subst in (1).

$T(n) = 2T(n/4) + 2$

$T(n) = 2\left[2T(n/4) + 2\right] + 2$

$T(n) = 4T(n/4) + 4 + 2.$

$T(n) = 2^2 T(n/2^2) + 2^2 + 2^1 \longrightarrow (2).$

when $n = n/4$

$T(n) = 2T(n/8) + 2$

$T(n) = 4\left[2T(n/8) + 2\right] + 4 + 2$

$= 8T(n/8) + 8 + 4 + 2$

$T(n) = 2^3 T(n/2^3) + 2^3 + 2^2 + 2^1 \longrightarrow (3)$

From (1), (2), (3), it is clear that

$$T(n) = 2^{k-1} T\left(n/2^{k-1}\right) + \sum_{i=1}^{k-1} 2^i \longrightarrow (4)$$

Since we know that $T(1) = 1 = \frac{n}{2^k}$.

$\Rightarrow n = 2^k$.

$$T(n) = \frac{2^k}{2} T\left(\frac{2n}{2^k}\right) + \sum_{i=1}^{k-1} 2^i$$

Ex no: 4
Date: 23-03-2023

Name: M.Rohith
3122 21 5001 085

$$T(n) = \frac{n}{2} T\left(\frac{2n}{n}\right) + 2\frac{(2^{k-1}-1)}{2-1}$$

$$= \frac{n}{2} T(2) + 2\left(\frac{2^k}{2}\right) - 2$$

$$= \frac{n}{2}(1) + 2^k - 2 \qquad [\because T(2) = 1]$$

$$= \frac{n}{2} + n - 2$$

$$= \frac{3n}{2} - 2$$

$$\Rightarrow T(n) = \frac{3n}{2} - 2.$$