CS-UY 1134 - Spring 2017 1134 Midterm 1

Justin Huang

TOTAL POINTS

25 / 100

QUESTION 1

1 Question 1 0 / 10

- 0 Correct
- **10 Wrong**
- 4 Doesn't show that c and n0 satisfy the inequality for all n>n0
- 2 Not accurate proof

QUESTION 2

2 Question 2 7 / 10

- 0 Correct..... lst1 = [[20, 30] , [4, 5]] ..... lst2 = [[20, 30] , [40, 50]]
- 2 lst1 [0] wrong
- 3 lst1 [1] wrong
- **3 lst2 [0] wrong**
- 2 lst2 [1] wrong
- 1 brackets wrong

QUESTION 3

3 Question 3 8 / 16

- 0 Correct. O(n^2) and O(n)
- **4 runtime wrong for part 1**
- **4 wrong explanation for part 1**
- 2 not good enough explanation for part 1
- 4 runtime wrong for part 2
- 4 wrong explanation for part 2
- 2 not good enough explanation for part 2

QUESTION 4

4 Question 4 5 / 14

- 0 Correct
- **2 Part 1 recursion tree wrong (should not have two branches)**
- **3 Part 1 runtime wrong**
- **2 Part 1 calculations wrong**
- **2 Part 2 recursion tree wrong (should not have two branches)**
- 3 Part 2 runtime wrong
- 2 Part 2 calculations wrong

QUESTION 5

5 Question 5 5 / 30

- 0 Correct
- **15 Not in-place solution; function should NOT return. Do not make a new array to return.**
- 10 Not linear runtime.
- 0 should be isinstance(elem, str), not isinstance(elem, string)
- **10 You cannot use other data structures other than ArrayStack and the passed-in list.**
- 15 Illegible
- 2 Invalid Python syntax
- 12 Resulting list is incorrect
- 2 Function should not have a return statement.
- 0 No need to rewrite the ArrayStack class. It's already given to you.
- 30 Wrong.
- 5 Too many unnecessary operations, clunky solution.
- 0 Correct, although using an ArrayStack would have been a cleaner solution.
- 30 No answer
- 30 wrong algorithm

QUESTION 6

6 Question 6 0 / 20

- 0 Correct
- **20 Wrong**
- 4 Not linear time
- 16 Recursive step is wrong

- **3** Not accurate
- **6** Does not accumulate the list of indices
- **15** Use of a global list is not allowed (if you call this function more than once the list would already have items from previous calls)
- **2** Returned list contains the elements not the indices
- **4** Wrong base
- **4** Indices in returned list in decreasing order

Name: Justin Huang Net ID: jh5387

Question 1 (10 points)
Consider the following definition of Big-O (same as was given in class):

Definition: Let $f(n)$ and $g(n)$ be two functions mapping positive integers to positive real numbers. We say that $f(n) = O(g(n))$ if there exists a positive real constant c, and a positive integer constant n_0, such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

Prove by definition that: $3n^2 + 2n - 3 = O(n^2)$

Note: Remember that you must come with constants c and n_0 in your answer.

Proof:

The running time is always the highest exponent because as n grows larger the other values don't affect running time anymore.

$C \cdot n_0^2 > C \cdot n_0 - C$

Question 2 (10 points)
What is printed when the following Python code is executed?

```
lst1 = [[2, 3], [4, 5]]
lst2 = lst1.copy()

lst1[0][0] = 20
lst1[0][1] = 30

lst2[1] = [10*k for k in lst2[1]]

print("lst1 = ", lst1)
print("lst2 = ", lst2)
```

[[20, 3], [4, 5]]
[[20, 30], [4, 5]]

Output:

lst1 = [[20, 30], [4, 5]]

lst2 = [[2, 3], [40, 50]]

Name: Justin Huang Net ID: jh5387

Question 3 (16 points)
Consider the following functions definitions:

```
def list_of_chars_to_string(lst):
    s_res = "" Θ(1)
    for char in lst: Θ(n)
        s_res = s_res + char Θ(1)
    return s_res Θ(1)

def string_to_list_of_chars(s):
    l_res = [] Θ(1)
    for char in s: Θ(n)
        l_res.append(char) Θ(1)
    return l_res Θ(1)
```

Notes:

- The function `list_of_chars_to_string(lst)` gets a list whose elements are single characters, and returns the string with those characters.
  For example, the call: `list_of_chars_to_string(['a', 'b', 'c'])`, would return the string `'abc'`.
- The function `string_to_list_of_chars(s)` gets a string, and returns a list whose elements are the characters of the input string.
  For example, the call: `string_to_list_of_chars('abc')`, would return the list `['a', 'b', 'c']`.

1) Let `lst` be a list of n characters. What is the running time of `list_of_chars_to_string(lst)`? Give a short explanation of your answer.

$T(n) = \theta($ n $)$

The for loop has to iterate over n elements, and takes the most time. Every other line costs only Θ(1) to run and thus doesn't affect the total running time.

2) Let `s` be a string of n characters. What is the running time of `string_to_list_of_chars(s)`? Give a short explanation of your answer.

$T(n) = \theta($ n $)$

This is also Θ(n) because the for loop costs Θ(n) time to run. All the other lines only cost Θ(1) and doesn't affect the total running time.

Name: Justin Huang Net ID: jh5387

Question 4 (14 points)

You are given two recursive implementations for the binary search algorithm, that searches for a given `elem` value in a sorted sequence (These functions return true if `elem` is one of the elements of the sequence, or false if it is not):

```
def binary_search1(srt_lst, elem):
    if(len(srt_lst) == 1):
        return (srt_lst[0] == elem)
    else:
        mid_ind = len(srt_lst) // 2
        if(srt_lst[mid_ind] == elem):
            return True
        elif(elem < srt_lst[mid_ind]):
            return binary_search1(srt_lst[ : mid_ind], elem)
        else: # elem > srt_lst[mid_ind]
            return binary_search1(srt_lst[mid_ind+1 : ], elem)
```

O(log(n))

```
def binary_search2(srt_lst, low, high, elem):
    if (low == high):
        return (srt_lst[low] == elem)
    else:
        mid_ind = (low + high) // 2
        if (srt_lst[mid_ind] == elem):
            return True
        elif (elem < srt_lst[mid_ind]):
            return binary_search2(srt_lst, low, mid_ind-1, elem)
        else:  # elem > srt_lst[mid_ind]
            return binary_search2(srt_lst, mid_ind+1, high, elem)
```

O(log n)

Note: The implementations differ in the parameters we pass to these functions:

- In the first version, we pass only the list (all the elements in the list should be taken in to account for the search).
- In the second version, in addition to the list, we pass two indices: `low` and `high` (`low` ≤ `high`), which indicate the range of indices of the elements that should to be considered.
  The initial values (for the first call) passed to `low` and `high` would represent the range of the entire list.

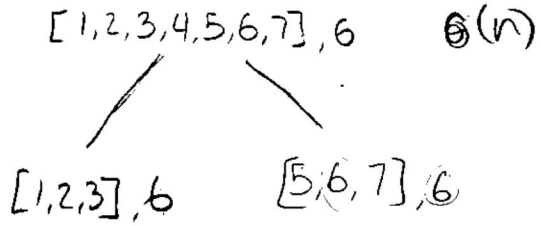
Analyze the running time of the implementations above. For each version:

1) Draw the recursion tree that represents the execution process of the function, and the cost of each call.
2) Conclude the total (asymptotic) running time of the function.

Name: Justin Huang Net ID: jh5387

a) For analyzing `binary_search1`, let `srt_lst` be a list of n elements.

i. Draw the **recursion tree** for `binary_search1(srt_lst, elem)`



ii. **Conclude the running time** of `binary_search1(srt_lst, elem)`

$T(n) = \theta($ log (n) $)$

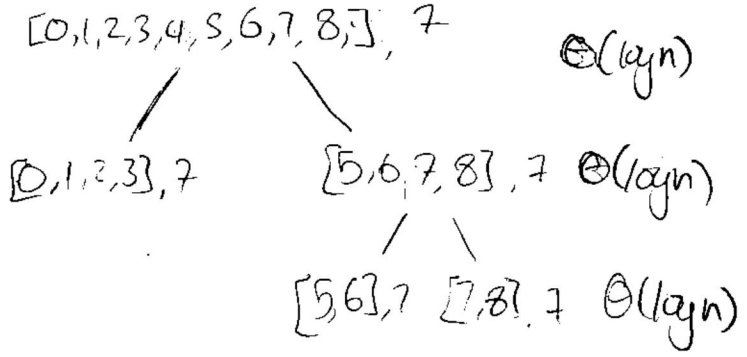
Calculations:

len (srt_lst)//2 = Θ (log (n))

Name: Justin Huang Net ID: jh5387

b) For analyzing `binary_search2`, let `srt_lst` be a list of n elements.

i. Draw the **recursion tree** for `binary_search2(srt_lst, 0, n-1, elem)`



ii. **Conclude the running time** of `binary_search2(srt_lst, 0, n-1, elem)`

$T(n) = \theta($ log(n) $)$

Calculations:

(low + high) // 2 == Θ(log(n))

Name: Justin Huang Net ID: jh5387

Question 5 (30 points)

Implement the function: **def** `words_keep_numbers_flip(lst)`
This function gets a list `lst` that contains words (of type `str`), and numbers (of type `int`).
When called, it should **reorder** (in-place) the elements of `lst` so that:

1. All the words come before all the numbers
2. The words will keep their original relative order
3. The numbers will be placed in a reversed order (relative to their original order).

For example, if `lst = ["Keep", 3, "this", "order", 2, 1]`, after calling `words_keep_numbers_flip(lst)`, `lst` will be:
`["Keep", "this", "order", 1, 2, 3]`

Implementation requirements:

1. Your function should run in **linear time**. That is, if there are n items in `lst`, calling `words_keep_numbers_flip(lst)` will run in $\theta(n)$.

2. For the memory used, in addition to `lst`, you **can use one `ArrayStack`** object, which has methods `push`, `pop`, `top`, `is_empty` and `__len__` as described in class. In addition to that, you are allowed to use only $\theta(1)$ memory.

Note: Write your implementation on the next page.

Name: Justin Huang Net ID: jh5337

```
def words_keep_numbers_flip(lst):
    full_lst = []
    num_lst = []
    for item in (lst):
        if item is str:
            full_lst.appord(item)
        else:
            num_lst.apprnd (item)
    for value in num_lst:
        full_lst.append(num_lst.pop())
    return full_lst
```

Name: Justin Huang Net ID: jh5387

Name: Justin Huang Net ID: jh5387

Question 6 (20 points)

Give a **recursive** implementation for the function:
`def find_all(lst, low, high, elem)`

The function is given:
- a list `lst`, to search in
- two indices: `low` and `high` (`low` ≤ `high`), which indicate the range of indices that need to be considered
- an element `elem`, to search for

The function should create and return a list, containing all the indices in the range `low`, `low+1`, ..., `high`, where `elem` appears.
The indices should come **in an ascending order**.

For example, if `lst = [7, 3, 2, 4, 2, 1, 1, 2, 5, 0]`, the call `find_all(lst, 3, 8, 2)`, will return: `[4, 7]`.

Implementation requirements:
Your function should run in **linear time**. That is, if n is the size of the range `low`, `low+1`, ..., `high`, calling `find_all(lst, low, high, elem)` will run in $\theta(n)$.

Note: Write your implementation on the next page.

Name: Justin Huang Net ID: jh5387

```
def find_all(lst, low, high, elem):
    count_lst = []
    if lst[high] = elem:
        return lst[high]
    else:
        rest_lst = find_all(lst, low, high, elem)
        for item in rest_lst:
            if lst[low] == elem:
                count_lst.append(low)
                low t= 1
        return count_lst
```

not a smaller instance

Name: Justin Huang Net ID: jh5387.

EXTRA PAGE IF NEEDED

Note question numbers of any questions or part of questions that you are answering here.
Also, write "ANSWER IS ON LAST PAGE" near the space provided for the answer.