CSC236 – Problem Set 6

There are two components of this problem set. The preliminary question is not marked or submitted: it is there as a suggested exercise that you should do early to make sure that you're on track. The problem set itself is what you will submit for marks.

Get in the habit of starting work early – the less time you give yourself, the more stressed you'll find yourself each week!

To avoid suspicions of plagiarism: at the beginning of your submission, **clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, that you consulted**.

The PDF file you submit must be typed, scanned handwritten submissions will not be marked.

Preliminary: Not Marked

This question is an opportunity for you to check your understanding of the topics and practice writing formal solutions. This is a valuable *learning opportunity* – if you see that you're at a loss, get help quickly!

For each of the following recurrences, do one of two things. If the master theorem applies, indicate which case applies and state the asymptotic runtime obtained using the master theorem. If the master theorem does not apply, briefly indicate why it does not apply and go no further.

A. $T(n) = 5T(2n/5) + n$

B. $T(n) = 4T(n/2) + 15n^3 + 4n^2 + n + 4$

C. $T(n) = T(n/2)$

Problem Set: due November 4, 2016 22:00, required filename: ps6sol.pdf

Answer each question completely, always justifying your claims and reasoning. Your solution will be graded not only on correctness, but also on clarity.

Answers that are technically correct that are hard to understand will not receive full marks. Mark values for each question are contained in the [square brackets].

You may work in groups of up to THREE to complete these questions.

1. [**6**] Here is a version of binary search that returns the index of the element if it is found, or `-1` if the element is not found.

```
def binary_search(A, x):
   '''
   Pre: list A is sorted in ascending order and has no duplicate elements
   Post: return the index of x in A if exists; return -1 otherwise.
   '''
   if len(A) == 0:
      return -1
   m = len(A)//2
   if A[m] == x:
      return m
   elif A[m] > x:
      return binary_search(A[0..m-1], x)
   else:
      result = binary_search(A[m+1..len(A)-1], x)
      if result == -1:
         return -1
      else:
         return result + m + 1
```

(a) Write the recurrence for the worst-case runtime $T(n)$ of the algorithm, and use the master theorem to find the asymptotic upper-bound on $T(n)$. State clearly which case of the master theorem applies.

(b) Prove that this algorithm is correct.

2. **[6]** Below is the `mystery` algorithm that we worked on in Problem Set 4.

```
1   def mystery(lst):
2       if len(lst) <= 1:
3           return
4       if lst[0] > lst[-1]:
5           lst[0], lst[-1] = lst[-1], lst[0]
6       if len(lst) >= 3:
7           split = len(lst) // 3
8           mystery(lst[0..len(lst) - split - 1])
9           mystery(lst[split..len(lst) - 1])
10          mystery(lst[0..len(lst) - split - 1])
```

We analyzed that the recurrence of the worst-case runtime of this algorithm is the following.

$$T(n) = \begin{cases} c, & \text{if } n \leq 2 \\ 3T(2n/3) + d, & \text{if } n \geq 3 \end{cases}$$

Some students have already realized that this algorithm is in fact a **sorting** algorithm. So in this problem set we will formalize our understanding of this interesting sorting algorithm.

(a) Find the asymptotic upper-bound on the worst-case runtime of `mystery` using the master theorem. State clearly which case of the master theorem applies.

(b) State the proper precondition and postcondition for the `mystery` function. Note: "proper" precondition means that it is **necessary** and **sufficient** for the algorithm to work correctly. In particular, don't add unnecessary conditions.

(c) Prove that `mystery` is correct according to the precondition and postcondition that you specified in (b). Note: Be careful when finding the possible program paths, and state clearly which lines of code are executed for each program path (use the line numbers).