Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Assignment 1 SOLUTION(3% - 12 points)

CSI2110/CSI2510 (Fall 2021)

Due: Thursday Sep 23, 11:59PM

Late assignment policy : ***1min-24hs late are accepted with 30%off; no assignments accepted after 24hs late.***

Question 1. **[6 points]**

Decide if each of the following statements is true or false and give a proof. For a true statement you need to identify the values for the constants c and n_0 as used in the definitions of big-O, Ω and Θ. For a false statement you need to justify why finding those constants is not possible.

a) $\log_{10}(n^2)$ is O($\log_2 n$) TRUE

$\log_{10}(n^2) = 2\log_2 n = 2\ \frac{\log_2 n}{\log_2 10} = \frac{2}{\log_2 10}\log_2 n \leq \frac{2}{\log_2 10}\log_2 n$, for all $n \geq 1$.

So $\log_{10}(n^2) \leq \frac{2}{\log_2 10}\log_2 n$, for all $n \geq 1$.

In this proof we used $c = \frac{2}{\log_2 10}$ and $n_0 = 1$.

Other constants are possible and can yield right answers; for example c=2 would work.

b) $n \times (10\,n^2 - 2\sqrt{n})$ is $\Omega(n^3)$ TRUE

$n \times (10\,n^2 - 2\sqrt{n}) = 10\,n^3 - 2\,n\sqrt{n}$

$\geq 10\,n^3 - n^3$, for all $n \geq 2$ (here we use $2\,n\sqrt{n} \leq n\,n\,n = n^3$)

$= 9n^3$, for all $n \geq 2$.

So $n \times (10\,n^2 - 2\sqrt{n}) \leq 9\,n^3$ for all $n \geq 2$.

In this proof we used $c = 9$ and $n_0 = 2$.

Other constants and different proofs are possible and can yield right answers.

c) $(n \sin n)^2 + 100$ is $\Theta(n^2)$ FALSE

We first show that $(n \sin n)^2 + 100$ is NOT $\Omega(n^2)$.

The proof is by contraction, so we assume $(n \sin n)^2 + 100$ is $\Omega(n^2)$.

Then there exists constans $c > 0$ and $n_0 \geq 1$ such that

$(n \sin n)^2 + 100 \geq c\,n^2$ for all $n \geq n_0$.

Take a number $\overline{n}$ *l*arge enough, say $\overline{n} \geq \max(n_0, 100/\sqrt{c})$ and such that $\overline{n}$ is multiple of π. In this case, $\sin \overline{n} = 0$. So

$$0 + 100 = (\overline{n} \sin \overline{n})^2 + 100$$
$$\geq c\, \overline{n}^2$$
$$\geq c \left(100/\sqrt{c}\right)^2 = 10000.$$

Thus reach a contradiction: $100 \geq 10000$.
This concludes the proof that $(n \sin n)^2 + 100$ is NOT $\Omega(n^2)$.
But is the function was $\Theta(n^2)$ it would be $\Omega(n^2)$, which is not the case. So we conclude $(n \sin n)^2 + 100$ is NOT $\Theta(n^2)$.
NOTE: $(n \sin n)^2 + 100$ is a function that always between 100 and $n^2 + 100$ and we cannot use a simple polynomial that yields both a big-O and Ω. We can prove that $(n \sin n)^2 + 100$ is $\Omega(1)$, and $(n \sin n)^2 + 100$ is $O(n^2)$ (try this), but can't get tighter polynomials.

Question 2. **[6 points]**

Suppose the only edit that can be done on strings is to replace 1 character. Given two strings having the same length `n`, the following function checks if they are one edit (or zero edit) away.

Examples:
marley, barley -> true
chip, chin -> true
lex, lox -> true
mule, maze -> false
aabb,abba -> false
abcdef, xbyzvw -> false

```
boolean oneEdit(String s1, String s2) {
// precondition s1.length() == s2.length()
  boolean foundDifference=false;
  n = s1.length();
  for (i=0; i< n; i++) {
      if (s1.charAt(i) != s2.charAt(i)){
          if (foundDifference) {
              return false;
          }
          foundDifference=true;
      }
  }
  return true;
}
```

(a) (4 points) Give the time complexity (also called running time) of this algorithm as a function of *n* using the big-O notation, for both the **worst case** and the **best case**. Give the simplest possible expression inside each big-Oh and the smallest possible function inside the big-O. Justify how you obtained the worst case and best case time complexities.

For the **worst case running time**, we note that the loop execute at most n times. All statements before and after the loop take constant time, i.e. O(1).
The loop executes at nost n times and the stuff inside the loop takes constant time. So in total the loop takes O(n). Counting all together O(1) plus O(n) we get that the worst case running time W(n) is O(n).
For the best case running time, we claim that for every n we can find some valid input for which the loop only runs 2 times. Indeed, consider for each n>=2 strings s1 and s2 with length n, and having the first two characters the same (e.g. s1="aabb….b" s2="aacc…c"); such inputs will have foundDifference=true when i=0 and end the loop and return false when i=1. So the best case running time is B(n)=c, a constant or in other words the bestcase running time B(n) is O(1).

(b) (2 points) Now, the definition of one edit has been modified, so that any number between 0 and 100 replacements can be done in a single edit. Modify the above function to detect whether no more than 100 replacements have been done on the string; continue assuming both strings have the same length *n*. Your algorithm must have the same **big-Os** as the given algorithm for the **worst case** and for the **best case** running times**;** justify why this is the case.

Note: All pairs of strings in the previous example would have the new algorithm return true. An example that returns false would have to have at least 101 unmatching characters.

You The idea here is to count the number of non matching positions and return false as soon as this becomes greater than 100. In this sense the previous problem is equivalent to using 1 instead of 100 in the program below.

```
boolean oneEditNew(String s1, String s2) {
// precondition s1.length() == s2.length()
  int countDifference=0;
  n = s1.length();
  for (i=0; i< n; i++) {
      if (s1.charAt(i) != s2.charAt(i)){
          if (countDifference>100) {
               return false;
          }
          countDifference++;
      }
  }
  return true;
}
```

Justification:

The worst case running time is the same as the previous, so we can use the same proof to show this is O(n).

The best case running time is different in that the loop can run at most 101 times (instead of 2), but this is still a constant, i.e. O(1).