Due: 2 August 2024

There are two problems listed below. The theory needed to solve each problem will be discussed at least a week before its due date, and often even earlier.

If you are stuck or confused by any of the problems, please post on Piazza. You may also ask your tutor or lecturer. You are allowed to discuss the problems with your peers and refer to online materials, but you are not allowed to share solutions or copy materials from any source. You may find the academic integrity rules at `https://academicintegrity.cs.auckland.ac.nz/`.

To get full marks you need to **show all working** unless a question explicitly states not to.

For each written question, you should submit via Canvas a PDF file containing your answer. A typed solution is preferred, but a scanned handwritten submission is acceptable if it is neatly written (if it's hard to read, it will not be marked). If typing the assignment, do the best you can with mathematical symbols. For exponents, write something like `2^n` if using plain text. Use LaTeX if you really want it to look good.

Answers to programming questions must be submitted via the automated marker system at `https://www.automarker.cs.auckland.ac.nz/student.php`.

Please try to submit your assignments no later than 5 min before the due time. Late submissions up to 3 days after the deadline will be accepted but will incur a 20% penalty. No further extensions will be granted.

Best of luck, and enjoy the problems!

Last edited: 2024/07/23 21:59:11 +12'00'

Problem 1 (16 marks). Calculate the number of elementary operations in the worst case for the following algorithms (justify your answer). You only need to count operations stated explicitly as "*C* elementary operations", and you may ignore arithmetic operations, comparisons, etc. You do not need to solve any recurrences. If you cannot compute an answer exactly, give reasonable upper and lower bounds instead.

```
Function roticeol(n):
    for i ← 1 to 2n do
        for j ← i to 3n do
            C elementary operations
```

```
Function umnesson(n):
    i ← 0
    f ← 1
    while f < n^n do
        i ← i + 1
        f ← f · i
        C elementary operations
```

```
a ← list of n elements
Function sifyinia(n):
    for i ← 1 to n/2 do
        a[2i − 1] ← 3 · i
        a[2i] ← i − 1
        C elementary operations
    for i ← 1 to n do
        if a[i] = n then
            return                    // Immediately ends the loop if it runs
        C elementary operations
```

```
Function horilexp(n):
    for i ← 1 to n − 1 do
        horilexp(i)
        C elementary operations
```

Problem 2 (4 marks). Comment on the following conversation with a LLM. Which parts of the answer are correct? Which are errors? How can the errors be fixed? Justify your answers. Remember to cite any external sources of information you may use and make sure these are trustworthy.

User How many comparisons does the following algorithm make in the average case? Count only comparisons involving an element of a. You may assume that each element of a is either 0 or 1, chosen uniformly.

```
Function FindOne(a):
  For i <- 1 to n:
    If a[i] = 1:
      Return i
  Return 0
```

LLM The given algorithm, `FindOne(a)`, searches for the first occurrence of the value `1` in the array `a`. It iterates through each element of the array and checks if the element is equal to `1`. If it finds a `1`, it returns the index of that element; otherwise, it returns `0`.

Let's analyze the average case:

1. The probability of `a[i]` being `1` is `0.5` (since each element is either `0` or `1`, chosen uniformly).
2. For each element `a[i]` where `a[i] = 1`, the expected number of comparisons is `(0.5 * i)`.

Now let's calculate the total expected number of comparisons:

- For `i = 1`, the expected number of comparisons involving `a[1]` (if it is `1`) is `0.5 * 1 = 0.5`.
- For `i = 2`, the expected number of comparisons involving `a[2]` (if it is `1`) is `0.5 * 2 = 1`.
- And so on, up to `i = n`.

Summing up all these expected comparisons and adding one final comparison for the case when no `1` is found, we get the total expected number of comparisons:

$$\text{Total expected comparisons} = \frac{1}{2} + \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 3 + \ldots + \frac{1}{2} \cdot n + 1$$

For example, if `n = 10`, the total expected comparisons are approximately 28.50.