SoftEng 284 The University of Auckland

Assignment 1

Due: Wednesday, August 4th, by 10pm. *Semester 1, 2021*

There are **four** problems listed below. To get full credit for this assignment you need to complete all of them!

If you are stuck or confused by any of the problems, feel free to post on Piazza! You can also contact the tutors and the lecturer. Note that sometimes we can't answer all your questions (we don't want to be giving away the answers) but we will help you as much as we can.

You must show your working to get full marks You should submit via Canvas a single PDF file containing the answers to the written questions. A scanned handwritten submission is acceptable if and only if it is clearly legible. Hard to read work may 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: https://www.automarker.cs.auckland.ac.nz/student.php.

Please try to submit your assignments no later than 5 min before the due time. Even though the time is a universal thing your watch and Canvas built-in clock could show the different time. It is quite possible that you might be on time on your watch and late on Canvas. To avoid these kind of situation submit the assignments at least 5 min before the due time.

Please note that late assignments cannot be marked under any circumstances. (With that said: if you find yourself unable to complete this assignment due to illness, injury, or personal/familial misfortune, please email Richard as soon as possible.)

Best of luck, and enjoy the problems!

1. Running times (10 marks).

   Suppose you are given 3 algorithms A_1, A_2 and A_3 solving the same problem. You know that in the worst case the running times are

   $$T_{A1}(n) = 500n, \qquad T_{A2}(n) = 10n \log_{10}(100n), \qquad T_{A3}(n) = 0.5n^2$$

   .

   You may assume the minimum input size is 10. Answer the following questions. You must justify your answers.

   (a) Which algorithm is the fastest for very large inputs? Which algorithm is the slowest for very large inputs?

   (b) For what problem sizes range is A_1 the best algorithm to use? Answer the same question for A_2 and A_3.

   (c) Suppose that we run each algorithm on a large problem instance of size n. Then we feed in an input of size $100n$ and re-run the algorithms on the new input. What would you expect to happen to the running times of each algorithm?

   (d) Suppose that we have limited computation resource that only allows us to do at most 10^9 elementary operations. And we wish to process as much input as we can. Which algorithm should we choose?

2. Asymptotic notations (10 marks).

   Answer each of the following questions.

   (a) Suppose $T(n) = 15n + 10\sqrt{n}$. Show that $T(n)$ is not $O(\sqrt{n})$ using the definition of Big-O only.

   (b) Suppose $T(n) = \sqrt{5}n^3 \log 15n + 0.3n^2 + \sqrt{15n}$. Show that $T(n)$ is both $\Theta(n^3 \log n)$ and $O(n^4)$ using the definition of Big-O only.

   (c) Suppose you have functions $f(n)$ and $g(n)$ such that $f(n)$ is $O(g(n))$. Determine whether we can conclude that $\log(f(n))$ must be $O(\log(g(n)))$? Justify if your answer is 'yes' or give a counter example (with explanation) if your answer is 'no'.

   (d) Suppose you have functions $f(n)$ and $g(n)$ such that $f(n)$ is $O(g(n))$. Determine whether we can conclude that $2^{f(n)}$ must be $O(2^{g(n)})$? Justify if your answer is 'yes' or give a counter example (with explanation) if your answer is 'no'.

3. Pseudo-code analysis (10 marks)

   (a) How many elementary operations does the following pseudo-code have? Here we are only considering the number of elementary operation e within the main loop-body (i.e. you can ignore all other operations such as additions, multiplications, condition checks, etc.) Show all working.

```
function Meh (positive integer n)
    for (int i = 1; i ≤ n; i = i * 2)
        for (int j = 0; j < n; j++)
            if i == j
                for (int k = 1; k < n^3; k+ = n)
                    ... constant number C of elementary operations e
                end for
            else
                ... constant number C of elementary operations e
            end if
        end for
    end for
```

   (b) Show that the number of elementary operations of the following pseudo-code is in order of $\Theta(n \log n)$. Hint: you may find the approximation formula $n! \approx \sqrt{2\pi n}e^{-n}n^n$ useful.

```
function Huh (positive integer n)
    for (int i = 1; i ≤ n; i++)
        for (int j = 1; j < i; j = j * 2)
            ... constant number C of elementary operations e
        end for
    end for
```

4. (10 marks) **Programming question:**

   - **Question:** You're consulting for a small investment company, and they have the following type of problem that they want to solve. They are doing a simulation in which they look at n consecutive days of a given stock price. Let's number the days $i = 1, 2, \cdots, n$. For each day i they have a price $p(i)$. They want to maximize their profit by choosing a **a single day** to buy the stock and choosing a **a different day in the future to sell**. For example, suppose $n = 4$ and $p(1) = 9$, $p(2) = 1$ $p(3) = 5$ and $p(4) = 2$. Then the algorithm should return buy on day 2 and sell on day 3 to make a profit of $5 - 1 = 4$. If no profit can be made (e.g. $p(i)$ is a decreasing sequence), then the algorithm should return 0 as the maximum profit. You may assume $n \geq 2$ and is a power of 2.

- **Input:** Input consists several lines of integers. The first integer in each line denotes n (the number of days in the sequence $p(i)$) followed by n integers (stock price for n days) in range between 1 and 100000 separated by spaces. The end of the input is marked by a line consisting of a single 0.
- **Output:** For each sequence in the input, print one line consisting of a single integer that is the maximum. profit.
- **Language:** You can use Java, C,C++, PyPy, Python 3, Go, C#, Rust, F#, Javascript.
- **Note:** There is a limit of 15 submission attempts for this question. This is to prevent people from spamming the automarker (i.e. please don't use the automarker as a debugger, you should test your program carefully before submitting it to the automarker). You should test your program with different inputs, a small sample input and its corresponding output will be on Canvas. Just getting the correct output with the sample is not enough to ensure the correctness of your program.

  Please do not share your code with other students. However, I'm happy if you want to share test cases. I have uploaded a simple Python program to randomly generate input on Canvas as well. You are welcome to use it or develop your own test cases.

  There will be two sets of input for you to submit your program on the automarker. Each has a time limit that your program has to finish within. Something like a brute-force approach probably won't work on the harder input. Do not be overly obsessive with your algorithm implementation. Remember what we discussed in class, optimizing the algorithm itself rather than optimizing the implementation can often lead to more performance gains. The easy test case is worth 8 points and the hard test case is worth 2 points so you should at least develop a correct algorithm before you go on to improve it. Number of submissions are counted separately for the two cases (i.e. you can make 15 submissions each).

  The last submission submitted before the assignment deadline will be the one marked.

  Good luck and have fun.
- **Other useful information:** You can assume that input will come from standard input (stdin) in a stream that represents one string per line. Output should be sent to standard output.

  Your can only submit a single program file (containing all nonstandard classes you use, e.g. you can have nested classes if you use Java).

  If your submission was put in the queue before the assignment due time then it will be accepted. All submissions after the assignment due time will not be considered. Exceptions: people who have an extension.

  Start early! Lots of students will be submitting their work closer to the deadline so it may take 30 min before your program will be executed and you will see the results.

  Your output should exactly match the one in the system for the submission to be correct. So be careful with the printing. No extra symbols! It may look the same on the first glance but may have a different end of line character or extra space.

  Please test at the command-line like the following.

```
python3 task1.py < canvas.in > myout1.txt
diff myout1.txt canvas.out1
```

  If you are on a Windows platform you may need to download diff.exe or use alternatives fc or comp.