CS3230 Semester 2 2024/2025

Assignment 02
Recurrences and Master Theorem

Due: Sunday, 2nd Feb 2025, 11:59 pm SGT.

Instructions:

- Canvas Assignment Submission page: `Assignments/Assignment 2`.
- Please upload PDFs containing your solutions (hand-written & scanned, or typed) by the due date.
- Name the file **Assignment2_SID.pdf**, where SID should be replaced by your student ID.
- You may discuss the problems with your classmates at a high level only. You should write up your solutions on your own (any copying from your co-students or usage of Internet or AI tools is not allowed). Please note the names of your collaborators or any other sources in your submission; failure to do so would be considered plagiarism.
- Question listed as "graded for correctness" (worth 6 points) require complete answers. Other questions (worth 1 point each) will be graded only based on reasonable attempts. However, you should still do these questions, as they are practice questions, which would be useful for exams as well as for your knowledge.

1. (6 points; graded for correctness) Solve the following recurrence relations. Provide as tight a bound as possible. If you use the master theorem, state the case that applies, along with a short reasoning why the case applies. Otherwise, give a detailed proof (using any of the methods). Unless otherwise specified, you can assume base cases, $T(n)$ for $n \le$ some constant, to be $\Theta(1)$. For ease of notation, floors/ceilings are omitted (as mentioned in class, asymptotically, they don't make much difference for the following questions). Thus, when using a term such as $6n/7$ below, assume it means $\lfloor \frac{6n}{7} \rfloor$.

   (a) (2 points) $T(n) = 6T(n/3) + n^2$.

   (b) (2 points) $T(n) = 9T(n/2) + 6n^3 + 4$.

   (c) (2 points) $T(n) = T(n/7) + T(6n/7) + 5$. (Assume $7 \le T(n) \le 100$ for $n \le 7$.)

2. (1 point) Suppose that $f(1) = 0$ and $f(n) = 3f(n/3) + n$ when n is a power of 3. Show that $f(n) = n \log_3 n$ when n is a power of 3.

3. (1 point) Recall that the *greatest common divisor (GCD)* of two non-negative integers m, n is the greatest positive integer p such that p divides both m and n.

   Euclid gave the following algorithm for computing GCD (where it is assumed that $m \le n$):

```
$GCD(m,n)$
    If $m = 0$, then return $n$.
    Else return $GCD(n \bmod m, m)$
END
```

   Give as tight an upper bound as possible on the running time of the above algorithm in terms of n, the larger of the two inputs. You may assume that computing "$n \bmod m$" takes constant time for this question.