CS3000: Algorithms & Data — Summer 2025 — Laney Strange

APP 2
Due: May 13th, 2025 @ 11:30am via Gradescope

Name: Sample Solution

- APPs will be assigned towards the end of roughly two lectures each week. You'll put together a solution to a short problem that we'll all use in the following lecture. We'll have time set aside to do these in class, or you can work on your own.
- You may handwrite your solutions, or typeset them in LaTeXor another system.
- APPs will be graded on completeness. They must be submitted by 11:30am (just before lecture) on the due date. They will not be accepted late, but we drop 3 of them (out of 8 total).
- Collaboration is strongly encouraged for APPs!

Problem 1.

(a) Use the iteration method to solve the recurrence $T(n) = 2T(n/2) + n + c$, where c is a constant, with base case T(2) = 1.

Solution:

Iteration 1: $T(n) = 2T(n/2) + n + c$
Plug in: $T(n/2) = 2T(n/4) + n/2 + c$
Iteration 2: $= 2[2T(n/4) + n/2 + c] + n + c$
$= 4T(n/4) + 2n + 3c$
Plug in: $T(n/4) = 2T(n/8) + n/4 + c$
Iteration 3: $= 4[2T(n/8) + n/4 + c] + 2n + 3c$
$= 8T(n/8) + 3n + 7c$
...

The pattern emerges! So now we can generalize to the kth iteration:
$T(n) = 2^k T(n/2^k) + kn + (2^k - 1)c$

(b) Give a bound on the recurrence.

Choose a value of k to get us to the base case. We want $n/2^k = 2$, solving for k we get $k = \lg n - 1$. Plug this value into our kth iteration:
$T(n) = 2^{\lg n - 1} T(n/2^{\lg n - 1}) + n(\lg n - 1) + c(2^{\lg n - 1} - 1)$
$= n/2 \cdot T(n/n/2) + n \lg n - n + c(n/2) - c$
$= n/2 \cdot T(2) + n \lg n - n + \frac{1}{2}cn - c$
$= -\frac{1}{2}n + n \lg n + \frac{1}{2}cn - c$
$= \Theta(n \lg n)$