CPSC 320 2025S: Assignment 3

This assignment is due **Friday, July 25 at 7 PM**. Late submissions will not be accepted. Please follow these guidelines:

- Prepare your solution using LaTeXand submit a pdf file. Easiest will be to submit using the .tex file provided. For questions where you need to select a circle, you can simply change `\fillinMCmath` to `\fillinMCmathsoln` .
- Enclose each paragraph of your solution with `\begin{soln}Your solution here...\end{soln}`. Your solution will then appear in dark blue, making it a lot easier for TAs to find what you wrote.
- Submit the assignment via GradeScope. Your group must make a **single** submission via one group member's account, marking all other group members in that submission **using GradeScope's interface**.
- After uploading to Gradescope, link each question with the page of your pdf containing your solution.

> **SPECIAL NOTE ON GRADING FOR THIS ASSIGNMENT:** this assignment has two questions, each worth 12 points. If m is the number of points your assignment earns out of 24, your grade for this assignment will be given by $\min\{m/12, 1\}$. (That is, earning full credit on one question, or at least 12 points combined on both questions, will earn you 100% on the assignment.) In addition, if you earn at least 8/12 on **both** questions, we will add 3 bonus percentage points to your midterm exam grade. Also, note that **the material covered on this assignment is within the scope of midterm exam coverage**, so it is worth your time to put in some effort on these questions (even if you don't complete them) before the exam!

Before we begin, a few notes on pseudocode throughout CPSC 320: Your pseudocode should communicate your algorithm clearly, concisely, correctly, and without irrelevant detail. Reasonable use of plain English is fine in such pseudocode. You should envision your audience as a capable CPSC 320 student unfamiliar with the problem you are solving. You may **neither** include what we consider to be irrelevant coding details **nor** assume that we understand the particular language you chose. (So, for example, do not write `#include <iostream>` at the start of your pseudocode, and avoid language-specific notation like C/C++/Java's ternary (question-mark-colon) operator.)

Remember also to **justify/explain your answers**. We understand that gauging how much justification to provide can be tricky. Inevitably, judgment is applied by both student and grader as to how much is enough, or too much, and it's frustrating for all concerned when judgments don't align. Justifications/explanations need not be long or formal, but should be clear and specific (referring back to lines of pseudocode, for example). Proofs should be a bit more formal.

On the plus side, if you choose an incorrect answer when selecting an option but your reasoning shows partial understanding, you might get more marks than if no justification is provided. And the effort you expend in writing down the justification will hopefully help you gain deeper understanding and may well help you converge to the right selection :).

Ok, time to get started...

Group Members

Please list the CWLs of all group members here (even if you are submitting by yourself). We will deduct a mark if this is incorrect or missing.

1 Fun with Recurrences

Give an asymptotic solution (which should be a Θ-bound) to each of the recurrences below. You may use whatever solution method you wish (drawing out the tree, unrolling the recurrence, proof by induction, Master Theorem, etc.), but make sure you fully justify your answer.

1. [3 points] $T(n) = 2T(n-1) + 1$ for $n > 1$, $T(1) = 1$.

$$\begin{aligned} T(n) &= 2T(n-1) + 1 \\ &= 2(2T(n-2) + 1) + 1 \\ &= 2^2 T(n-2) + 2^1 + 2^0 \\ &= 2^3 T(n-3) + 2^2 + 2^1 + 2^0 \\ &\dots \\ &= 2^{n-1} T(1) + 2^{n-2} + \cdots + 2^2 + 2^1 + 2^0 \end{aligned}$$

So my claim is, $T(n) = \sum_{i=0}^{n-1} 2^i = \frac{2^{n-1+1}-1}{2-1} = 2^n - 1, n > 1$.

Proof: $T(2) = 2T(1) + 1 = 3 = 2^2 - 1$. Base case holds. Then $T(n+1) = 2T(n) + 1 = 2(2^n - 1) + 1 = 2^{n+1} - 1$. Thus, induction makes it hold.

Then, $\frac{2^n}{2} = 2^n - \frac{2^n}{2} \leq 2^n - 1 = T(n) \leq 2^n$. Hence, $T(n) \in \Theta(2^n)$.

2. [3 points] $T(n) = 3T\left(\frac{n}{2}\right) + e^n$ for $n > 1$, $T(1) = 1$.

For the 0th level of the tree, we are doing e^n amount of work then in the second we are getting, $3e^{\frac{n}{2}}$.

For any level of the tree we are going to be doing $3^i e^{\frac{n}{2^i}}$ work. The height of the tree is $\lg(n)$.

Hence, $T(n) = \sum_{i=0}^{\lg(n)} 3^i e^{\frac{n}{2^i}}$ for $n > 1$. Then, if we consider $\frac{T(n)}{e^n}$ we get:

$$\frac{T(n)}{e^n} = \sum_{i=0}^{\lg(n)} 3^i e^{n\left(\frac{1-2^i}{2^i}\right)} = 1 + 3e^{\frac{-n}{2}} + 3^2 e^{\frac{-3n}{4}} + \cdots + 3^{\lg(n)} e^{1-n}.$$

Now, observe that if $i \geq 1$ we get $2^i \geq 2 \implies 1 - 2^i \leq -1 < 0$, since $n > 1$ then $1 - n < 0$.

Hence, every term after $i = 1$ has e^{-an} for some $a > 0$ in other words, it decays exponentially.

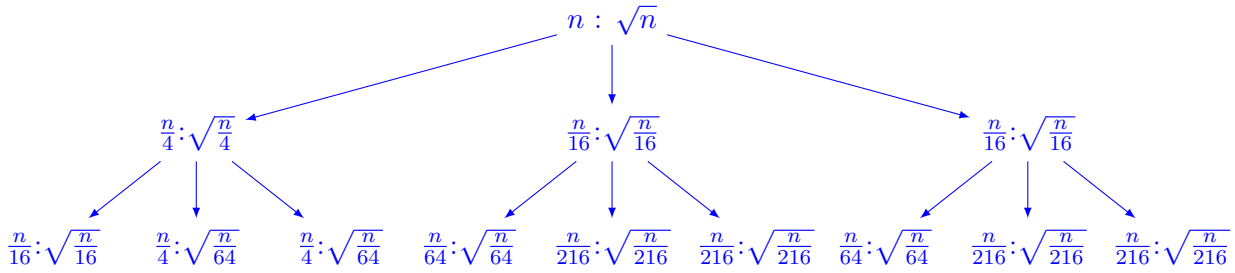
Then,

$$\lim_{n\to\infty} \frac{T(n)}{e^n} = \lim_{n\to\infty} 1 + 3e^{\frac{-n}{2}} + 3^2 e^{\frac{-3n}{4}} + \cdots + 3^{\lg(n)} e^{1-n} = 1.$$

Thus, $T(n) \in \Theta(e^n)$.

3. [3 points] $T(n) = T\left(\frac{n}{4}\right) + 2T\left(\frac{n}{16}\right) + \sqrt{n}$ for $n > 16$, $T(n) = 1$ for $n \leq 16$.

By drawing out the first three levels of the tree and its work needed to perform the subproblem, we see that:



If the top is level 0, then we see at level 1 we are doing $\sqrt{n} \cdot (\frac{1}{2} + \frac{1}{4} + \frac{1}{4}) = \sqrt{n}$ amount of work.

At level 2, we are again doing $\sqrt{n} \cdot (\frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16}) = \sqrt{n}$ amount of work.

Then for each level i we are doing $\sqrt{n}$ amount of work, then we have that $h = \log_{16}(n)$ is the amount of levels we have to work through.

Thus, $T(n) = \log_{16}(n) \cdot \sqrt{n} \in \Theta(\sqrt{n}\log(n))$.

4. [3 points] $T(m,n) = mT(m, \frac{n}{3}) + n$ for $m \geq 1$ and $n \geq 3$; $T(m,n) = 1$ for $m \geq 1$ and $n < 3$.

We consider cases on a fixed m. Once m is chosen, it remains the same for every recursive call.

We will show that each case corresponds to a certain case of the master thoerem, denote $f(n) = n$.

Case $m > 3$, then $\log_3(m) > 1$. Then set $\varepsilon = \log_3(m) - 1 > 0$ so that $\log_3(m) - \varepsilon = 1$.

Thus, $f(n) \in O(n^{\log_3(m)-\varepsilon})$. This corresponds to case 1 theorem, thus $T(n) \in \Theta(n^{\log_3(m)})$.

Case $m = 3$. Then $\log_3(m) = 1$. We see that $f(n) = n \in \Theta(n^{\log_3(m)}\log^0(n))$.

This corresponds to case 2 of the master theorem. Hence $T(m,n) \in \Theta(n\log(n))$.

Case $m \leq 2$. Then $0 < \log_3(m) < 1$, and hence we can set $\varepsilon = 1 - \log_3(m) > 0$ so that $\varepsilon + \log_3(m) = 1$.

Then we see that $n \in \Omega(n^{1+\varepsilon})$. Then observe that $m < 3 \implies \frac{m}{3} < 1$.

Now, take any δ, with $0 < \frac{m}{3} < \delta < 1$ so that $mf(\frac{n}{3}) = \frac{mn}{3} < \delta n = \delta f(n)$.

This corresonds to case 3 of the master theorem, thus $T(m,n) \in \Theta(n)$.

2 D + C = Profit

You own an online sales company called DCAuctions.com that sells goods both on auction and on a fixed-price basis. You want to use historical auction data to investigate your fixed price choices.

Over n minutes, you have a good's price in each minute of the auction. You want to find the largest *price-over-time stretch* in your data. That is, given an array A of n price points, you want to find the largest possible value of

$$f(i, d) = d \cdot \min(A[i], A[i+1], \ldots, A[i+d-1]),$$

where i is the index of the left end of a stretch of minutes, d is the duration (number of minutes) of the stretch, and the function f computes the duration times the minimum price over that period. (Prices are positive, $d \geq 0$, and for all values of i, $f(i, 0) = 0$ and $f(i, 1) = A[i]$.)

For example, the best stretch is underlined in the following price array: $[8, 2, \underline{9, 5, 6, 5}, 3, 1]$. Using 1-based indexing, the value for this optimal stretch starting at index 3 and running for 4 minutes is $f(3, 4) = 4 \cdot \min(9, 5, 6, 5) = 4 \cdot 5 = 20$.

1. [2 points] Describe a polynomial-time brute force algorithm to solve this problem.

2. [1 point] Suppose the minimum element in A is $A[k] = 4$, and suppose that A has length n. What's the best price stretch for all intervals that include $A[k]$? Briefly justify your answer.

3. [5 points] Using the insight from the previous question, give an efficient algorithm to find the best price stretch.

4. [2 points] Give and briefly justify a good asymptotic bound on the **worst-case** runtime of your algorithm.

5. [2 points] Give and briefly justify a good asymptotic bound on the **average-case** runtime of your algorithm.