CS251 - Data Structures and Algorithms
Homework 2, Fall 2024

Due Thursday, September 5, 2024, 11:59 PM EST

Instructions:

- Read all questions carefully before you start working on them. Do your best to understand the context of each question and identify the concepts covered in class that will help you to solve them. Use Ed Discussion, PSOs, and office hours to discuss doubts or guidance.
- The TAs will provide assistance through the mechanisms indicated in the syllabus, but they need you to do initial work and show some understanding of the topics.
- You must type your work using LaTeX(preferred), MS Word, or a similar quality text editor. We recommend Overleaf for typing in LaTeX(your Purdue account gives access to the Premium version).
- Handwritten work, including math, images, and diagrams, will not be graded (i.e., get 0 points). Sketch your diagrams using PowerPoint or an image editor.
- Submit your work as a single PDF file on Gradescope.
- Ensure each problem begins on a new page and follow the instructions for matching problems to page numbers. Correct submission and formatting are your responsibility; errors will lead to a 20% point deduction per improperly assigned page.
- You must provide detailed explanations for every answer (i.e., justify your answer), even if the problem description doesn't say so. Graders regard half of the points on correctness and the other half on explanations.
- Include any images, diagrams, or pictures in a vertical layout with a clean background. Submissions lacking professional presentation will not be eligible for regrade requests.
- There are no late submissions. Failure to submit by the deadline will result in a score of 0.
- For grading purposes, we will only grade your latest submission.
- Individual absences and extensions for a valid reason (e.g., medical, military, grief) should be approved by the Office of the Dean of Students. No rescheduling will be made unless the instructor(s) receive the respective ODOS notification.

On academic integrity: While all course activities are individual tasks, discussing with your study group the topics and strategies to solve the problems is allowed and encouraged. However, the work you submit must be entirely your own. Check Purdue's Academic Integrity for detailed information on the actions that constitute academic dishonesty. You are responsible to abide by the integrity guidelines.

Read the syllabus for more information on academic integrity. Academic dishonesty is reported to the Office of Student Rights and Responsibilities.

On using AI tools: Dependence on AI might offer short-term gains but can jeopardize long-term success and understanding. Remember, AI tools won't be available during midterms and the final exam. Consider the broader implications of over-reliance on AI: you aim to be a skilled professional, not someone limited by the capabilities of these tools. Read the syllabus for the policies of this course related to AI tools.

Question 1

Consider the following algorithm:

```
function A(n : $\mathbb{Z}_{\geq 0}$)
    if $n = 0$ then
        return 0
    end if
    if n is even then
        return n + A(n - 2)
    else
        return A(n - 1)
    end if
end function
```

(a) State the mathematical recurrence relation followed by the algorithm.

(b) Prove by induction on n that the algorithm correctly computes the sum of even numbers from 0 to n for $n \in \mathbb{Z}_{\geq 0}$.

Solution:

(a)

$$A(n) = \begin{cases} 0 & \text{if } n = 0 \\ A(n-2) + n & \text{if } n \text{ is even} \\ A(n-1) & \text{if } n \text{ is odd} \end{cases}$$

(b) We want to prove that the function $A(n)$ computes the sum of even numbers from 0 to n.

Base Case:

- **Case 1: $n = 0$:**

$$A(0) = 0$$

  The sum of all even numbers up to 0 is 0.

- **Case 2: $n = 1$:**

$$A(1) \text{ evaluates to } A(0) = 0$$

  The sum of all even numbers up to 1 is 0.

Therefore, the base case holds true.

Inductive Step:

Assume that $A(k)$ correctly computes the sum of even numbers from 0 to k for all smaller k. We need to show that $A(k+1)$ is also correct.

- **Case 1: $k + 1$ is even:**
  If $k + 1$ is even, then $A(k + 1)$ computes:

$$A(k+1) = (k+1) + A((k+1) - 2)$$

  By the inductive hypothesis, $A(k - 1)$ correctly computes the sum of all even numbers up to $k - 1$. Therefore:

$$A(k+1) = (k+1) + \text{sum of all even numbers up to } (k-1)$$

  This is equal to the sum of all even numbers up to $k + 1$.

- **Case 2: $k+1$ is odd:**
  If $k+1$ is odd, then $A(k+1)$ computes:

  $$A(k+1) = A(k)$$

  By the inductive hypothesis, $A(k)$ correctly computes the sum of all even numbers up to k. Since $k+1$ is odd, the sum of all even numbers up to $k+1$ is the same as the sum of all even numbers up to k.

  Therefore, $A(k+1)$ correctly computes the sum.

Question 2

Consider the algorithm below:

```
function FUN(n : $\mathbb{Z}_{\geq 1}$ power of 2)
    if $n = 1$ then
        return 1
    end if
    temp = 0
    for $i = 1$ to $n$ do
        temp = temp + 1
    end for
    return temp + FUN(n/2) + FUN(n/2)
end function
```

(a) State the mathematical recurrence relation followed by FUN.

(b) Solve the recurrence relation to obtain the respective non-recursive expression.

Solution:

Note: The input must be a power of 2 for both the recurrence relation and its non-recursive one to return the same result.

(a)

$$FUN(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2FUN\left(\frac{n}{2}\right) + n & \text{if } n > 1 \end{cases}$$

(b) Given the recurrence relation:

$$FUN(n) = 2FUN\left(\frac{n}{2}\right) + n$$

First, expand the relation by substituting $FUN\left(\frac{n}{2}\right)$:

$$\begin{aligned} FUN(n) &= 2\left[2FUN\left(\frac{n}{2^2}\right) + \frac{n}{2}\right] + n \\ &= 2^2FUN\left(\frac{n}{2^2}\right) + 2 \cdot \frac{n}{2} + n \\ &= 2^2FUN\left(\frac{n}{2^2}\right) + n + n \\ &= 2^2FUN\left(\frac{n}{2^2}\right) + 2n \end{aligned}$$

Expand again by substituting $FUN\left(\frac{n}{2^2}\right)$:

$$\begin{aligned} FUN(n) &= 2^2\left[2FUN\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right] + 2n \\ &= 2^3FUN\left(\frac{n}{2^3}\right) + 2^2 \cdot \frac{n}{2^2} + 2n \\ &= 2^3FUN\left(\frac{n}{2^3}\right) + n + 2n \\ &= 2^3FUN\left(\frac{n}{2^3}\right) + 3n \end{aligned}$$

After k expansions, we have:

$$FUN(n) = 2^k FUN\left(\frac{n}{2^k}\right) + kn$$

To solve this, consider the base case $FUN(1) = 1$.
Let $FUN\left(\frac{n}{2^k}\right) = FUN(1)$. Then, $\frac{n}{2^k} = 1 \iff n = 2^k \iff k = log_2(n)$ and:

$$\begin{aligned} FUN(n) &= 2^{\log_2 n} FUN(1) + \log_2 n \cdot n \\ &= n \cdot 1 + n \log_2 n \\ &= n + n \log_2 n \end{aligned}$$

Question 3

To solve specific problems in Computer Science disciplines, it is essential to consider all of the input value permutations. The following algorithm is a brute-force approach to finding all permutations of a multiset (i.e., a set that allows replicated values) of size $n \in \mathbb{Z}_{\geq 0}$ represented as an array.

```
function GenerateSubsets($A$ : array, $n : \mathbb{Z}_{\geq 0}$, $C$ : array)
    if $n = 0$ then
        return $[C]$
    end if
    $excluding \leftarrow$ GenerateSubsets($A, n-1, C$)
    $including \leftarrow$ GenerateSubsets($A, n-1$, Concatenate($C, [A[n-1]]$))
    return Concatenate($excluding, including$)
end function
```

Note:

- The notation $[x]$ (with no variable name next to it) means the declaration of an anonymous array of capacity one and initialized with item x. For example, $[A[i]]$ means we declare an anonymous array of capacity one and initialize it with the item from A at index i.
- The function Concatenate(A, B) returns a new array resultant from the concatenation of arrays A and B.

The first call of the algorithm is $S \leftarrow$ GenerateSubs(A, n, C), where A is an array of size n, and C is an empty array.

(a) Indicate the recursive runtime expression $T(n)$ for the cost of running GenerateSubsets in terms of the size of the multiset. Assume the cost of the function Concatenate is already included in the respective cost value for the recursive step.

(b) Find the respective non-recursive $T(n)$ using substitutions.

(c) Prove by induction on n that the non-recursive $T(n)$ describes the cost of running GenerateSubsets in terms of the size of the multiset.

Solution:

(a) Let C_b be the cost of the operations run in the base case, and C_r be the cost of the operations run in the recursive step. Then:

$$T(0) = C_b$$
$$T(n) = C_r + 2T(n-1)$$

(b) Using substitutions:

$$\begin{aligned}
T(n) &= C_r + 2T(n-1)\\
&= C_r + 2(C_r + 2T(n-2)) = C_r + 2C_r + 2^2T(n-2)\\
&= C_r + 2C_r + 2^2(C_r + 2T(n-3)) = C_r + 2C_r + 2^2C_r + 2^3T(n-3)\\
&= C_r + 2C_r + 2^2C_r + 2^3(C_r + 2T(n-4)) = C_r + 2C_r + 2^2C_r + 2^3C_r + 2^4T(n-4)\\
&= \cdots\\
&= \sum_{i=0}^{k-1} 2^iC_r + 2^kT(n-k)\\
&= C_r\frac{2^k-1}{2-1} + 2^kT(n-k)\\
&= C_r(2^k-1) + 2^kT(n-k)\\
&\text{Recursion finishes when } n-k=0\text{, then } n=k\\
&= C_r(2^n-1) + 2^nT(n-n)\\
&= C_r(2^n-1) + 2^nT(0)\\
&= C_r(2^n-1) + 2^nC_b\\
&= (C_r+C_b)2^n - C_r
\end{aligned}$$

(c) Proof by induction on n of $T(n) = (C_r + C_b)2^n - C_r$:

- Base case ($n = 0$): $T(0) = (C_r + C_b)2^0 - C_r = C_r + C_b - C_r = C_b$
- Inductive step ($n > 0$): Assume $T(n)$ is true for $n' < n$ (Inductive Hypothesis)

$$\begin{aligned}
T(n) &= C_r + 2T(n-1)\\
&= C_r + 2((C_r + C_b)2^{n-1} - C_r)\\
&= C_r + 2\left((C_r + C_b)\frac{2^n}{2} - C_r\right)\\
&= C_r + (C_r + C_b)2^n - 2C_r\\
&= (C_r + C_b)2^n - C_r
\end{aligned}$$

Question 4

Alice and Bob play a game with piles of stones. There are an even number of piles arranged in a row, and each pile i has a positive integer number of stones $piles[i]$.

The objective of the game is to end with the most stones. The total number of stones across all the piles is odd, so there are no ties.

Alice and Bob take turns, with Alice starting first. For each turn, a player takes the entire pile of stones from either the beginning or the end of the row.

This continues until there are no more piles left, at which point the person with the most stones wins.

(a) If Alice always picks the pile with the largest number of stones (either from the beginning or the end), will she always win?

(b) Assuming both players play optimally, prove by induction that Alice will win each time.

(c) Will Alice always win if the number of piles was odd instead?

Solution:

(a) Assume $piles = [3, 10, 1, 1]$.
Alice will pick 3 and Bob will pick 10.
Alice will now pick 1 and Bob will pick 1.
Alice's score = 4
Bob's score = 11
Bob wins.
Proof by contradiction.

(b) **Base Case** $(n = 2)$**:**
With two piles, Alice will pick the larger pile. Since the total number of stones is odd, Alice will always have more stones than Bob, guaranteeing her win.

Inductive Hypothesis:
Assume Alice wins for any even number of piles $n = 2k$.

Inductive Step $(n = 2k + 2)$**:**
Consider $n = 2k + 2$ piles.

- **Case 1:**
If the sum of stones in even-indexed piles is greater than the sum in odd-indexed piles, Alice will pick the first pile (index 0). Bob is then forced to pick from odd-indexed piles (either index 1 or $n - 1$), leading to a smaller case with $2k$ piles. By the inductive hypothesis, Alice wins.
- **Case 2:**
If the sum of stones in odd-indexed piles is greater, Alice will pick the last pile (index $n - 1$). Bob will then be forced to pick from even-indexed piles (either index 0 or $n - 2$), leading to a smaller case with $2k$ piles. By the inductive hypothesis, Alice wins.

Thus, in both cases, Alice wins by reducing the game to a smaller instance where she can again apply the same strategy. Hence, by induction, Alice always wins when the number of piles is even.

(c) *Proof by contradiction:* Consider the case when $piles = [2, 10, 1]$.
Regardless of which end Alice picks, Bob will always win. Thus, Alice does not always win when the number of piles is odd.