Group HW #6

Michael Ku, Dhvan Shah, Pranav Bonthu

November 17, 2025

Problem 1

Let $P(n)$ be the proposed statement that $2^{(n-1)}+2\cdot4^{(n-1)}$ is a closed form solution for the recurrence relation for the number of length n codewords you can make if you must have an even number of 0s. We want to prove that P(n) is true for integers $n \geq 1$. The recurrence relation is defined as

$$q_n = 2(q_{n-1}) + 4^{n-1}.$$

Lets look at the proposed statement of $P(1)$:

$$P(1) = 2^{(1-1)} + 2 \cdot 4^{(1-1)} = 2^0 + 2 \cdot 4^0 = 3.$$

When looking at the recurrence relation, q_1 is equal to 3. This means that when $n = 1$, the recurrence relation and the proposed closed-form solution are equal.

As our inductive hypothesis, assume that $P(n)$ holds for SOME $k \geq 1$. We will look at the case of $P(k+1)$ and q_{k+1}.

$$q_{k+1} = 2(q_k) + 4^{(k+1)-1}.$$

Through induction, we say $q_k = P_k$. So:

$$q_{k+1} = 2(2^{k-1}) + 2(4^{k-1}) + 4^{(k+1)-1}.$$

With algebra, we can simplify to:

$$q_{k+1} = 2(2^{k-1}) + 2(4^{k-1}) + 4^k,$$

$$q_{k+1} = 2 \cdot 2^{k-1} + 2 \cdot 2(4^{k-1}) + 4^k,$$

$$q_{k+1} = 2 \cdot 2^{k-1} + 4(4^{k-1}) + 4^k,$$

$$q_{k+1} = 2 \cdot 2^{k-1} + 4^k + 4^k,$$

$$q_{k+1} = 2^k + 2 \cdot 4^k,$$

$$q_{k+1} = 2^{(k+1)-1} + 2 \cdot 4^{(k+1)-1} = P(k+1).$$

Since the base case of $P(1)$ holds and since the $(k+1)$st case holds true, $P(n)$ must be true for all positive integers where $n \geq 1$.

- **Who Contributed:** Pranav Bonthu was the main contributor, doing the formal write-up of the solution. Mikey Ku and Dhvan Shah compared their solutions to Pranav's and provided feedback and edits.
- **Resources**: The textbook, class notes
- **Main points:** Based on group discussions, we felt that the main point of this problem was to understand how to prove that the closed form and recurrence relation are equal to each other (using induction).

Problem 2

Let $P(n)$ be the proposed statement that $n!$ represents the product of all "choose" states when a pile of n candies is split into two smaller piles (r and s) and $\binom{r+s}{s}$ is taken. The piles keep splitting until there are n piles of 1 candy, where at each split a choose statement is formed (the total sum of candy from the two piles choose the amount of candy in one of the piles).

Below are the initial conditions for the problem:

$$n = 1 : 1, \qquad n = 2 : 2, \qquad n = 3 : 6, \qquad n = 4 : 24$$

Consider $P(1)$:

$$P(1) = 1! = 1.$$

This shows that when $n = 1$, $P(1)$ holds. As our inductive hypothesis, assume that $P(k)$ holds for some $k \geq 1$. We now consider $P(k+1)$. Since $k+1$ can be split into r and s (both less than $k+1$ and satisfying $r + s = k + 1$), we obtain the choose term

$$\binom{r+s}{s} = \binom{k+1}{s}.$$

Because r and s are both less than $k+1$, the inductive hypothesis gives that the product of choose statements for the pile of size r is $r!$ and similarly the product for size s is $s!$.

Therefore, the total product of choose statements for splitting $k+1$ candies is:

$$\binom{r+s}{s} \cdot r! \cdot s!.$$

Using algebra:

$$\binom{k+1}{s} \cdot r! \cdot s! = \frac{(k+1)!}{r!\, s!} \cdot r! \cdot s! = (k+1)!.$$

Thus, $P(k+1)$ holds. Since the base case $P(1)$ holds and the inductive step is valid, $P(n)$ is true for all integers $n \geq 1$.

- **Who Contributed:** Pranav Bonthu was the main contributor, doing the formal write-up of the solution. Mikey Ku and Dhvan Shah compared their solutions to Pranav's and provided feedback and edits.
- **Resources**: The textbook, class notes
- **Main points:** Based on group discussions, we felt that the main point of this problem was to understand how induction can be used to prove iterative problems.

Problem 3

Question:
Part (a):
Recall your inclusion/exclusion knowledge to find an expression for the number of onto functions from S (s elements) to a set T (with some t elements). To get full credit, you cannot simply use the formula from the book. You must set up a big set A that contains all functions from S to T, and then carefully define subsets A_i that contain "bad" (not onto) functions, and then explain how you are applying inclusion/exclusion to those subsets. Put your final answer in the form "$t^s - .....$".

Solution:

To solve this question, we will use the principle of inclusion-exclusion to take the set of all functions from S to T and then remove the ones that are not onto. We start by defining our big set A as the set of all functions from S to T. Each element in S has t possible choices in T, so there are t^s total functions in A. Now, we define subsets A_i for each element i in T, where each A_i represents the set of "bad" functions that do not map any element of S to i. These functions are not onto because they completely miss the element i in T. What we want, then, is to count all the functions that are not in the union of these bad sets, or $|A| - |\bigcup_i A_i|$. This is where inclusion-exclusion comes in. We can start by considering what happens when we prevent one value of T from being used. If we block out one element of T, then each of the s elements in S only has $(t-1)$ choices. That means there are $(t-1)^s$ such functions, and since there are $\binom{t}{1}$ ways to choose which element of T to block, we get $\binom{t}{1}(t-1)^s$ total functions that miss exactly one element.
Next, we continue this process by increasing the number of elements that we prevent from being mapped to. If we block two elements of T, there are $(t-2)^s$ functions that map S into the remaining $t-2$ elements, and there are $\binom{t}{2}$ ways to choose which two to exclude. The same logic applies for blocking three, four, or more elements.
Using inclusion-exclusion, we alternate the signs each time to make sure we are not double-counting or double-removing overlapping cases. Putting this all together, the number of onto functions from S to T is:

$$t^s - \binom{t}{1}(t-1)^s + \binom{t}{2}(t-2)^s - \binom{t}{3}(t-3)^s + \cdots + (-1)^t\binom{t}{t}(t-t)^s.$$

This expression starts with all possible functions t^s, subtracts those that miss at least one element of T, adds back those that miss at least two, and continues this alternating pattern to correctly

count all onto functions from S to T. This can also be written as:

$$f_t = \sum_{i=0}^{t} (-1)^i \binom{t}{i} (t-i)^s.$$

Part (b):
Find a recurrence relation to count f_t = the number of onto functions from S with s elements to a set T with t elements. Although philosophically you could recur on either or both s and t, I want you to find a recurrence relation that assumes that s is fixed and t is the variable that can vary. This means that you can recur on t but not on s. There are multiple correct RRs, but please keep working on options until you find a RR that starts with "$t^s - .....$".

Solution:
Based on the previous part, we can say that the number of onto functions from S to T is given by the inclusion–exclusion formula:

$$f_t = \sum_{i=0}^{t} (-1)^i \binom{t}{i} (t-i)^s.$$

To better understand how this behaves and to look for a recurrence pattern, let's compute the first few values of f by changing the value of t and holding s constant.

- If $t = 1$:
$$f_1 = 1^s = 1.$$
- If $t = 2$:
$$f_2 = 2^s - \binom{2}{1}(1^s) = 2^s - 2.$$
- If $t = 3$:
$$f_3 = 3^s - \binom{3}{1}(2^s) + \binom{3}{2}(1^s) = 3^s - 3(2^s) + 3.$$
- If $t = 4$:
$$f_4 = 4^s - 4(3^s) + 6(2^s) - 4.$$
- If $t = 5$:
$$f_5 = 5^s - 5(4^s) + 10(3^s) - 10(2^s) + 5.$$

Now let's analyze the pattern among these results. Between $t = 1$ and $t = 2$, we see that the expression starts with t^s and then subtracts a term involving $\binom{t}{1}$ multiplied by a previous power. When we move to $t = 3$, we notice that 2^s reappears with a coefficient based on $\binom{3}{1}$, and the constant term 3 relates to $\binom{3}{2}$ and f_1. Similarly, for $t = 4$ and $t = 5$, each new expression brings back powers that appeared in earlier f_i terms, each multiplied by the corresponding binomial coefficient. This observation suggests that each f_t depends on the previous values $f_{t-1}, f_{t-2}, \ldots$ through descending binomial coefficients. Conceptually, every function from S to T either uses all t elements (and is onto) or misses at least one of them. If a function misses exactly i elements, there are $\binom{t}{i}$ ways to choose which are missed, and the remaining functions correspond to f_{t-i}.

Putting this reasoning together, we arrive at the following recurrence relation that looks something like this:

$$f_t = t^s - \binom{t}{1} f_{t-1} - \binom{t}{2} f_{t-2} - \binom{t}{3} f_{t-3} - \cdots - \binom{t}{t-1} f_1.$$

Turning this into a more formal summation, we get a final answer that looks like this:

$$f_t = t^s - \sum_{i=1}^{t-1} \binom{t}{i} f_{t-i}, \quad \text{with } f_1 = 1 \text{ , } s \geq 1 \text{ and } t \geq 1$$

This recurrence starts with t^s and subtracts the overcounted cases where one or more of the t elements are not used, matching the pattern we observed from the earlier examples.

Part (c):
Write a sentence or two that compares and contrasts the closed-form solution that you got via inc/exc and the RR that you got that counts the same thing. What is the major difference in how these two solutions count onto functions?

Solution:
Both the closed-form solution from inclusion–exclusion and the recurrence relation begin by considering all possible functions from S to T (i.e., t^s total) and then subtracting the cases that are not onto. The key difference lies in how they handle overcounting. In the closed-form expression, inclusion–exclusion explicitly corrects for overlaps by alternately adding and subtracting terms to account for functions missing one or more elements of T. In contrast, the recurrence relation inherently incorporates this correction through previous computed values f_i, each of which already accounts for overlaps in smaller cases. As a result, the recurrence builds onto functions incrementally, while the closed form resolves all cases simultaneously.

- **Who Contributed:** Mikey Ku was the main contributor, doing the formal write-up of the solution. Pranav Bonthu and Dhvan Shah compared their solutions to Mikey's and provided feedback and edits.
- **Resources**: The textbook, class notes
- **Main points:** Based on group discussions, we felt that the main point of this problem is to practice using inclusion-exclusion on a closed form and RR, to undertstand their similarities and differences.

Problem 4

Setup: In the Tower of Hanoi puzzle, suppose our goal is to transfer all n disks from Peg 1 to Peg 3, but we cannot move a disk directly between Pegs 1 and 3. Each move of a disk must be a move involving Peg 2. As usual, we cannot place a disk on top of a smaller disk.

Part A: Find a recurrence relation for the number of moves required to solve the puzzle for n disks with this added restriction.

Let a_n be the minimum number of moves to solve the puzzle for n disks.

We can start by finding some base cases. To solve the puzzle for $n = 1$ disk, we move the disk from Peg 1 to Peg 2, and then from Peg 2 to Peg 3. This means $a_1 = 2$.

We can do the same for $n = 2$ disks.

1. First, we need to move Disk 1 from Peg 1 to Peg 3. This is just an $n = 1$ problem, which we know takes a_1 moves.
2. Now, move Disk 2 from Peg 1 to Peg 2 (1 move).
3. Next, we must move Disk 1 from Peg 3 to Peg 1 so we can move Disk 2. This is just an $n = 1$ problem in reverse, which also takes a_1 moves.
4. Then, move Disk 2 from Peg 2 to Peg 3 (1 move).
5. Finally, move Disk 1 from Peg 1 to Peg 3, which again takes a_1 moves.

Adding this all up:

$$a_2 = a_1 + 1 + a_1 + 1 + a_1 = 3a_1 + 2$$

We can generalize this process for n disks. To move n disks from Peg 1 to Peg 3:

1. Move the top $n - 1$ disks from Peg 1 to Peg 3. This takes a_{n-1} moves.
2. Move the nth disk from Peg 1 to Peg 2. This takes 1 move.
3. Move the top $n - 1$ disks from Peg 3 to Peg 1. This takes a_{n-1} moves.
4. Move the nth disk from Peg 2 to Peg 3. This takes 1 move.
5. Move the top $n - 1$ disks from Peg 1 to Peg 3. This takes a_{n-1} moves.

This process is valid because the nth disk is the largest, so it only moves to an empty peg, and the moves of the $n - 1$ stack are valid by our recursive definition.

Writing the above sequence as a recurrence relation:

$$a_n = a_{n-1} + 1 + a_{n-1} + 1 + a_{n-1} = 3a_{n-1} + 2$$

$$\text{Where } a_1 = 2$$

Part B: Prove the closed form solution of the recurrence relation using induction

Let $P(n)$ be the proposition that $a_n = 3^n - 1$ for the recurrence $a_n = 3a_{n-1} + 2$ and $a_1 = 2$.

Base Case: We can show that $P(1)$ holds true:

$$a_1 = 2$$

$$3^1 - 1 = 2$$

The proposition holds for $n = 1$.

We can also check if $P(2)$ holds true:

$$a_2 = 3a_1 + 2 = 3(2) + 2 = 8$$

$$3^2 - 1 = 9 - 1 = 8$$

The proposition also holds for $n = 2$.

Inductive Hypothesis: Assume $P(k)$ holds true for some $k \geq 1$. That is, assume $a_k = 3^k - 1$.

Inductive Step: We want to show that $P(k+1)$ holds true, meaning we want to show that $a_{k+1} = 3^{k+1} - 1$.

We start with the recurrence relation for a_{k+1}:

$$a_{k+1} = 3a_k + 2$$

Now, we use our inductive hypothesis to substitute the formula for a_k:

$$\begin{aligned} a_{k+1} &= 3(3^k - 1) + 2 \\ &= 3 \cdot 3^k - 3 + 2 \\ &= 3^{k+1} - 1 \end{aligned}$$

This result, $a_{k+1} = 3^{k+1} - 1$, is exactly what we wanted to show.

Since $P(1)$ is true and we have shown that $P(k)$ implies $P(k+1)$, by the Principle of Mathematical Induction, $P(n)$ is true for all $n \geq 1$.

Part C: How many different arrangements are there of the n disks on three pegs so that no disk is on top of a smaller disk?

We can solve this by considering each disk independently. Each of the n disks can be placed on any of the 3 pegs, which means each disk has 3 choices.

Since the peg choice for each disk is independent, we can use the product rule to find the total number of arrangements: $3 \cdot 3 \cdot \cdots \cdot 3$ (n times) $= 3^n$.

We don't have to worry about the stacking order on a single peg, because the rules of the puzzle state that the disks must be ordered from largest on the bottom to smallest on the top. Once we have assigned a set of disks to a peg, there is only one valid way to stack them.

Part D: Show that every allowable arrangement of the n disks occurs in the solution of this variation of the puzzle.

In Part C, we showed that there are 3^n total valid arrangements for the n disks.

In Part B, we proved that the algorithm described in Part A requires $a_n = 3^n - 1$ moves to get from the starting configuration to the final configuration

A sequence of $3^n - 1$ moves, starting from the starting configuration, will pass through a total of $(3^n - 1) + 1 = 3^n$ states.

Since our solution path visits 3^n unique states, and there are only 3^n possible valid states in total, our solution must visit every possible allowable arrangement exactly once.

- **Who Contributed:** Dhvan Shah was the main contributor, doing the formal write-up of the solution. Pranav Bonthu and Michael Ku compared their solutions to Dhvan's and provided feedback and edits.
- **Resources**: The textbook, class notes
- **Main points:** Based on group discussions, we felt that the main point of this problem was to understand how to use induction to prove the closed form solution of a recurrence relation

Improvement Goal: We didn't have an improvement goal for this week
Next Week: Next week our improvement goal is to be done before Saturday night.