MAT2540, Classwork8, Spring2026

8.1 Applications of Recurrence Relations

1. Recurrence Relation and its solution

Recurrence Relation is a rule for determining subsequent terms from those that precede them and a sequence is called a solution of a recurrence relation if its terms satisfy the recurrence relation.

2. Example of the Modeling With Recurrence Relations: Rabbits and the Fibonacci Numbers.

A pair of rabbits does not breed until they are 2 months old. After they are 2 months old, each pair of rabbits produces another pair each month, as shown in the table. Find a recurrence relation for the number of pairs of rabbits on the island after n months, assuming that no rabbits ever die.

Reproducing Pairs (at least 2 months old)	Young Pairs (less than 2 months old)	Month	Reproducing Pairs	Young Pairs	Total Pairs
	1	1	0	1	1
	1	2	0	1	1
1	2	3	1	1	2
1	2 3	4	1	2	3
1 2	3 4 5	5	2	3	5
1 2 3	4 5 6 7 8	6	3	5	8

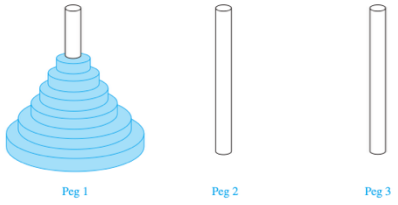
$a_1=1,\ a_2=1,\ a_n=a_{n-1}+a_{n-2}$

$f_1=1,\ f_2=1,\ f_n=f_{n-1}+f_{n-2}$

3. Example of the Modeling With Recurrence Relations: The Tower of Hanoi Puzzle.

Tower of Hanoi consists of three pegs mounted on a board together with disks of different sizes. Initially these disks are placed on the first peg in order of size, with the largest on the bottom (as shown in Figure). The rules of the puzzle allow disks to be moved one at a time from one peg to another as long as a disk is never placed on top of a smaller disk. The goal of the puzzle is to have all the disks on the second peg in order of size, with the largest on the bottom.

Let H_n denote the number of moves needed to solve the Tower of Hanoi puzzle with n disks. Set up a recurrence relation for the sequence $\{H_n\}$.



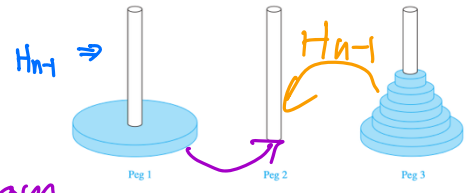
$H_0 \Rightarrow$

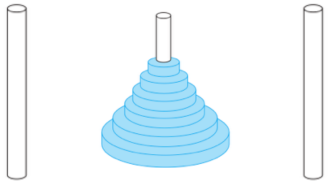
How to find the difference from the step before the final step (H_{n-1}) to the final step (H_n)

① We need H_{n-1} steps to get n-1 disks from peg 1 to peg 2

② We need 1 steps to get the bigest disk from peg 1 to peg 2

③ We need H_{n-1} steps to move n-1 disks from peg 3 to peg 2





$\Rightarrow$ Total move is $H_n = H_{n-1} + 1 + H_{n-1} \Rightarrow H_n = 2H_{n-1} + 1$

4. Example of the Modeling With Recurrence Relations:

Find a recurrence relation and give initial conditions for the number of bit strings of length n that do not have two consecutive 0s.

$a_1 = 2$, $a_2 = 3$

1 or 0 — 10, 01, 11, ~~00~~

n bits

Let a_n denote the number of bit string of length n that do not have two consecutive 0s

We assume $n \geq 3$. (so that the string has at least 3 bits)

a_n can be divided into those that end in 1 or 0

			Number of bit strings of length n with no two consecutive 0s:
End with a 1:	Any bit strings of length n-1 with no two consecutive 0s	1	a_{n-1}
End with a 0:	Any bit string of length n-2 with no two consecutive 0s	1 0	a_{n-2}
		Total:	

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

5. From 4., how many such bit strings are there of length five?

$a_1 = 2, a_2 = 3, a_3 = a_1 + a_2 = 5,$

$a_4 = a_2 + a_3 = 8,$

$a_5 = a_4 + a_3 = 13.$

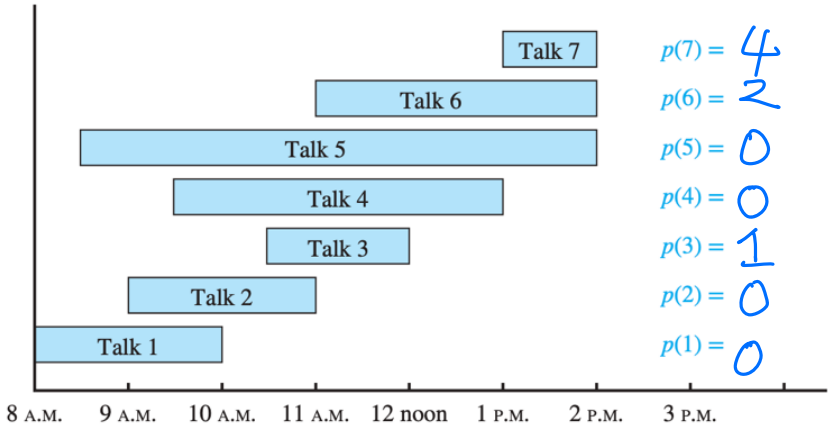
6. Dynamic programming

Dynamic programming is a powerful algorithmic technique used to solve complex problems by breaking them down into simpler, overlapping subproblems and storing the results to avoid redundant calculations.

7. Suppose we have n proposed talks with preset start and end times. Talk j begins at time s_j (where s stands For start) and ends at time e_j (where e stands for end) for $j = 1, \ldots, n$. We define a function $p(j)$:

$$p(j) = \begin{cases} \text{largest integer } i,\ i<j \text{ for which } e_i \leq s_j \\ 0 \end{cases}$$

Consider seven talks with these start times and end times, as illustrated in Figure. Find the values of function $p(j)$ for $j = 1, \ldots, 7$.



$p(7) = 4$

$p(6) = 2$

$p(5) = 0$

$p(4) = 0$

$p(3) = 1$

$p(2) = 0$

$p(1) = 0$

We say that two talks are **compatible** if they can be part of the same schedule, that is, if the times they are scheduled do not overlap. In this example, talk 3 and talk 1 are compatible, so is talk 6 with talk 2, and talk 7 with talk 4.

possible optimal schedule:

① Talk 1, 3, 7 ② Talk 2, 6 ③ Talk 4, 7