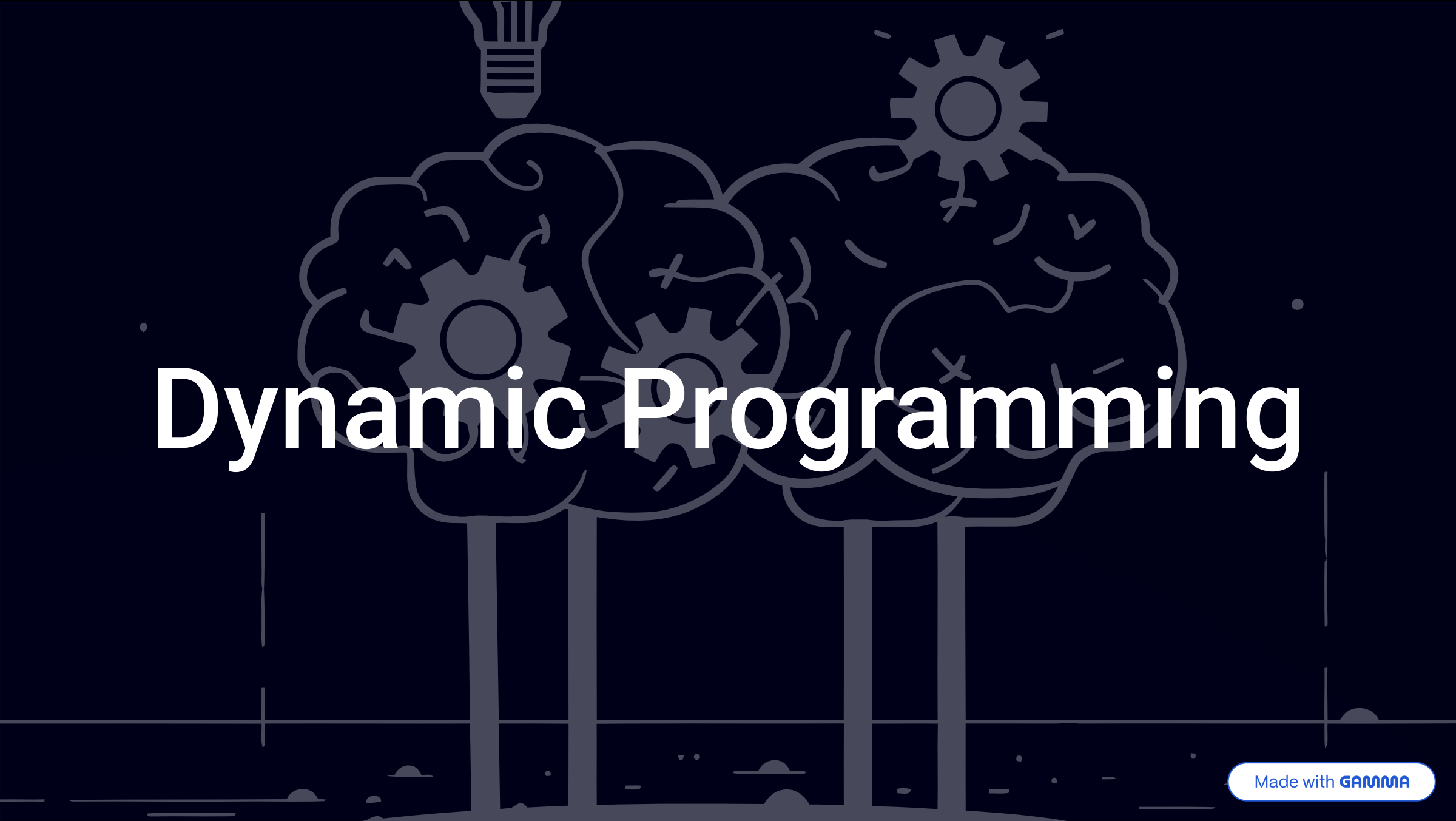


Dynamic Programming

Introduction

Dynamic Programming (DP) is one of the most powerful techniques every programmer should know.

Many people think it’s complicated, but it’s simply a way of thinking to solve big problems efficiently.

When to Use DP

1	When the solution of a problem depends on solutions of smaller parts.
2	Those smaller parts might also depend on even smaller ones.

Core Idea

Integrate Solutions

Combine subproblem solutions to solve main

Subproblems

Break the main problem into smaller parts

Atomic Tasks

Decompose subproblems to simplest tasks

Main Problem

Identify the large, overarching problem

- **Break a big problem into smaller subproblems.**
- **Subproblems can be broken down further until they reach the simplest form.**
- **The solution of the main problem depends on the solutions of its subproblems.**

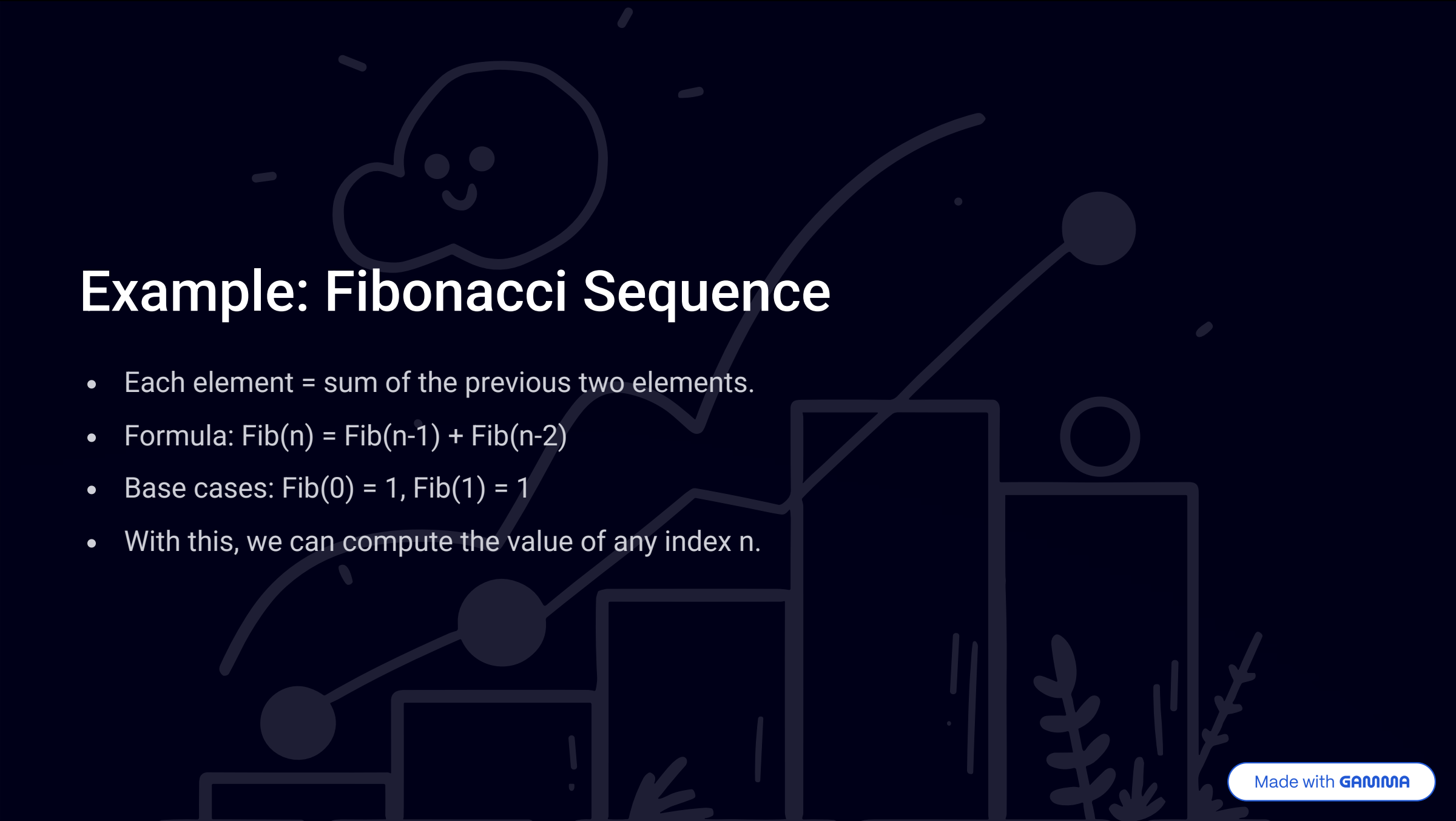
Two Approaches

Top-Down:

- Start from the main problem and break it into smaller ones.
- Usually implemented using recursion.

Bottom-Up:

- Start from the smallest subproblem and build up to the final solution.
- Usually implemented using loops.



Example: Fibonacci Sequence

- Each element = sum of the previous two elements.
- Formula: Fib(n) = Fib(n-1) + Fib(n-2)
- Base cases: Fib(0) = 1, Fib(1) = 1
- With this, we can compute the value of any index n.