Challenge 0: Top-Down Approach

Journey to your best program-2024

The principles of the Top-Down Approach

Starting with the Big Picture

Focus first on the overall goal or purpose of the project

Iterative Refinement

Continuously adjust and refine strategies based on feedback

Avoiding Focus on Details

Prevent getting overwhelmed by details too early in the process

Prioritizing and Planning

Determine importance of tasks,organizing efforts effectively

Breaking Down

Decompose the larger goal into smaller, more manageable

Focusing on Core Concepts First

Problem-solving, understand fundamental concepts

Effective Communication

Ensure everyone understands the overall vision and their role

Flexibility and Adaptability:

Be open to adjusting plans as new information or challenges arise

Agenda

1 **What's Top Down Approach**

What's Top Down && Bottom Up approach

How Top-Down && Bottom Up approach works

critical differences

2 **Why does it so important**

3 **How to Apply it in program**

4 **An Example of Top-down Approach in Programming**

bbv

What’s Top Down approach

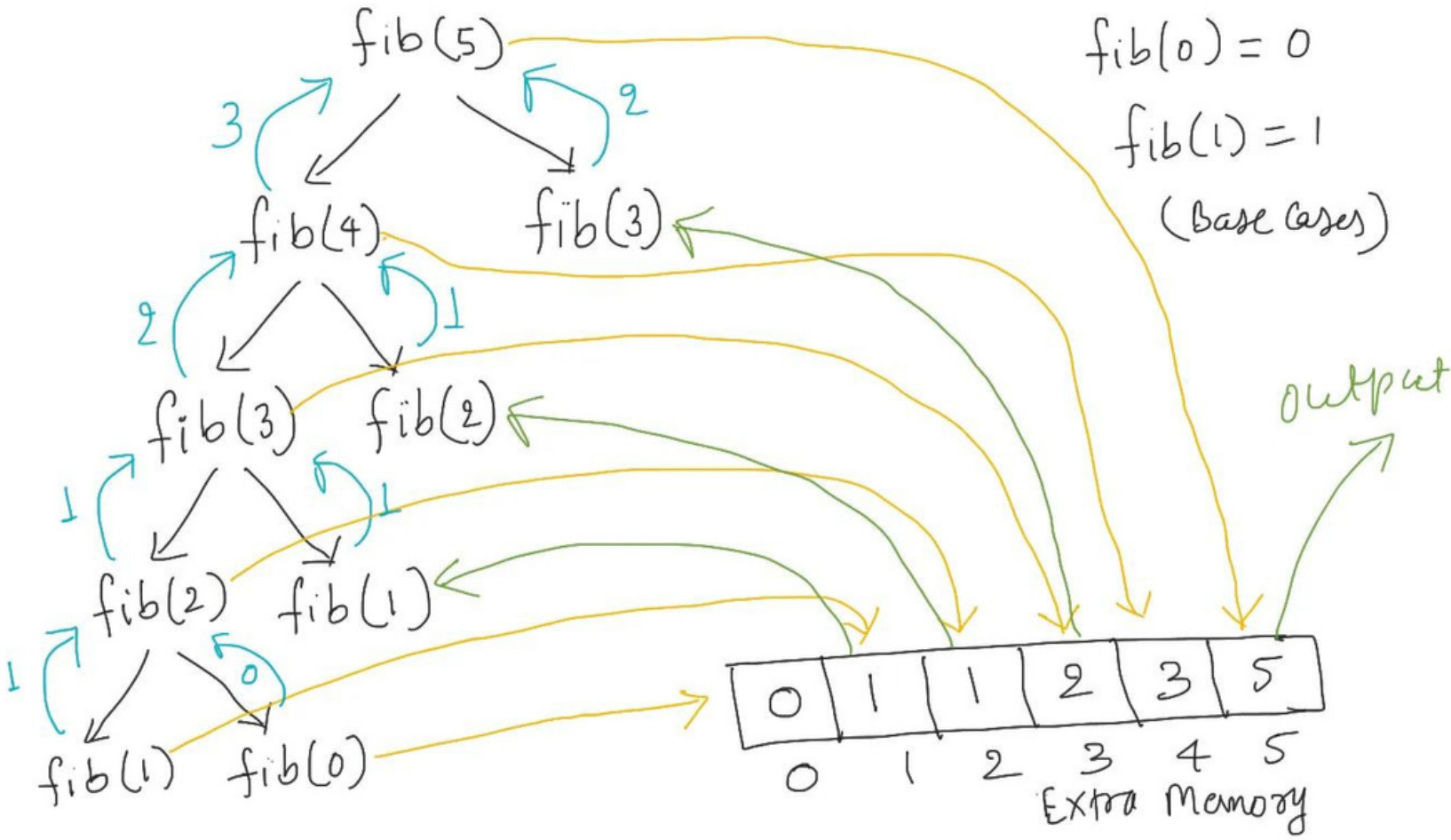
The Top-Down Approach is a strategic method that begins with identifying overarching goals and objectives, and then breaks these down into smaller, detailed components. It's widely used in project management and problem-solving to ensure that all actions align with the main objectives.

What’s Bottom Up approach

The Bottom-Up Approach is a method where you start with the smallest or simplest elements of a problem and gradually integrate them to form the complete solution. It's iterative, detail-focused, and often used in software development, where small sub-problems are solved first and their solutions are combined to address larger issues.

How top-down approach works?

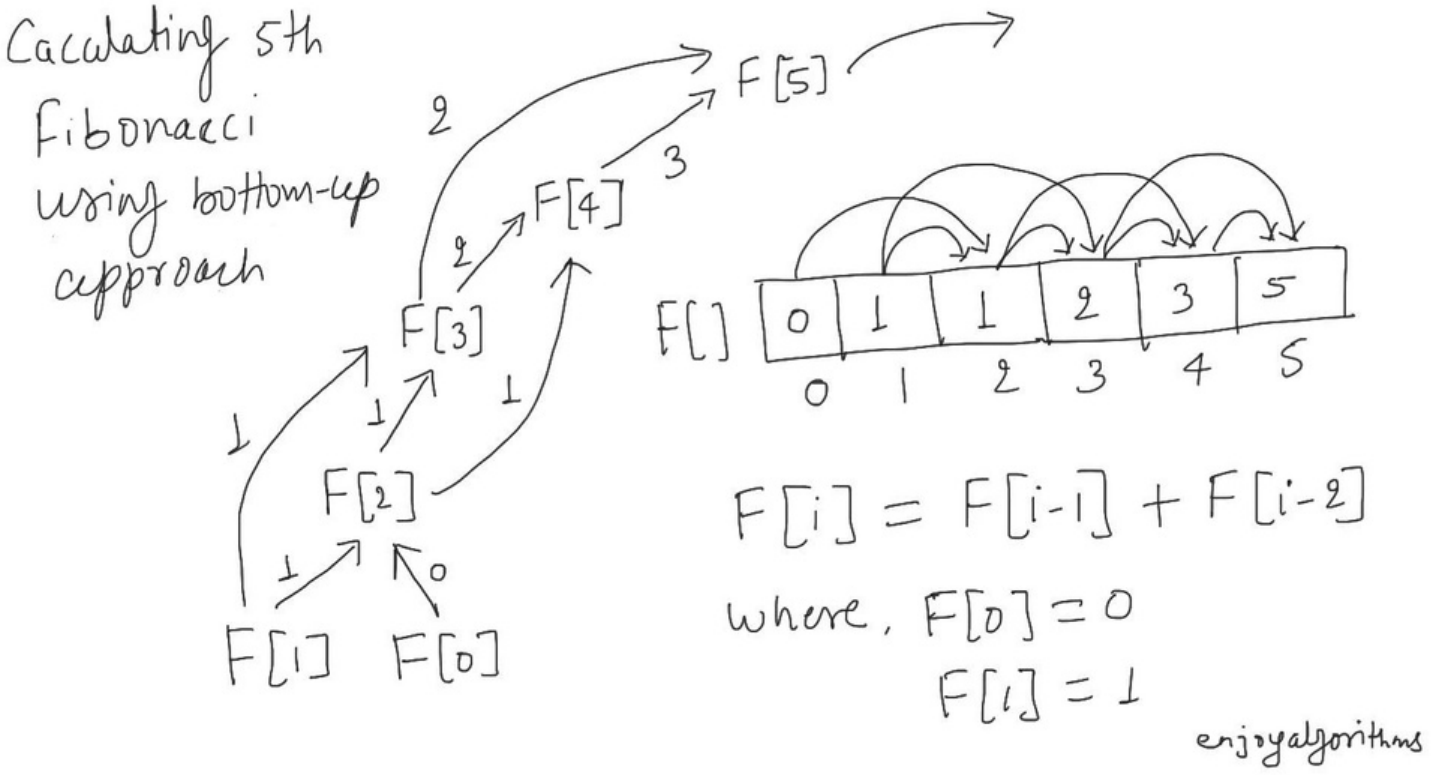
In the top-down approach, we implement the solution naturally using recursion but modify it to save the solution of each subproblem in an array or hash table. This approach will first check whether it has previously solved the subproblem. If yes, it returns the stored value and saves further calculations. Otherwise, top-down approach will calculate sub-problem solutions in the usual manner. We say it is the memoized version of a recursive solution, i.e., it remembers what results have been computed previously.



Visualization: Finding the 5th Fibonacci using top-down approach

How bottom-up approach works?

On another side, bottom-up approach is just the reverse but an iterative version of the top-down approach. It depends on a natural idea: solution of any subproblem depends only on the solution of smaller subproblems. So bottom-up approach sorts the subproblems by their input size and solves them iteratively in the order of smallest to largest. In other words, when solving a particular subproblem, bottom-up approach will first solve all of the smaller subproblems its solution depends upon and store their values in extra memory.

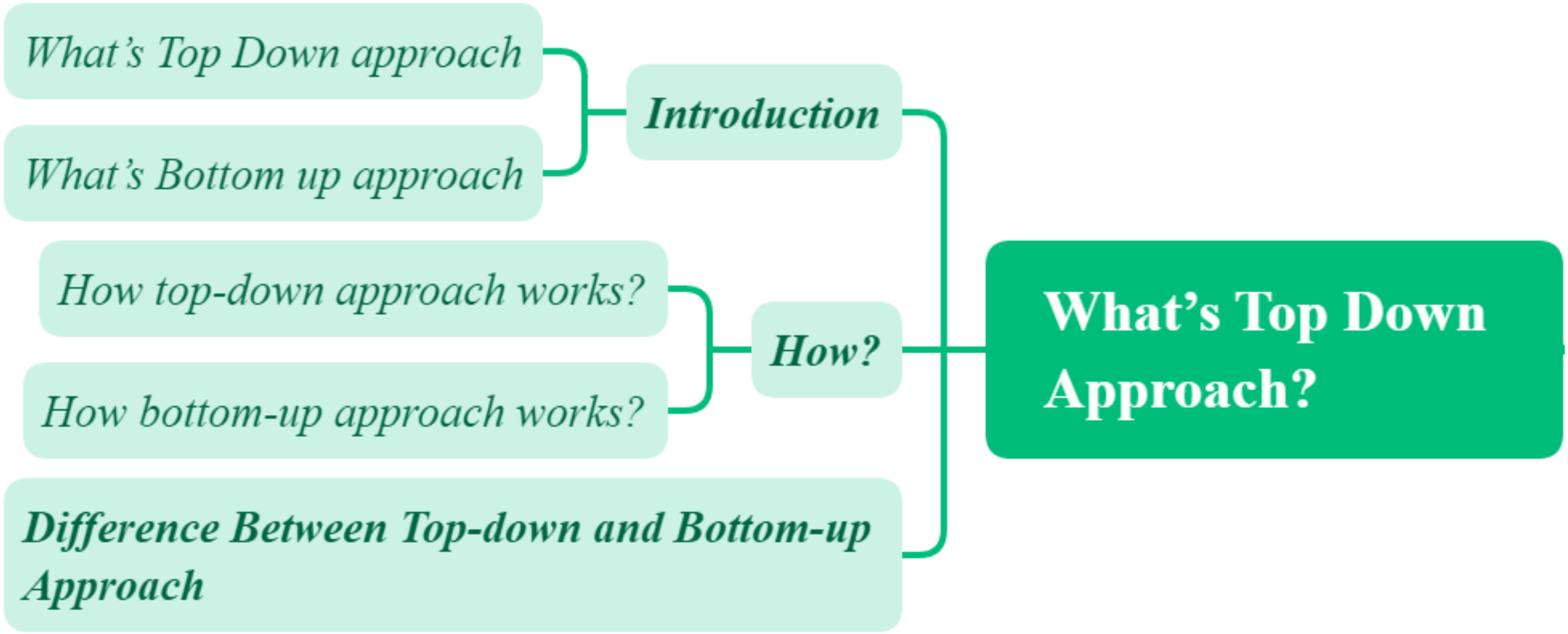


Visualization: Finding the 5th Fibonacci using bottom-up approach

Critical differences

Criteria	Top-Down Approach	Bottom-Up Approach
Problem-Solving Approach	Recursive, solving problems by breaking them down into smaller sub-problems.	Iterative, builds up the solution from smaller sub-problems.
Implementation Complexity	Easier, often involves adding an array or lookup table for memoization.	More complex, requires defining an iterative order and handling boundary conditions.
Performance	Slower due to the overhead of recursive calls.	Faster, as it avoids the overhead of recursive calls.
Space Overhead	Higher, due to the recursion call stack; risk of stack overflow in deep recursion.	Lower, generally doesn't involve recursion and uses iterative constructs.
Time Complexity Aspects	Same as bottom-up, except in cases with limited recursion.	Asymptotically the same as top-down.
Problem Solving Order	Starts from the large input size, solving from the base case upwards.	Begins with the base case, building solutions up to larger sub-problems.
Optimization Opportunities	Less scope for optimization, especially in reducing space complexity.	Greater potential for optimization in time and space complexity.

SUMMARY BY MINDMAP



Why does it so important

The Top-Down Approach is essential for ensuring that projects and problem-solving efforts are goal-oriented, well-organized, and strategically planned, leading to more effective and successful outcomes.

01 **Clear Vision and Objectives**
Starting with a clear understanding of the overarching goals

02 **Streamlined Planning**
Simplifies planning and makes complex projects more manageable

03 **Efficient Prioritization**
Tasks based on their relevance to the overarching goal

04 **Resource Optimization**
Aligning them with key project areas.

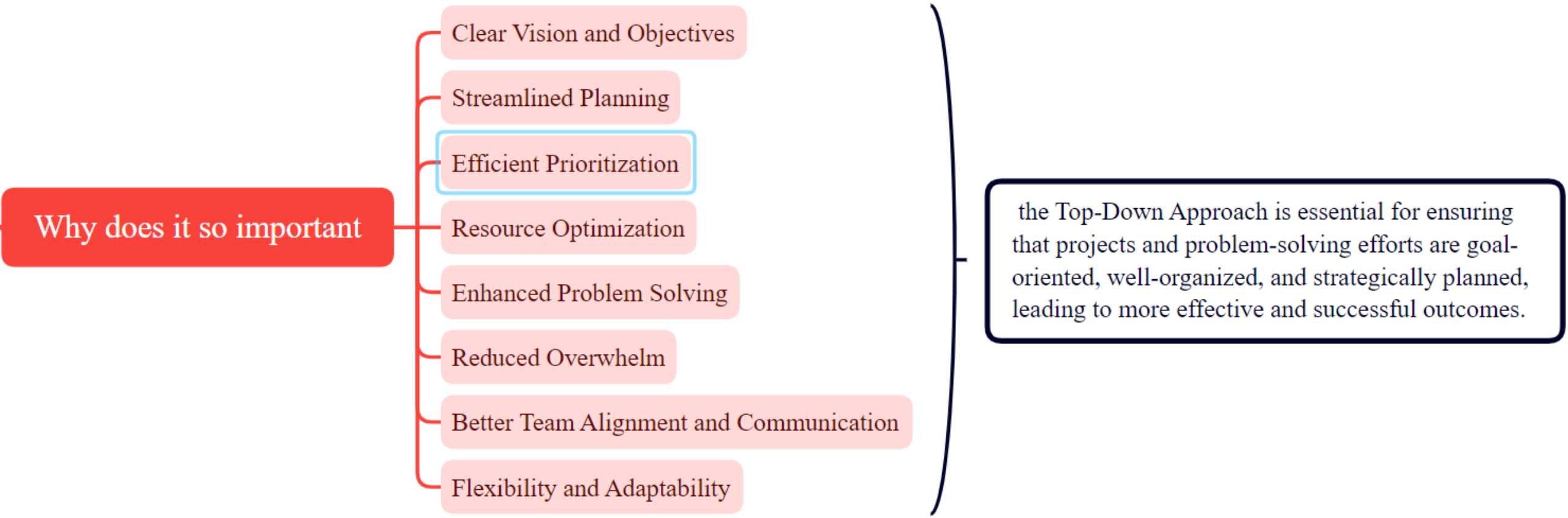
05 **Enhanced Problem Solving**
Focusing on the big picture first identify the root causes of issues

06 **Reduced Overwhelm**
It prevents getting mired in details too early, maintaining focus

05 **Better Team Communication**
Ensures everyone involved understands the ultimate goals

06 **Flexibility and Adaptability**
Reviews and adjustments based on new insights or feedback

SUMMARY BY MINDMAP



How to apply it in programing

1

Break down the method's logic into steps using comments, as shown in the example below.

2

Generate dependent methods, classes, enums, etc., used in stage 1
For now, just generate empty dependent methods or classes and don't bother implementing them during this stage.

3

Once have the code skeleton, implement dependent methods one by one and run the unit test.

bbv

AN EXAMPLE OF TOP-DOWN APPROACH IN PROGRAMMING

bbv

1. Task Model

```
public class Task
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }

    public Task(int id, string title, string description)
    {
        Id = id;
        Title = title;
        Description = description;
    }

    public override string ToString()
    {
        return $"Task ID: {Id}, Title: {Title}, Description: {Description}";
    }
}
```

AN EXAMPLE OF TOP-DOWN APPROACH IN PROGRAMMING

bbv

2. Task Manager

```
public class TaskManager
{
    private List<Task> tasks = new List<Task>();
    private int nextId = 1;

    public void AddTask(string title, string description)
    {
        tasks.Add(new Task(nextId++, title, description));
    }

    public void ViewTasks()
    {
        foreach (var task in tasks)
        {
            Console.WriteLine(task);
        }
    }

    public bool DeleteTask(int id)
    {
        var task = tasks.Find(t => t.Id == id);
        if (task != null)
        {
            tasks.Remove(task);
            return true;
        }
        return false;
    }
```

AN EXAMPLE OF TOP-DOWN APPROACH IN PROGRAMMING

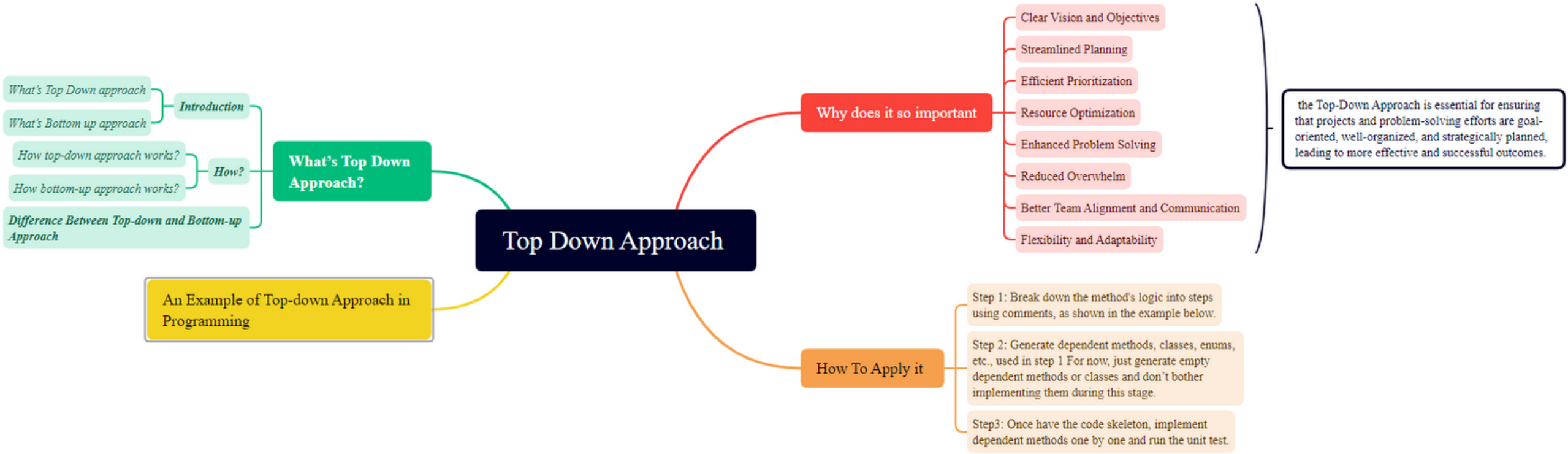
bbv

3. Main Programing

```
static void Main()
{
    bool running = true;
    while (running)
    {
        Console.WriteLine("1. Add Task\n2. View Tasks\n3. Delete Task\n4. Exit"
        Console.Write("Choose an option: ");
        var option = Console.ReadLine();

        switch (option)
        {
            case "1":
                AddTask();
                break;
            case "2":
                ViewTasks();
                break;
            case "3":
                DeleteTask();
                break;
            case "4":
                running = false;
                break;
            default:
                Console.WriteLine("Invalid option.");
                break;
```

SUMMARY


Top Down Approach
What's Top Down Approach?
Introduction
What's Top Down approach
What's Bottom up approach
How?
How top-down approach works?
How bottom-up approach works?
Difference Between Top-down and Bottom-up Approach
An Example of Top-down Approach in Programming
Why does it so important
Clear Vision and Objectives
Streamlined Planning
Efficient Prioritization
Resource Optimization
Enhanced Problem Solving
Reduced Overwhelm
Better Team Alignment and Communication
Flexibility and Adaptability
the Top-Down Approach is essential for ensuring that projects and problem-solving efforts are goal-oriented, well-organized, and strategically planned, leading to more effective and successful outcomes.
How To Apply it
Step 1: Break down the method's logic into steps using comments, as shown in the example below.
Step 2: Generate dependent methods, classes, enums, etc., used in step 1 For now, just generate empty dependent methods or classes and don't bother implementing them during this stage.
Step3: Once have the code skeleton, implement dependent methods one by one and run the unit test.

Thanks for listenning