

AoA Stimulation User Guide

Sum of Subsets Problem – Backtracking Approach

Group Members:

- 16010123012 – Aaryan Sharma
- 16010123017 – Aditey Kshirsagar
- 16010123023 – Aditya Baheti

Division: A

Batch: A1

Website: <https://sum-of-subsets.vercel.app/>

User Guide Documentation

1. SIMULATION OVERVIEW

This interactive visualization simulates the **Sum of Subsets Problem** using the **Backtracking Algorithm**. The objective is to find all subsets of given numbers that sum up to a target value.

Users can:

- Define the input set and target sum
- Step through the backtracking process
- Visually follow how subsets are explored
- View feasible solutions in real-time

The screenshot shows a web browser window with the URL `https://sum-of-subsets.vercel.app`. The page title is "Calorie-Based Meal Planner". The main content area is titled "Plan Your Meal" and includes a description: "The algorithm will find all possible meal combinations that add up to exactly the target calories. This demonstrates the **Sum of Subsets** algorithm, which uses backtracking to find all valid combinations."

The interface features a form for adding food items with columns for "Food Name", "Calories", and "Icon". Below the form, there is a list of food items with their respective calorie counts: Apple (100 kcal), Sandwich (250 kcal), Salad (150 kcal), Yogurt (200 kcal), and Protein Bar (300 kcal). Each item has a red "X" icon in the top right corner, indicating it can be removed. At the bottom, there is a "Target Calories" field set to "500 kcal" and two buttons: "Run Full Algorithm" and "Step-by-Step".

| Food Name | Calories | Icon |
|------------|----------|-------|
| e.g. Apple | e.g. 100 | Apple |

Food Items:

- Apple (100 kcal)
- Sandwich (250 kcal)
- Salad (150 kcal)
- Yogurt (200 kcal)
- Protein Bar (300 kcal)

Target Calories: 500 kcal

Buttons: Run Full Algorithm, Step-by-Step

2. KEY FEATURES

Problem Customization:

- Define your own input set (up to 10 elements)
- Enter a custom target sum
- Clear and reset input for new simulations

The screenshot shows a web application titled "Calorie-Based Meal Planner". The main interface is a "Plan Your Meal" form. At the top, it explains the algorithm: "The algorithm will find all possible meal combinations that add up to exactly the target calories. This demonstrates the **Sum of Subsets** algorithm, which uses backtracking to find all valid combinations." Below this, there is a table with columns for "Food Name", "Calories", and "Icon". The table contains the following items: Milk (150 kcal, Dairy icon), Sandwich (250 kcal, Sandwich icon), Salad (150 kcal, Salad icon), Yogurt (200 kcal, Yogurt icon), Protein Bar (300 kcal, Protein Bar icon), Banana (100 kcal, Banana icon), and Apple (100 kcal, Apple icon). Each item has a red "X" icon in the top right corner. To the right of the table is a "+ Add Food" button. Below the table, there is a "Target Calories:" label followed by a green box containing "500 kcal". At the bottom right, there are two buttons: "Run Full Algorithm" and "Step-by-Step".

Visualization Features:

- Step-by-step subset generation
- Highlighting current subset being evaluated
- Feasibility check animation for each subset
- Clear indicators for valid/invalid combinations

Controls:

- **Start:** Begin the simulation
- **Step:** Move through each recursive call manually
- **Prev/Next:** View the previous or next subset of the particular subset currently being checked.

Algorithm Execution

Execution Log

Exploring index 0, subset: [], sum: 0

Exploring index 1, subset: [Sandwich], sum: 250

Exploring index 2, subset: [Sandwich,Salad], sum: 400

Exceeded target (sum = 600 > 500)

Exceeded target (sum = 700 > 500)

Valid plan found:
[Sandwich,Salad,Banana]

Exploring index 3, subset: [Sandwich,Yogurt], sum: 450

Exceeded target (sum = 750 > 500)

Previous

Next Step

Results Summary

24

Total Recursive Calls

2

Valid Meal Plans

7

Pruned Paths

Valid Meal Combinations

1

Sandwich + Salad + Banana

500 kcal

2

Yogurt + Protein Bar

500 kcal

target calories: 500 kcal

Full Recursive Algorithm

Step-by-Step

Algorithm Execution

Step-by-Step Execution

Exploring index 0, subset: [], sum: 0

Exploring index 1, subset: [Sandwich], sum: 250

Exploring index 2, subset: [Sandwich,Salad], sum: 400

Previous

Next Step

Results Summary

3

Steps Executed

0

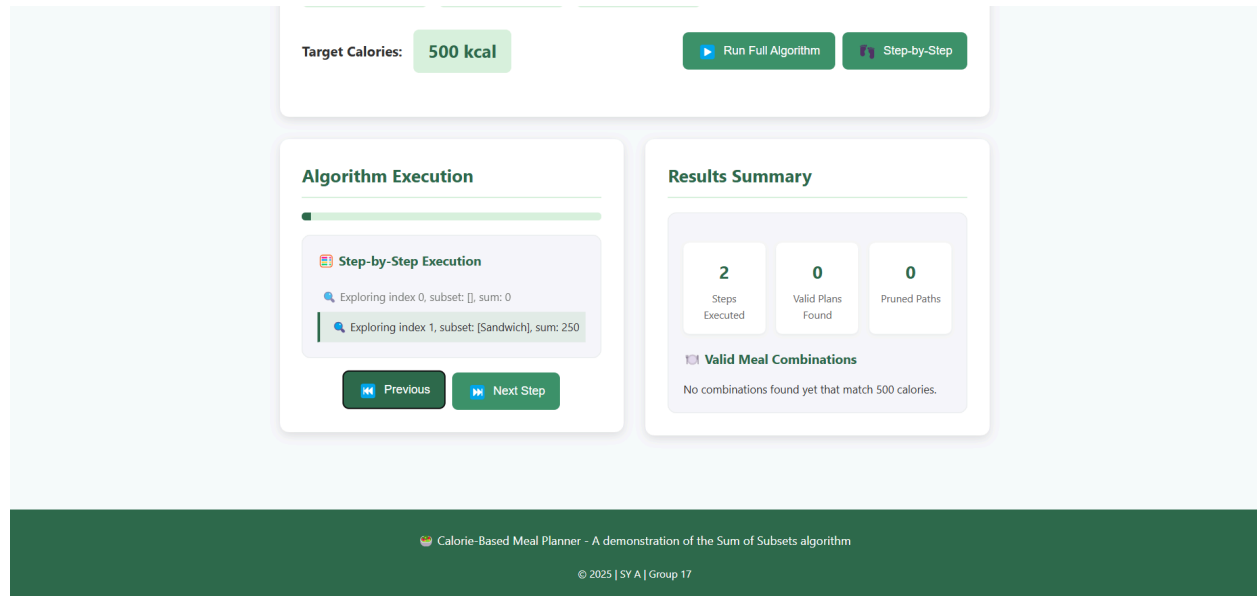
Valid Plans Found

0

Pruned Paths

Valid Meal Combinations

No combinations found yet that match 500 calories.



Solution Display:

- Valid subsets displayed in output section
- Real-time subset value display
- Count of valid solutions found
- Total recursive calls shown

Results Summary

64

Total
Recursive
Calls

8

Valid Meal
Plans

19

Pruned
Paths

Valid Meal Combinations

1 Sandwich + Salad + Banana
500 kcal

2 Sandwich + Salad + Apple
500 kcal

3 Sandwich + Banana + Milk
500 kcal

Results Summary

3 Sandwich + Banana + Milk
500 kcal

4 Sandwich + Apple + Milk
500 kcal

5 Salad + Yogurt + Milk
500 kcal

6 Salad + Banana + Apple +
Milk
500 kcal

7 Yogurt + Protein Bar
500 kcal

8 Protein Bar + Banana + Apple
500 kcal

3. HOW IT WORKS

Algorithm Flow:



- 1. Initialization:**
 - User enters set and target sum
 - System initializes recursive backtracking function
- 2. Backtracking Execution:**
 - Recursively include/exclude each element
 - Keep track of running sum
 - Store subset if it matches the target
- 3. Visualization:**
 - Current subset is highlighted
 - Real-time sum calculation
 - Valid subsets shown as output
 - Visual transition for decisions


User Interaction:


- Input set & target via form
- Control buttons for simulation flow
- Output updates as subset checks are completed


Plan Your Meal


The algorithm will find all possible meal combinations that add up to exactly the target calories.
This demonstrates the **Sum of Subsets** algorithm, which uses backtracking to find all valid combinations.


| Food Name | Calories | Icon | |
|---|---------------------------------------|--|--|
| <input type="text" value="e.g. Apple"/> | <input type="text" value="e.g. 100"/> |  Apple ▼ |  Add Food |


**Sandwich**
250 kcal


**Salad**
150 kcal

**Yogurt**
200 kcal


**Protein Bar**
300 kcal


**Banana**
100 kcal

**Apple**
100 kcal

**Milk**
150 kcal

Target Calories: **500 kcal**

 Run Full Algorithm

 Step-by-Step

4. TECHNICAL ASSUMPTIONS

- Input array size ≤ 10
- Minimum Calories is 100kcal.
- Target sum and elements must be integers
- Only unique subsets shown (no duplicates)
- UI optimized for desktop resolutions
- Chrome/Edge browsers recommended

5. IMPLEMENTATION DETAILS

Technologies Used:

- **HTML/CSS** for structure, style and responsiveness
- **JavaScript** for algorithm logic

Performance Considerations:

- Complexity: $O(2^n)$ subset generation
- Optimized rendering with React state management
- Recursive visual steps avoid blocking UI
- Memory-efficient subset tracking