Modeling Daily Healthy Meal Composition Algorithm Using Recursive Constraints and Combination with Repetition

Helena Kristela Sarhawa - 13524109
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: hkristela03@gmail.com , 13524109@std.stei.itb.ac.id

***Abstract*—Food is one of the fundamental human needs as a primary source of energy. When the composition of food consumed is not properly controlled, it may become a source of health problems. Planning a healthy and nutritionally balanced daily menu often poses a significant challenge. This paper proposes an algorithmic model for daily meal planning that considers both nutritional requirements and menu diversity over a given time period. The model utilizes the principle of combination with repetition to generate all possible meal compositions that satisfy the minimum and maximum nutritional thresholds based on the user's age. Furthermore, recursive constraints are applied to ensure that each daily menu differs from the previous day. Mathematical induction is employed to formally verify the correctness of the proposed model.**

Keywords—daily healthy menu, combination with repetition, recursive constraint, mathematical induction, meal planning

I. Introduction

Food is a fundamental human need that serves as a source of energy for performing daily activities. Excessive consumption or an unbalanced intake of nutrients can pose health risks, such as obesity, diabetes, or malnutrition. Therefore, maintaining a balanced and nutritious diet is essential and should be carefully managed.

In daily life, people often face difficulties in planning daily meals that meet nutritional requirements while remaining varied from day to day. Monotony in meal choices may decrease the motivation to maintain healthy eating habits. Moreover, not everyone has the time or expertise to manually calculate the nutritional content of meals.

This paper offers a solution through the design of an algorithmic model capable of automatically generating daily healthy meal compositions. The model applies principles of Discrete Mathematics, particularly combination with repetition, to construct all possible menu combinations from a predefined set of food ingredients. To ensure variation between days, recursive constraints are employed to prevent the selection of identical menus in succession. Mathematical induction is also applied as a means of proving that the model remains valid over a defined time period. With the development of this paper, it is expected that the model can serve as a foundation for creating practical and health-oriented meal planning systems, both for everyday use and in the form of digital meal planner applications. Furthermore, the model is expected to be applicable for widespread use, fostering healthier eating habits within the community.

II. Theoretical Foundations

A. *Recursive Functions*

Recursion refers to an object defined in terms of itself. A recursive function is a rule for generating a value based on a previous value. A recursive function consists of two parts: the base case and the recurrence. The base case contains a value of the function that is explicitly defined. The base serves to terminate the recursion and acts as its stopping point. Meanwhile, the recurrence part defines the function in terms of itself by using the values of preceding elements. Examples of such preceding elements include a_0, a_1, a_2, …. a_{n-1}.

$$f(n) = \begin{cases} 3 & ,n = 0 \quad \text{basis} \\ 2f(n-1)+4 & ,n > 0 \quad \text{rekurens} \end{cases}$$

Fig. 2.1 Example of Recursive Function (Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/10-Deretan,%20rekursi-dan-relasi-rekurens-(Bagian1)-2024.pdf)

Recursive functions are used to ensure that the menu on day n is not identical to the menu on day n-1. The recursive function will form recursive constraints in the day-to-day iterations.

B. *Combinatorics*

Combinatorics is a branch of mathematics that deals with counting the number of ways or possible arrangements of objects without listing all possibilities one by one.

Combinatorics is based on two basic counting principles: the rule of product and the rule of sum. The rule of product is used when calculating the number of ways in which multiple conditions occur simultaneously. On the other hand, the rule of sum is used when calculating the number of ways in which multiple conditions may occur alternatively. The application of both rules is illustrated as follows:

a. Rule of Product

   Condition 1: a ways

   Condition 2: b ways

   Condition 1 **and** Condition 2: a × b ways

b. Rule of Sum

   Condition 1: a ways

   Condition 2: b ways

   Condition 1 **or** Condition 2: a + b ways

A permutation is the number of distinct sequences for arranging objects. A permutation is a specific application of the rule of product. A permutation of r from n elements is the number of possible sequences of r elements chosen from n distinct elements, assuming each element is unique.

$$P(n,r) = n(n-1)(n-2)\ldots(n-(r-1)) = \frac{n!}{(n-r)!}$$

Fig. 2.2 Definition of Permutation (Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/18-Kombinatorika-Bagian1-2024.pdf)

A specific form of permutation is called a combination. In a combination, all elements are treated as identical. Thus, the order in which elements appear is disregarded, as each element is considered equal. A combination of r from n elements is the number of ways to choose r elements from n elements without considering the order of selection.

$$\frac{n(n-1)(n-2)\ldots(n-(r-1))}{r!} = \frac{n!}{r!(n-r)!} = C(n, r)$$

Fig. 2.3 Definition of Combination (Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/18-Kombinatorika-Bagian1-2024.pdf)

Combinations can also be applied with repetition. The previous definition of combinations only applies when each selection slot may be filled with at most one object. To allow for more than one object in a slot, the concept of combination with repetition is used. In the following definition, n denotes the number of available slots, and r denotes the number of objects to be distributed.

$$C(n + r - 1, r) = C(n + r - 1, n - 1)$$

Fig. 2.4 Definition of Combination with Repetition (Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/19-Kombinatorika-Bagian2-2024.pdf)

The concept of combination with repetition will be used to construct all possible combinations of food ingredients that meet nutritional constraints.

C. *Mathematical Induction*

Mathematical induction is a method for proving the truth of a statement that involves integers. Mathematical induction helps reduce the steps needed for proof into a limited number of steps. Without induction, all integers would have to be verified one by one, which is inefficient and time-consuming.

In this paper, simple mathematical induction will be used. Simple induction allows a statement to be proven using only two main steps. The first step is called the base case, and the second is the inductive step. The base case serves as the starting point at which the statement is accepted as true. Meanwhile, the inductive step demonstrates that the statement holds for element n + 1, assuming it is true for element n, where n is the base case. The inductive step uses an assumption that the statement is true—this assumption is called the inductive hypothesis. If both steps are proven valid, it can be concluded that the statement is true for all integers greater than the base case.

Mathematical induction works like a domino effect. To knock down an entire sequence of dominoes, one only needs to push the first piece so that the rest follow. With this mechanism, mathematical induction only needs to prove that the first element is valid and that the following one is also valid. In this paper, mathematical induction will be used to prove that the menu selection model is correct for any number of days.

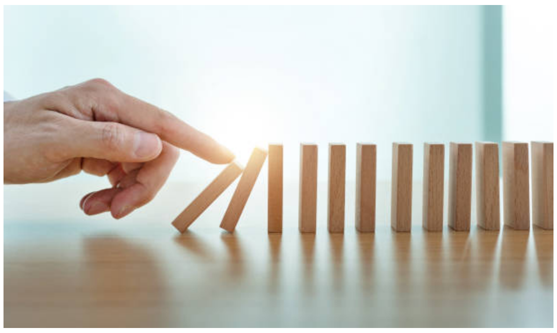


Fig. 2.5 Domino Effect (Source: https://images.app.goo.gl/3xepumGFCtJaNUcm8)

III. Model Design

A. *Problem Definition*

The main problem addressed in this model is how to construct the composition of a daily food menu that is healthy and nutritionally balanced, while taking into account the differing nutritional needs of each individual. These nutritional needs are influenced by age and gender factors. Therefore, the developed model must be able to adjust the menu composition based on the user profile.

This model refers to Angka Kecukupan Energi (AKE) as stipulated in Peraturan Menteri Kesehatan Republik Indonesia Nomor 28 Tahun 2019. The document provides the recommended daily nutritional requirements based on age and gender categories.

TABLE 3.1

Category	Age/Condition	Energy (kcal)	Protein (g)	Fat (g)
Infant	0–5 months	550	9	31
Infant	6–11 months	800	15	35
Toddler	1–3 years	1350	20	45
Child	4–6 years	1400	25	50
Child	7–9 years	1650	40	55
Boy	10–12 years	2000	50	65
Girl	10–12 years	1900	55	65
Boy	13–15 years	2400	70	80
Girl	13–15 years	2050	65	70
Boy	16–18 years	2650	75	85
Girl	16–18 years	2100	65	70
Man	19–29 years	2650	65	75
Woman	19–29 years	2250	60	65
Man	30–49 years	2550	65	70
Woman	30–49 years	2150	60	60
Man	50–64 years	2150	65	60
Woman	50–64 years	1800	60	50
Man	65–80 years	1800	64	50
Woman	65–80 years	1550	58	45
Man	80+ years	1600	64	45
Woman	80+ years	1400	58	40

TABLE 3.2

Category	Age/Condition	Carbohydrate (g)	Fiber (g)	Water (ml)
Infant	0–5 months	59	0	700
Infant	6–11 months	105	11	900
Toddler	1–3 years	215	19	1150
Child	4–6 years	220	20	1450
Child	7–9 years	250	23	1650
Boy	10–12 years	300	28	1850
Girl	10–12 years	280	27	1850
Boy	13–15 years	350	34	2100
Girl	13–15 years	300	29	2100
Boy	16–18 years	400	37	2300
Girl	16–18 years	300	29	2150
Man	19–29 years	430	37	2500
Woman	19–29 years	360	32	2350
Man	30–49 years	415	36	2500
Woman	30–49 years	340	30	2350
Man	50–64 years	340	30	2500
Woman	50–64 years	280	25	2350
Man	65–80 years	275	25	1800
Woman	65–80 years	230	22	1550
Man	80+ years	235	22	1600
Woman	80+ years	200	20	1400

The model will receive input from the user:

a. age (in months or years) and

b. gender.

Based on the information provided by the user, the model will extract the nutritional requirements for six primary nutritional elements, namely energy (kcal), protein (grams), fat (grams), carbohydrates (grams), fiber (grams), and water (ml). The next step, from a set of food ingredients each of which has a defined nutritional content per serving unit, the model will generate combinations of these ingredients such that the total nutritional content of the menu falls within the target range and the resulting menu differs from day to day.

B. Food Ingredient Data Model

Once the daily nutritional requirements are defined, the next step is to construct a data model containing a set of food ingredients as the basic components of menu formation. Each food ingredient has a nutritional value per serving unit, which is used as the building block in the composition of the daily menu.

Each food ingredient is represented as a six-dimensional vector, for example, b_i = [calories$_i$, protein$_i$, fat$_i$, carbohydrate$_i$, fiber$_i$, water$_i$]. Thus, one food menu can be formed from one or more combinations of food ingredients, and the total nutritional value can be calculated through the summation of the component vectors.

The following table presents examples of food ingredient data along with their nutritional contents per serving unit:

TABLE 3.3

Name	Energy (kcal)	Protein (g)	Fat (g)
White rice	175	3.0	0.3
Fried tofu	132	7.8	10.0
Boiled spinach	18	2.0	0.4
Boiled carrot	33	0.8	0.1
Apple	52	0.3	0.2
Fried tempeh	118	6.8	6.0
Grilled chicken breast	165	31.0	3.6
Boiled potato	87	2.0	0.1
Banana	89	1.1	0.3
Boiled broccoli	35	2.4	0.4
Whole wheat bread	247	13.0	4.2
Boiled water spinach	19	2.1	0.2
Grilled beef	250	26.0	15.0
Orange	47	0.9	0.1
Papaya	43	0.5	0.3

TABLE 3.4

Name	Carbohydrate (g)	Fiber (g)	Water (ml)
White rice	40.6	0.2	72
Fried tofu	2.4	0.3	65
Boiled spinach	3.4	2.2	91
Boiled carrot	7.7	3.0	88
Apple	14.0	2.4	85
Fried tempeh	7.5	1.3	64
Grilled chicken breast	0.0	0.0	70
Boiled potato	20.1	1.8	77
Banana	23.0	2.6	74
Boiled broccoli	7.2	3.3	89
Whole wheat bread	41.0	6.0	35
Boiled water spinach	3.7	1.9	90
Grilled beef	0.0	0.0	55
Orange	11.8	2.4	87
Papaya	10.8	1.7	88

C. *Menu Representation as Integer Vectors*

After defining the daily nutritional requirements and the food ingredient data model, the next step is to represent the food menu as a mathematical structure that enables algorithmic processing. In this model, each daily menu is represented as an n-dimensional integer vector, where n is the number of available food ingredients. Suppose there are n food types (b_1, b_2, ..., b_n), a daily menu M is represented as $M = [x_1, x_2, ..., x_n]$ where x_i represents the number of servings of food ingredient (b_i) used in that menu. A value of $x_i = 0$ indicates that the i-th ingredient is not used in that menu. This reflects that one menu uses only a subset of all available ingredients.

Each food ingredient (b_i) has six nutritional attributes. Thus, the menu vector M will be operated with a nutritional matrix to obtain the total nutritional content of the menu. Let A be a 6 × 30 matrix containing energy, protein, fat, carbohydrate, fiber, and water values for each food item. Then $T = A \cdot M^T$ will produce a 6-dimensional vector T that holds the total nutritional contents of menu M, namely T = [Total Energy, Total Protein, ..., Total Water].

An example of the application is as follows. Four food ingredients are selected, namely rice (b_1), chicken (b_2), spinach (b_3), and papaya (b_4) out of a total of 30 food items. A possible menu generated is $M = [2, 1, 1, 1, 0, 0, ..., 0]$, which means the menu consists of 2 servings of rice, 1 serving of chicken, 1 serving of spinach, and 1 serving of papaya, while the remaining 26 ingredients are not used. The total nutritional content is calculated by summing the product of each portion with the respective nutritional content of each ingredient.

The use of vector representation allows for the evaluation of nutritional feasibility through simple linear algebra operations, generation of all possible menus using the principle of combination with repetition, and the application of constraints between daily menus.

D. *Daily Menu Generation Model*

After the menu is represented as a vector of serving quantities of food ingredients, the next step is to construct an algorithmic model to generate daily menus that meet nutritional requirements and maintain inter-day variation. The menu generation model is carried out in three main stages, namely enumeration of all valid candidate menus, application of recursive constraints between days, and scheduling of menus for a specified number of days.

a. Step 1: Enumerating Valid Menu Combinations

The first step is to generate all possible combinations of food ingredients that form a single menu. Since one food ingredient may be used more than once, the principle of combination with repetition is applied. Each candidate menu is represented as a vector $M = [x_1, x_2, ..., x_n]$ composed of the number of servings of each food item. For each candidate menu, the total nutritional content is calculated with $T = A \cdot M^T$, then filtered with the condition that $T_j \leq max_j$ for all components j (where j is a nutritional attribute considered), where T_j is the total amount of nutrient j from the menu, and max_j is the upper bound of daily nutritional needs based on Angka Kecukupan Energi (AKE) that has been increased by a tolerance of 1.25% from the standard AKE value. Menus that meet these criteria will be stored in the candidate menu set.

b. Step 2: Recursive Constraints Between Days

To maintain menu variation between days, recursive constraints are applied. This means that the menu selection for day t depends on the menu of day t–1 and may not be identical in vector structure.

Given two menus M^t dan M^{t-1}, the constraint applied is that the menus must not be identical ($M^t \neq M^{t-1}$).

c. Step 3: Recursive Menu Scheduling

After all candidate menus are stored in the set M, the scheduling of menus for N days is performed. This is done recursively, by selecting one menu from M that does not violate the constraint with the previous menu, and continuing the process until the desired number of days is reached.

Listing 1. Function schedule_menu

```
def schedule_menu(N, valid_menus, window=400):
    result = []
    for t in range(N):
        for menu in valid_menus:
            is_unique_recently = 
all(is_different_enough(menu, result[prev]) for prev 
in range(max(0, t - window), t))
            if t == 0 or is_unique_recently:
                result.append(menu)
```

```
                    break
            return result
```

E. Algorithm Overview

The following flowchart illustrates the steps carried out by the algorithm in a structured manner.

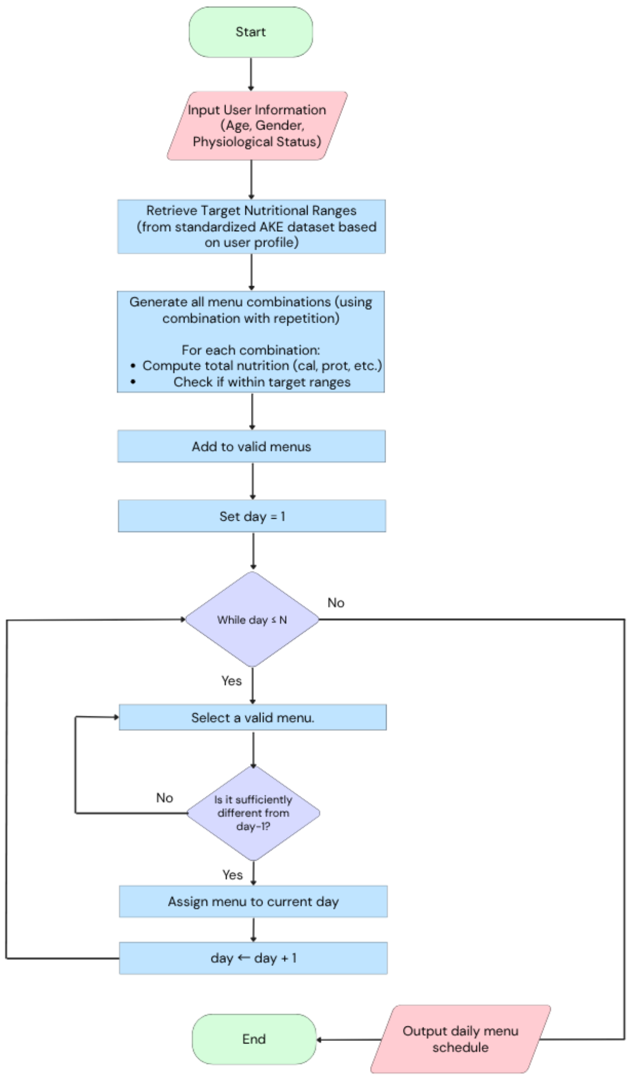

Start
Input User Information (Age, Gender, Physiological Status)
Retrieve Target Nutritional Ranges (from standardized AKE dataset based on user profile)
Generate all menu combinations (using combination with repetition)
For each combination:
• Compute total nutrition (cal, prot, etc.)
• Check if within target ranges
Add to valid menus
Set day = 1
While day ≤ N
No
Yes
Select a valid menu.
Is it sufficiently different from day-1?
No
Yes
Assign menu to current day
day ← day + 1
End
Output daily menu schedule

Fig. 3.1 Flowchart

IV. Simulation and Results

A. Implementation Overview

The program is implemented in Python using a modular structure. Two external data files are used as inputs:

a. food_ingredients.csv: Contains food ingredients along with their nutritional values (energy, protein, fat, carbohydrate, fiber, and water) per unit. The full list of ingredients and their corresponding nutritional information is provided in Table 3.3 and Table 3.4, which serve as the basis for evaluating the nutritional content of each generated menu during the simulation.

b. nutrition_requirements_with_range.csv: Contains daily nutritional requirement ranges for various user profiles based on age, gender, and physiological status. These ranges, as detailed in Table 3.1 and Table 3.2, are used as constraints in the menu generation algorithm to ensure that each simulated menu meets the personalized dietary needs of the user.

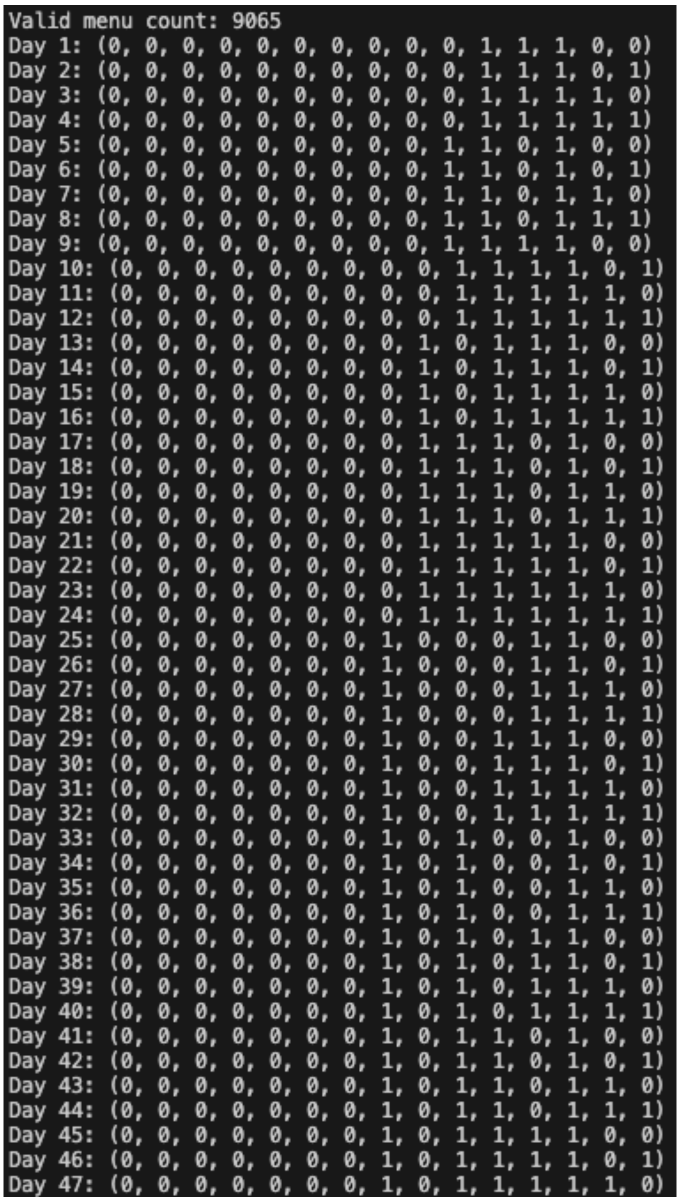
B. Code Implementation

Listing 2. Daily Menu Scheduling Program Based on Nutritional Constraints

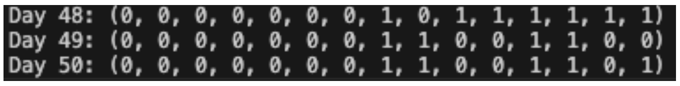
```
import pandas as pd
from itertools import product


# Read external CSV files
nutrition_df                                                    =
pd.read_csv("nutrition_requirements_with_range.csv")
food_df = pd.read_csv("food_ingredients.csv")


# Get the nutritional target bounds from user profile
def get_nutrition_bounds(category, age_condition):
    row = nutrition_df[
        (nutrition_df["Category"] == category) &
        (nutrition_df["Age/Condition"] == age_condition)
    ].iloc[0]
    bounds = {
        "Energy       (kcal)":        (row["Energy_min"],
row["Energy_max"]),
        "Protein        (g)":         (row["Protein_min"],
row["Protein_max"]),
        "Fat (g)": (row["Fat_min"], row["Fat_max"]),
        "Carbohydrate   (g)":   (row["Carbohydrate_min"],
row["Carbohydrate_max"]),
        "Fiber          (g)":           (row["Fiber_min"],
row["Fiber_max"]),
        "Water         (ml)":           (row["Water_min"],
row["Water_max"]),
    }
    return bounds


# Check if a menu is valid according to nutrition bounds
def is_valid_menu(menu, food_data, bounds):
    total = {k: 0 for k in bounds}
    for idx, portion in enumerate(menu):
        for nutrient in bounds:
            total[nutrient]                                  +=
food_data.iloc[idx][nutrient] * portion
    for nutrient, (_, max_val) in bounds.items():
        if total[nutrient] > max_val:
            return False
    return True


# Check if two menus are sufficiently different (at least 1 item differs)
def is_different_enough(menu1, menu2):
    return sum([abs(a - b) for a, b in zip(menu1, menu2)]) >= 1


# Schedule menus for N days, ensuring no identical days
def schedule_menu(N, valid_menus, window=400):
    result = []
    for t in range(N):
        for menu in valid_menus:
            is_unique_recently = all(is_different_enough(menu, result[prev]) for prev in range(max(0, t - window), t))
            if t == 0 or is_unique_recently:
                result.append(menu)
                break
    return result


def has_all_required_categories(menu, food_data, required_categories):
    selected = [food_data.iloc[i]["Category"] for i, portion in enumerate(menu) if portion > 0]
    return all(cat in required_categories)


# Main function to generate daily healthy menu schedule
def generate_menu_schedule(category, age_condition, total_days, max_portion=1):
    bounds = get_nutrition_bounds(category, age_condition)
    food_data = food_df.copy()
    n = len(food_data)
    required_categories = ["Carbohydrate", "Protein", "Vegetable"]
    valid_menus = []
    max_check = 20000
    min_ingredients_used = 3
    for i, combo in enumerate(product(range(max_portion + 1), repeat=n)):
        if i >= max_check:
            break
        if sum(combo) == 0:
            continue
        if sum(1 for x in combo if x > 0) < min_ingredients_used:
            continue
        if not has_all_required_categories(combo, food_data, required_categories):
            continue
        if is_valid_menu(combo, food_data, bounds):
            valid_menus.append(combo)
        if len(valid_menus) >= 20000:
            break
    print("Valid menu count:", len(valid_menus))
    return schedule_menu(total_days, valid_menus)
```

C. Simulation Results

The simulation was conducted for various user profiles, each defined by different age and gender combinations that determine their respective nutritional requirements. The output of each simulation is a daily menu represented as a vector, where each element corresponds to the portion count of a specific food ingredient. The position of each element in the vector follows the order of ingredients as listed in the food_ingredients.csv file. For example, if the third element in the vector is 2, it means that the third ingredient in the CSV file is included in the menu with 2 portions. This vectorized representation allows for efficient evaluation of nutritional values by multiplying each portion with the corresponding nutrient content per ingredient.

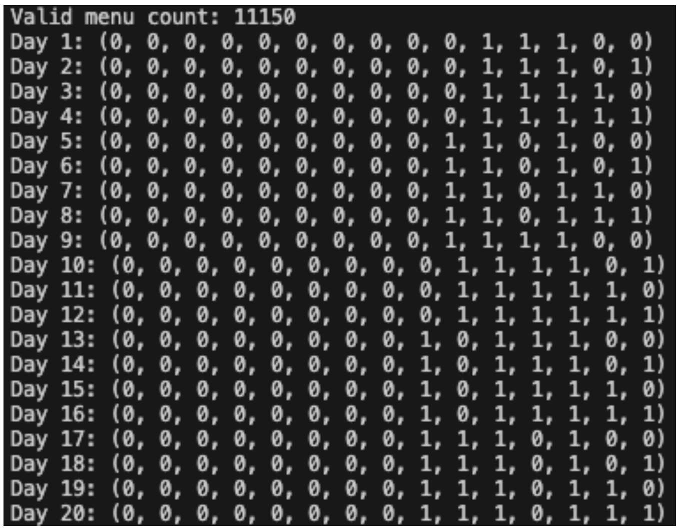


```
Valid menu count: 9065
Day 1: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)
Day 2: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1)
Day 3: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0)
Day 4: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
Day 5: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0)
Day 6: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1)
Day 7: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0)
Day 8: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1)
Day 9: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0)
Day 10: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1)
Day 11: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0)
Day 12: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
Day 13: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0)
Day 14: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1)
Day 15: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0)
Day 16: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1)
Day 17: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0)
Day 18: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1)
Day 19: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0)
Day 20: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1)
Day 21: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0)
Day 22: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1)
Day 23: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0)
Day 24: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1)
Day 25: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0)
Day 26: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1)
Day 27: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0)
Day 28: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1)
Day 29: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0)
Day 30: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1)
Day 31: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0)
Day 32: (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1)
Day 33: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0)
Day 34: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1)
Day 35: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0)
Day 36: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1)
Day 37: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0)
Day 38: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1)
Day 39: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0)
Day 40: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1)
Day 41: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0)
Day 42: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1)
Day 43: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0)
Day 44: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1)
Day 45: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0)
Day 46: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1)
Day 47: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0)
```



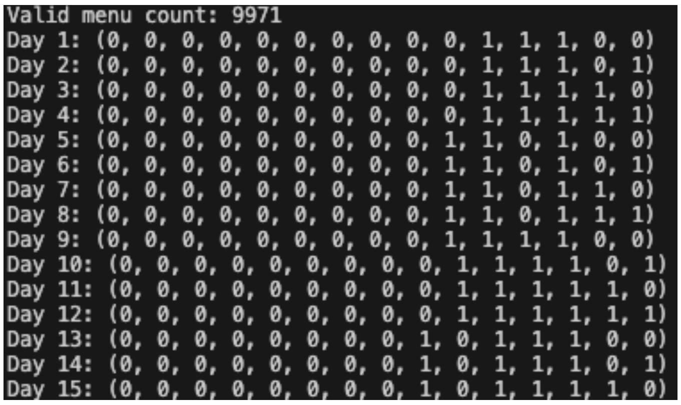
```
Day 48: (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1)
Day 49: (0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0)
Day 50: (0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1)
```

Fig. 4.1 Simulation Result for Woman Aged 30–49 Years



```
Valid menu count: 11150
Day 1: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)
Day 2: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1)
Day 3: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0)
Day 4: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
Day 5: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0)
Day 6: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1)
Day 7: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0)
Day 8: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1)
Day 9: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0)
Day 10: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1)
Day 11: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0)
Day 12: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
Day 13: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0)
Day 14: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1)
Day 15: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0)
Day 16: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1)
Day 17: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0)
Day 18: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1)
Day 19: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0)
Day 20: (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1)
```

Fig. 4.2 Simulation Result for Boy Aged 16–18 Years



```
Valid menu count: 9971
Day 1: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)
Day 2: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1)
Day 3: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0)
Day 4: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
Day 5: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0)
Day 6: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1)
Day 7: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0)
Day 8: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1)
Day 9: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0)
Day 10: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1)
Day 11: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0)
Day 12: (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
Day 13: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0)
Day 14: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1)
Day 15: (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0)
```

Fig. 4.3 Simulation Result for Man Aged 50–64 Years

V. MODEL VALIDATION

To ensure that the schedule_menu(N, valid_menus) algorithm can generate a valid and non-identical daily food menu for every day over a period of N days, we apply a simple mathematical induction proof on the number of days N.

a. Base Case (N = 1)

For N = 1, the algorithm begins by selecting the first available menu from the pre-generated set valid_menus. Each menu in this set has already been verified for nutritional validity (within upper bounds only) and completeness of required food categories (e.g., carbohydrate, protein, vegetable) through the function is_valid_menu. Since there is no previous menu to compare against, the uniqueness condition automatically holds. Therefore, the algorithm can generate one valid menu for the first day, and the base case is satisfied.

b. Inductive Hypothesis

Assume that for some k ≥ 1, the algorithm successfully schedules valid and non-identical menus for k days, resulting in a set of menus {M_1, M_2, ..., M_k}, where each menu satisfies all nutritional and categorical constraints and differs from the others in at least one component.

c. Inductive Step

We now show that the algorithm can also generate a valid menu for day k+1 that is not identical to any of the previous k menus. At iteration k+1, the schedule_menu function iterates through all candidates in valid_menus, checking whether each candidate menu differs sufficiently from recent selections using the is_different_enough function. Because the set valid_menus is large (up to 20,000 elements) and generated to include various valid combinations, the algorithm is highly likely to find a new valid menu that satisfies all constraints and differs from menus in previous days. Once such a menu is found, it is appended to the result list. Therefore, by the principle of mathematical induction, the algorithm is able to construct a schedule of N valid and non-identical daily menus for any N ≥ 1, as long as the candidate set is sufficiently large and diverse.

VI. CONCLUSION

This paper develops an algorithmic model for generating daily healthy meal plans based on individual nutritional needs and menu variation across days. Combination with repetition is used to enumerate all possible menus that meet daily nutritional limits, while recursive constraints are applied to prevent identical menus on consecutive days. The implementation is carried out in Python using a modular structure and two external data sources: a list of food ingredients with nutritional values and nutritional requirements categorized by age and gender.

Simulations show that the algorithm can produce daily meal schedules that are nutritionally valid and non-identical across days. However, results also indicate a limitation in the selection process. Despite the large number of valid menus, those selected for the initial days tend to follow similar food composition patterns. This is due to a selection strategy that processes menus in order without any shuffling mechanism.

Further development is needed to improve the quality of the generated menu schedule, particularly in terms of variety in ingredient composition. Approaches such as controlled random selection or distribution of menu structures based on vector similarity may be considered to ensure the algorithm is not only nutritionally valid but also adaptive to long-term variation needs.

VII. Appendix

The attached document in CSV format contains the nutritional content of food ingredients and the nutritional requirements for various user profiles. A video explaining this paper can be accessed here.

VIII. Acknowledgment

Upon the completion of this paper, the author extends sincere appreciation to:

1. God Almighty, for His blessings and guidance throughout the writing process;
2. Dr. Rinaldi Munir, M.T., lecturer of the K-1 IF1220 Discrete Mathematics course, for his valuable guidance and knowledge; and
3. The author's parents, for their unwavering support and encouragement.

References

[1] Kementerian Kesehatan Republik Indonesia, Direktorat Jenderal Kesehatan Masyarakat, Direktorat Gizi Masyarakat. 2018. *Tabel Komposisi Pangan Indonesia*. Jakarta: Kementerian Kesehatan RI.

[2] Munir, Rinaldi. 2024. "Deretan, rekursi, dan relasi rekurens (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/10-Deretan,%20rekursi-dan-relasi-rekurens-(Bagian1)-2024.pdf . Accessed 18 June 2025 at 18:08.

[3] Munir, Rinaldi. 2024. "Induksi matematika (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/08-Induksi-matematik-bagian1-2024.pdf . Accessed 19 June 2025 at 13:03.

[4] Munir, Rinaldi. 2024. "Kombinatorika (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/18-Kombinatorika-Bagian1-2024.pdf . Accessed 18 June 2025 at 19:48.

[5] Munir, Rinaldi. 2024. "Kombinatorika (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/19-Kombinatorika-Bagian2-2024.pdf . Accessed 19 June 2025 at 12:28.

[6] Wardani, Agustin Tri. "Berapa Kebutuhan Nutrisi Per Hari? Simak Panduan Berikut Ini..." *Kompas.com*, https://health.kompas.com/read/24A07063000468/berapa-kebutuhan-nutrisi-per-hari-simak-panduan-berikut-ini-. Accessed 19 June 2025 at 15:19.

Statement of Originality

I hereby declare that this paper I have written is my own work, not an adaptation or translation of someone else's paper, and not plagiarism.

Bandung, 20th June 2025

[illegible]

Helena Kristela Sarhawa
13524109