Chapter 7
Examples of the Main Sub-Disciplines

ABSTRACT

The chapter explores the significant subdisciplines of combinatorics through concrete examples. Problems such as combinatorial design applied to smartphones and restaurant menus, Kirkman's fifteen-student puzzle, and the use of Steiner systems in agriculture are presented. The chapter also discusses Gardner's square packing problem, restaurant code assignment using coding theory, and dining room layout planning using geometric combinatorics. The chapter includes examples of graph coloring for radio frequency assignment, the art gallery problem, and the use of Stirling numbers in algebraic combinatorics. Finally, graph theory examples are provided for scheduling home deliveries and using Hasse diagrams in project management and partition representation. It is noted that Hasse diagrams are widely used in discrete mathematics to visualize ordering relationships between elements without explicitly showing all redundant connections. These examples highlight how combinatorics offers powerful tools for solving optimization and organization problems in diverse contexts.

EXAMPLES OF THE MAIN SUB-DISCIPLINES

Design is the central core of human creation

Steve Jobs

Exploring all the sub-disciplines of *Combinatorial Design* means delving into a field of vast complexity. However, it's possible to illustrate the topic's scope by limiting the discussion to a few concrete examples.

Combinatorial Design is a valuable methodology for designing and optimizing systems and products (Colbourn & Van Oorschot, 1989; Stinson, 2008). A practical example is the design of a new smartphone. First, we need to identify the main variables that influence the product's features, such as processor type, RAM size, camera resolution, and battery life. Using Combinatorial Design, we can explore different combinations of these variables. In a lab setting, performance can be tested, and based on the goals, the most promising combination can be selected.

To calculate the total of possible combinations, we multiply the number of options for each variable.

For example, if the variables are:

Processor: (A, B, C)

RAM: (4GB, 8GB, 16GB)

Camera: (12MP, 24MP)

Battery: (3000mAh, 4000mAh)

The total number of possible combinations is 36. The final evaluation can then be done using a formula that weights the various components, considering factors such as cost.

Program 57 smartphone design

DOI: 10.4018/979-8-3373-3089-1.ch007

```
import itertools
# Definition of variables and costs
processors = {
'A': 55, # Cost in monetary units
'B': 60,
'C': 50
}
ram_memory = {
4: 20,
8: 40,
16: 60
}
cameras = {
12: 30,
24: 60
}
batteries = {
3000: 30,
4000: 40
}
# Generate combinations
combinations = list(itertools.product(processors.keys(), ram_memory.keys(), cameras.
keys(), batteries.keys()))
# Print all combinations
print("All possible combinations:")
for combination in combinations:
print(combination)
# Function to simulate performance evaluation
def evaluate_combination(combination):
processor, ram, camera, battery = combination
# Calculate costs
total_cost = (
processors[processor] +
ram_memory[ram] +
cameras[camera] +
batteries[battery]
)
# Calculate performance
performance = (
ram * 10 +
(camera / 12) * 20 +
(battery / 3000) * 15 +
(5 if processor == 'A' else 3 if processor == 'B' else 1)
)
# Calculate performance-to-cost ratio
ratio = performance / total_cost if total_cost > 0 else 0
return ratio
# Evaluate all combinations
results = {}
```