# Heuristics

Lab Section 6

# For students with Programming background

- Take input from the knap file which contains
  - First line as the total weight of all items
  - Rest of the lines as Items with weight as the first attribute and value as the second attribute
  - Read these attributes, take the weight integer from the user, and find the best items that approximately make that weight meaning having almost that weight those items have the maximum value of all the possible items
  - For example if we have items with weight and values = { 5: 10, 4: 20, 20: 100, 10:25} , now if the user gives a weight of 30, the program will select the best items with maximum value: {5: 10, 4:20, 20:100}
  - Solve the above problem using the genetic algorithm, the details of which are available in the next few slides

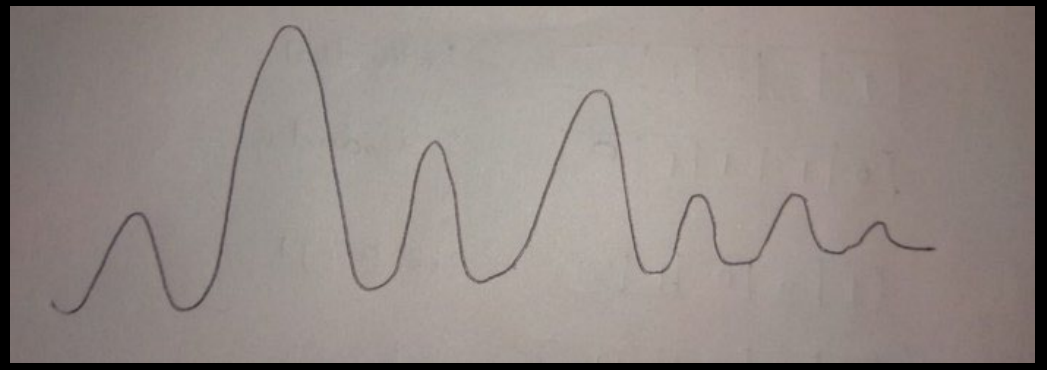
# For students with Programming background

- In essence:

# Today's Lab

- We will explore these exercises and concepts:
- Genetic algorithm introduction
- Genetic algorithm implementation
- Use GA to solve Knapsack problem

# Genetic Algorithm – Intro

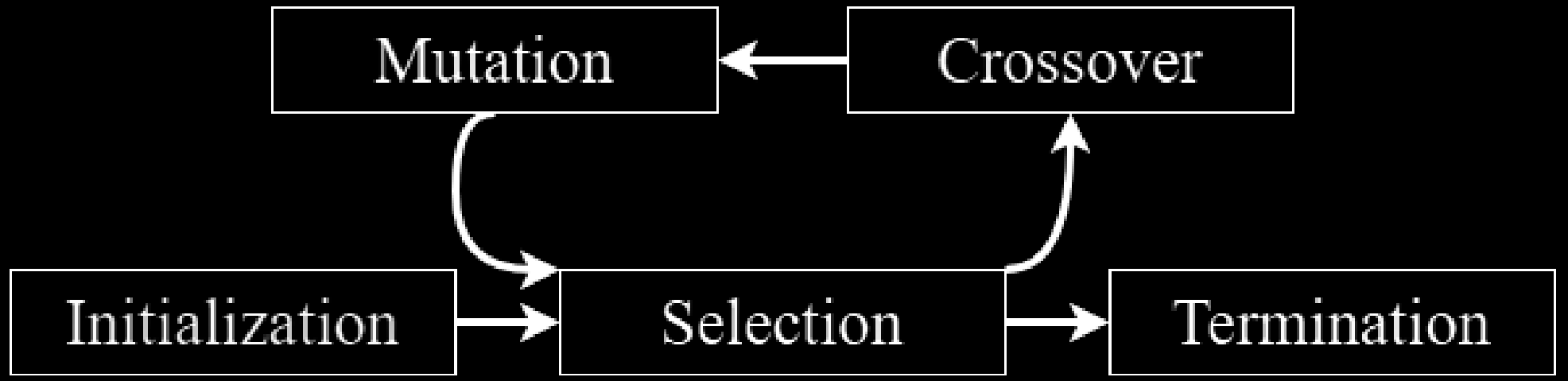


- Sometimes we are given a large search space:
  - Brute force will not work because it will take too long
  - Techniques like gradient descent will have the problem of local optimum
  - In these problems, random exploration of search spaces could be used to reach the global maxima
  - Genetic algorithm is one such algorithm (**evolutionary algorithm** (**EA**), a generic population-based metaheuristic optimization algorithm.)
  - It is based on Darwin's theory of "Survival of the Fittest"
  - In GA we create random changes to the current solutions through *Selection*, *Crossover*, and *Mutation* to create new solutions until we reach the best solution.

# Genetic Algorithm - Flow

- In the Algorithm,
- a population is initialized with a set of candidate solutions.
- Each candidate is distinguished by a specific set of properties called the genotype or the chromosome.
- An objective function calculates the fitness for each individual.
- The individuals with the highest fitness are selected from the current generation and are crossed to form new individuals.
- These individuals may then undergo mutation to form a new generation.
- This cycle is repeated until a satisfactory solution is obtained, or the maximum number of generations is exhausted

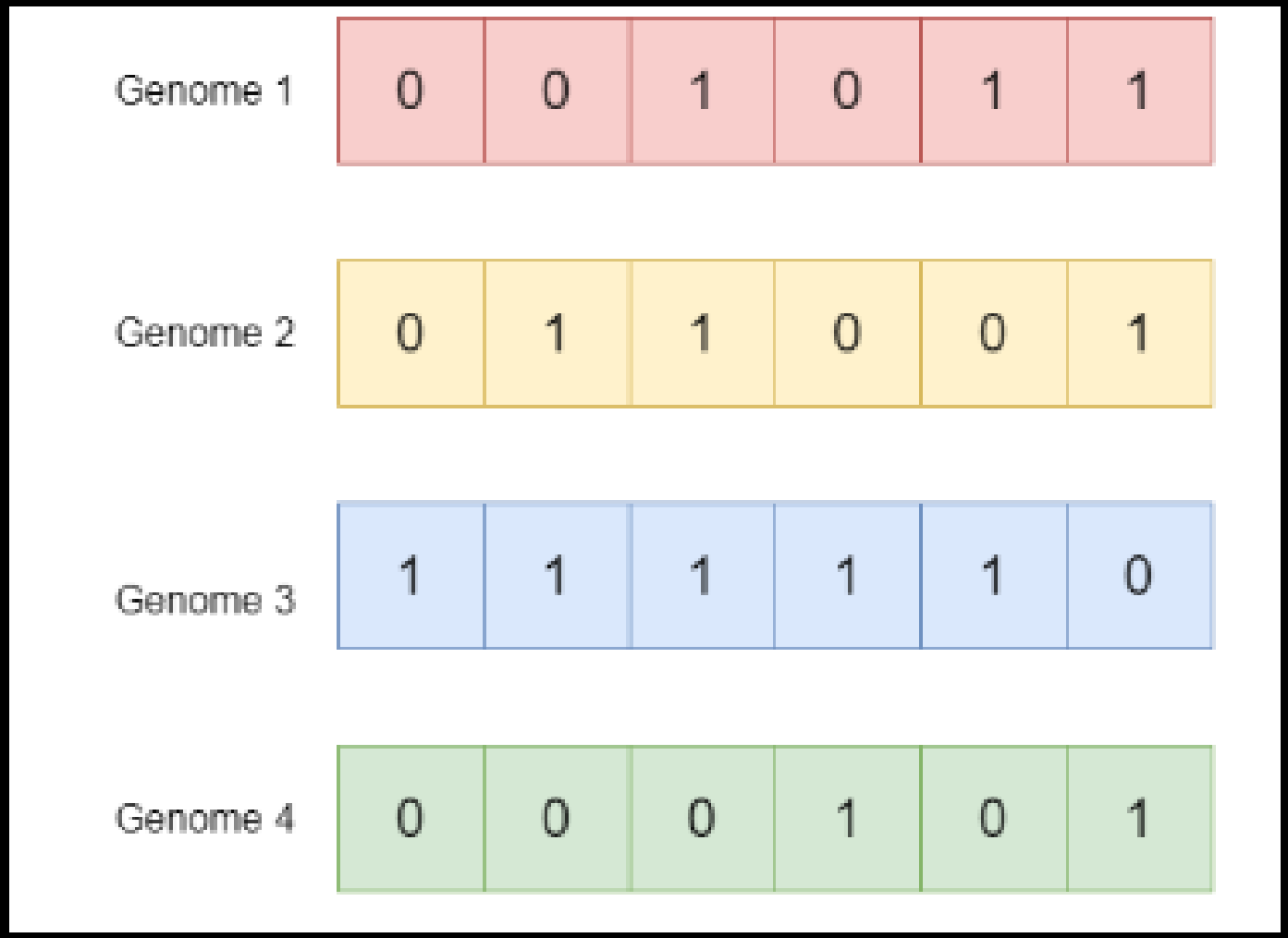
# Genetic Algorithm – Flow Diagram



# Step 1: Initialization - generate chromosomes

- Define an *initial population* that contains a certain number of solutions.
- Each solution is called an individual.
- Each individual solution is encoded as a chromosome
- The chromosomes or the genotype of the individuals in the initial population are typically generated at random.
- various ways of chromosome encoding, we will only use binary encoding.

# Initial Population



# Step 2: Selection using fitness funtion

- First, the fitness score of each individual is calculated by an objective function, which is decided based on the respective problem.
- This fitness score is an indication of how suitable the respective solution is for the problem.
- During each successive generation a portion of the current population with relatively higher fitness scores is allowed to move to the next stage.
- Various schemes may be used in the selection stage, such as tournament selection, rank-based selection, roulette wheel selection, elitism selection, etc

# Selection

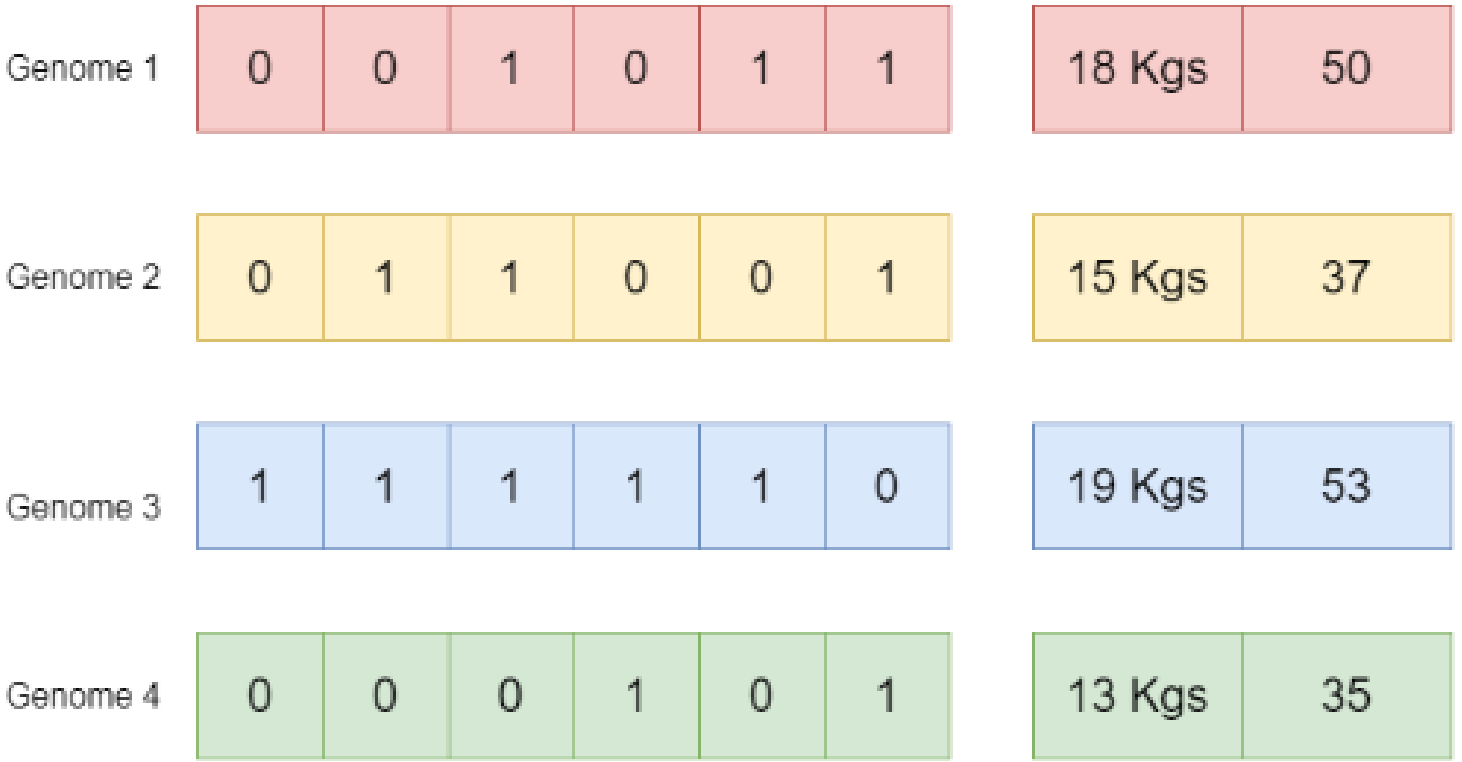



*Fig: Selection*

# Step 3: Crossover

- Crossover involves the reproduction of a new generation from the selected individuals.
- It is a genetic operation used to combine the genetic information of two parents in order to generate new offspring.

# Crossover

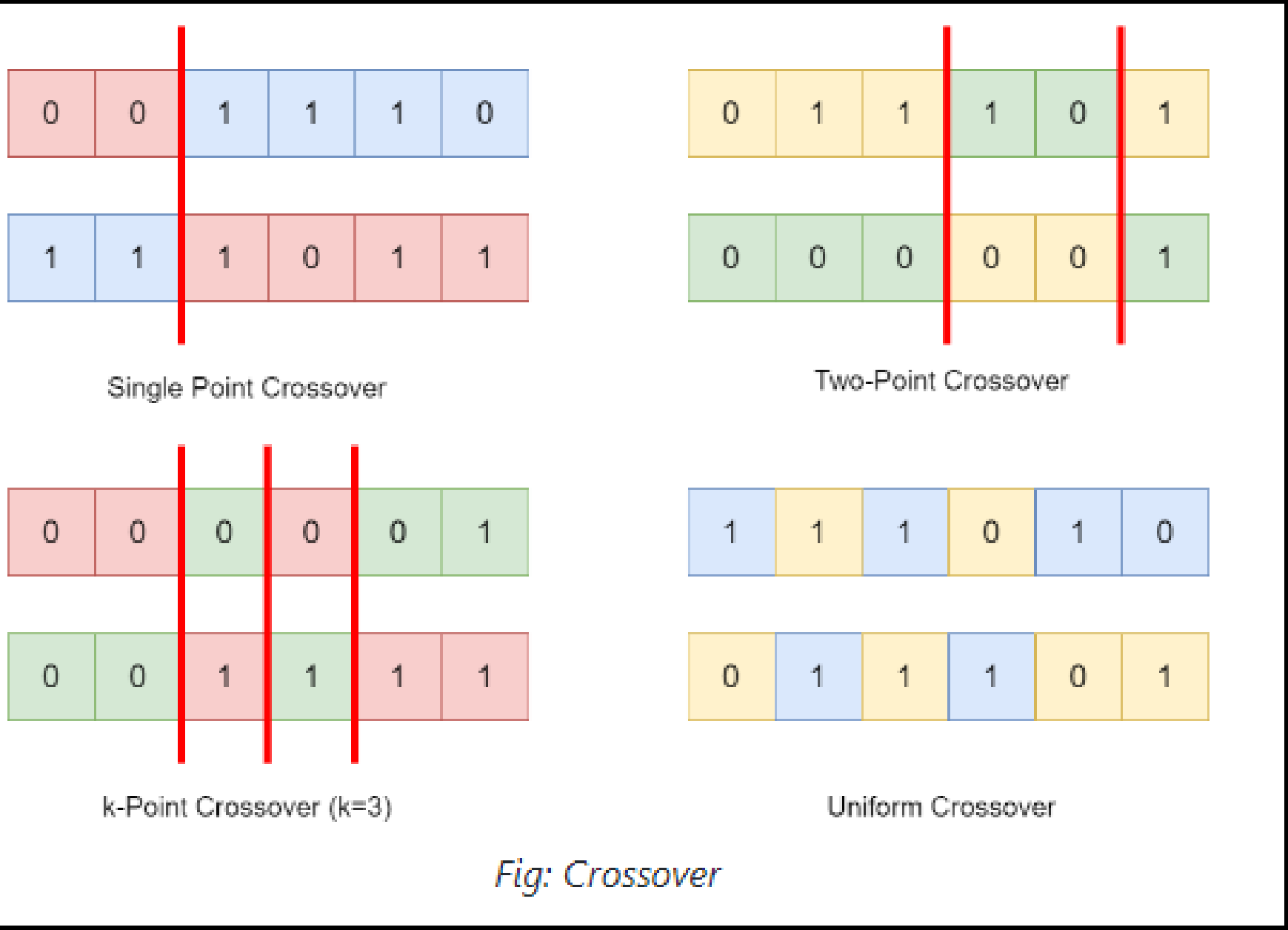



*Fig: Crossover*

# Step 4: Mutation

- Mutation is the genetic operation used to maintain genetic diversity after the crossover stages.
- It is analogous to biological mutation and alters the chromosomes of the children using one or more schemes such as:
  - bit-flip mutation, swap mutation, scramble mutation, etc.
- In simple terms, mutation may be defined as a small random tweak in the chromosomes, to get a new solution.

# Mutation

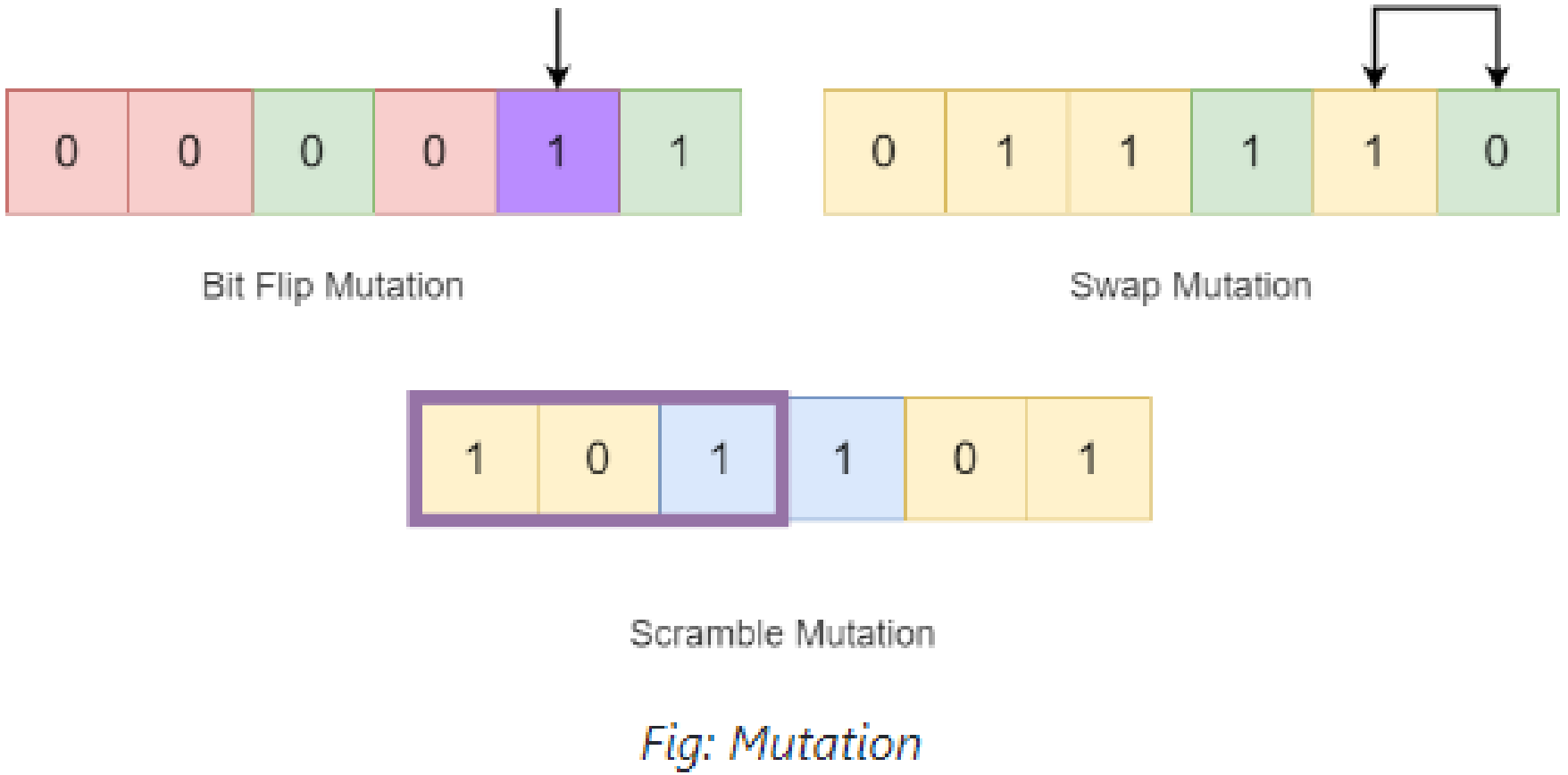



*Fig: Mutation*

# Step 5: Termination

- The cycle of selection, crossover, and mutation is repeated until one of the following conditions is met:

1. A solution is found that satisfies the threshold condition.
2. The maximum number of iterations/cycles has been exhausted.
3. The high-ranking solutions' fitness scores have become stagnant and show no or little improvement.
4. Allocated computation/time resources have been exhausted.

# Termination

**WINNER GENOME**

| | | | | | | |
|---|---|---|---|---|---|---|
| Genome: | 1 | 0 | 0 | 1 | 1 | 1 |
| Weights: | 1 | 3 | 4 | 5 | 6 | 8 |
| Values: | 3 | 5 | 12 | 15 | 18 | 20 |

| | | |
|---|---|---|
| Total Weight & Value | 20 Kgs | 56 |

*Fig: Termination*

# For Programming Novice

- Take input from the knap file which contains
  - First line as the total weight of all items
  - Rest of the lines as Items with weight as the first attribute and value as the second attribute
  - Read these attributes and print the first ten items.
  - Also, print the total weight.

# Knapgenetic.py

- Go over the corresponding code file.

# End