CS 581

Advanced Artificial Intelligence

February 7, 2024

Announcements / Reminders

- Please follow the Week 04/05 To Do List instructions (if you haven't already)

- Midterm exam is approaching

- Written Assignment #01: due on Saturday 02/10 at 11:59 PM CST
- Programming Assignment #01: due on Sunday 03/03 at 11:59 PM CST

Plan for Today

- Solving problems by Searching
  - Evolutionary Algorithms
    - Genetic Algorithm
    - Genetic Programming

Evolutionary Algorithms [Wikipedia]

An evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm.

An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions (see also loss function).

Evolution of the population then takes place after the repeated application of the above operators.

Genetic Algorithm: Components

Representation	
Recombination	
Mutation	
Parent Selection	
Survival Selection	

Simple Genetic Algorithm

Representation	Bit strings
Recombination	1-point crossover
Mutation	Bit flip
Parent Selection	Fitness proportional
Survival Selection	Generational

Evolutionary Algorithms: More on Representations

Chromosome: Representation

Individuals / chromosomes can be represented as a string of values.

Typically:

Binary									
0	1	0	0	1	1	1	0	1	1

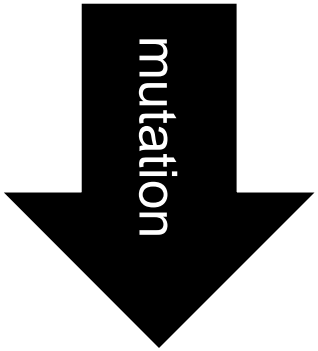
Integer									
2	1	11	2	3	78	1	0	111	33

Floating-point									
2.0	1.5	1.1	0.2	3.3	7.8	1.	0.0	11.1	3.3

Binary Representation Issues

Chromosome A

Variable/Gene = 512									
1	0	0	0	0	0	0	0	0	0



Chromosome A after mutation

Variable/Gene = 0									
0	0	0	0	0	0	0	0	0	0

Small, one bit, change can lead to drastic fitness changes. Solution: use Gray Code.

Gray Code

The reflected binary code (RBC), also known as reflected binary (RB) or Gray code (named after Frank Gray – Bell Labs), is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).

Decimal	Binary	Gray	Decimal	Binary	Gray
0	0000	0000	7	0111	0100
1	0001	0001	8	1000	1100
2	0010	0011	9	1001	1101
3	0011	0010	10	1010	1111
4	0100	0110	11	1011	1110
5	0101	0111	12	1100	1010
6	0110	0101	13	1101	1011

Gray Code | Hamming Distance

The Hamming distance between two equal-length strings of symbols is the number of positions at which the corresponding symbols are different.

Gray code: subsequent numbers → Hamming distance of 1

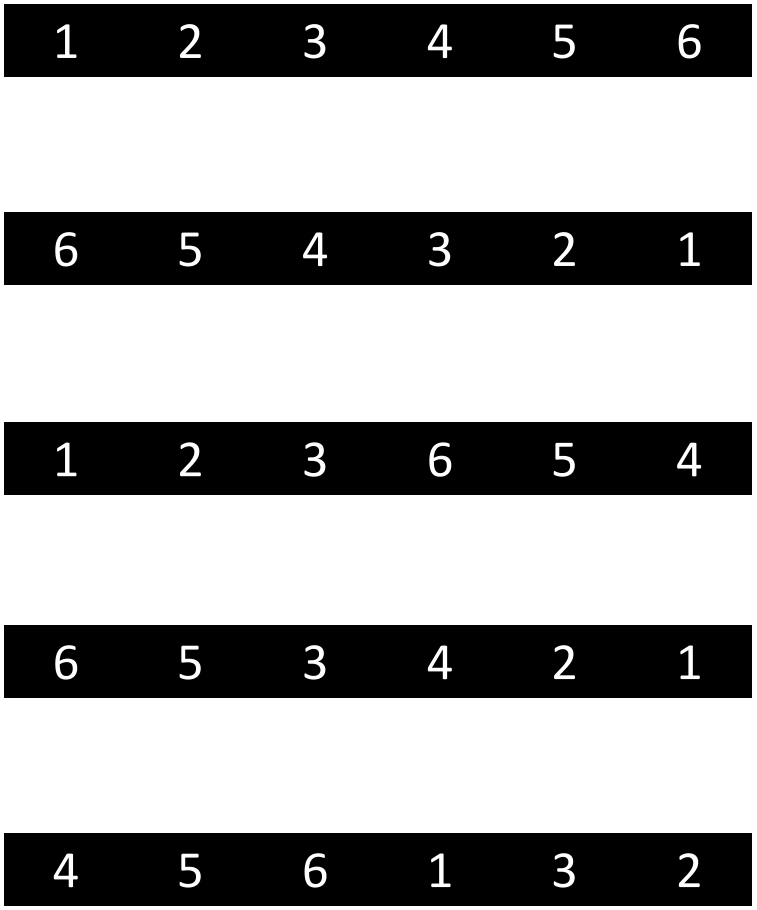
Decimal	Binary	Gray	Decimal	Binary	Gray
0	0000	0000	7	0111	0100
1	0001	0001	8	1000	1100
2	0010	0011	9	1001	1101
3	0011	0010	10	1010	1111
4	0100	0110	11	1011	1110
5	0101	0111	12	1100	1010
6	0110	0101	13	1101	1011

Integer Representation

- Some problems naturally map (8-queens) to integer representations
  - solution is an assignment variable = integer value
- Unrestricted: any value is permissible
- Restricted: only values from a certain set / domain
  - for example {0, 1, 2, 3} for {North, East, South, West}
- Consider:
  - is there any relationship between values (e.g. 3 is more like 4 than 789 → ordinal relationship) or not ({North, East, South, West})
  - Choose your recombination / mutation strategy accordingly

Permutation Encoding

Population of individuals



In permutation encoding, every chromosome is a string of numbers that representing position in a sequence.

Permutation encoding is useful for ordering problems. For some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e. have valid sequence in it) for certain problems.

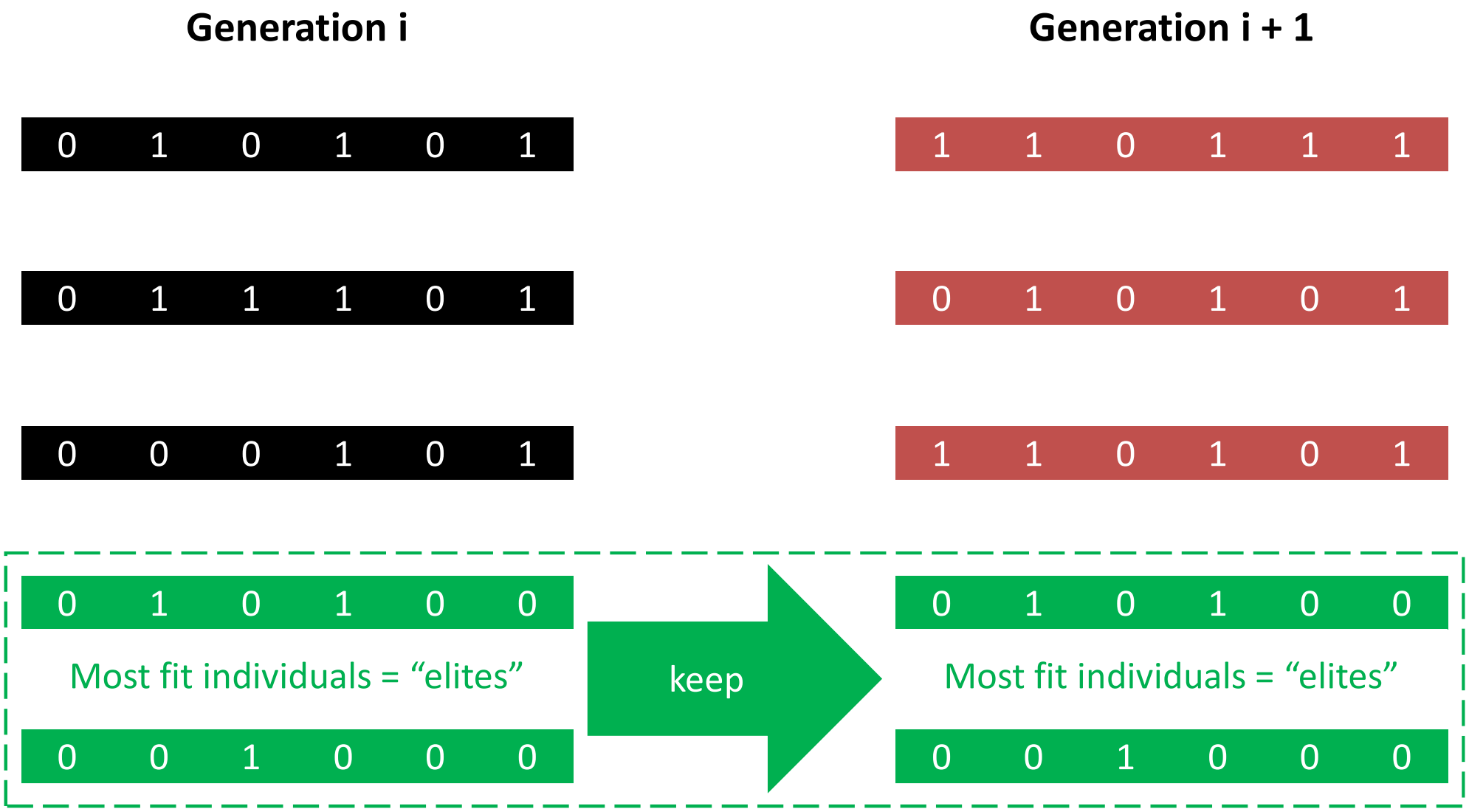
Genetic Algorithm:
Other Selection Mechanisms

Genetic Algorithm: Elitism

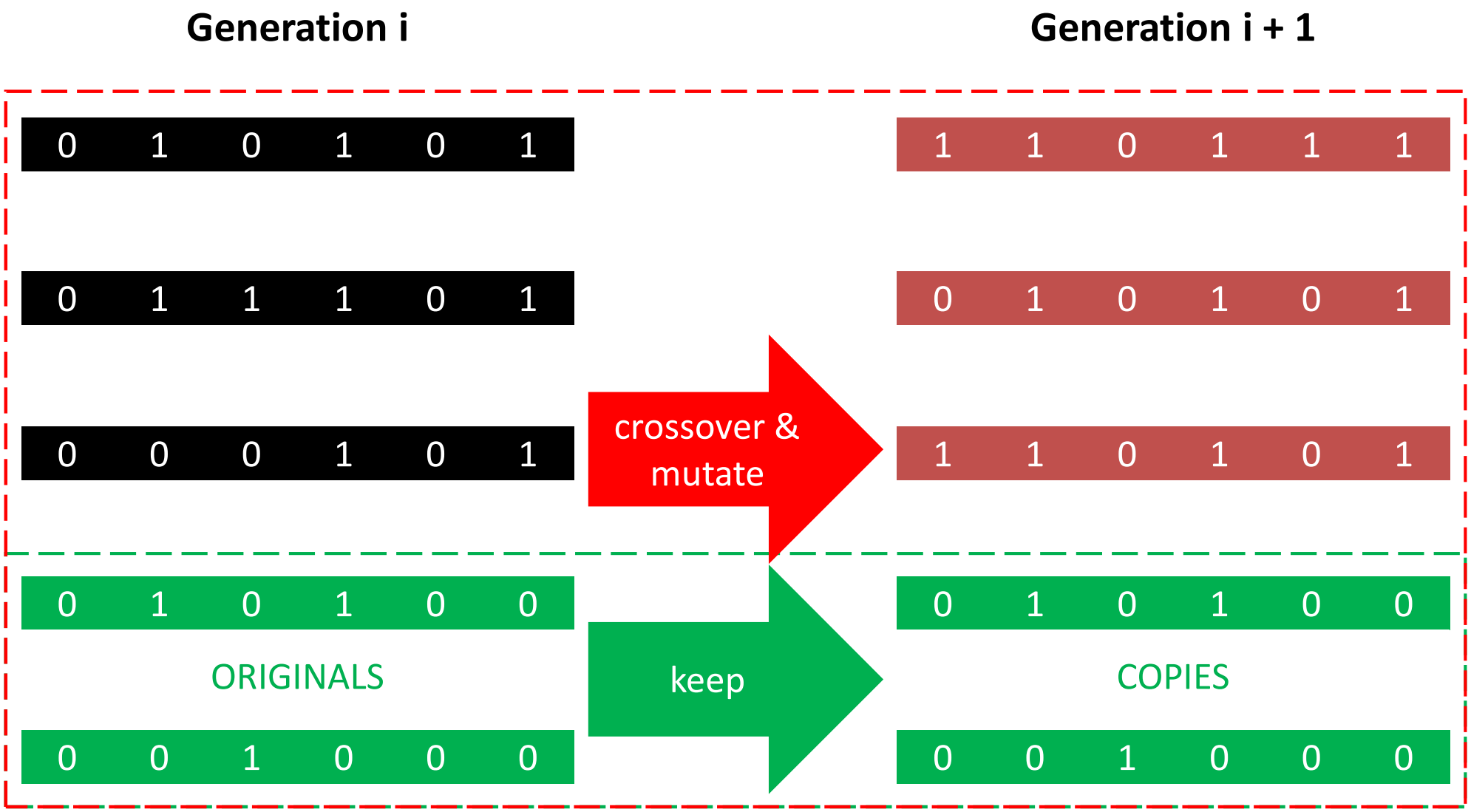
Elitism (Elitist selection) is a strategy in genetic algorithms (in practice: evolutionary algorithms in general) where one or more most fit individuals (**the elites**) in every generation, are inserted into the next generation **without undergoing any change**.

This strategy usually speeds up the convergence of the algorithm.

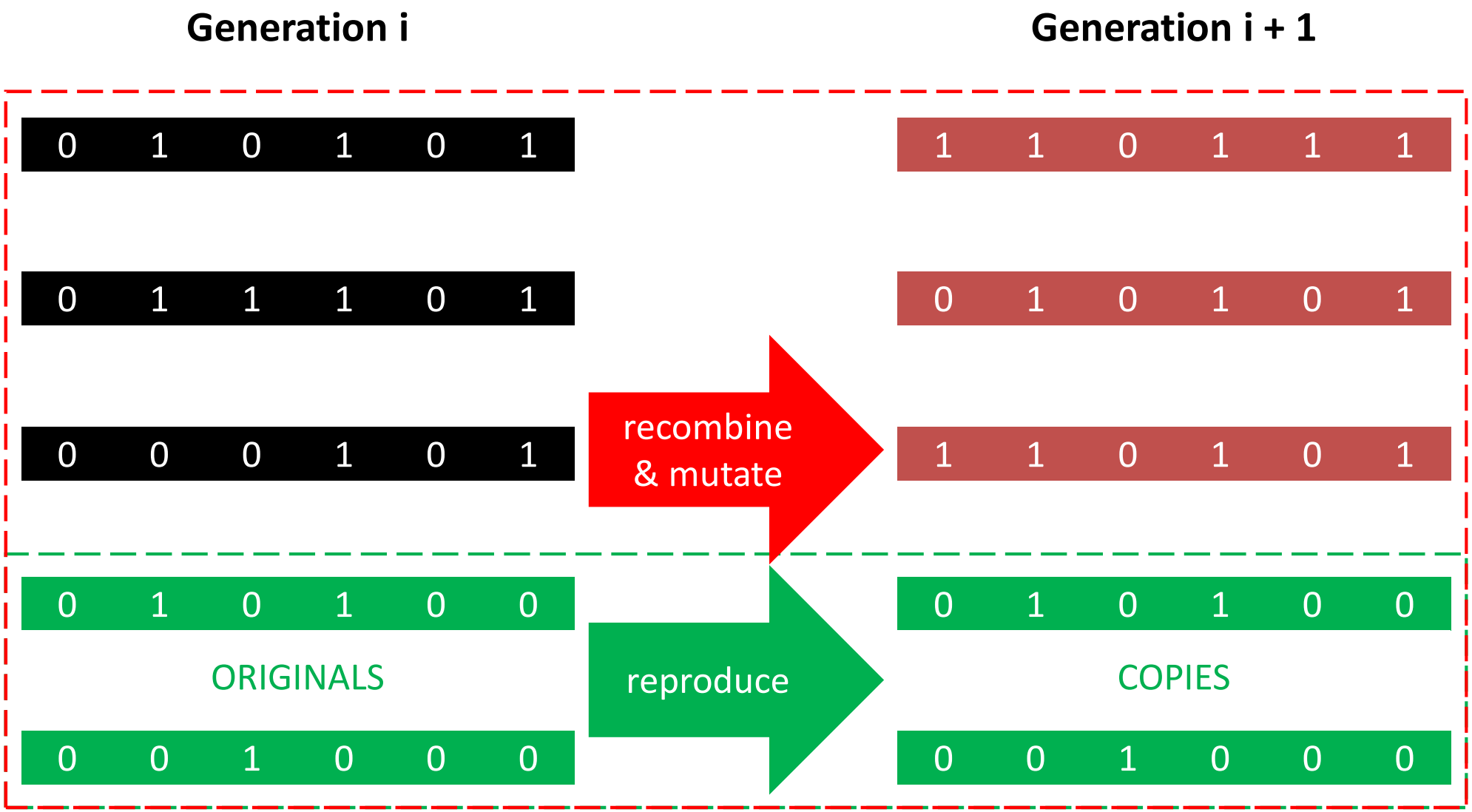
Genetic Algorithm: Elitist Selection



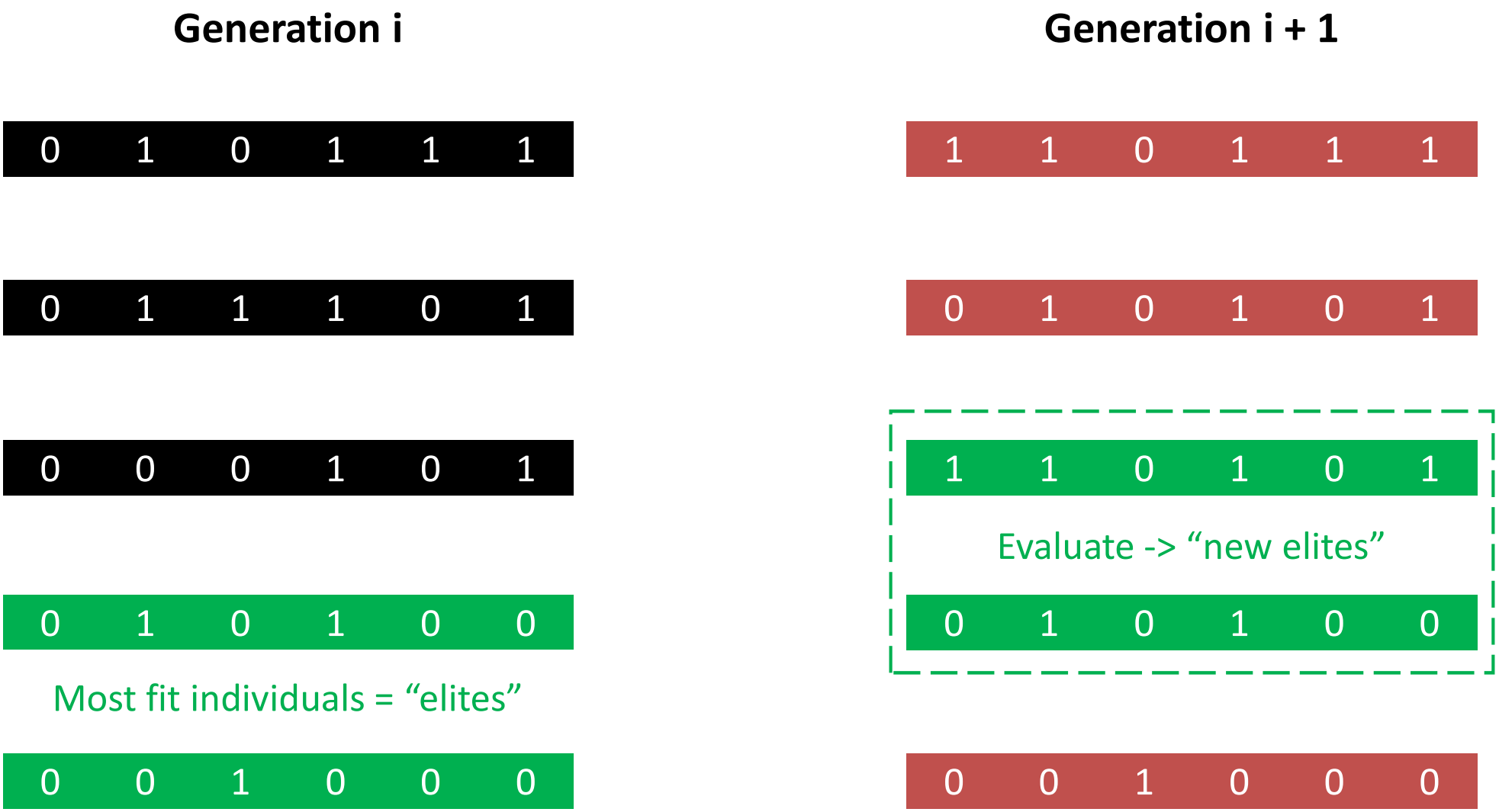
Genetic Algorithm: Elitist Selection



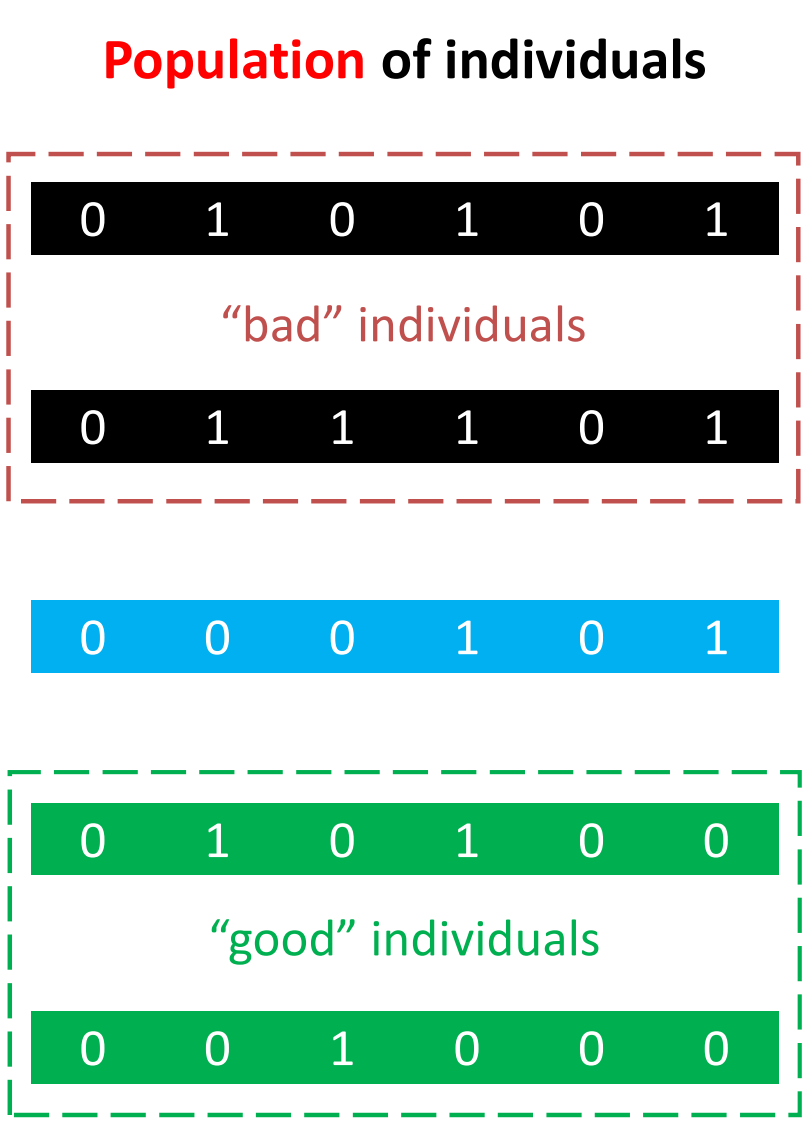
BTW: Reproduction Operator



Genetic Algorithm: Elitist Selection



GA: Steady State Selection



In every generation few chromosomes are selected (good - with high fitness) for creating a new offspring.

Then some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place.

The rest of population survives to new generation.

Fitness Proportional Selection

Individual	Genotype	Phenotype	Phenotype fitness	Fitness ratio [%]
X1	1 1 0 0	12	36	16.5
X2	0 1 0 0	4	44	20.2
X3	0 0 0 1	1	14	6.4
X4	1 1 1 0	14	14	6.4
X5	0 1 1 1	7	56	25.7
X6	1 0 0 1	9	54	24.8

Fitness Ratio: What if?

Individual	Genotype	Phenotype	Phenotype fitness	Fitness ratio [%]
X1	1 1 0 0	12	1	1.7
X2	0 1 0 0	4	1	1.7
X3	0 0 0 1	1	2	2.2
X4	1 1 1 0	14	1	1.7
X5	0 1 1 1	7	30	50.0
X6	1 0 0 1	9	25	42.7

Fitness Proportional Selection: Issues

Fitness Proportional Selection approach has problems:

- **<u>outstanding</u> individuals take over the entire population quickly; this is called premature convergence**
- **potential solutions**
- **when fitness values are all very close together, there is almost no selection pressure [= selection is almost uniformly random] → performance increases slowly**
- **Potential solutions:**
  - **windowing (slide and subtract some value based on history)**
  - **Goldberg's sigma scaling:** $f'(x) = \max(f(x)-(f_{avg} - 2 * \sigma_f), 0.0)$
  - **ranking**

Rank Based Selection

In rank selection, the selection probability does not depend directly on the fitness, but on the fitness rank of an individual within the population. The exact fitness values themselves do not have to be available, but only a sorting of the individuals according to quality.

- **Linear ranking:**

$$P(R_i) = \frac{1}{n}\left(sp - (2sp-2)\frac{i-1}{n-1}\right) \qquad 1 \le i \le n, \quad 1 \le sp \le 2 \quad \text{with} \quad P(R_i) \ge 0, \quad \sum_{i=1}^{n} P(R_i) = 1$$

  sp: **selection pressure (the degree to which the better individuals are favored: the higher the selection pressure, the more the better individuals are favored) which can take values between 1.0 (no selection pressure) and 2.0 (high selection pressure)**

- **Exponential ranking**

Solution: Rank Individuals

Individual	Genotype	Phenotype	Phenotype fitness	Fitness ratio [%]	Rank	New Ratio P(Rank)
X1	1 1 0 0	12	1	1.7	6	
X2	0 1 0 0	4	3	5.0	4	
X3	0 0 0 1	1	2	2.2	5	
X4	1 1 1 0	14	4	6.7	3	
X5	0 1 1 1	7	30	50.0	1	
X6	1 0 0 1	9	25	42.7	2	

Replace Worst

- In this scheme the worst k members of the population are selected for replacement
- This can lead to rapid improvements in the average population fitness, but can also cause premature convergence
  - it will focus on most fit individual
- Typically used with large populations

Age-Based Replacement

- Rather than look at fitness of the individual, pick the oldest (in iterations) first to replace
- A FIFO queue will be needed

Genetic Algorithm:
Other Recombination Mechanisms

“Cut and Crossfill” Crossover

1. Pick a random crossover point
2. Cut both parents in two segments after this position
3. Copy the first segment of Parent 1 into Child 1and the first segment of Parent 2 into Child 2
4. Scan Parent 2 from left to right and fill the second segment of Child 1 with values from Parent 2, skipping those that are already contained in it
5. Do the same for Parent 1 and Child 2

Multiparent Recombination

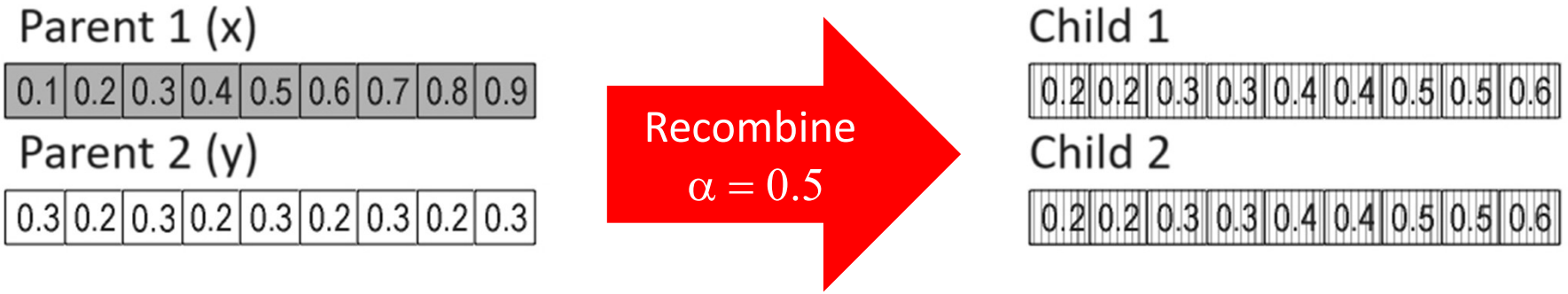
- The idea: recombine genes of more than 2 parents

- Some strategies:
  - allele frequency-based: p-sexual voting
  - segmentation and recombination of parents-based: the diagonal crossover
  - based on numerical operations on real-valued alleles: the center of mass crossover

Other Recombination Options

- **Integer representation:**
  - **same approaches as for binary**
- **Floating-point representation:**
  - **simple recombination**
  - **single arithmetic recombination**
  - **whole arithmetic recombination**

Child 1: α*x + (1-α)*y **and Child 2:** α*y + (1-α)*x



Genetic Algorithm:
Other Mutation Mechanisms

Other Mutation Mechanisms

Scramble mutation

A subset of genes is chosen and their values randomly shuffled / scrambled.

Inversion mutation

A subset of genes, a substring is selected and inverted.

Interchange (order changing) mutation

Randomly select two positions of the chromosome and interchange the values.

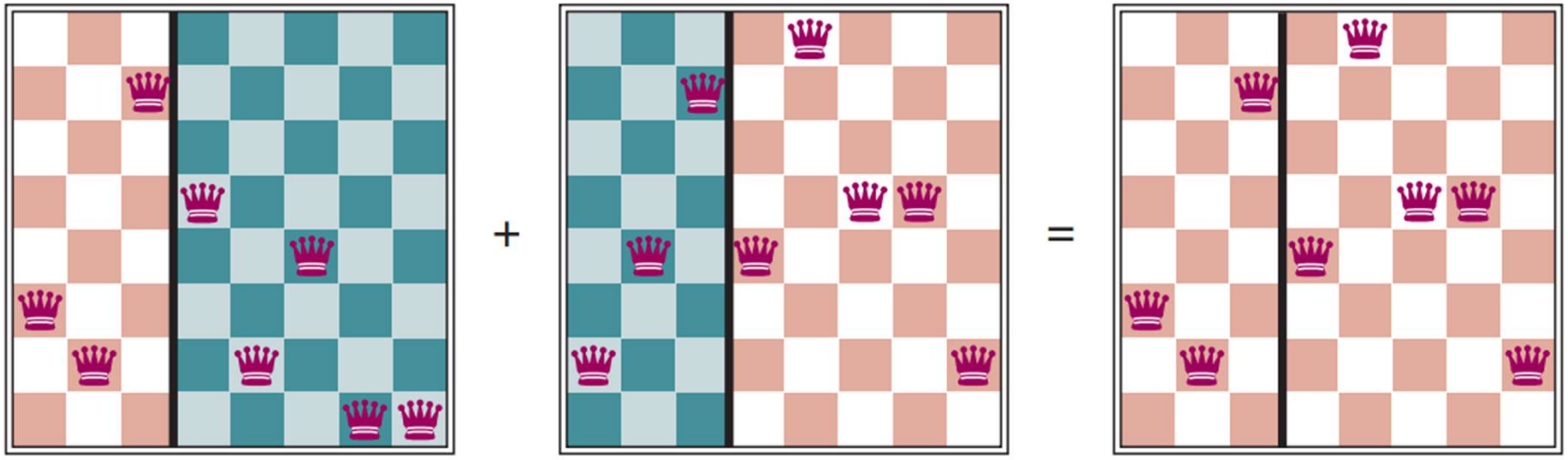
Genetic Algorithm: Selected Problems

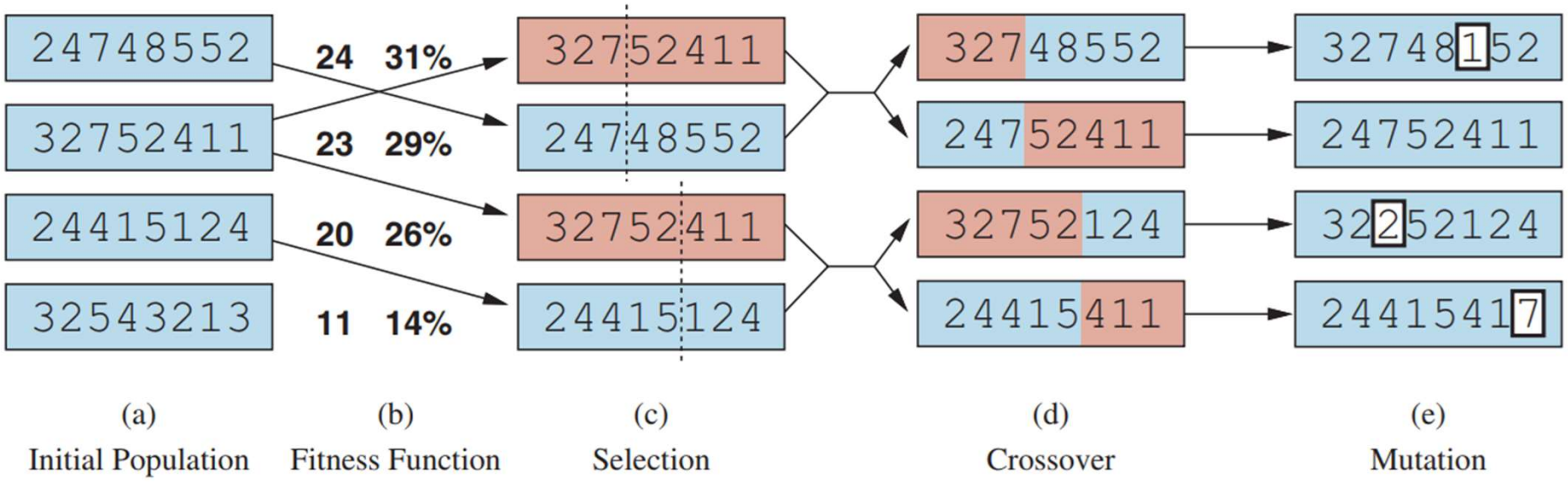
Simple Genetic Algorithm

Representation	Bit strings
Recombination	1-point crossover
Mutation	Bit flip
Parent Selection	Fitness proportional
Survival Selection	Generational

8-Queens Problem

Genetic Algorithm: 8-Queens Problem





8-Queens: Potential EA Approach

Representation	Permutations
Recombination	“cut and crossfill” crossover
Recombination probability	1.0
Mutation	Swap
Mutation probability	0.8
Parent selection	Best 2 out of random 5
Survival selection	Replace worst
Population size	100
Number of offspring	2
Initialization	Random
Termination condition	Solution or fitness function threshold

“Cut and Crossfill” Crossover

1. Pick a random crossover point
2. Cut both parents in two segments after this position
3. Copy the first segment of Parent 1 into Child 1and the first segment of Parent 2 into Child 2
4. Scan Parent 2 from left to right and fill the second segment of Child 1 with values from Parent 2, skipping those that are already contained in it
5. Do the same for Parent 1 and Child 2

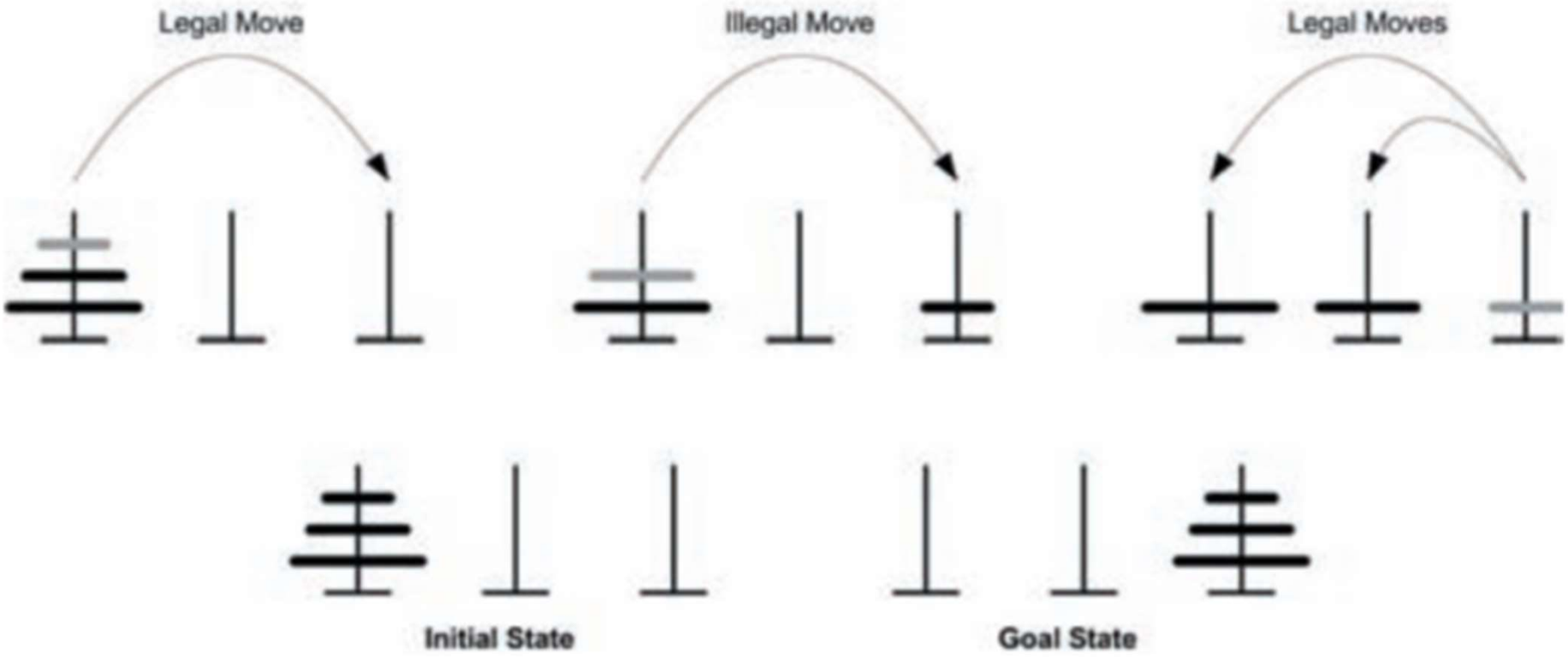
Knapsack Problem

Knapsack: Potential EA Approach

Representation	Binary string of length n
Recombination	1-point crossover
Recombination probability	0.7
Mutation	Each value inverted with mutation probability
Mutation probability	1/n
Parent selection	Best out of random 2
Survival selection	Generational
Population size	500
Number of offspring	500
Initialization	Random
Termination condition	No improvement in last 25 generations

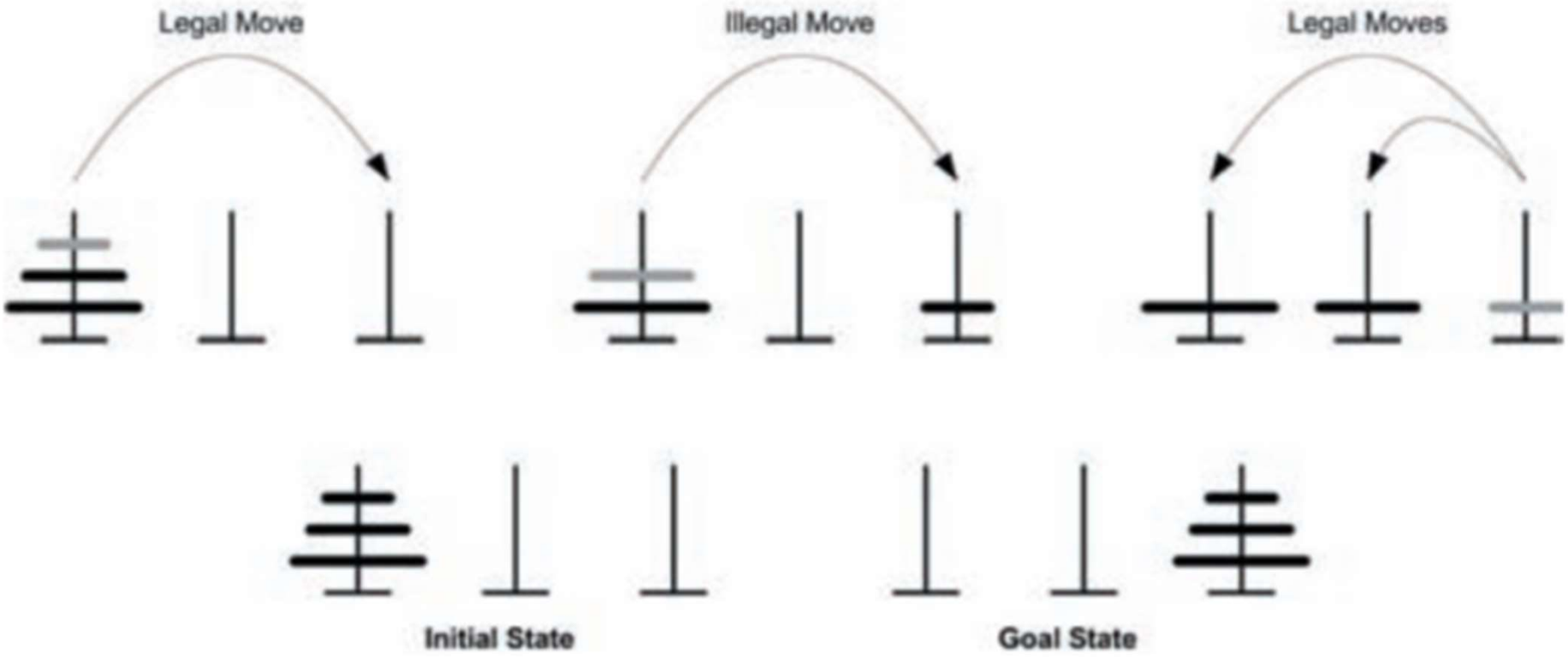
The Tower of Hanoi

The Tower of Hanoi Puzzle



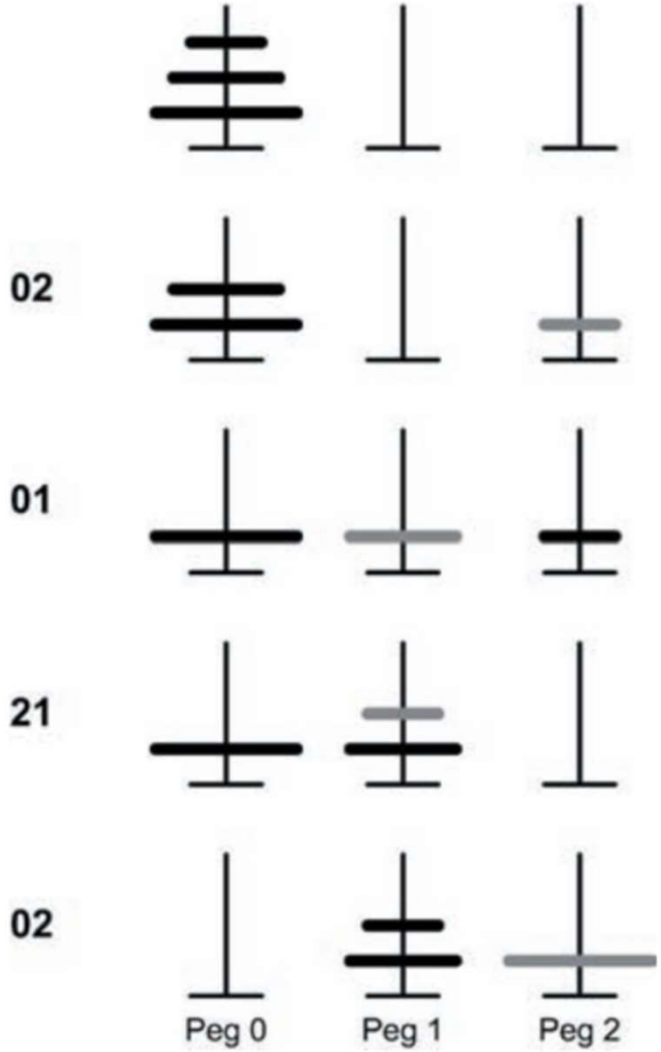
Solution: sequence of moves from peg to peg								
Move 0	Move 1	Move 2	Move 3	…	…	…	…	Move N

The Tower of Hanoi Puzzle with GA



Chromosome: every gene is a single move from peg to peg								
02	01	21	02	…	…	…	…	XY

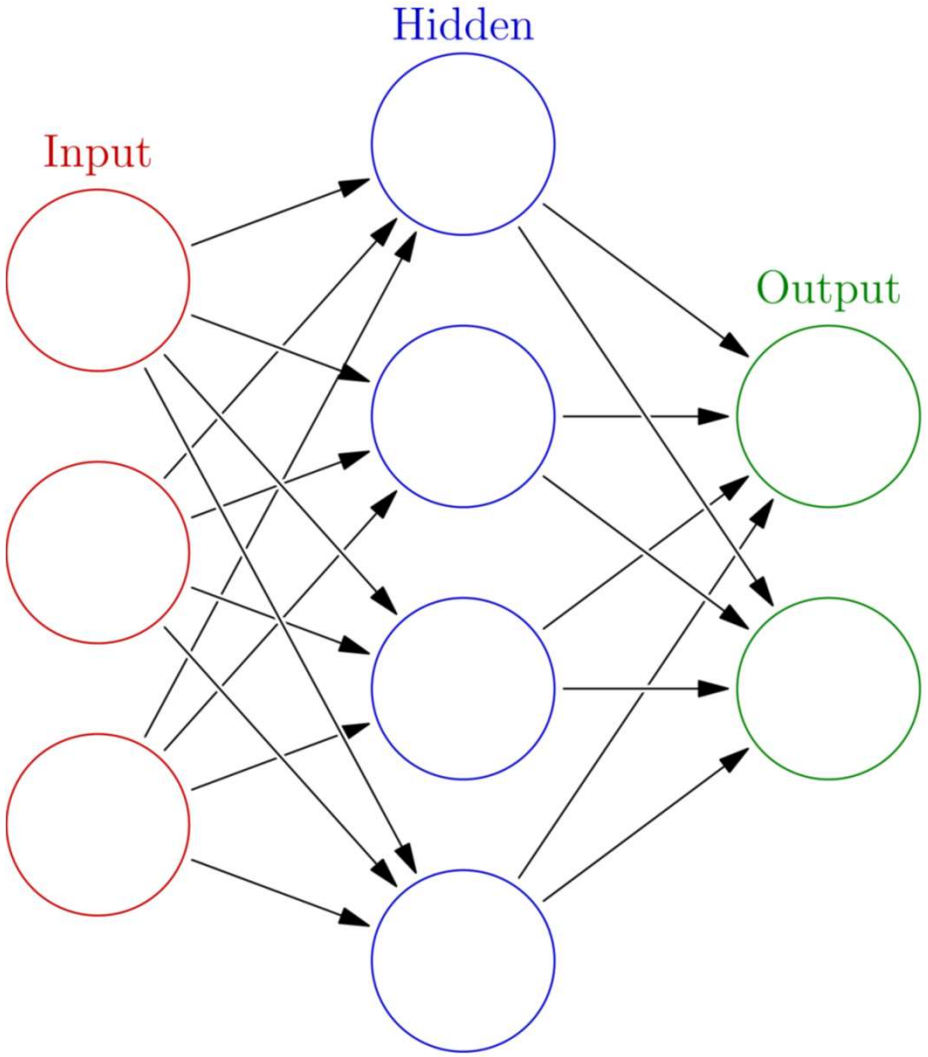
The Tower of Hanoi Puzzle: Solution



Chromosome: every gene is a single move from peg to peg								
02	01	21	02	…	…	…	…	XY

Neural Network Structure

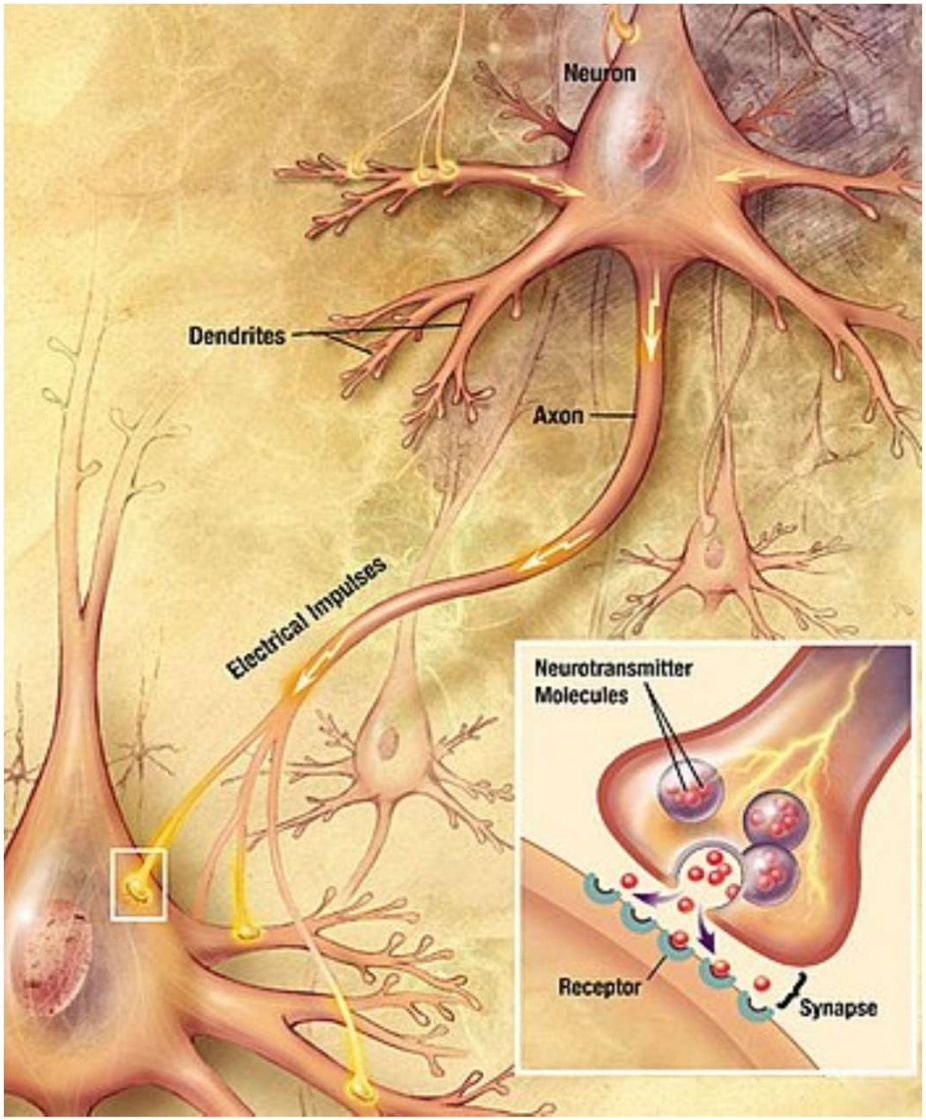
Artificial Neural Network (1943)



Source: https://en.wikipedia.org/wiki/Artificial_neural_network

First computational models of an **Artificial Neural Network** (loosely inspired by biological neural networks) were proposed by Warren McCulloch and Walter Pitts in 1943. Their ideas are a **key component of modern day machine and deep learning**.

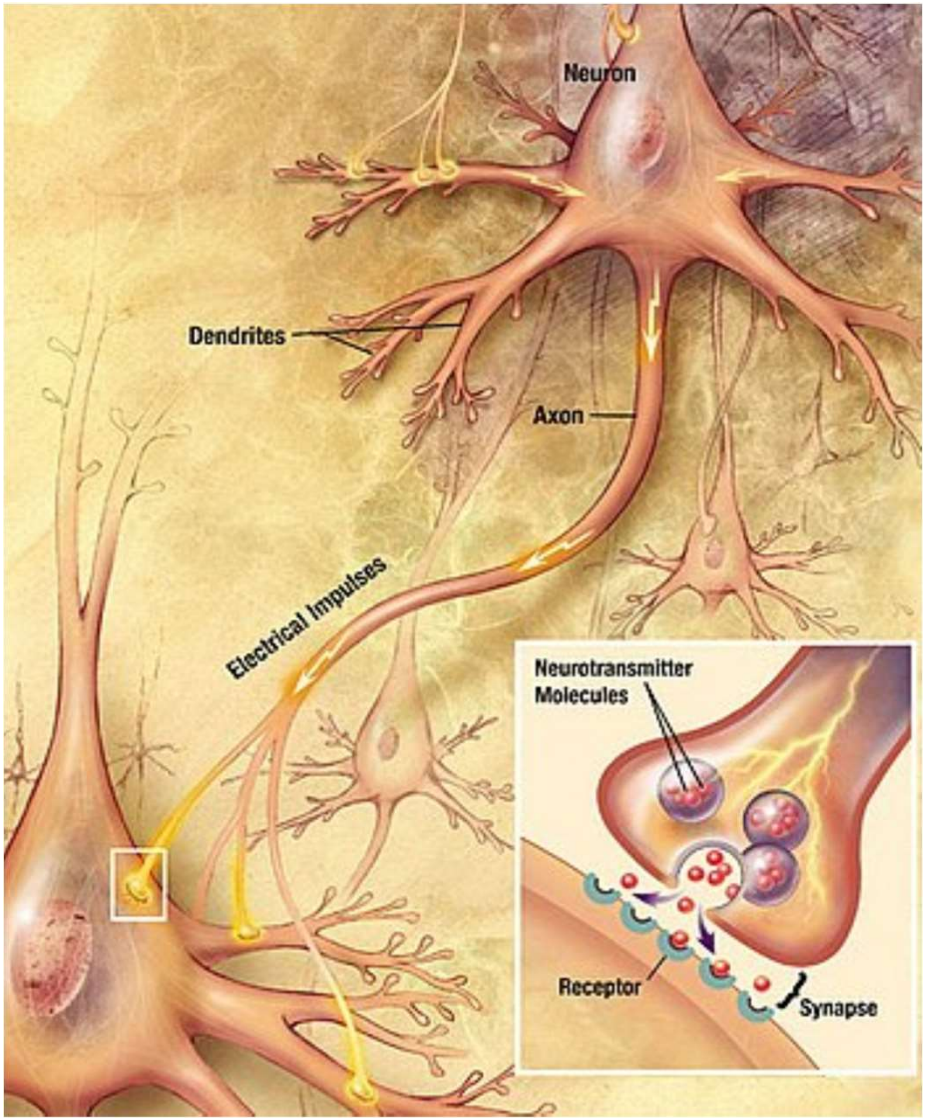
A Biological Neuron



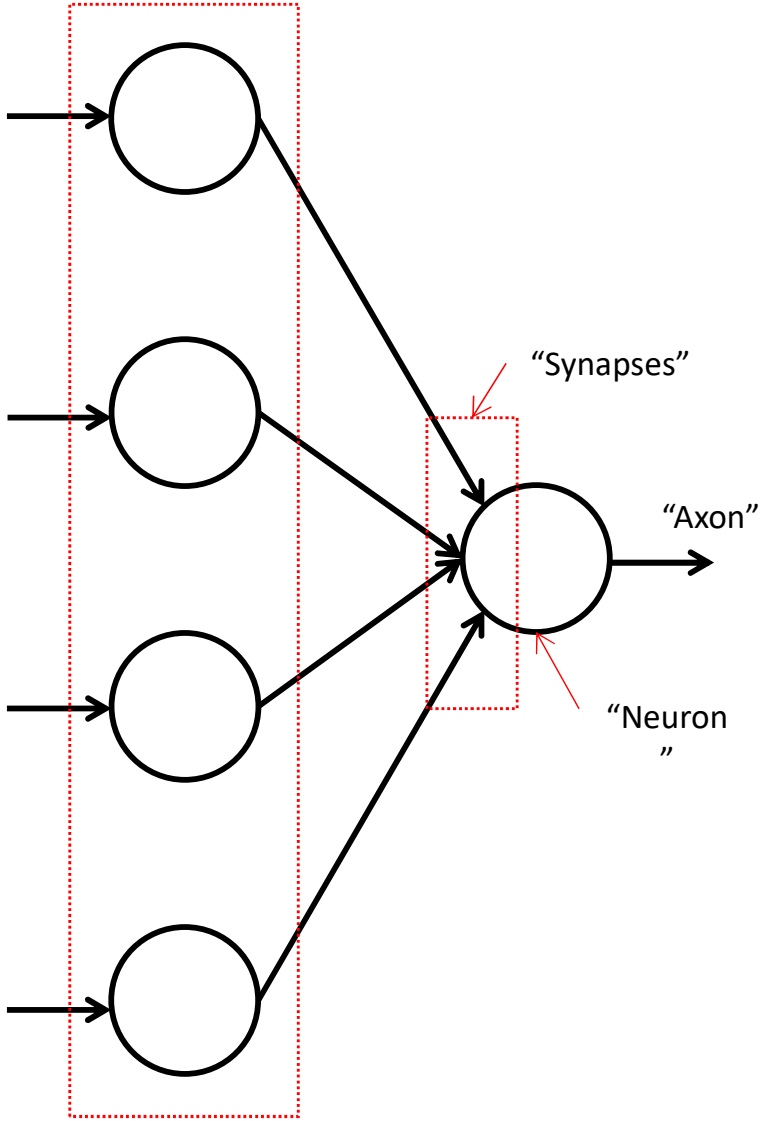
Source: https://en.wikipedia.org/wiki/Neuron

A **neuron** or nerve cell is an electrically excitable cell that **communicates with other cells via specialized connections called synapses**. Most **neurons receive signals via the dendrites and soma** and **send out signals down the axon**. At the majority of synapses, signals cross from the axon of one neuron to a dendrite of another.

Biological vs. Artificial Neuron



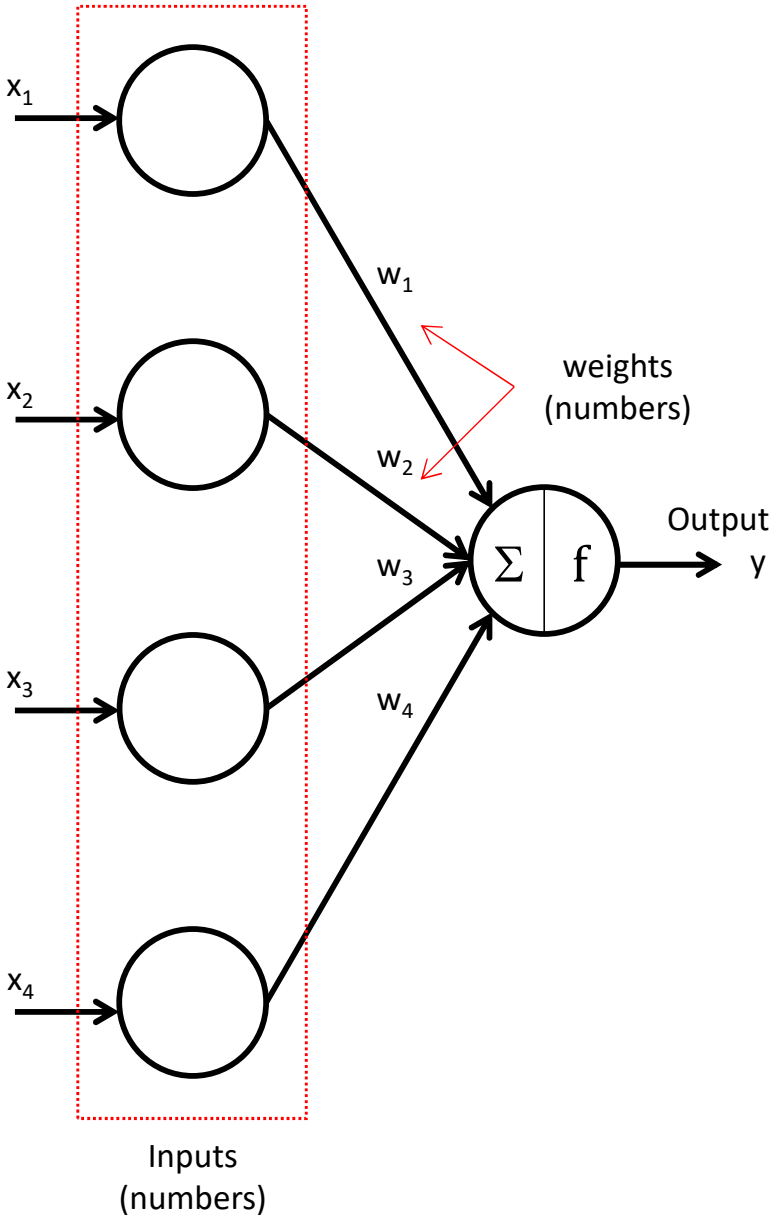
Source: https://en.wikipedia.org/wiki/Neuron



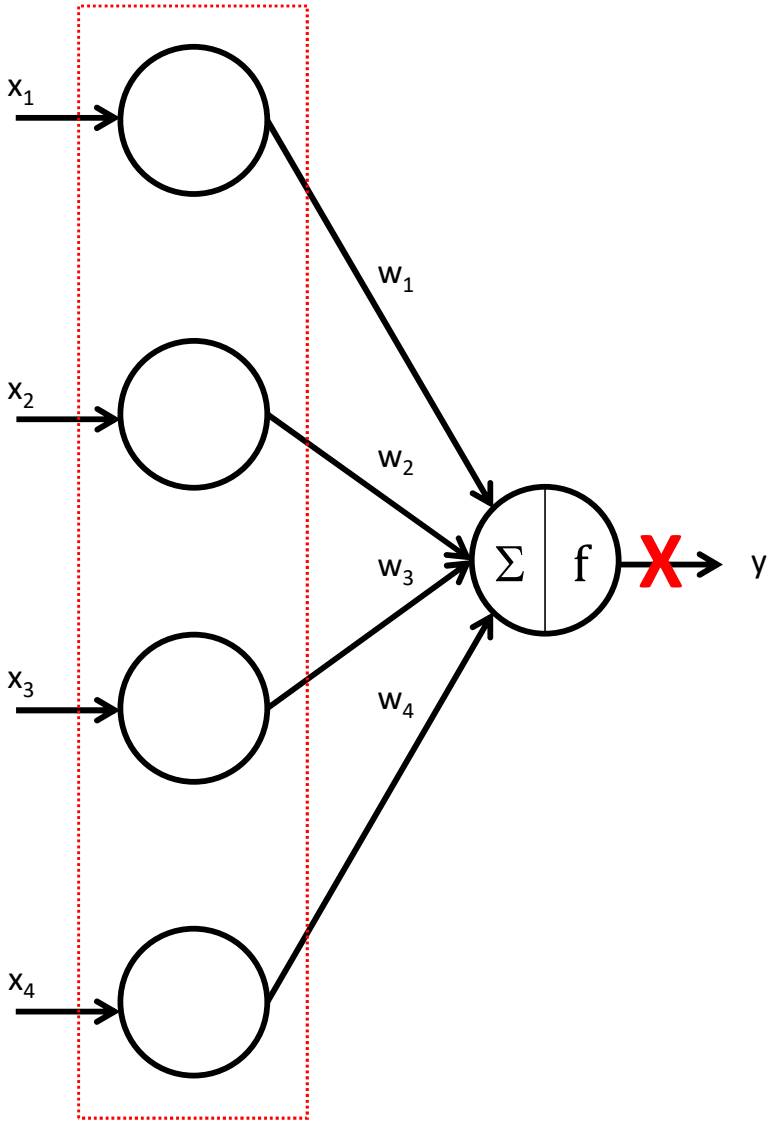
Artificial Neuron (Perceptron)

A (**single-layer**) **perceptron** is a model of a biological neuron. It is made of the following components:

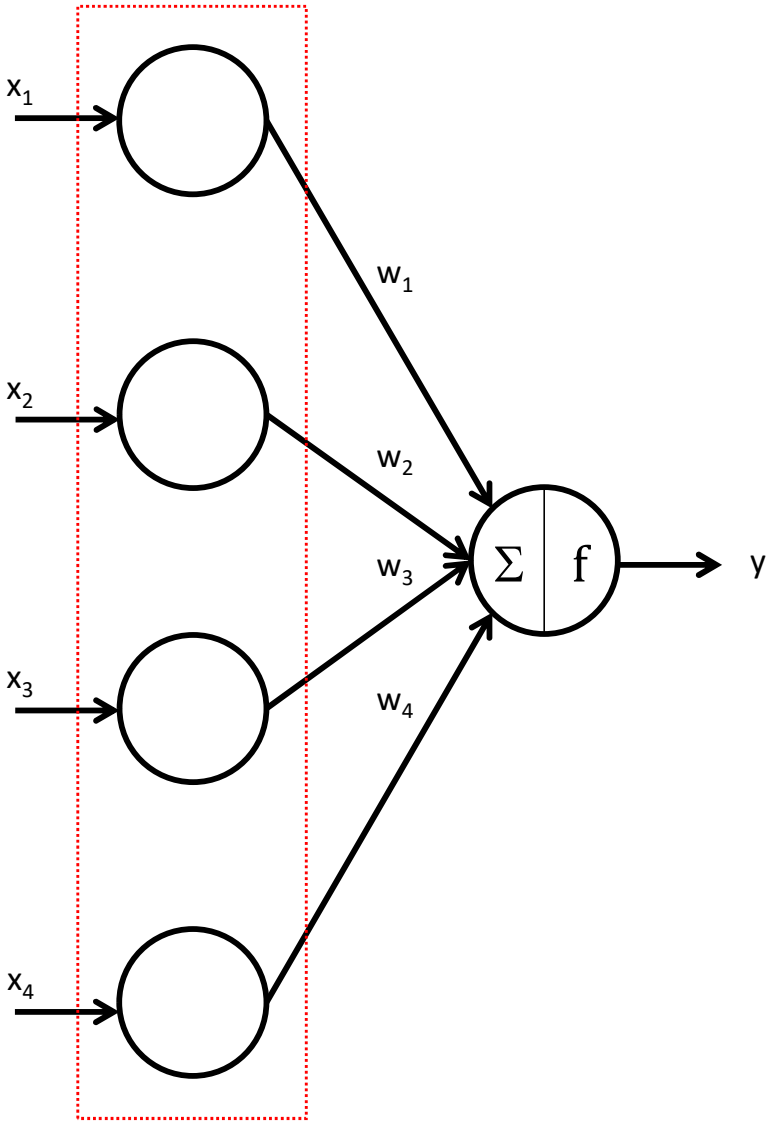
- inputs x_i - numerical values representing information
- weights w_i - numerical values representing how "important" corresponding input is
- weighted sum: $\Sigma\ w_i * x_i$
- activation function f that decides if the neuron "fires"



Artificial Neuron (Perceptron)

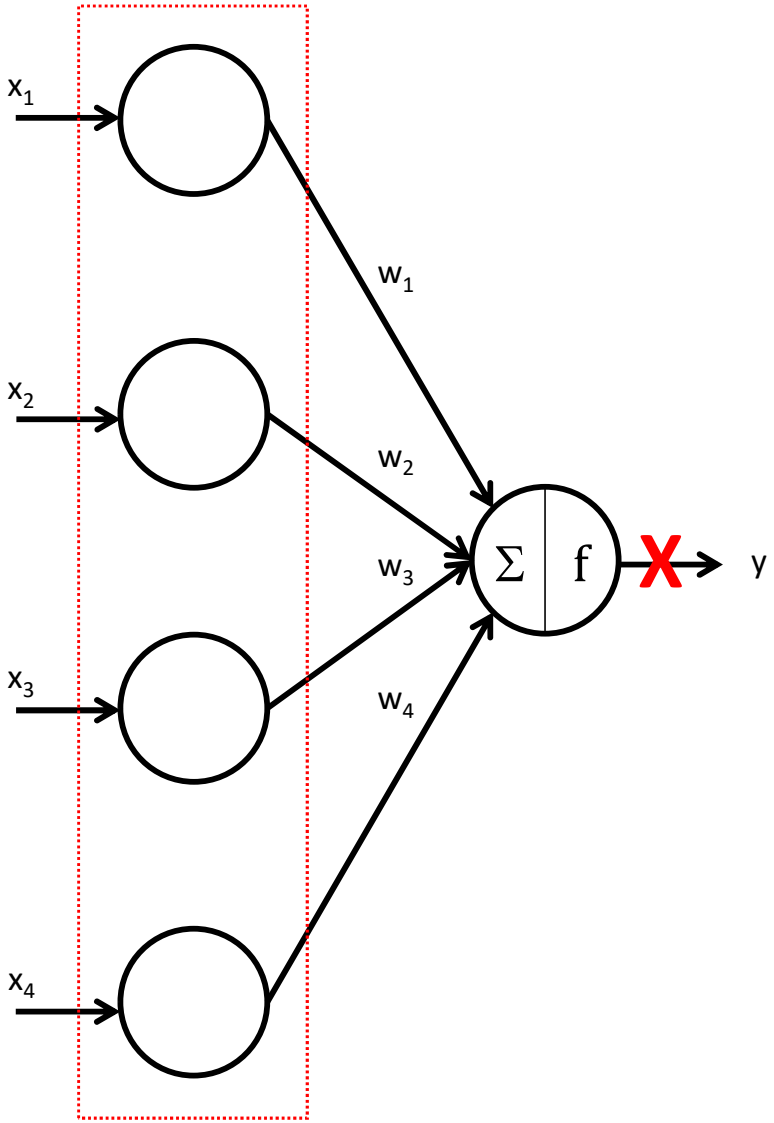


$\Sigma\ w_i * x_i < 0 \rightarrow f = 0$ → **DON'T "fire"**

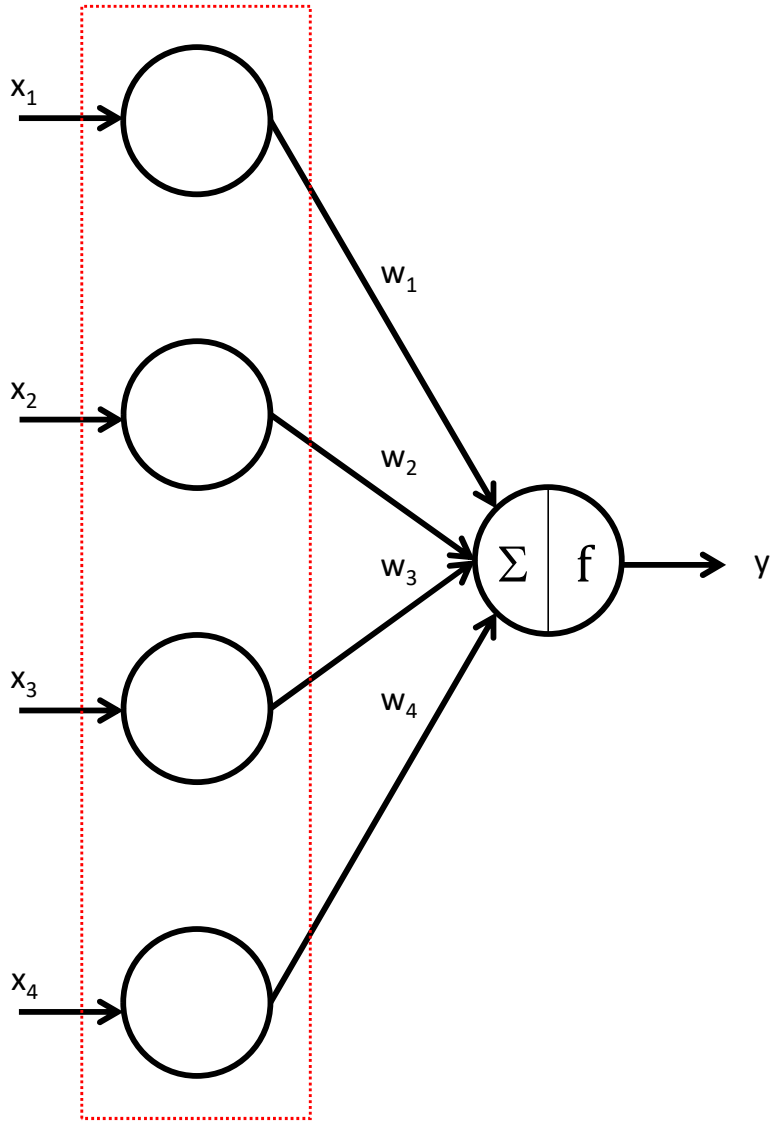


$\Sigma\ w_i * x_i \geq 0 \rightarrow f = 1$ → **"fire"**

Single-layer Perceptron as a Classifier

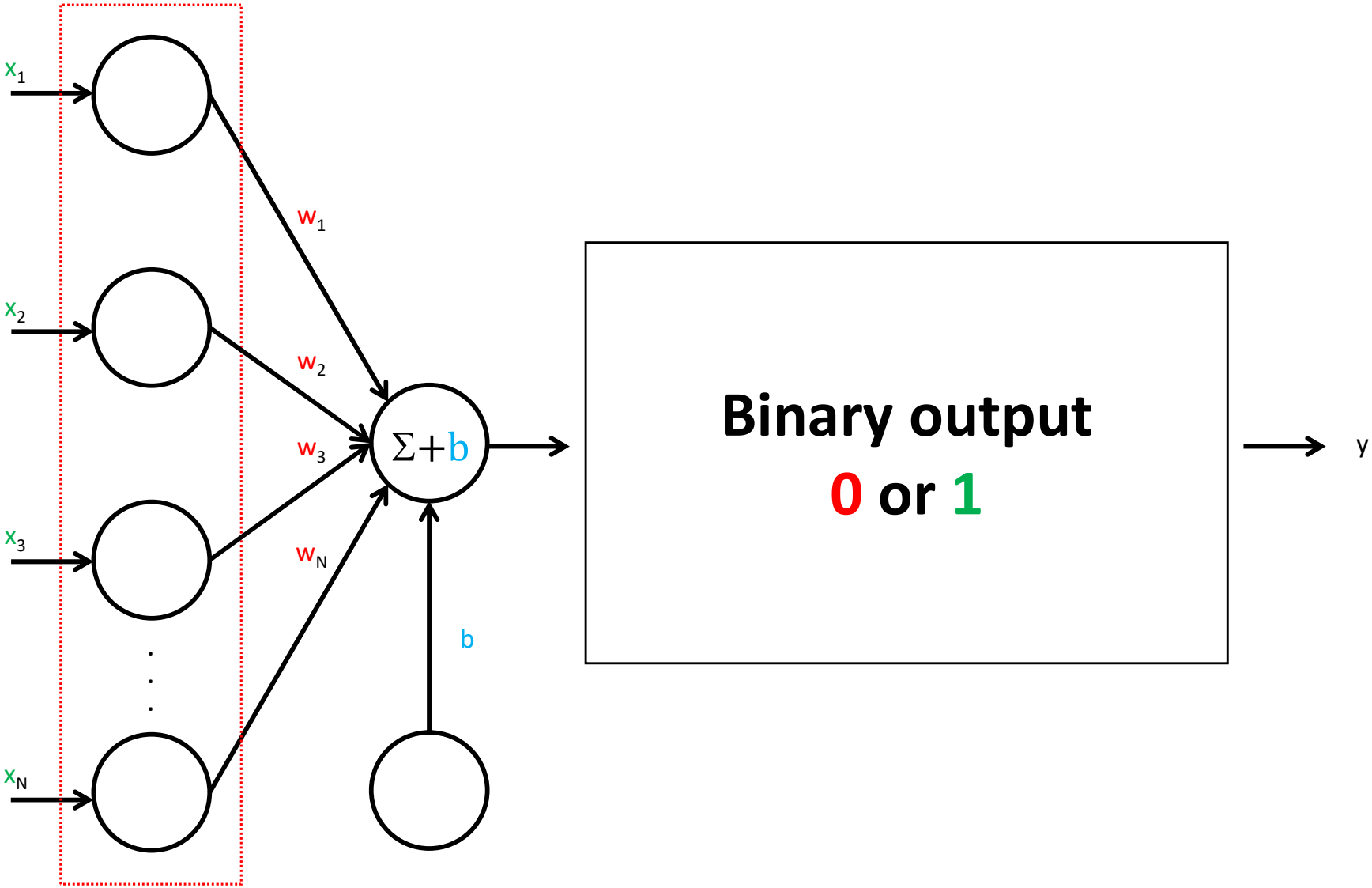


$\Sigma\, w_i * x_i < 0 \rightarrow f = 0 \rightarrow$ **NO**



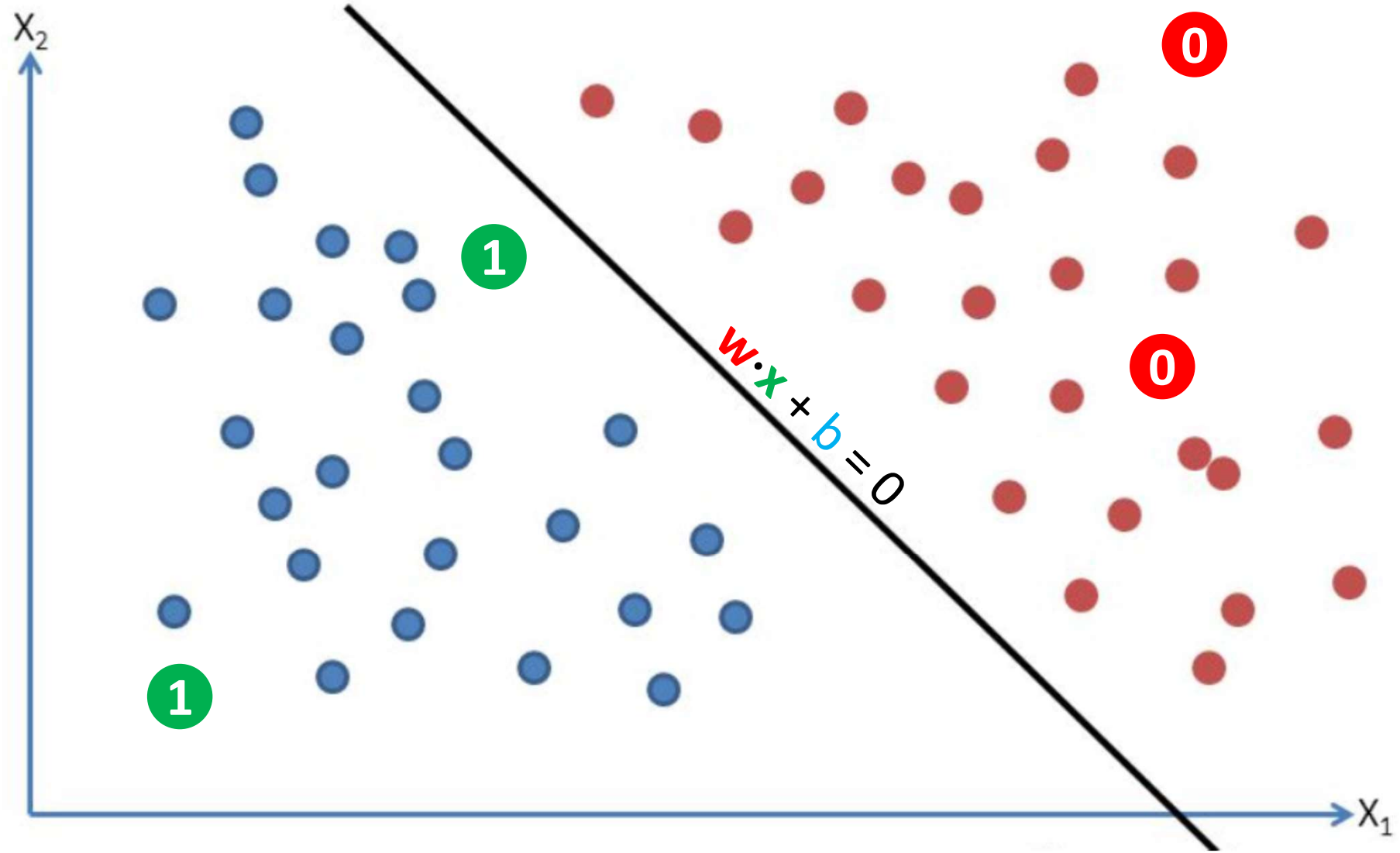
$\Sigma\, w_i * x_i \geq 0 \rightarrow f = 1 \rightarrow$ **YES**

Perceptrons = Linear Classifiers



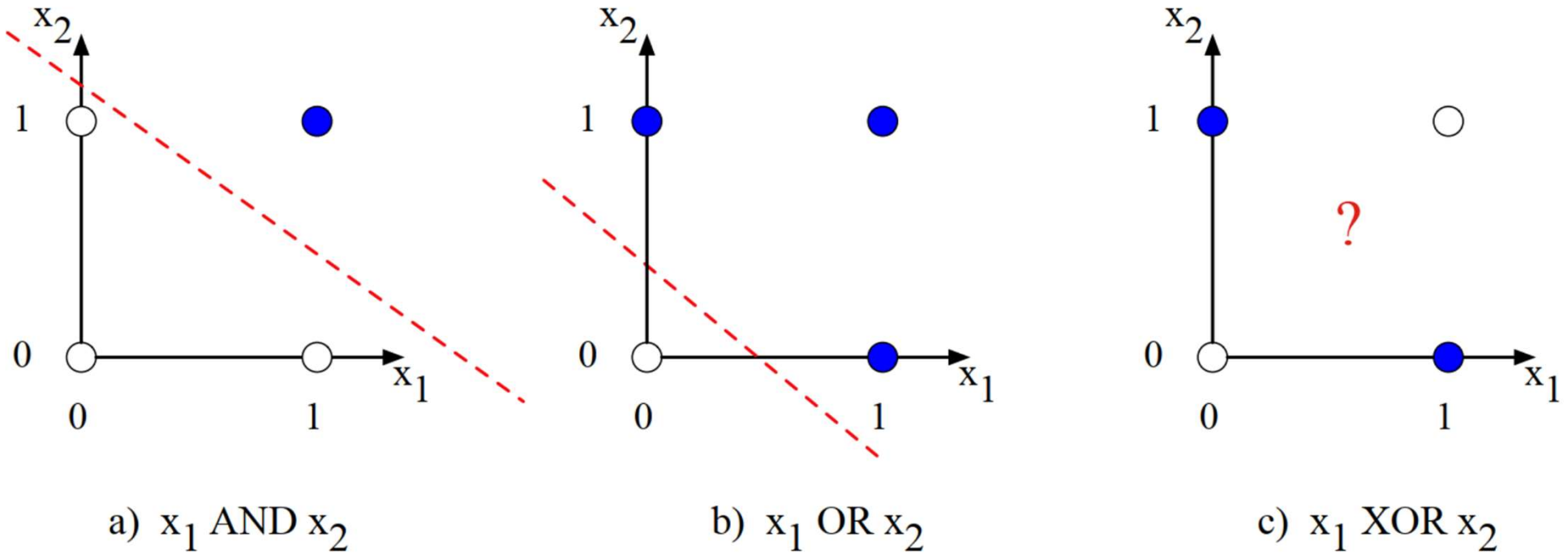
$\Sigma\ w_i * x_i + b \rightarrow f \rightarrow y$

Classification: Linear Separation



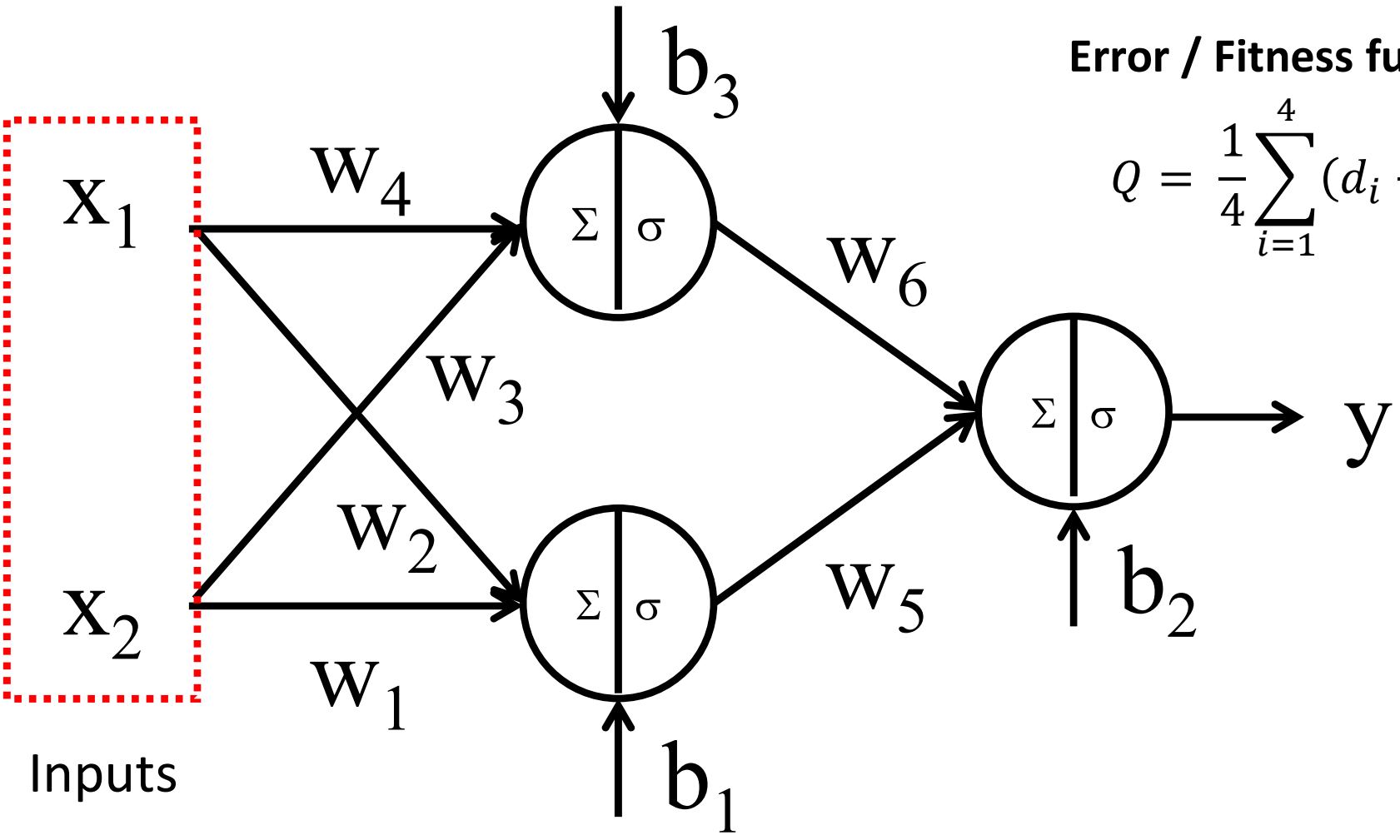
The **w·x** + b = 0 **line is the decision boundary**

XOR: Not a Linearly Separable f()



Logical XOR is an example of a function that is NOT linearly separable

XOR Gate With Neural Network

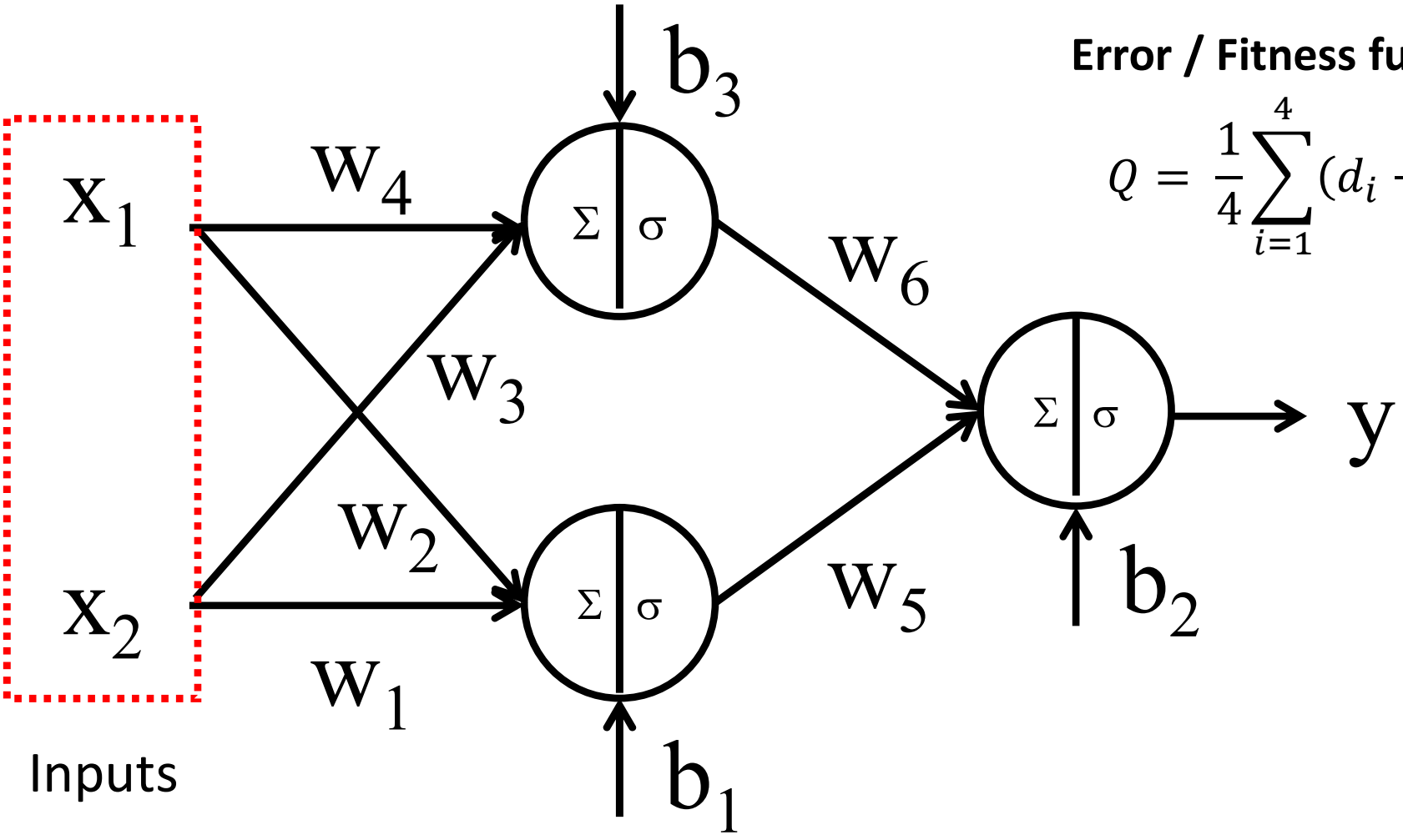


Error / Fitness function:

$$Q = \frac{1}{4}\sum_{i=1}^{4}(d_i - y_i)^2$$

$$y = \frac{1}{1 + e^{-\left(\frac{w_5}{1+e^{-(w_2*x_1+w_1*x_2+b_1)}} + \frac{w_6}{1+e^{-(w_4*x_1+w_3*x_2+b_3)}} + b_2\right)}}$$

Find Neural Network Weights With GA

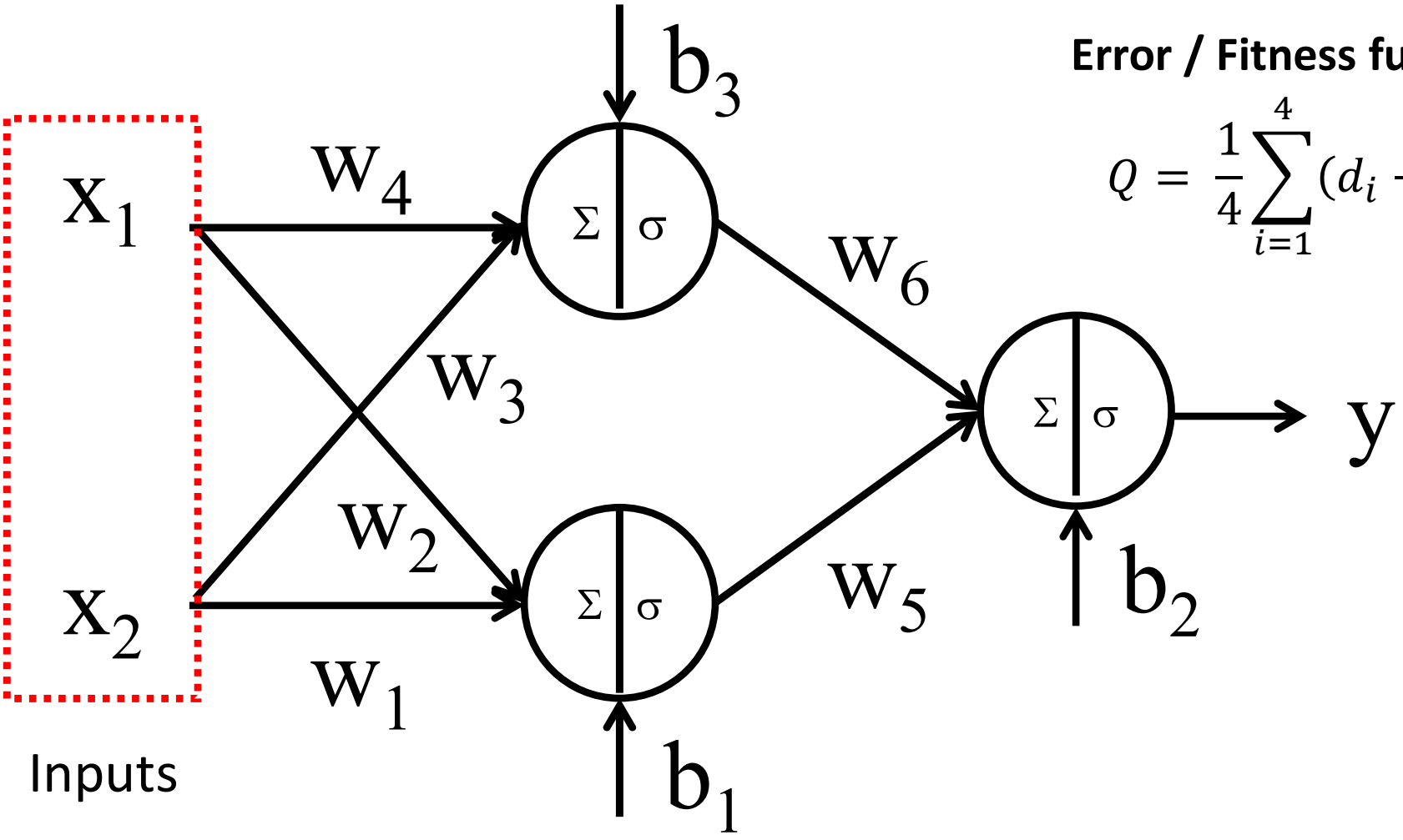


Error / Fitness function:

$$Q = \frac{1}{4}\sum_{i=1}^{4}(d_i - y_i)^2$$

$w_1 = ?, w_2 = ?, w_3 = ?, w_4 = ?, w_5 = ?, w_6 = ?, b_1 = ?, b_2 = ?, b_3 = ?$

Find Neural Network Weights With GA

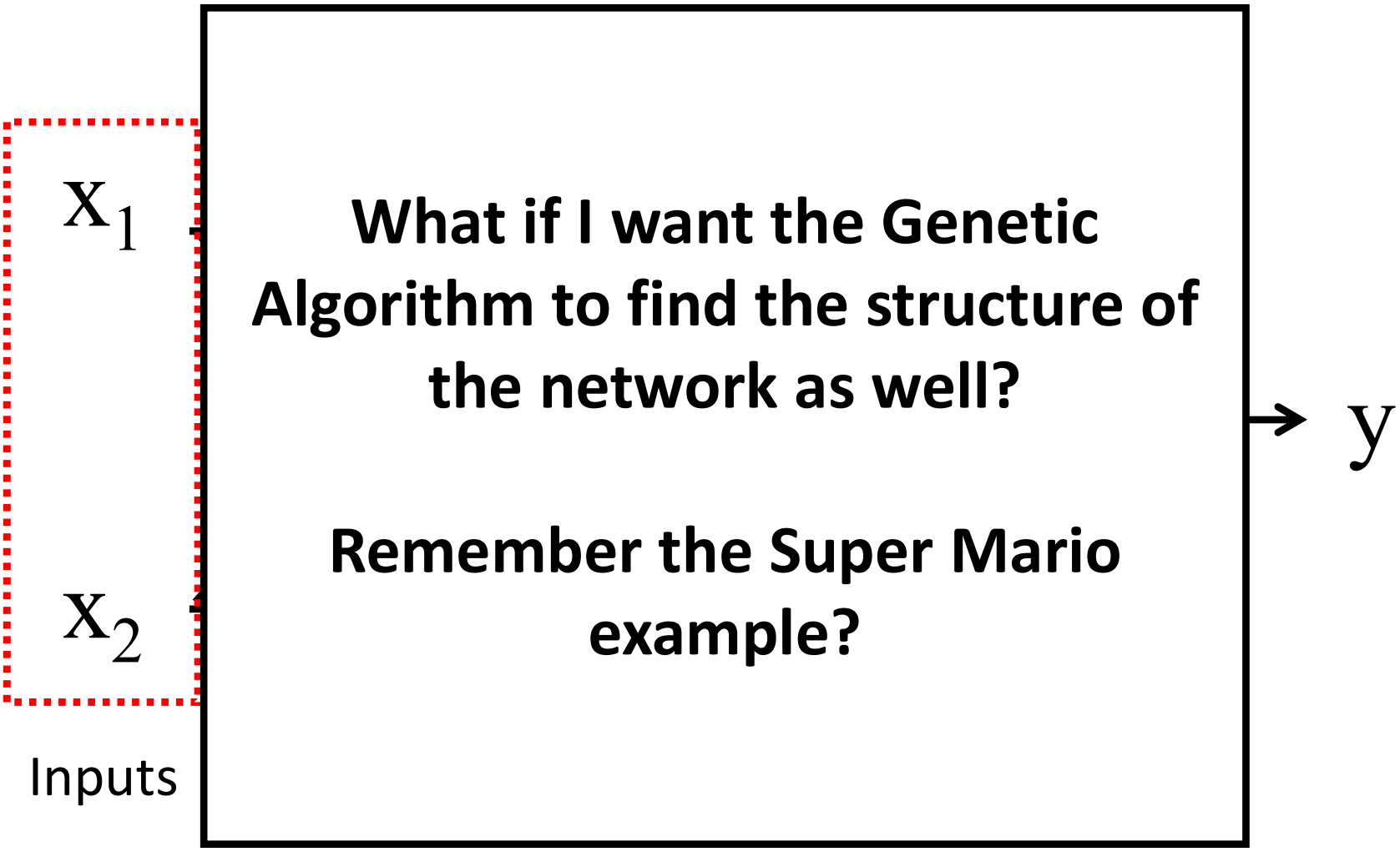


Error / Fitness function:

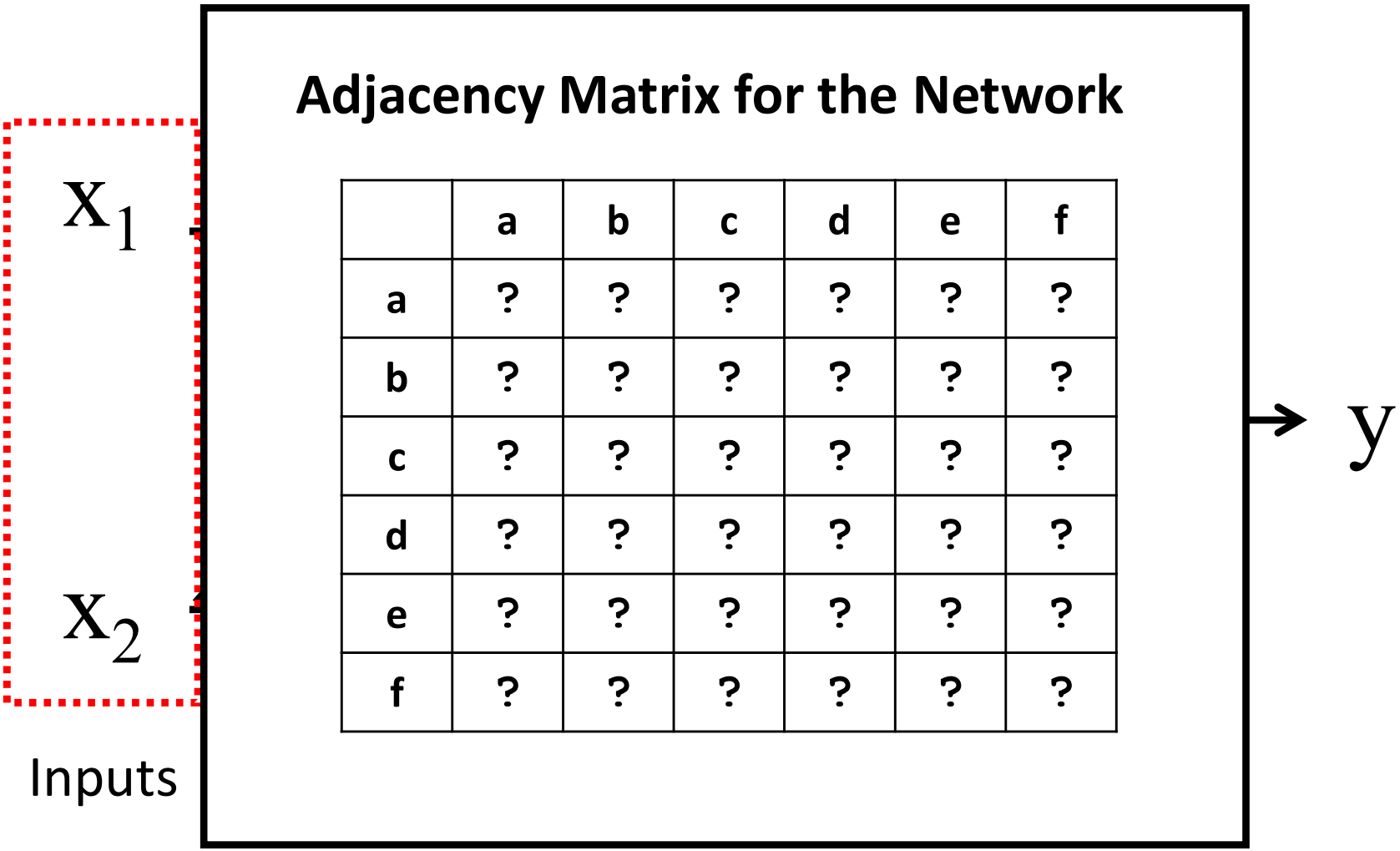
$$Q = \frac{1}{4}\sum_{i=1}^{4}(d_i - y_i)^2$$

Chromosome (5 bits per gene should suffice):								
w_1	w_2	w_3	w_4	w_5	w_6	b_1	b_2	b_3

Find Neural Network Structure /w GA

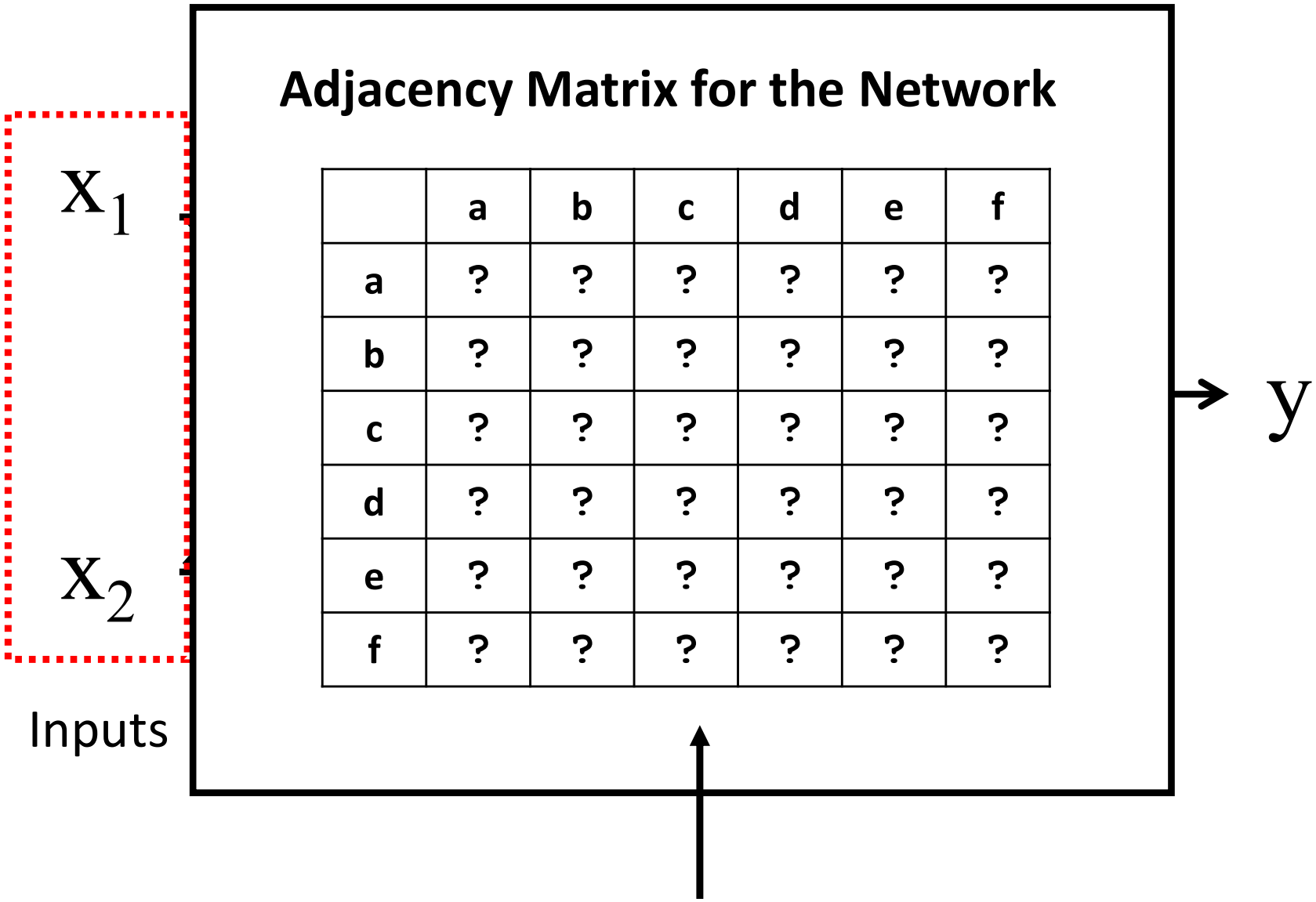


Find Neural Network Structure /w GA



	a	b	c	d	e	f
a	?	?	?	?	?	?
b	?	?	?	?	?	?
c	?	?	?	?	?	?
d	?	?	?	?	?	?
e	?	?	?	?	?	?
f	?	?	?	?	?	?

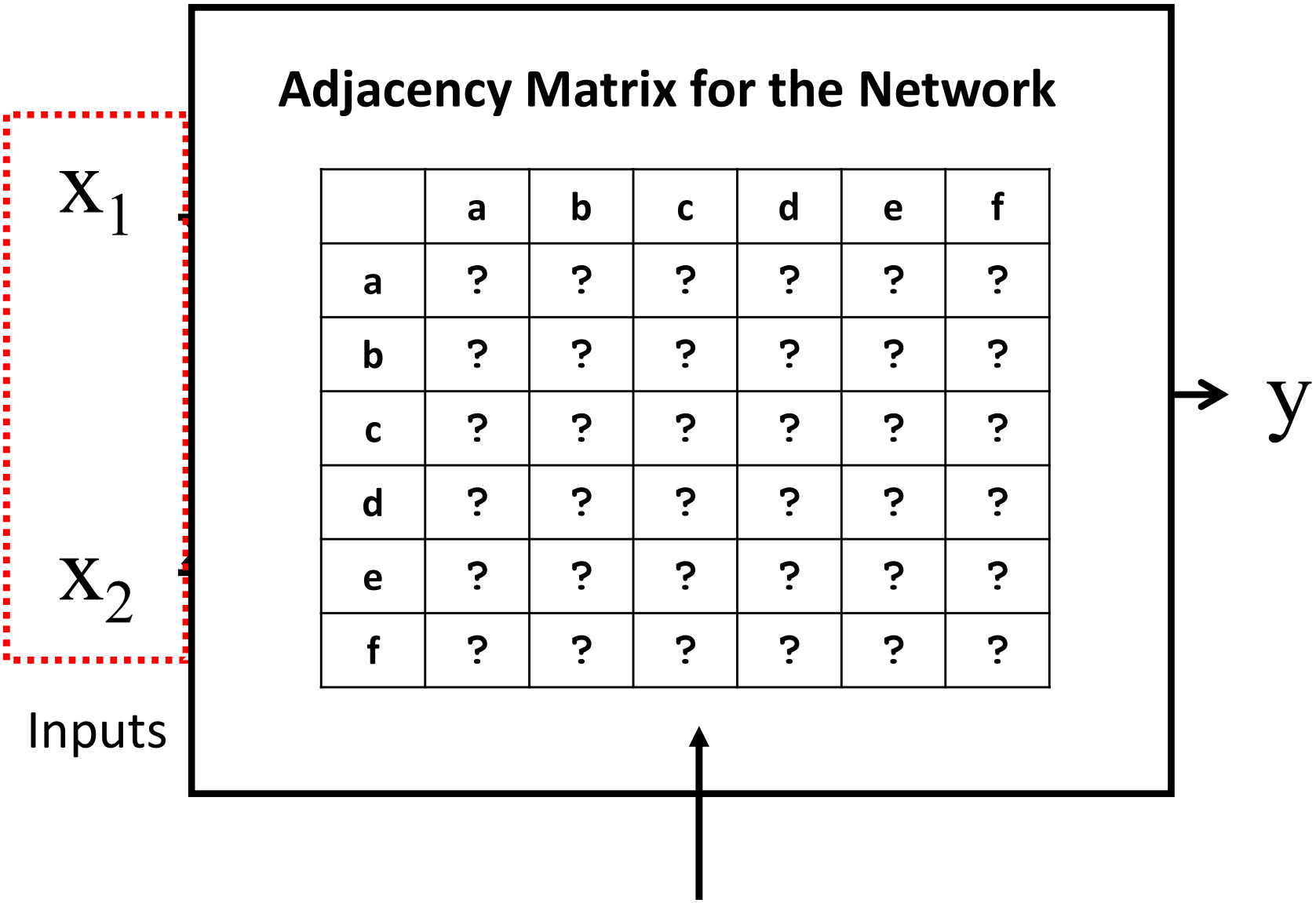
Find Neural Network Structure /w GA



	a	b	c	d	e	f
a	?	?	?	?	?	?
b	?	?	?	?	?	?
c	?	?	?	?	?	?
d	?	?	?	?	?	?
e	?	?	?	?	?	?
f	?	?	?	?	?	?

Turn this into a chromosome (for example every row a gene)

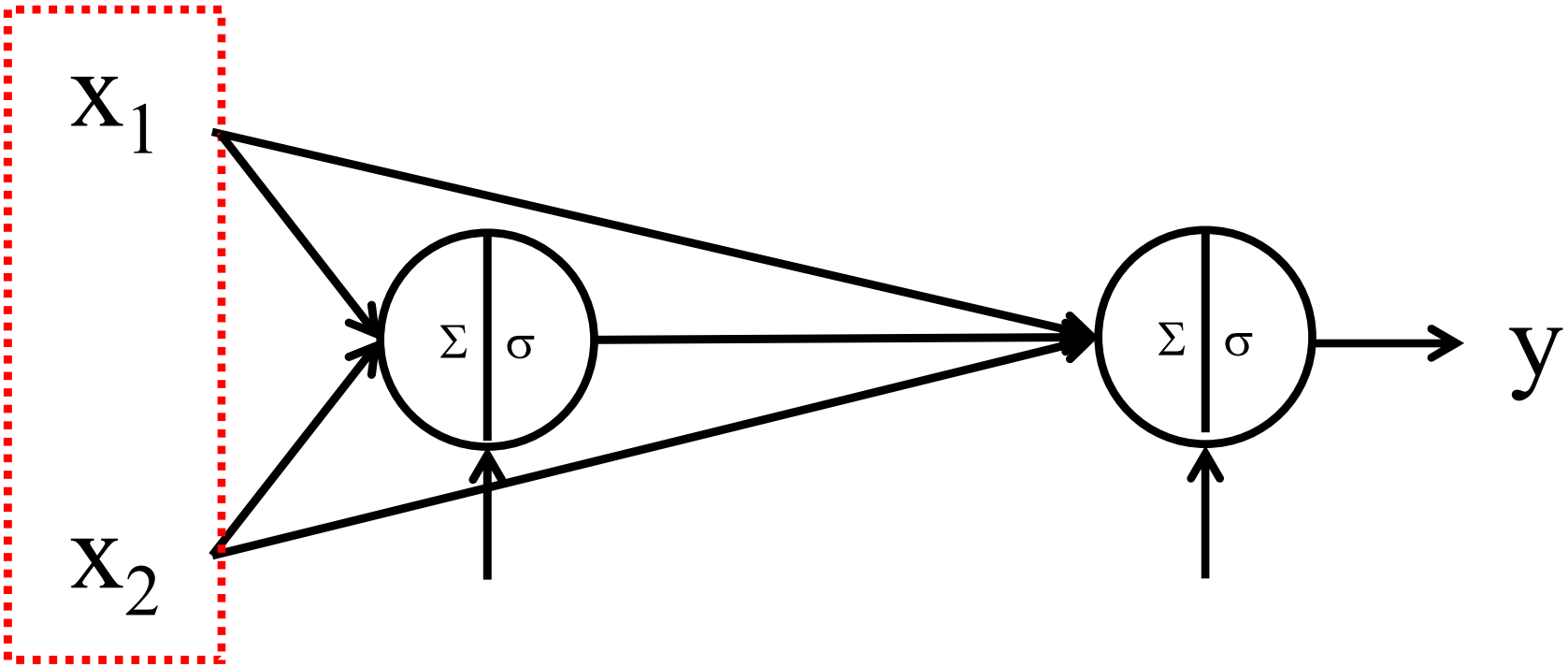
Find Neural Network Structure /w GA



	a	b	c	d	e	f
a	?	?	?	?	?	?
b	?	?	?	?	?	?
c	?	?	?	?	?	?
d	?	?	?	?	?	?
e	?	?	?	?	?	?
f	?	?	?	?	?	?

Set a limit on the number of nodes and change y formula

Potential GA Solution

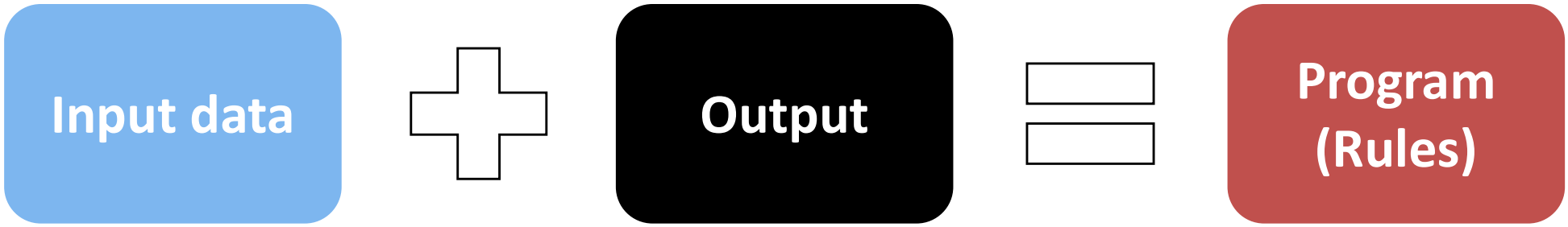


Genetic Programming

Traditional Programming vs …

Traditional programming:

…:



Genetic Programming

- Genetic programming (GP) is an automated method for creating a working computer program from a high-level problem statement of a problem.
- Genetic programming starts from a high-level statement of “what needs to be done” and automatically creates a computer program to solve the problem
- John Koza - "Genetic Programming: On the Programming of Computers by Means of Natural Selection"

Genetic Programming

- Goal: find a program that generates correct solution
  - based on input – correct output data
- Genotype: tree [GA: string]
- Phenotype: actual program [GA: f() value)]
- Evaluation:
  - run program
  - see if output data is expected

GA vs GP

Genetic Algorithm:

Genetic Programming:

Example Problem Description

Objective	Find a computer program with one input X for which the output Y is equal to the given data
Terminal set	$T = \{\text{variables, constants}\}$
Function/operator set	$F = \{\text{functions, operators}\}$
Initial population	Randomly created individuals built using elements from T and F.
Fitness function	$\lvert y_0' - y_0\rvert + \lvert y_1' - y_1\rvert + \ldots$ where y_i' is computed output and y_i is given output for x_i in the range $[-1,1]$
Termination condition	An individual emerges with the value of its fitness function is less than ε

Subtree Crossover

Consider a program in C programming language:

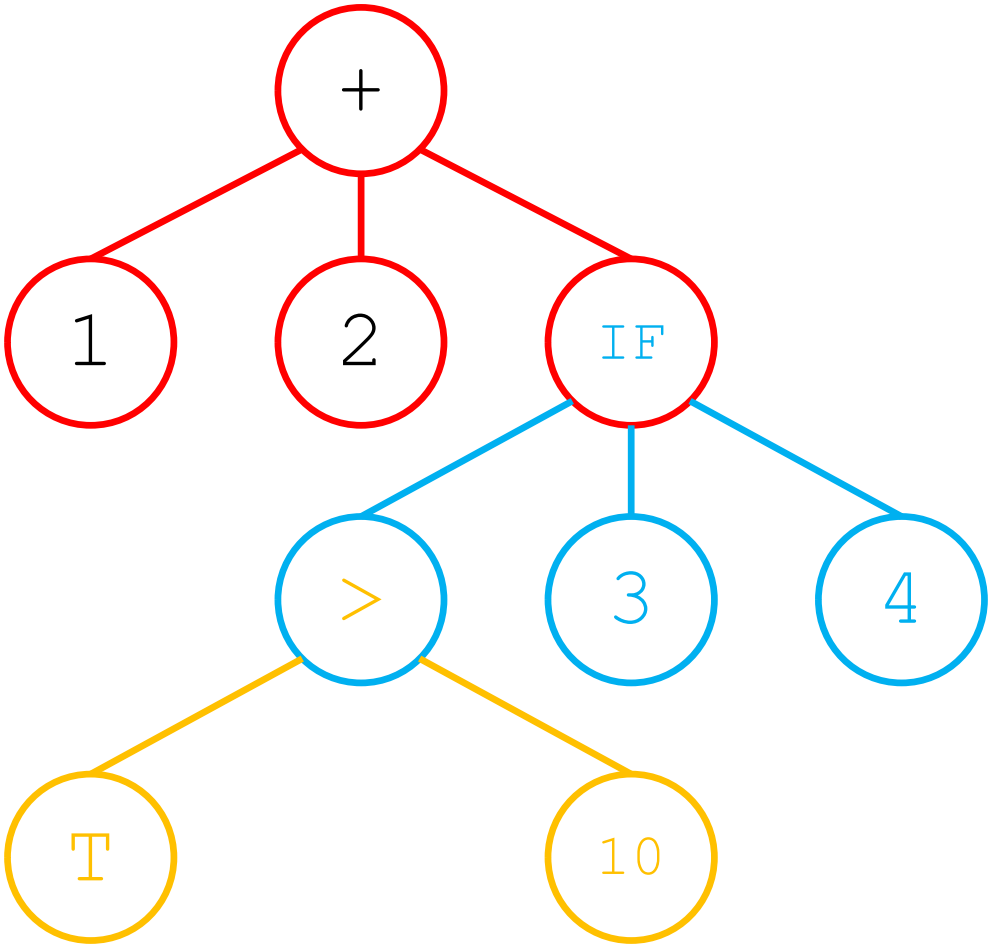
```
int foo (int time)
{
    int temp1, temp2;
    if (T > 10)
        temp1 = 3;
    else
        temp1 = 4;
    temp2 = temp1 + 1 + 2;
    return (temp2);
}
```

Equivalent expression:

```
(+ 1 2 (IF (> T 10) 3 4))
```

Program Tree

```
(+ 1 2 (IF (> T 10) 3 4))
```



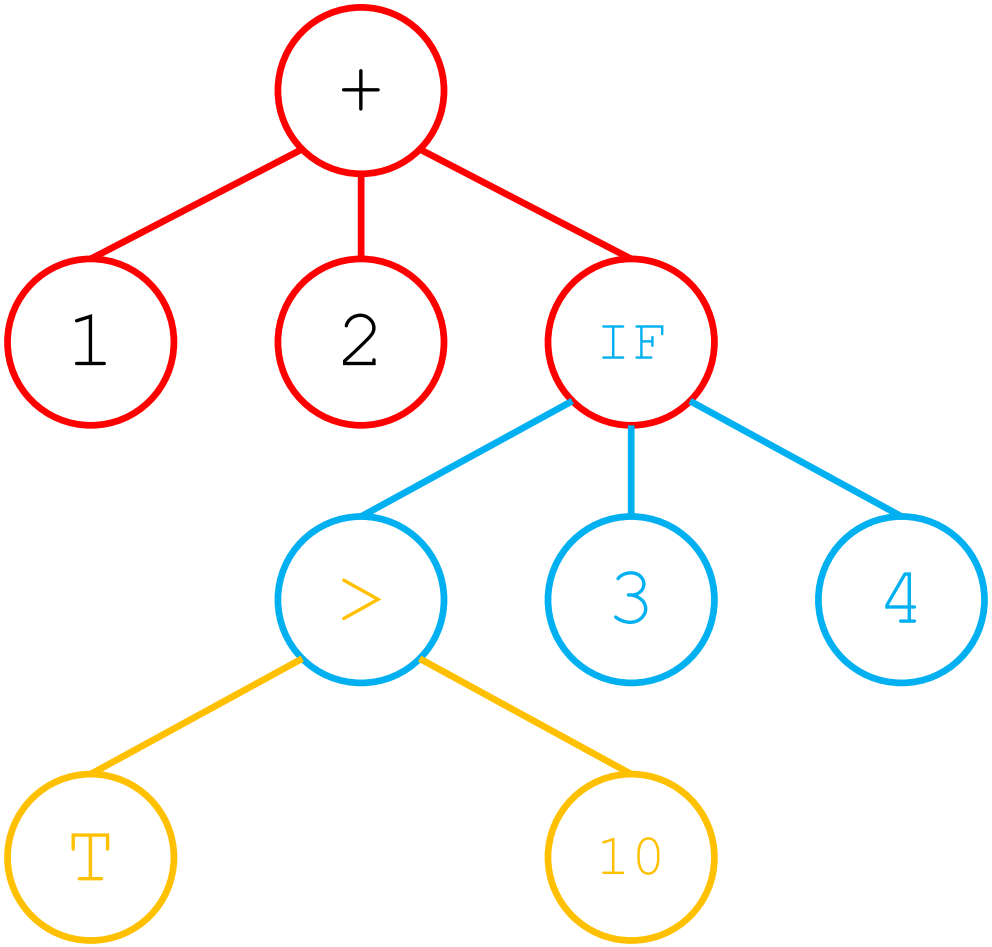
Functions/Operators vs. Terminals

- Functions / Operators
  - require arguments

- Terminals:
  - tree leaf
  - constants
  - variables
  - “external” function calls

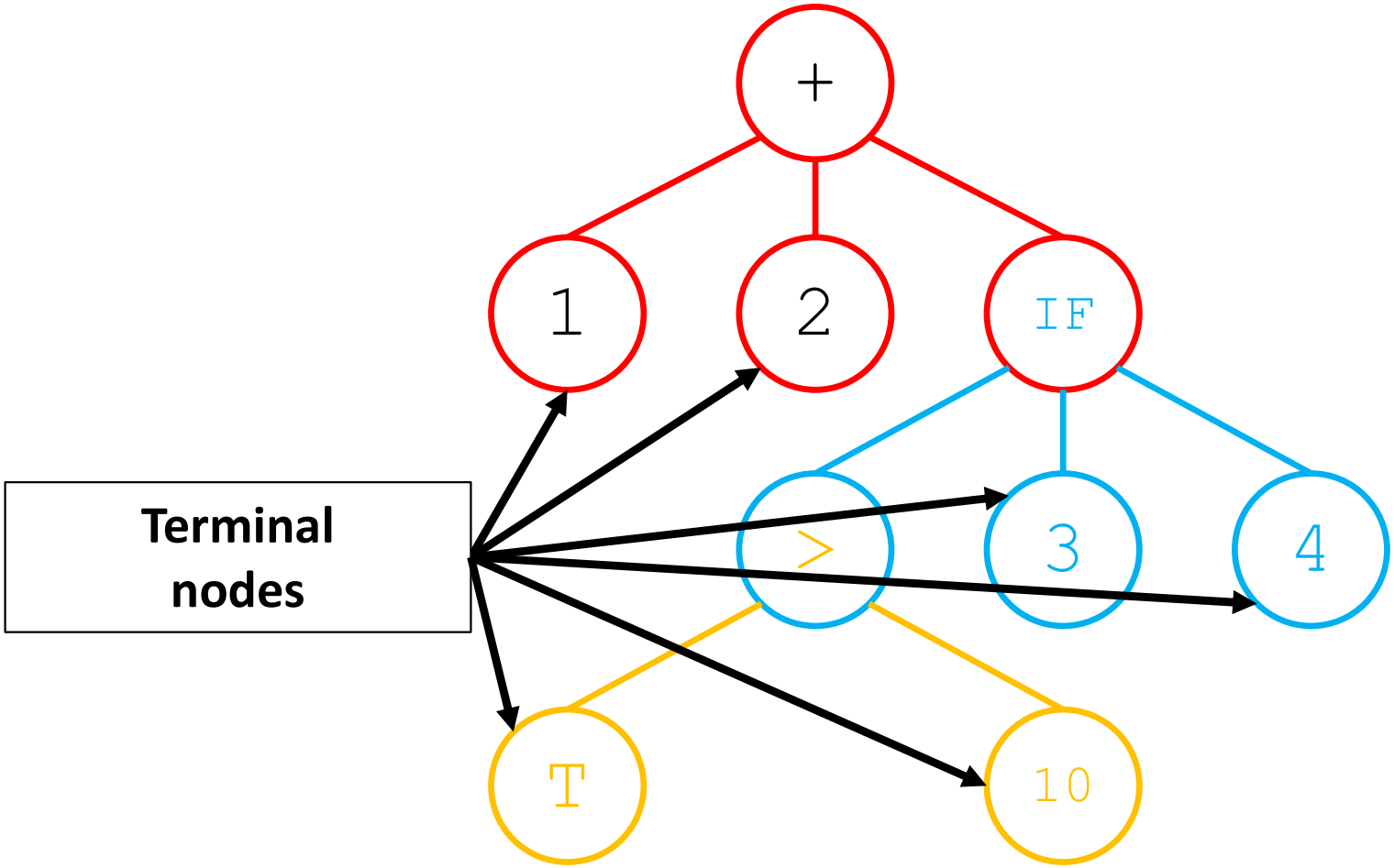
Program Tree

```
(+ 1 2 (IF (> T 10) 3 4))
```



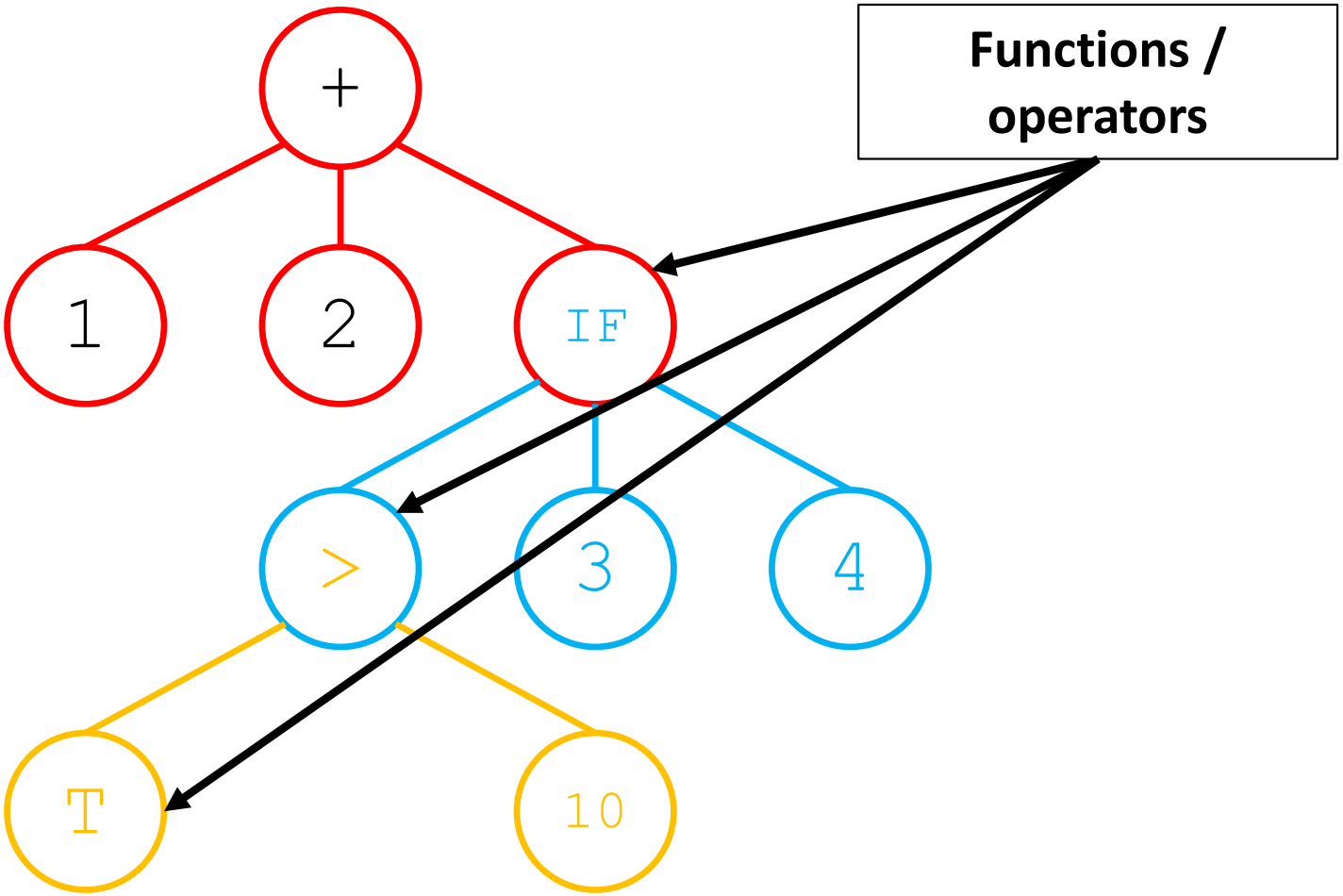
Program Tree

```
(+ 1 2 (IF (> T 10) 3 4))
```



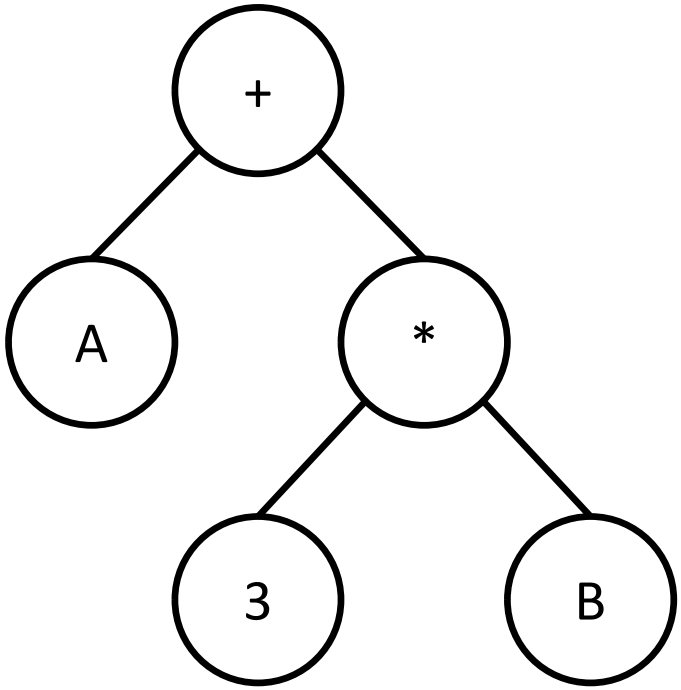
Program Tree

```
(+ 1 2 (IF (> T 10) 3 4))
```



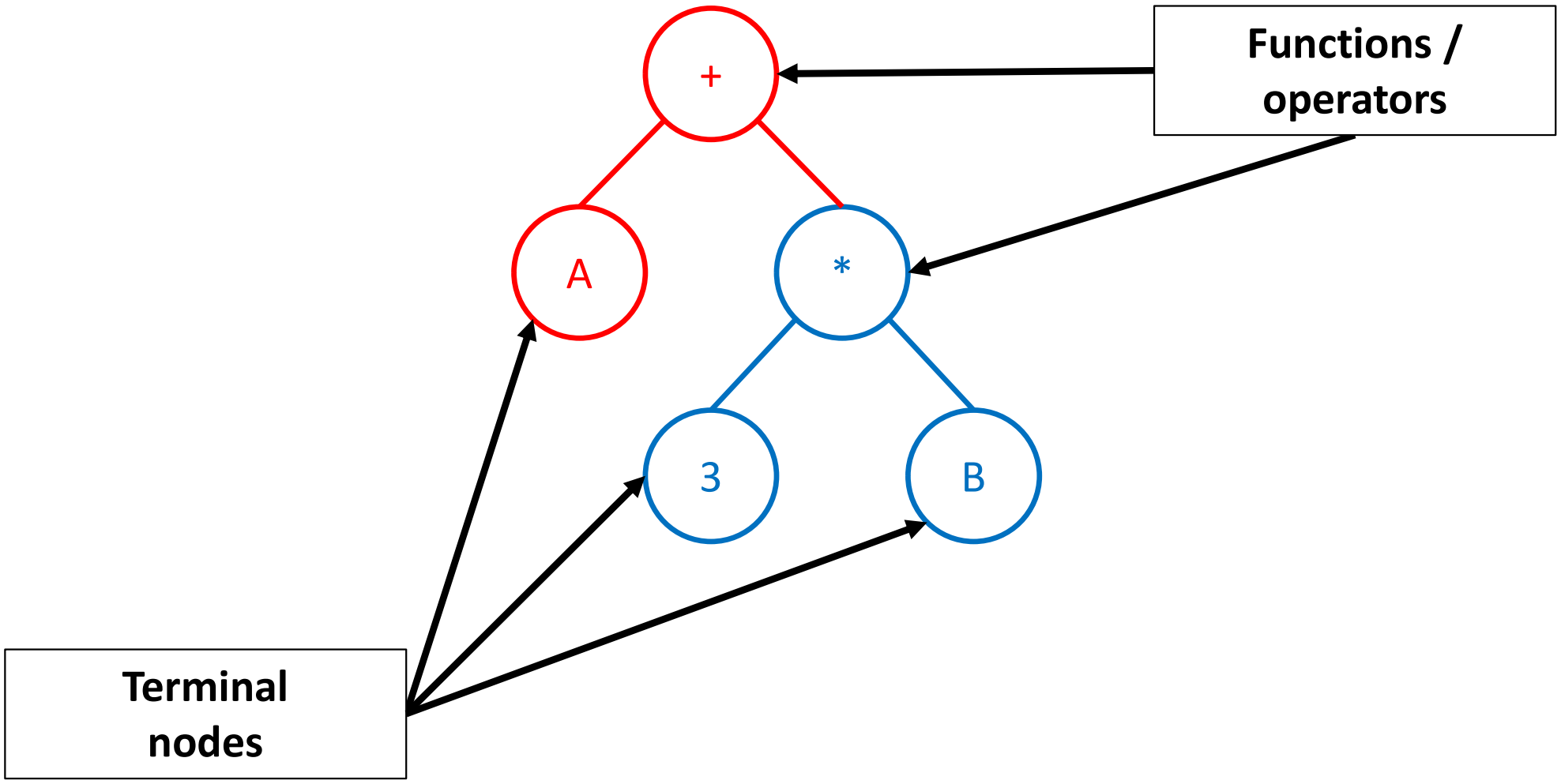
Abstract Syntax Trees

A + 3 * B



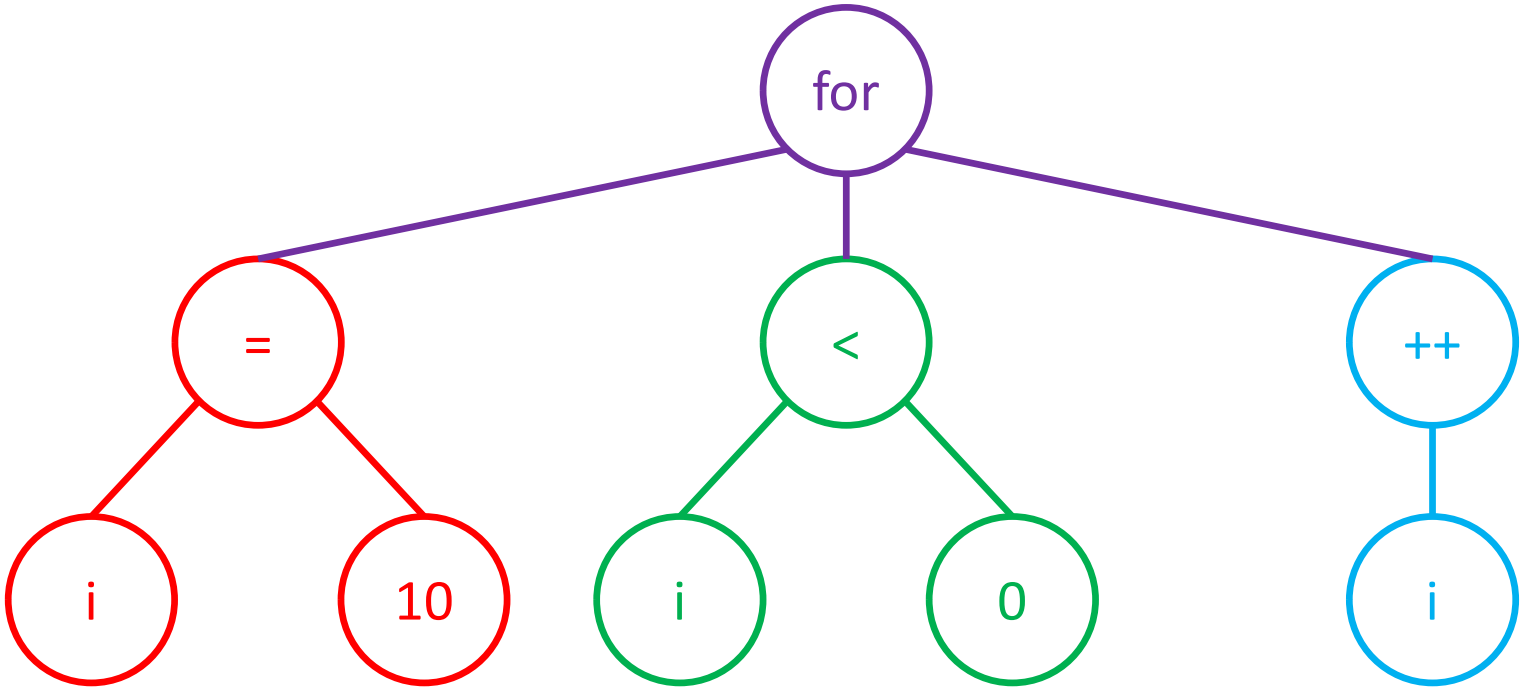
Abstract Syntax Trees

A + 3 * B = A + (3 * B) = (A + (3 * B))



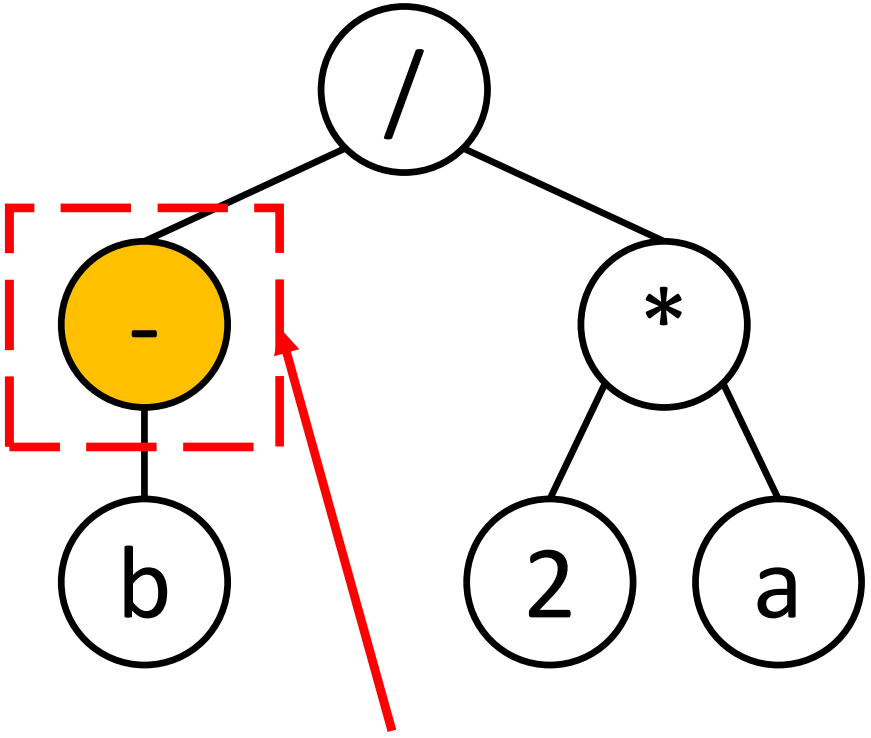
Abstract Syntax Trees

for(i = 0; i < 0; i++)



Genetic Programming: Mutation

$$\frac{-b}{2 * a}$$

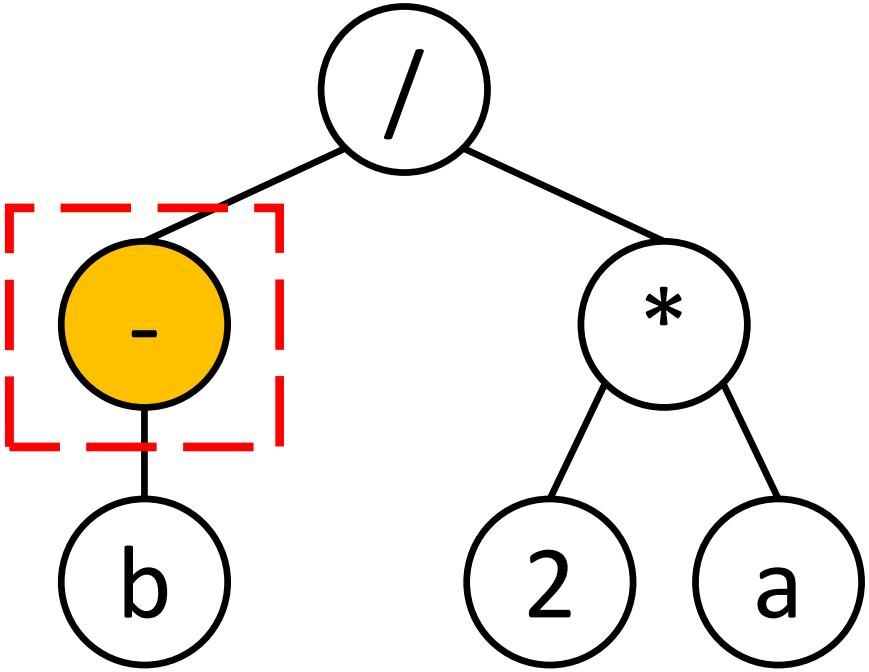


Pick a node for mutation randomly

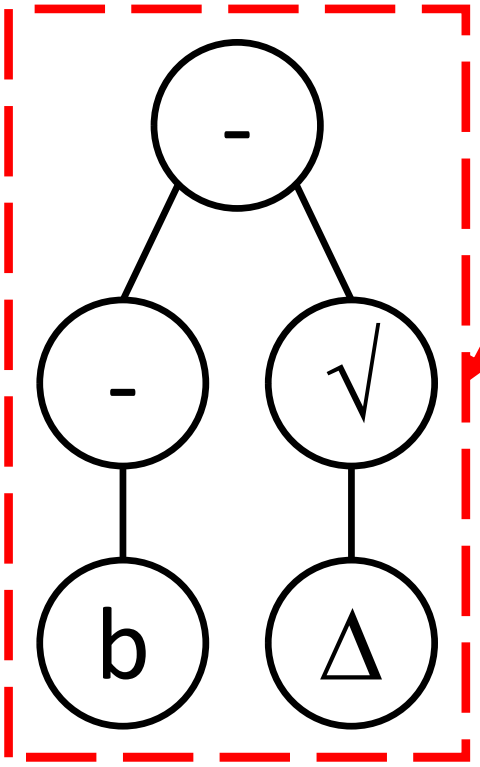
Individual B

Genetic Programming: Mutation

$$\frac{-b}{2 * a}$$



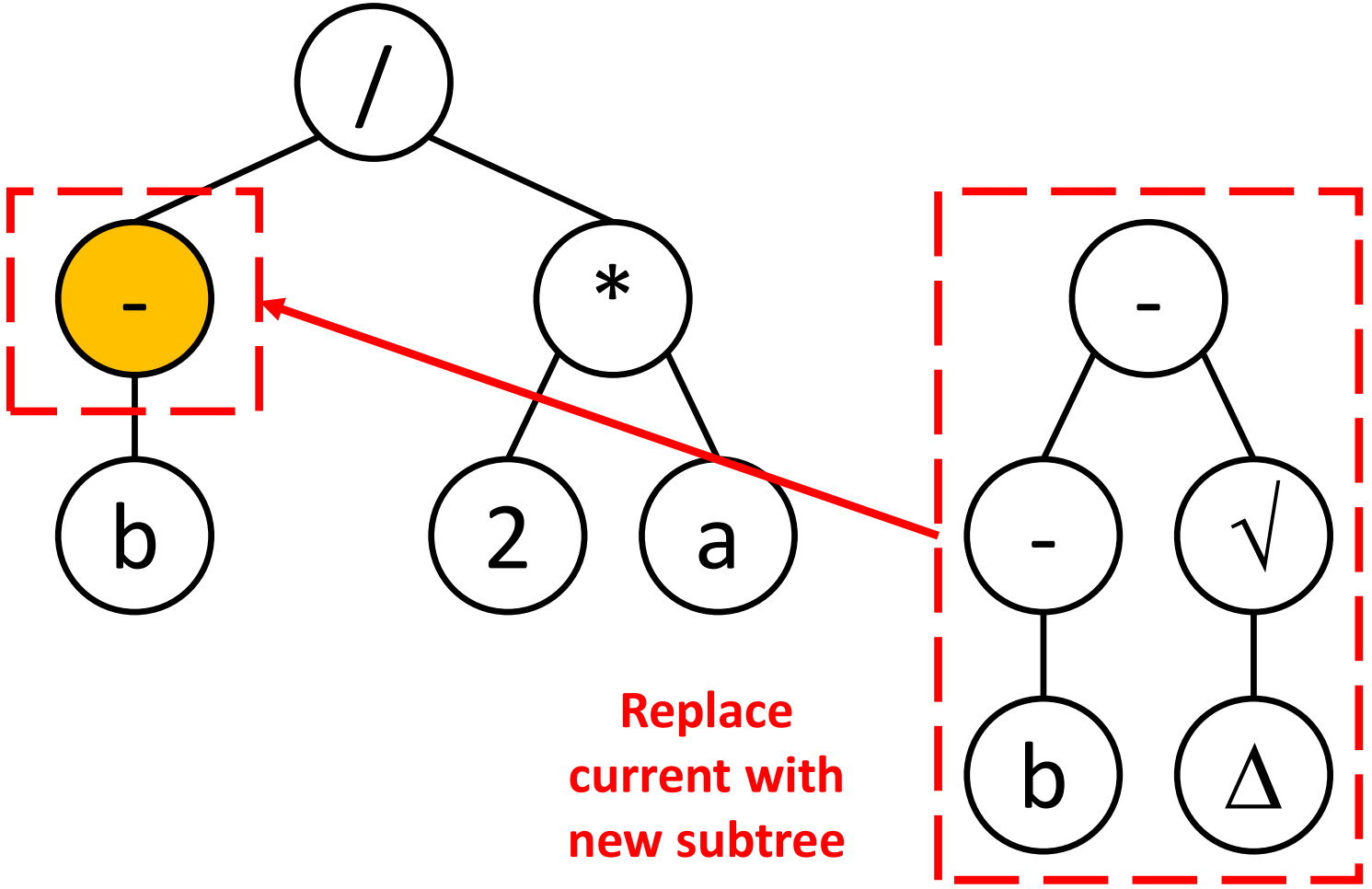
Generate random subtree



Individual B

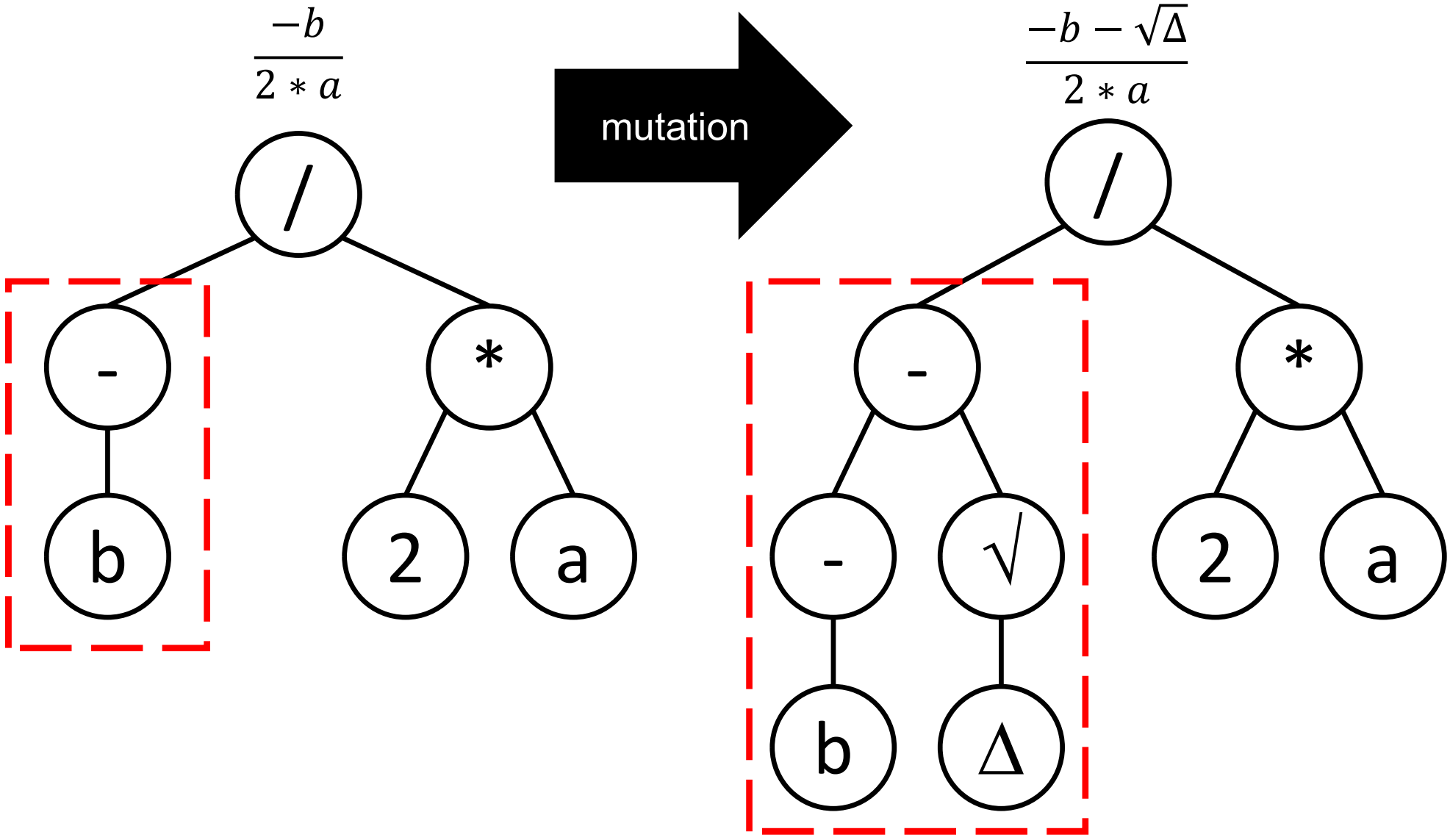
Genetic Programming: Mutation

$$\frac{-b}{2 * a}$$



Individual B

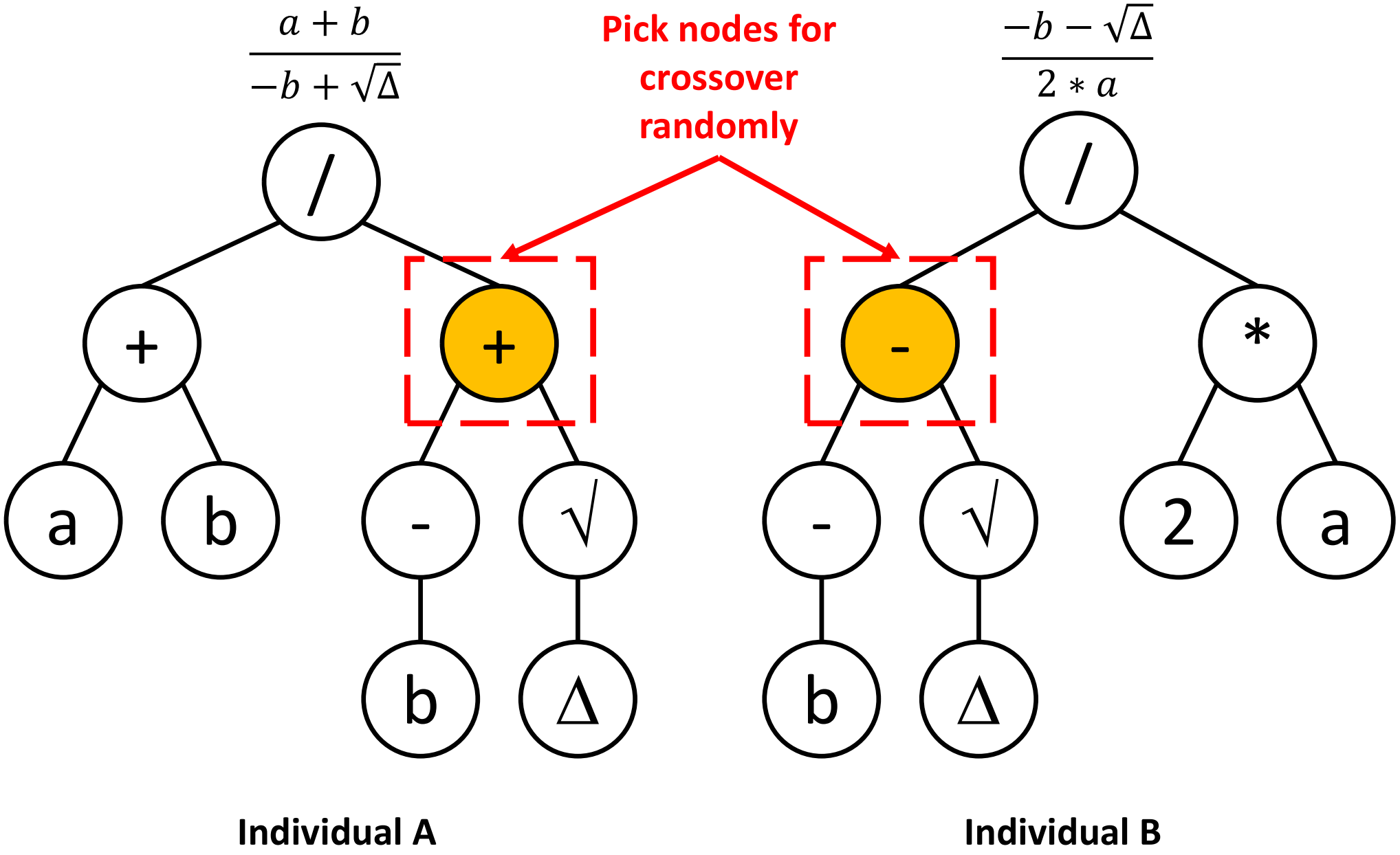
Genetic Programming: Mutation



Individual B

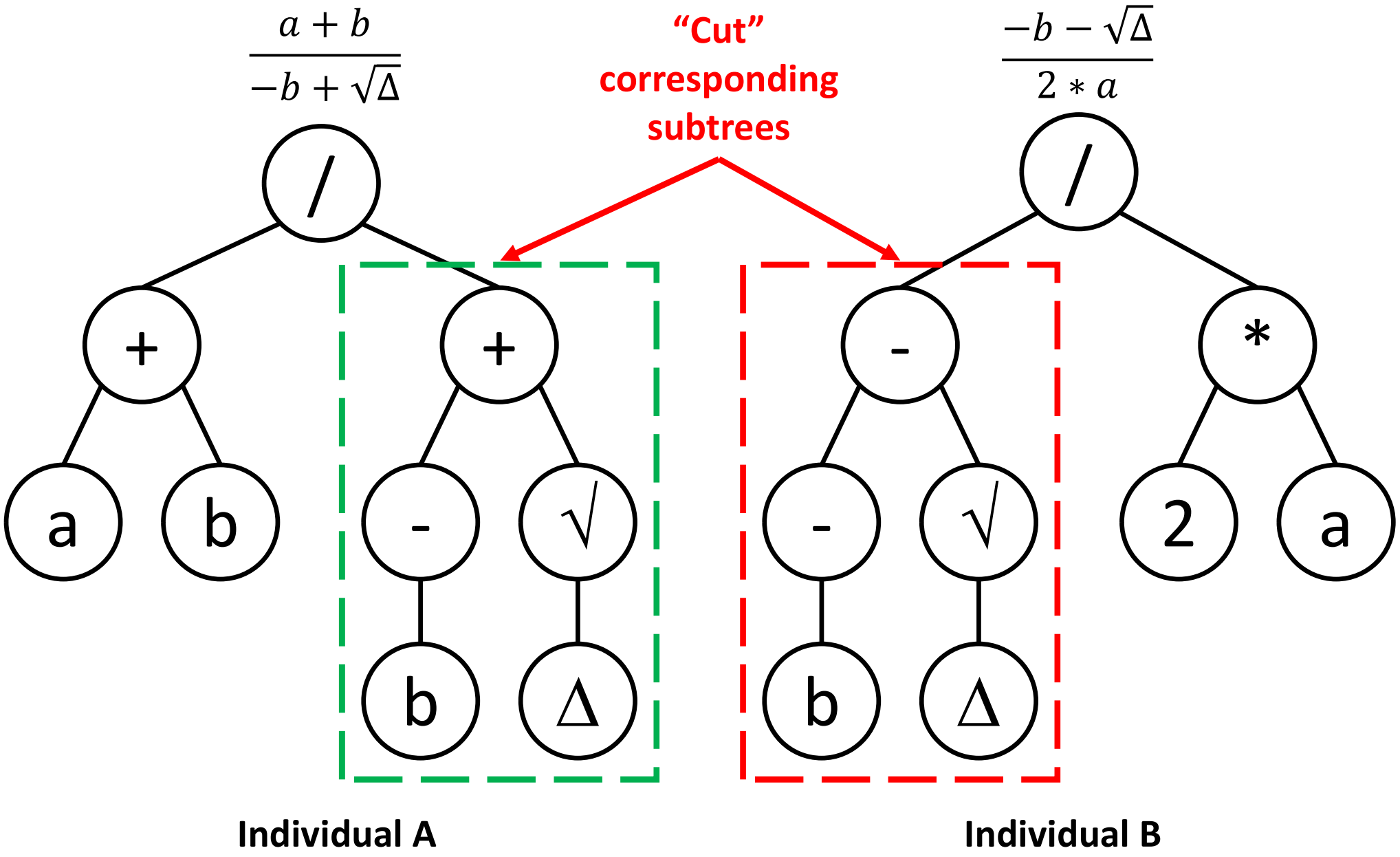
Individual B after mutation

Genetic Programming: Crossover

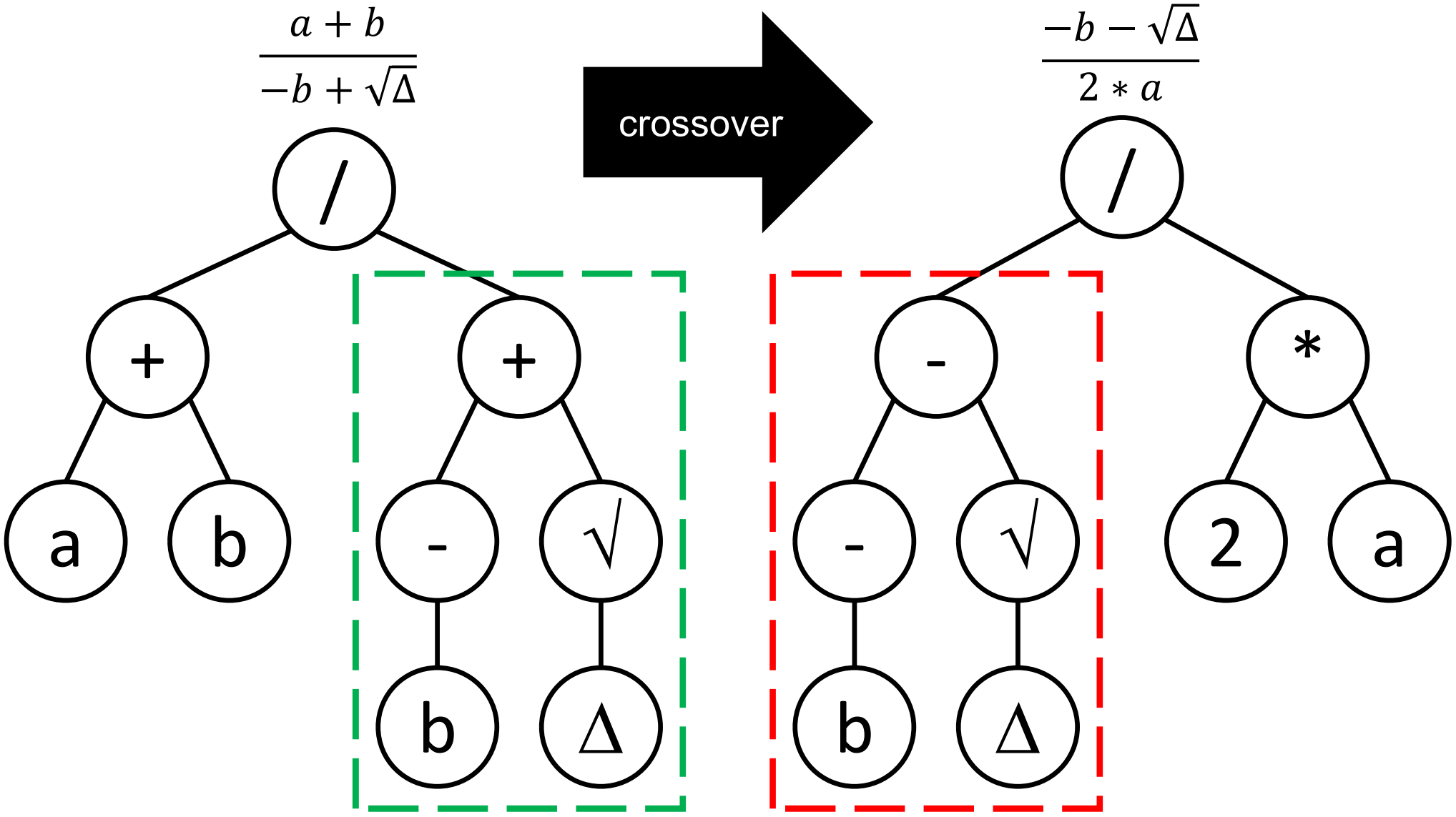


Individual A **Individual B**

Genetic Programming: Crossover

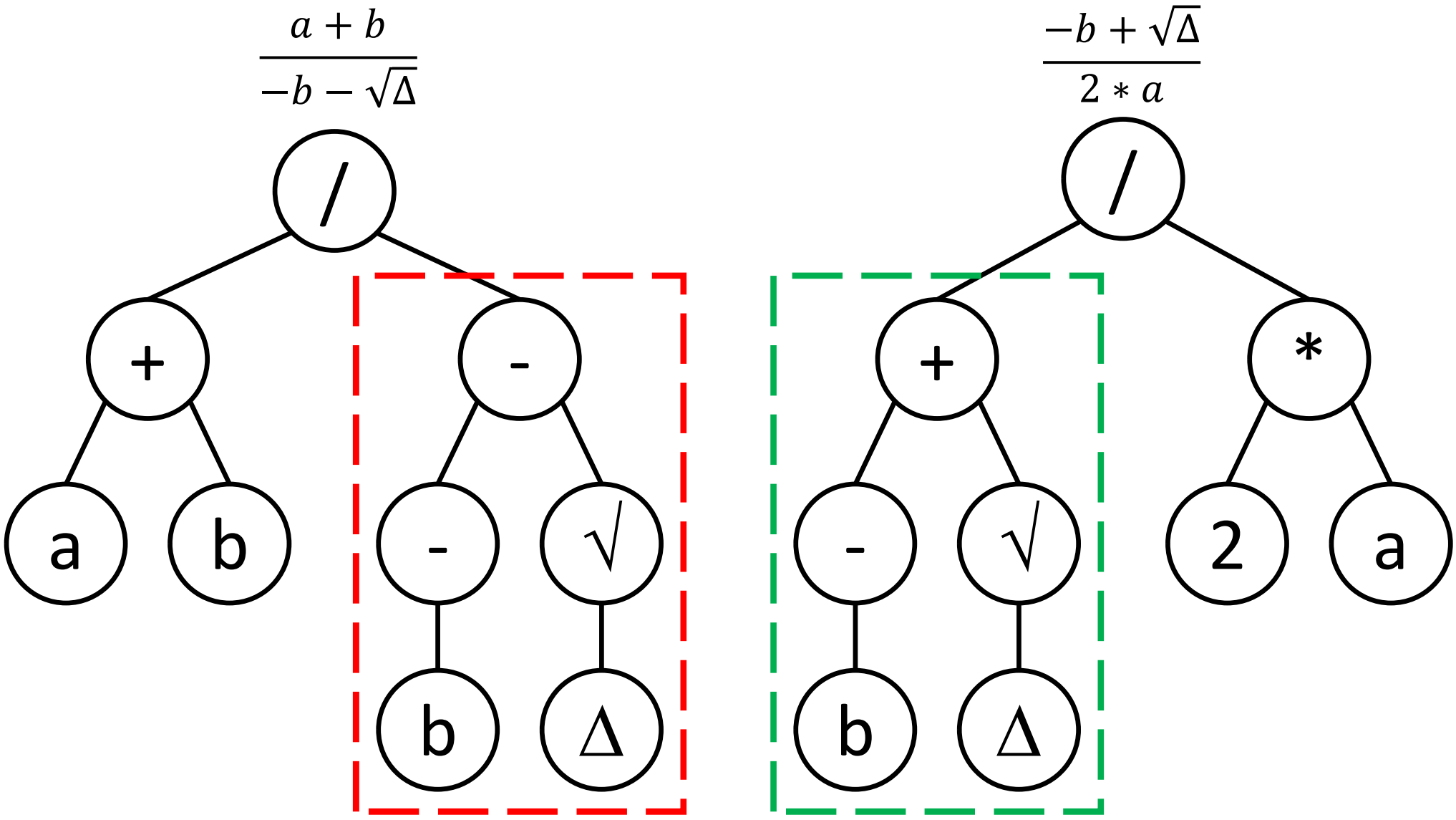


Genetic Programming: Crossover



Individual A **Individual B**

Genetic Programming: Crossover



Child 1 of individuals A and B

Child 2 of individuals A and B

Example Problem: Data

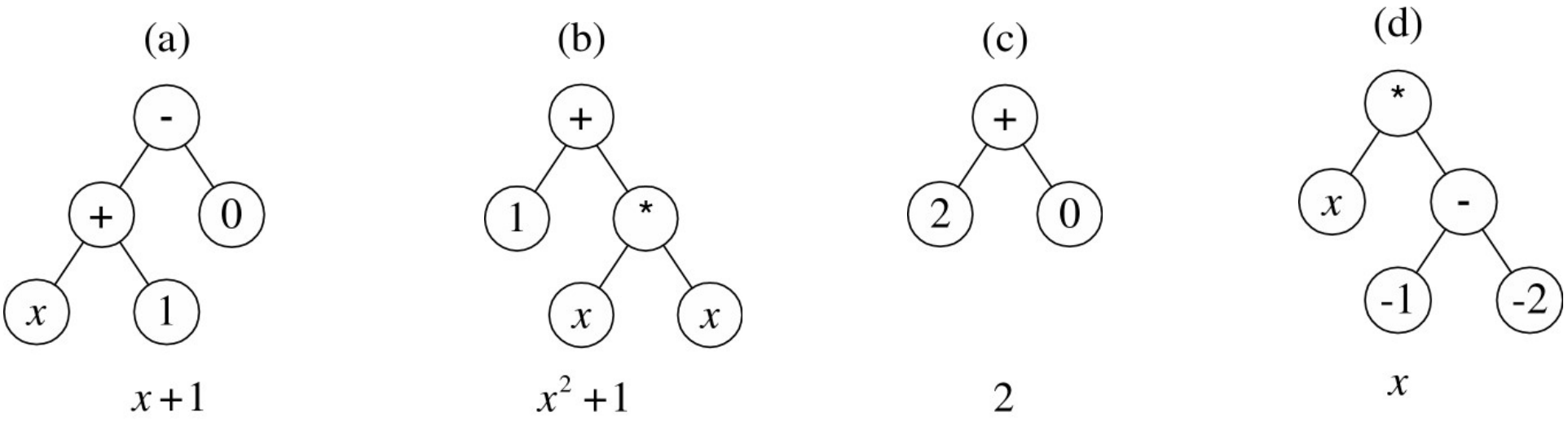
Input: Independent variable X	Output: Dependent variable Y
-1.00	1.00
-0.80	0.84
-0.60	0.76
-0.40	0.76
-0.20	0.84
0.00	1.00
0.20	1.24
0.40	1.56
0.60	1.96
0.80	2.44
1.00	3.00

Example Problem Description

Objective	Find a computer program with one input X for which the output Y is equal to the given data
Terminal set	$T = \{X, \text{Constants}\}$
Function/operator set	$F = \{+, -, *, /\}$
Initial population	Randomly created individuals built using elements from T and F.
Fitness function	$\lvert y_0' - y_0\rvert + \lvert y_1' - y_1\rvert + \ldots$ where y_i' is computed output and y_i is given output for x_i in the range $[-1,1]$
Termination condition	An individual emerges with the value of its fitness function is less than ε

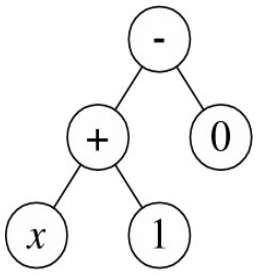
Example Problem: Generation 0

Initial population of four randomly created individuals:



Example Problem: Generation 1

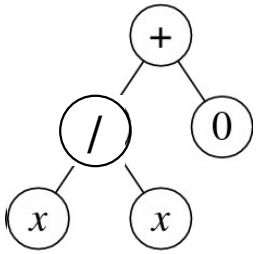
(a)



$x+1$

Copy of (a)

(b)

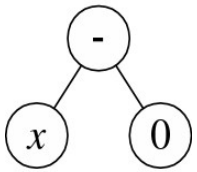


1

Mutant of (c)

picking "2" as mutation point

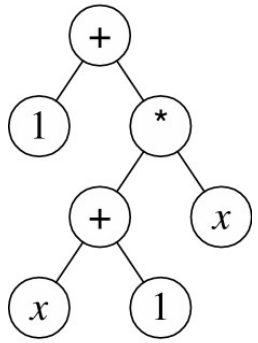
(c)



x

First offspring of crossover of (a) and (b) picking "+" of parent (a) and left-most "x" of parent (b) as crossover points

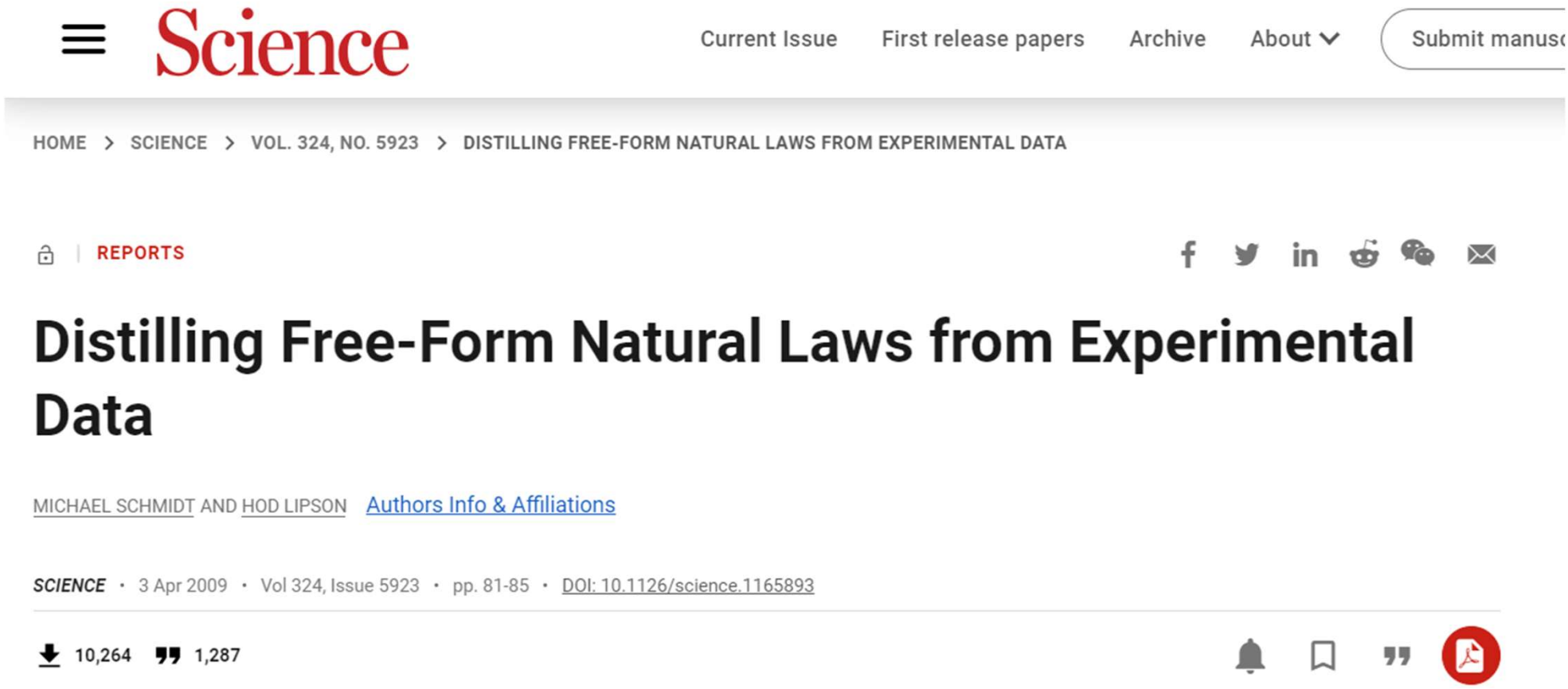
(d)



x^2+x+1

Second offspring of crossover of (a) and (b) picking "+" of parent (a) and left-most "x" of parent (b) as crossover points

Genetic Programming: Application



Genetic Programming: Video