Genetic Algorithms

Programming by the Seat of Your Genes!

Jeremy Fisher

@thisisroot

June 29, 2016

Genetic algorithms are a biologically inspired stochastic metaheuristic for combinatorial search and optimization.

Genetic algorithms are a biologically inspired ~~stochastic metaheuristic for combinatorial search and optimization~~ fancy method of trial-and-error.

Evolutionary Algorithm Complexity

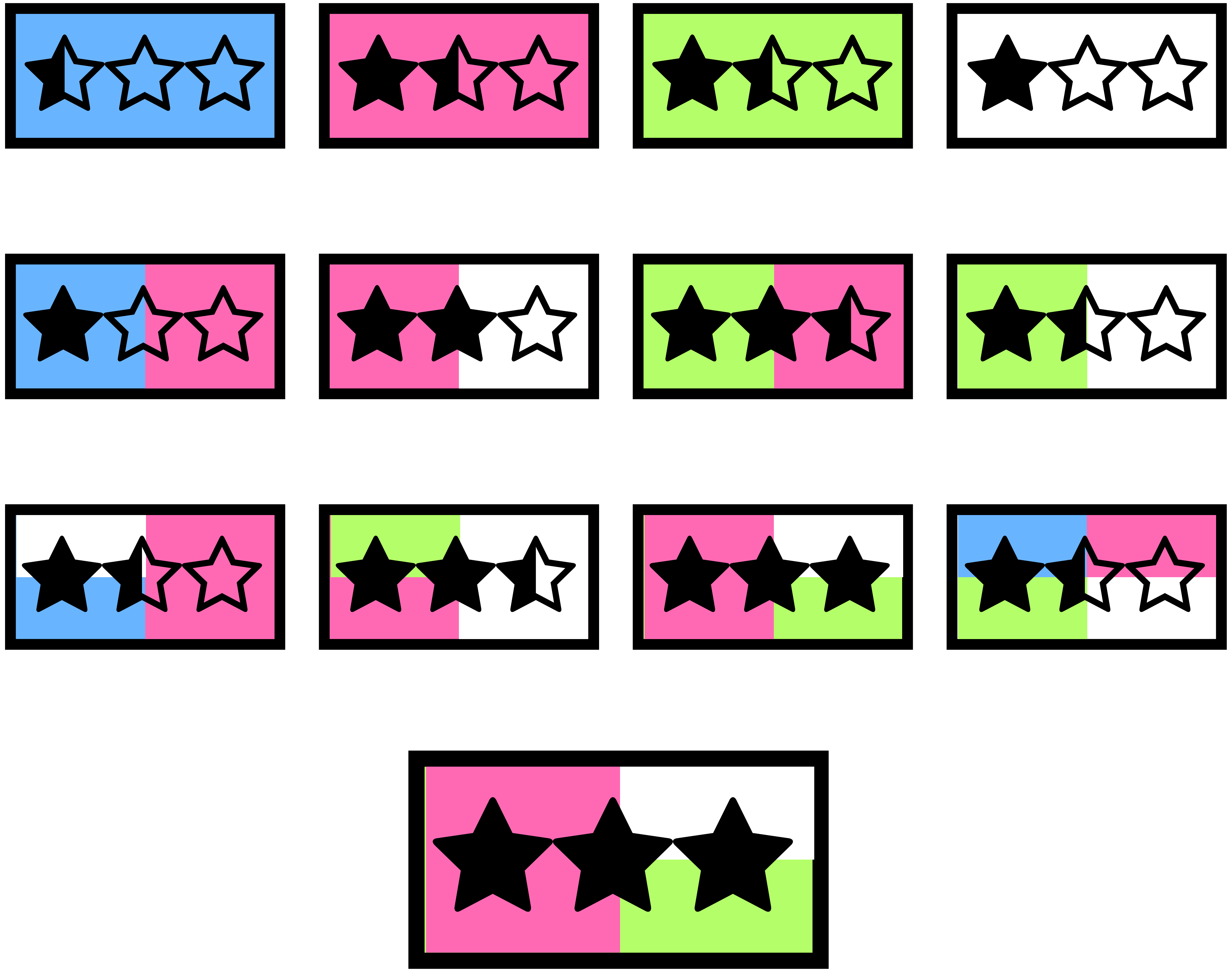
Beginner	Intermediate	Advanced
◦ Binary encoding ◦ Single Point Crossover ◦ Uniform Crossover ◦ Proportionate Selection ◦ Tournament Selection	◦ List Encoding ◦ Value Encoding ◦ Permutation Encoding ◦ Parallel GAs ◦ Ordered Crossover ◦ Partially Matched Crossover ◦ Cycle Crossover ◦ Tree Encoding ◦ Edge Recombination ◦ Distributed GAs ◦ Multi-Objective Fitness Functions ◦ Diploidism	◦ Coevolutionary Algorithms ◦ Genetics-Based Machine Learning ◦ Genetic Programming ◦ Estimation of Distribution Algorithms

Genetic Algorithms In The Wild

- Operations Research
- Sociology
- Game Theory
- Economics
- Financial Trading
- Biology

Genetic Algorithm Lifecycle

1. Create a population of random **chromosomes** (solutions).
2. Score each chromosome in the population for **fitness**.
3. Create a new generation through **mutation** and **crossover**.
4. Repeat until done.
5. Emit the fittest chromosome as the solution.

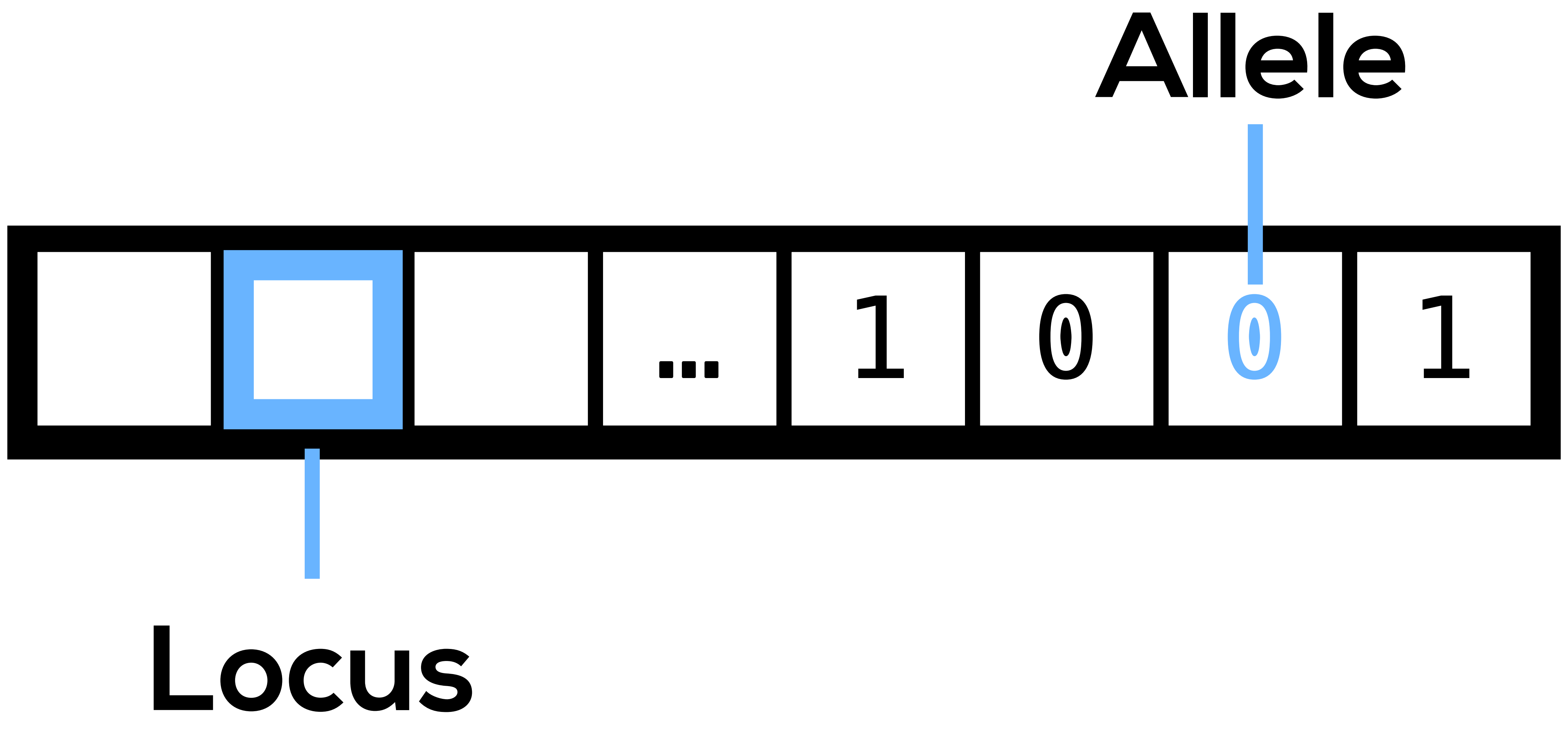


Terminology

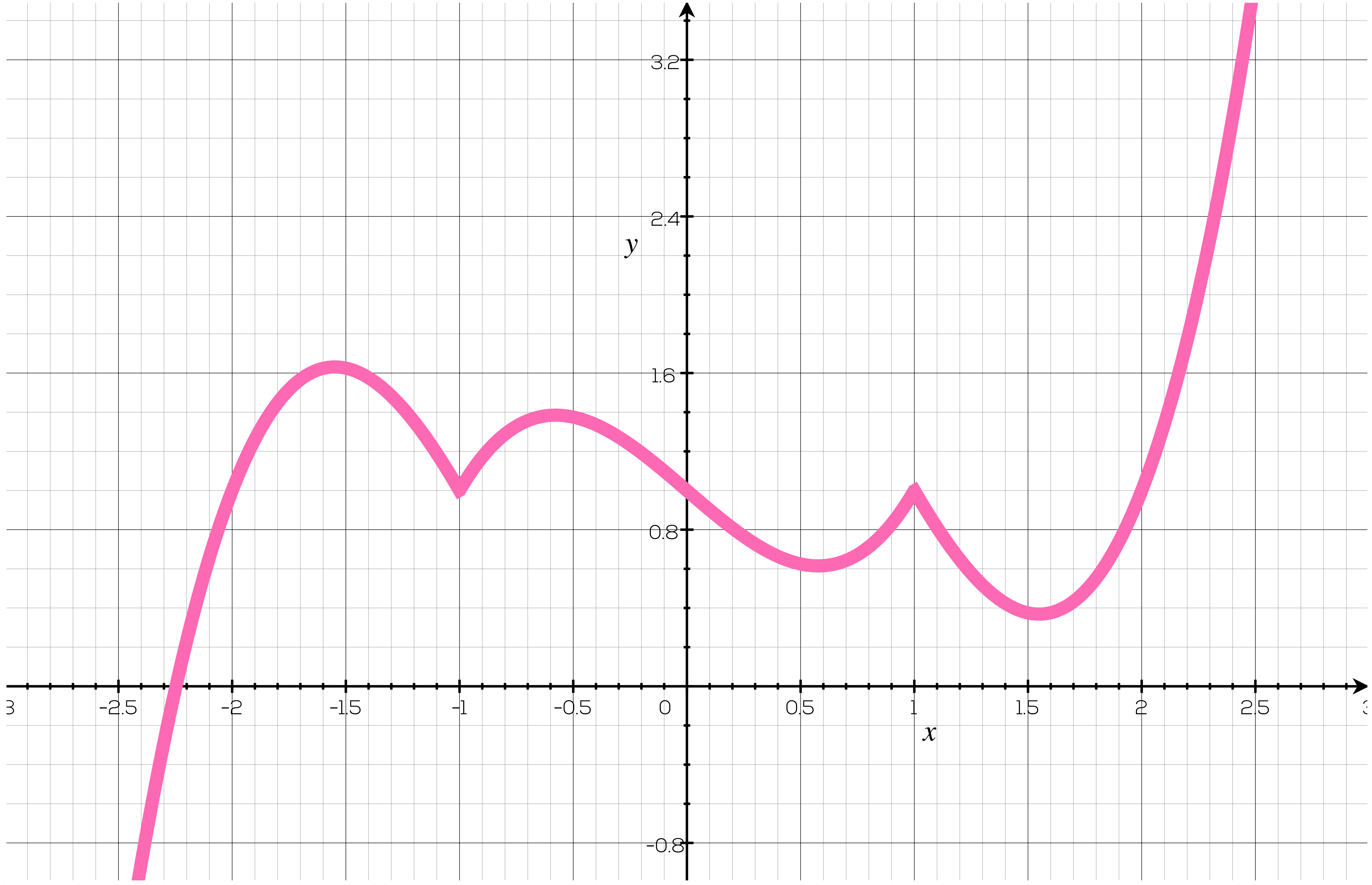
Chromosome

a.k.a.

Genotype, Organism, Creature, Member, Individual, Solution



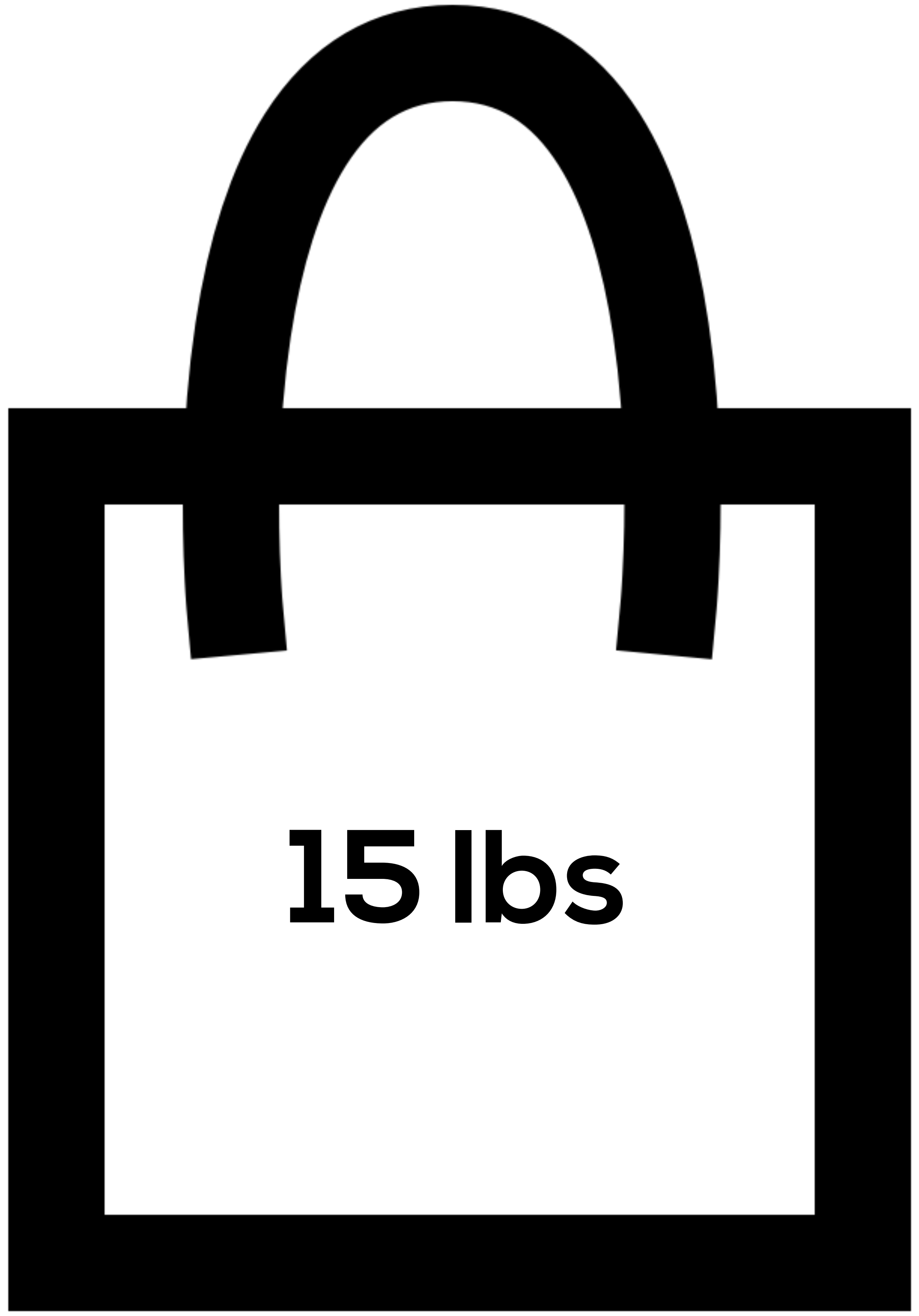
Terminology



Fitness Landscape

https://github.com/rawg/levis

0/1 Knapsack Problem



Given a set of items, each with a quantified weight and value ($), what combination of items will:

1. Fit inside a knapsack capable of carrying a fixed amount of weight?

2. Maximize the value of the knapsack's payload?

Encoding Schemes

Binary Encoding

1	0	0	1	0	1	1	0

Value Encoding

10	3	22	19	65	97	22	41

List Encoding

10	3	22	19

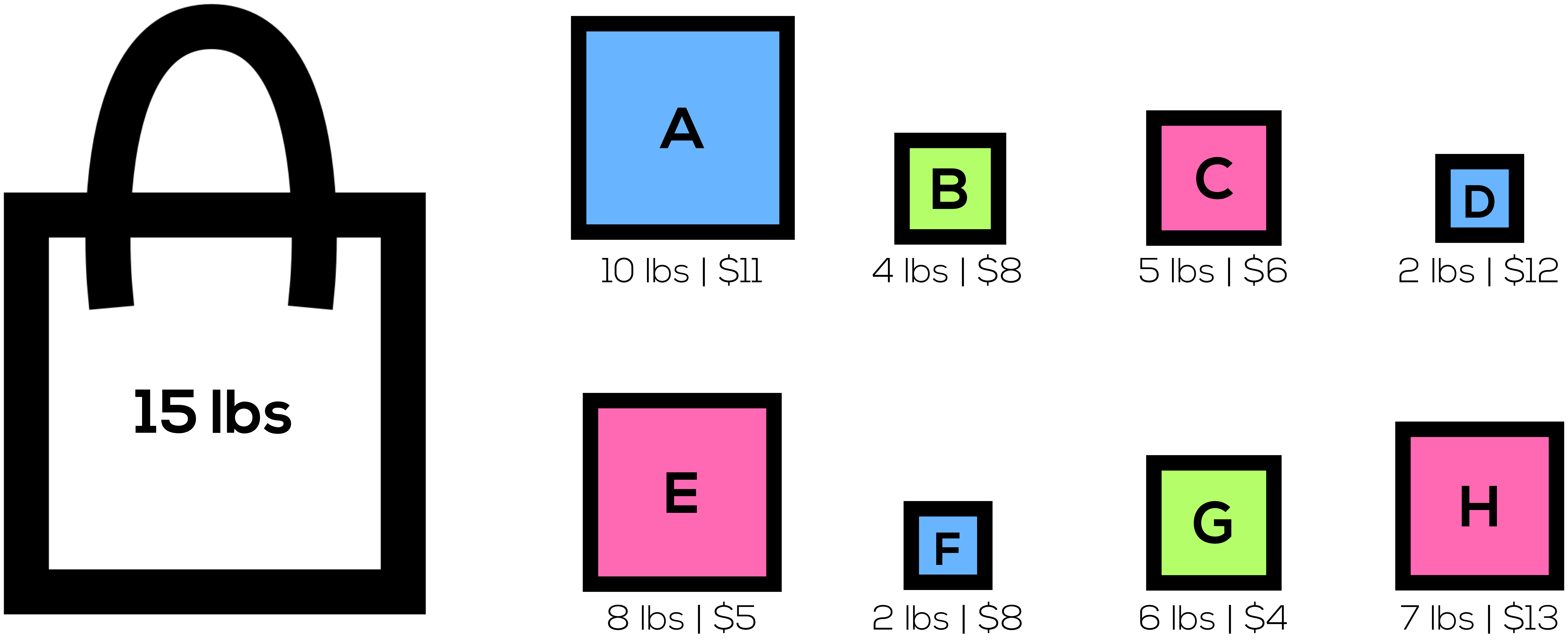
19	65	97	35	41	10

Permutation Encoding

A	B	C	D	E	F	G	H

G	C	F	A	B	D	E	H

0/1 Knapsack Problem



15 lbs

A
10 lbs | $11

B
4 lbs | $8

C
5 lbs | $6

D
2 lbs | $12

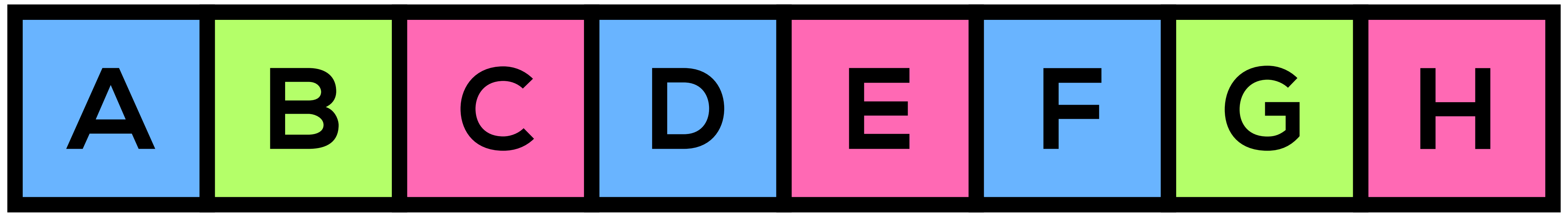
E
8 lbs | $5

F
2 lbs | $8

G
6 lbs | $4

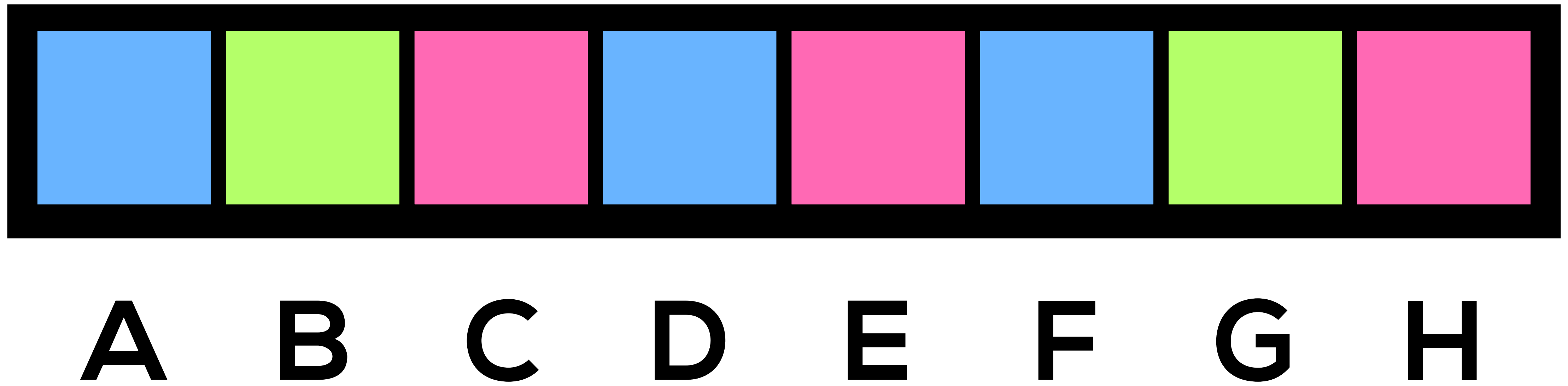
H
7 lbs | $13

Binary Encoding



A	B	C	D	E	F	G	H

Binary Encoding



Binary Encoding

Weight: 29 lbs

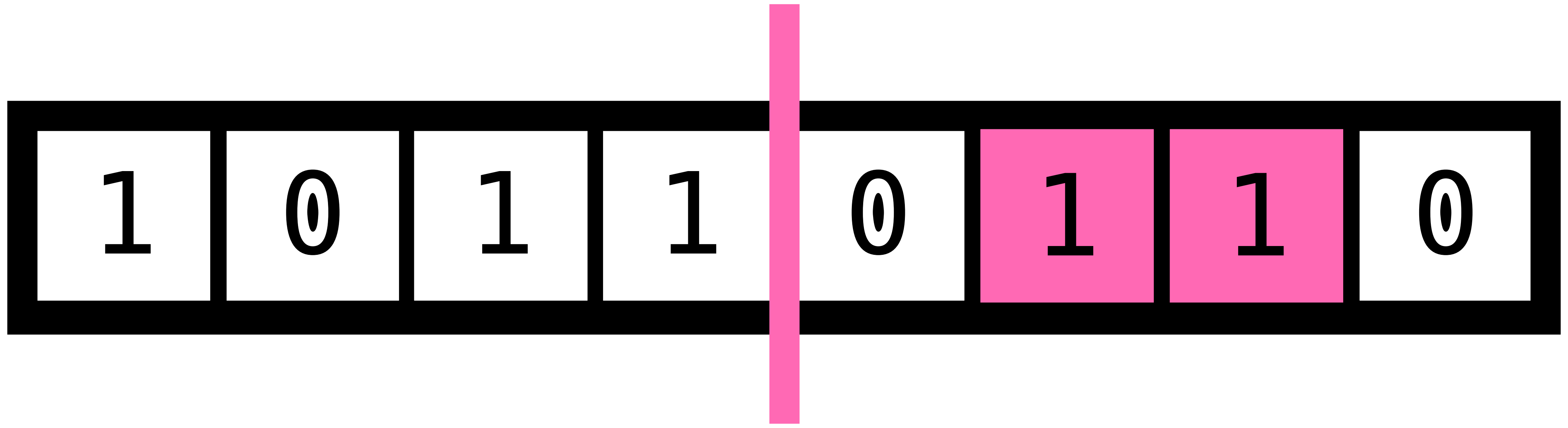
Value: $26

1	0	1	0	1	0	1	0
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
10 lbs $11	4 lbs $8	5 lbs $6	2 lbs $12	8 lbs $5	2 lbs $8	6 lbs $4	7 lbs $13

Initialize the Population

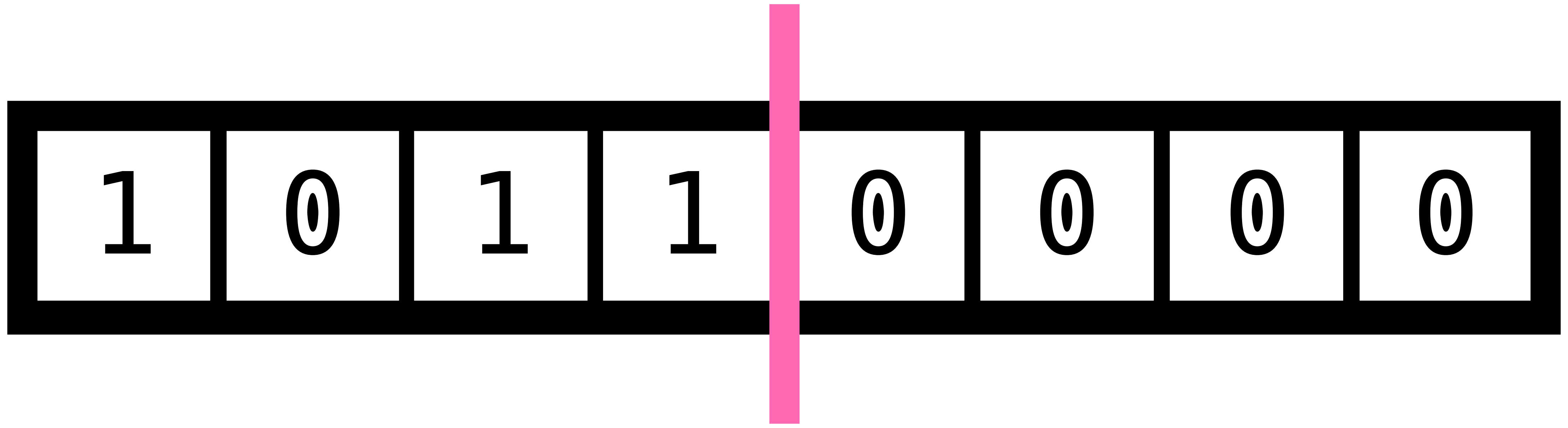
1	0	1	1	0	1	1	0

Initialize the Population



1	0	1	1	0	1	1	0

Initialize the Population



1	0	1	1	0	0	0	0

Initialize the Population

0	0	1	1	0	1	0	1

Fitness Function

- Maximize value
- Maximize weight to limit

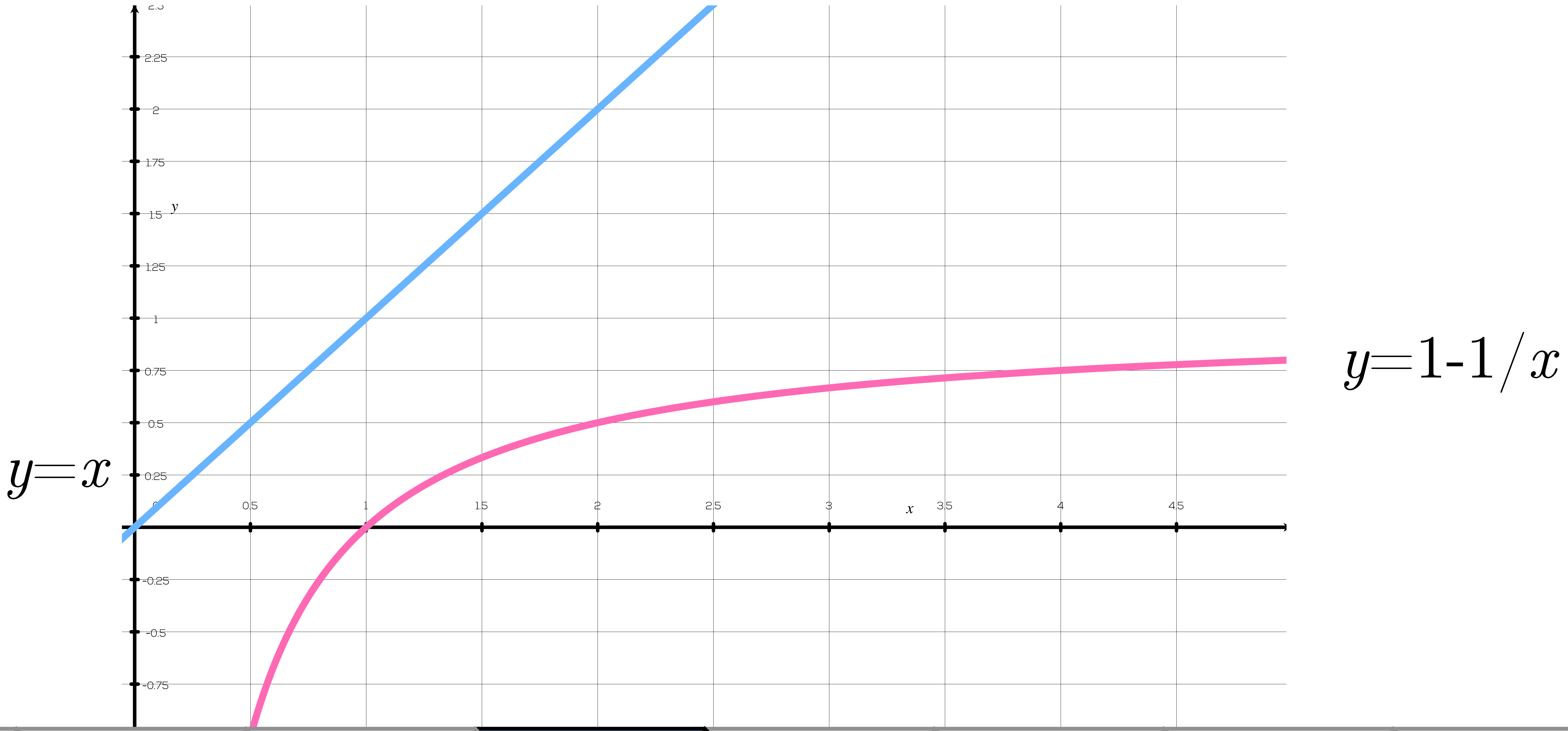
```
def score(self, chromosome):
  weight, value = self.assess(chromosome)

  if weight > self.max_weight:
    return 0.0

  if value <= 0:
    return 0.0

  wt = 1 / (1 + self.max_weight - weight)
  vl = 1 - 1 / value
  return wt + vl * self.value_bias
```

Fitness Function

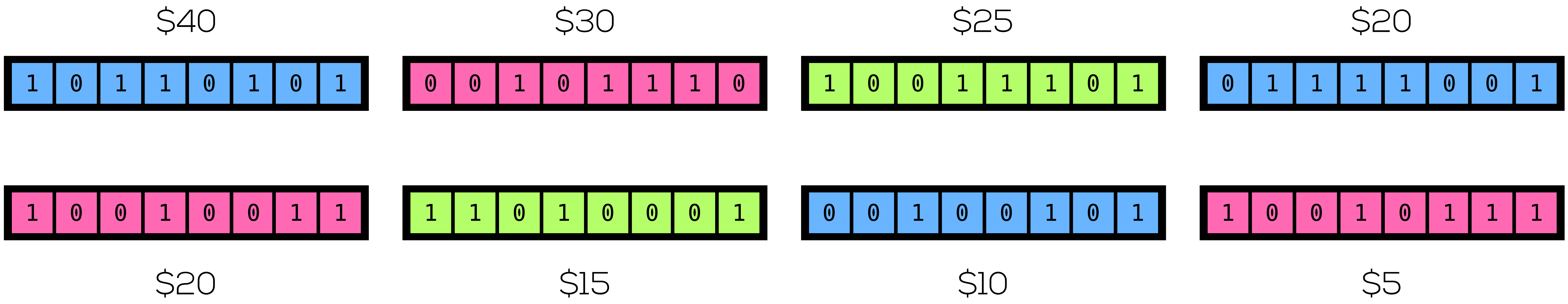
- Maximize value
- Don't exceed weight limit

```
def score(self, chromosome):
  weight, value = self.assess(chromosome)

  if weight > self.max_weight:
    return 0.0

  return 1 - 1 / value
```

Fitness Function



Proportionate Selection



$40

1 0 1 1 0 1 0 1

$30

0 0 1 0 1 1 1 0

$25

1 0 0 1 1 1 0 1

$20

0 1 1 1 1 0 0 1

1 0 0 1 0 0 1 1

$20

1 1 0 1 0 0 0 1

$15

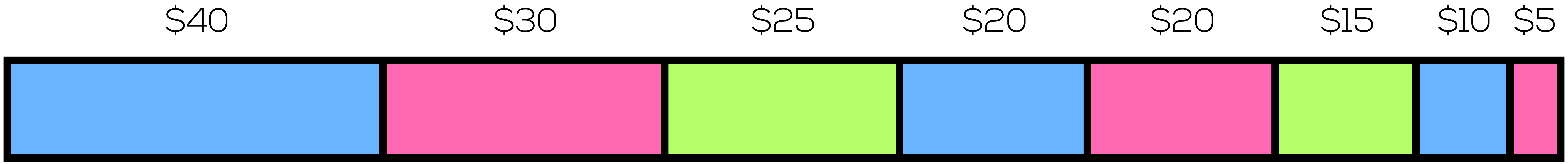
0 0 1 0 0 1 0 1

$10

1 0 0 1 0 1 1 1

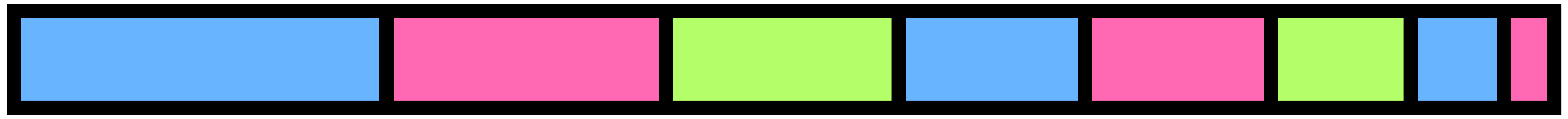
$5

Proportionate Selection

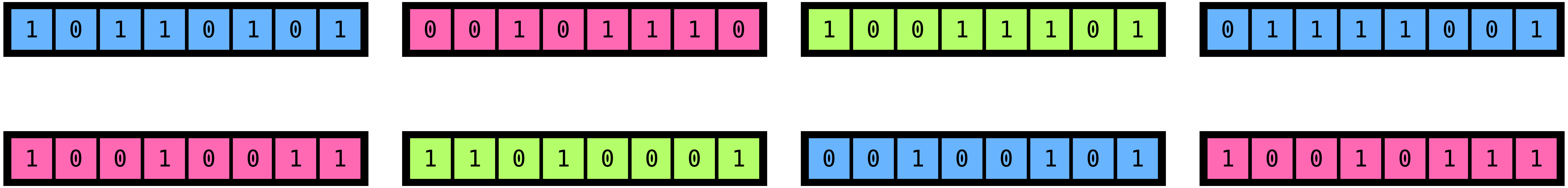


Proportionate Selection

- Most direct path to convergence
- Enables elitism



Tournament Selection



1 0 1 1 0 1 0 1

0 0 1 0 1 1 1 0

1 0 0 1 1 1 0 1

0 1 1 1 1 0 0 1

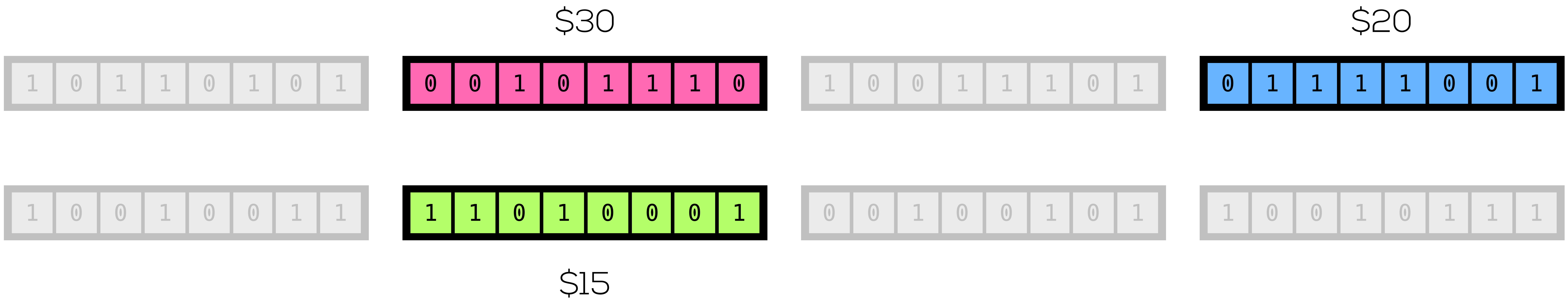
1 0 0 1 0 0 1 1

1 1 0 1 0 0 0 1

0 0 1 0 0 1 0 1

1 0 0 1 0 1 1 1

Tournament Selection



$30

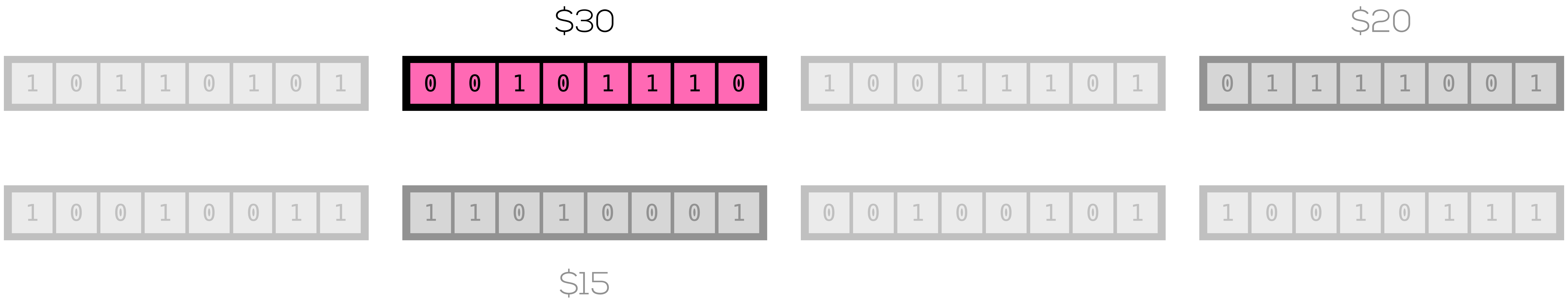
$20

1 0 1 1 0 1 0 1 | 0 0 1 0 1 1 1 0 | 1 0 0 1 1 1 0 1 | 0 1 1 1 1 0 0 1

1 0 0 1 0 0 1 1 | 1 1 0 1 0 0 0 1 | 0 0 1 0 0 1 0 1 | 1 0 0 1 0 1 1 1

$15

Tournament Selection



$30

$20

1 0 1 1 0 1 0 1

0 0 1 0 1 1 1 0

1 0 0 1 1 1 0 1

0 1 1 1 1 0 0 1

1 0 0 1 0 0 1 1

1 1 0 1 0 0 0 1

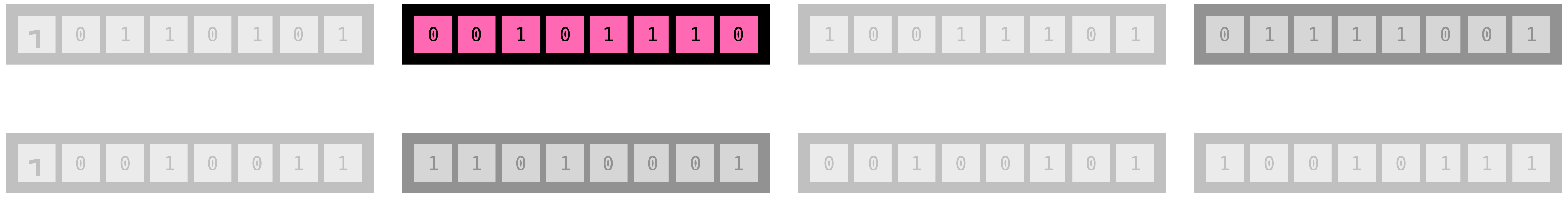
0 0 1 0 0 1 0 1

1 0 0 1 0 1 1 1

$15

Tournament Selection

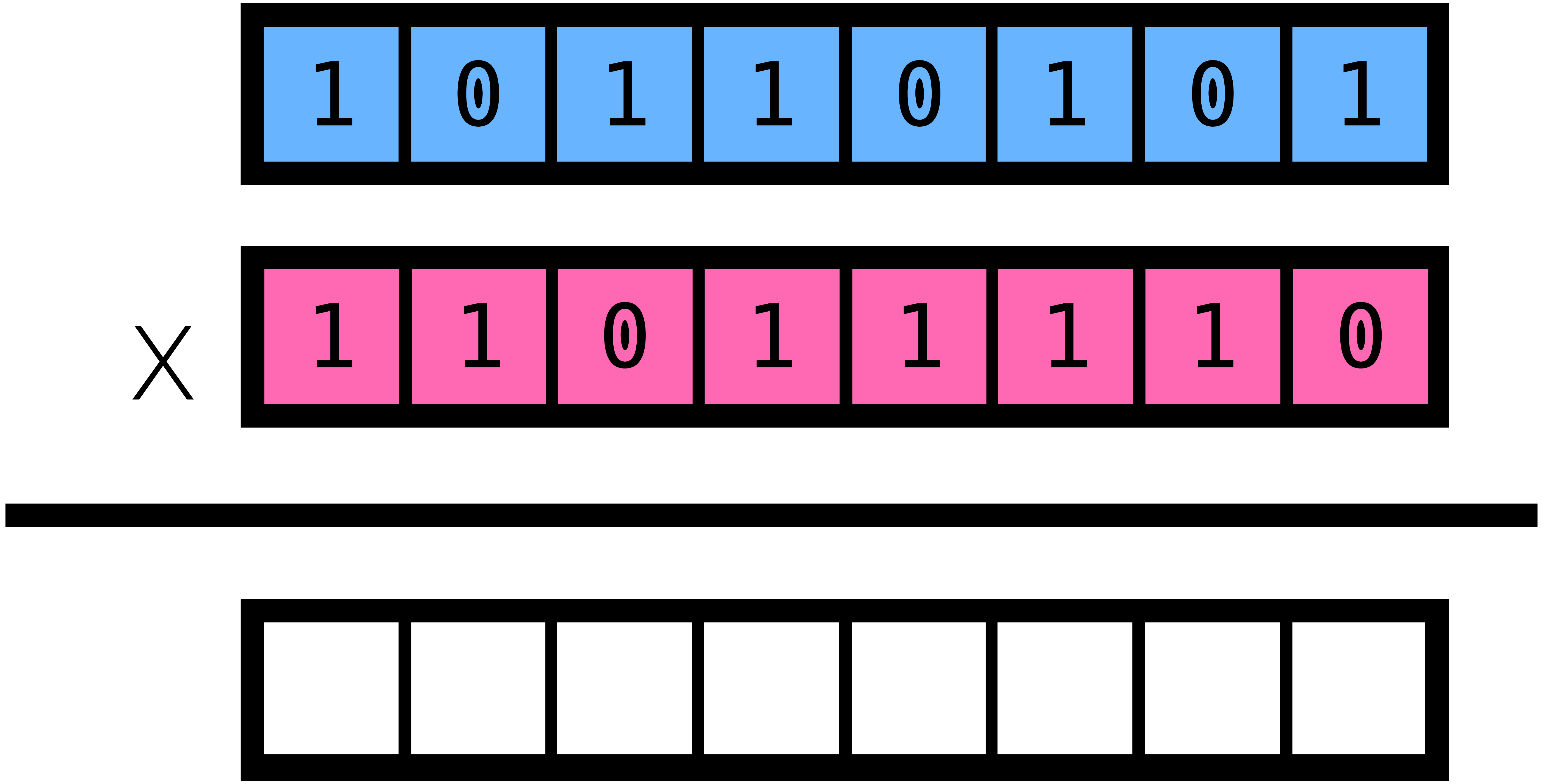
- Fewer fitness function invocations
- Built-in scaling



1 0 1 1 0 1 0 1 | 0 0 1 0 1 1 1 0 | 1 0 0 1 1 1 0 1 | 0 1 1 1 1 0 0 1

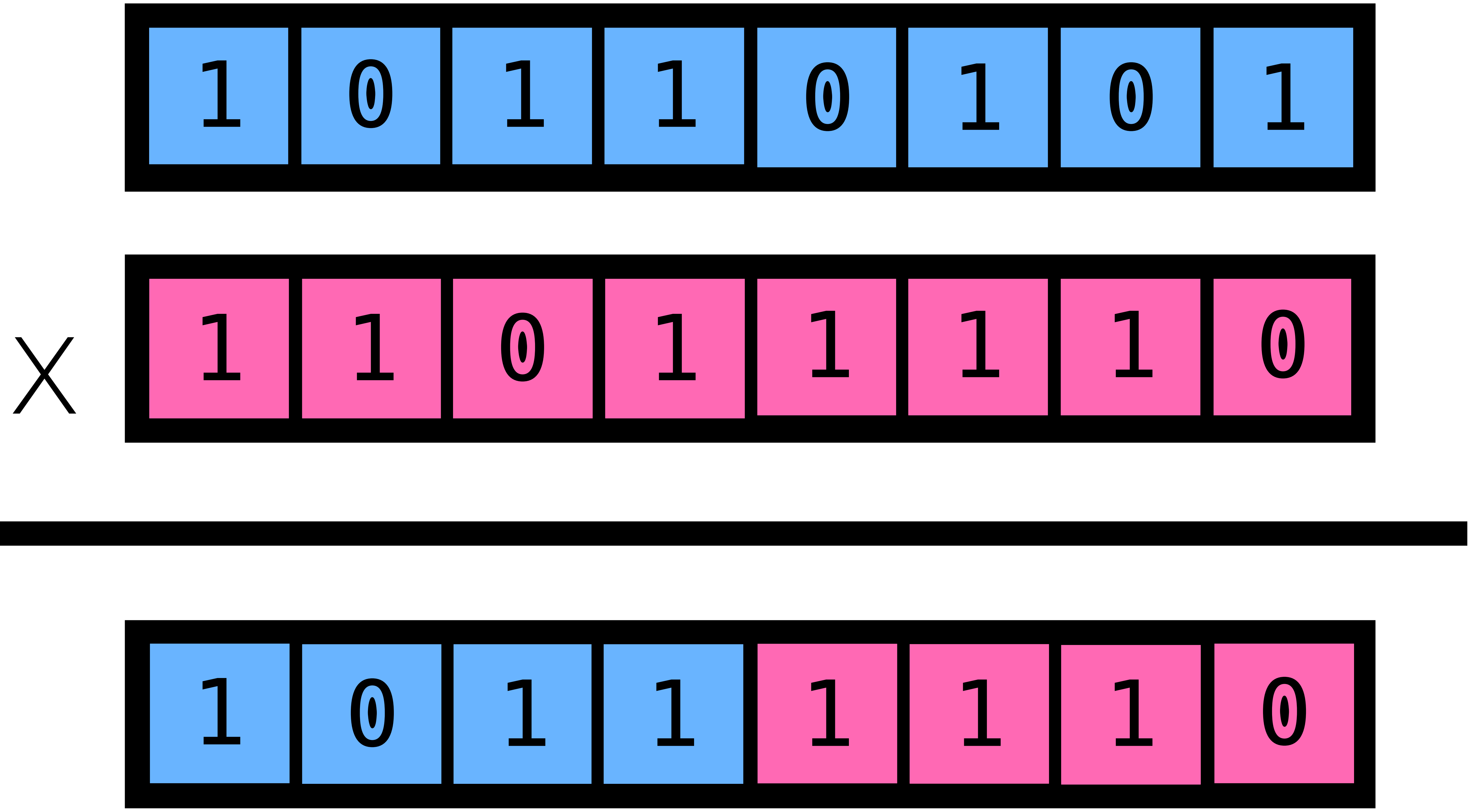
1 0 0 1 0 0 1 1 | 1 1 0 1 0 0 0 1 | 0 0 1 0 0 1 0 1 | 1 0 0 1 0 1 1 1

Single-Point Crossover



	1	0	1	1	0	1	0	1
X	1	1	0	1	1	1	1	0

Single-Point Crossover



1 0 1 1 0 1 0 1

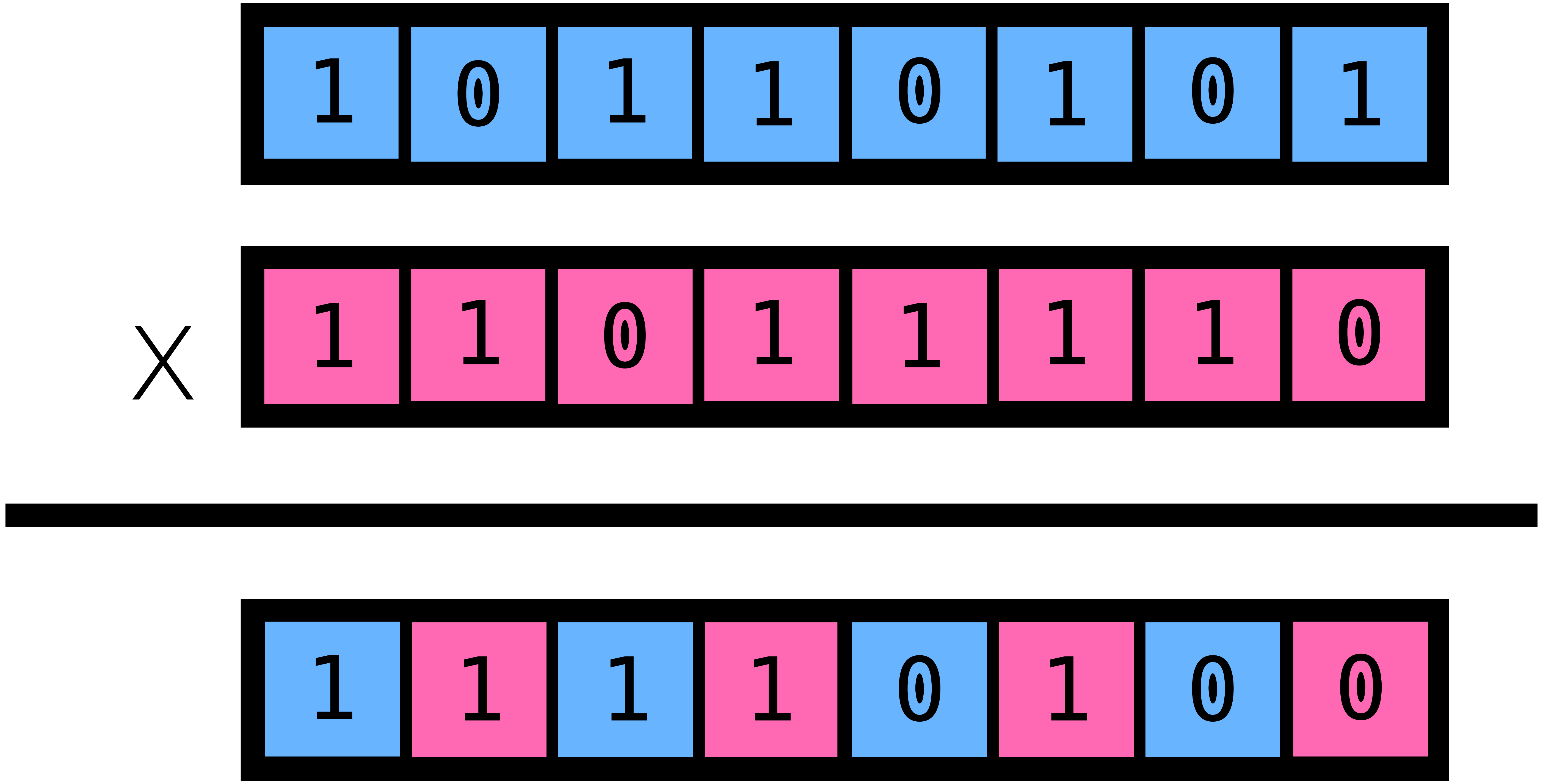
X 1 1 0 1 1 1 1 0

1 0 1 1 1 1 1 0

Uniform Crossover

	1	0	1	1	0	1	0	1
X	1	1	0	1	1	1	1	0

Uniform Crossover



1	0	1	1	0	1	0	1

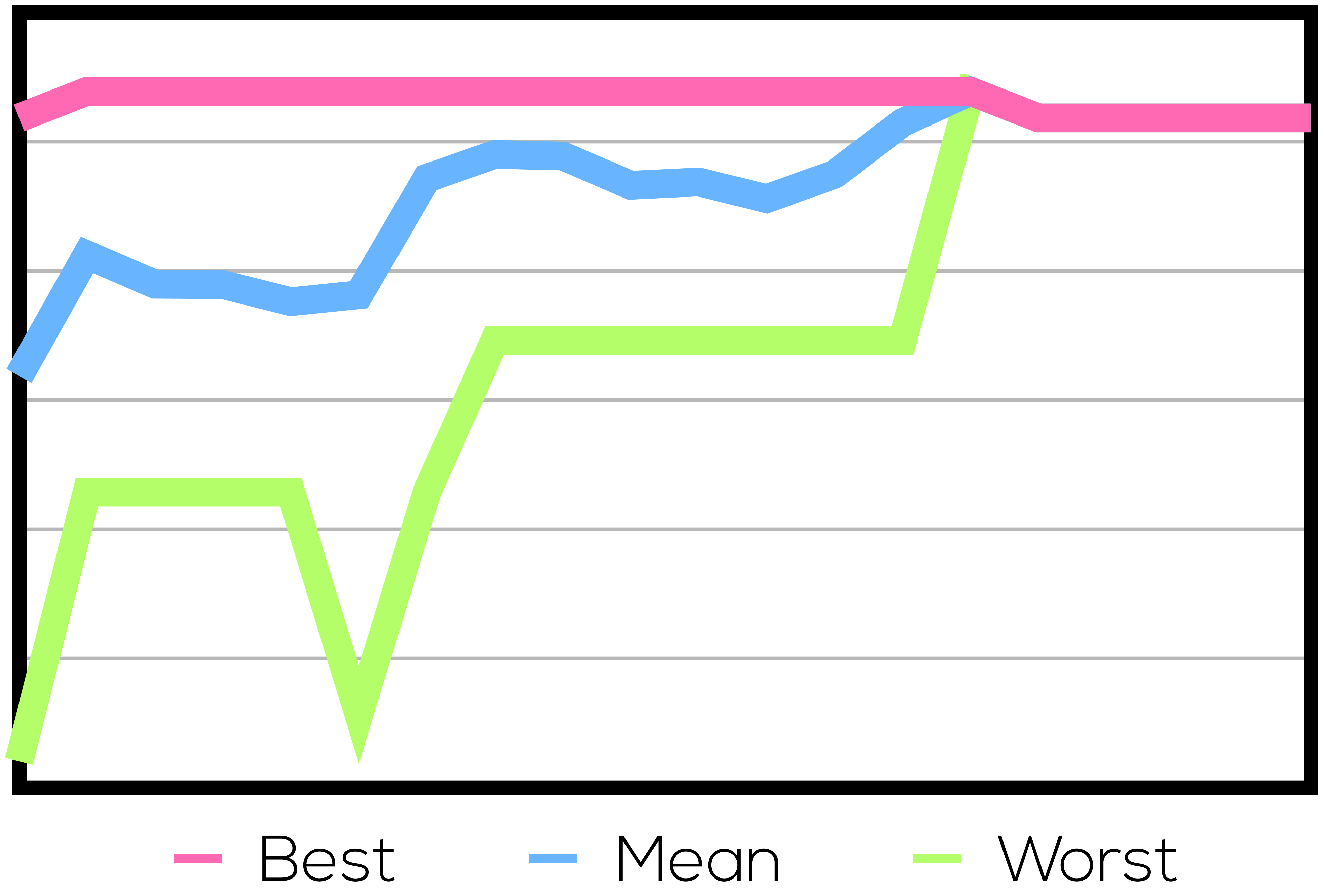
X

1	1	0	1	1	1	1	0

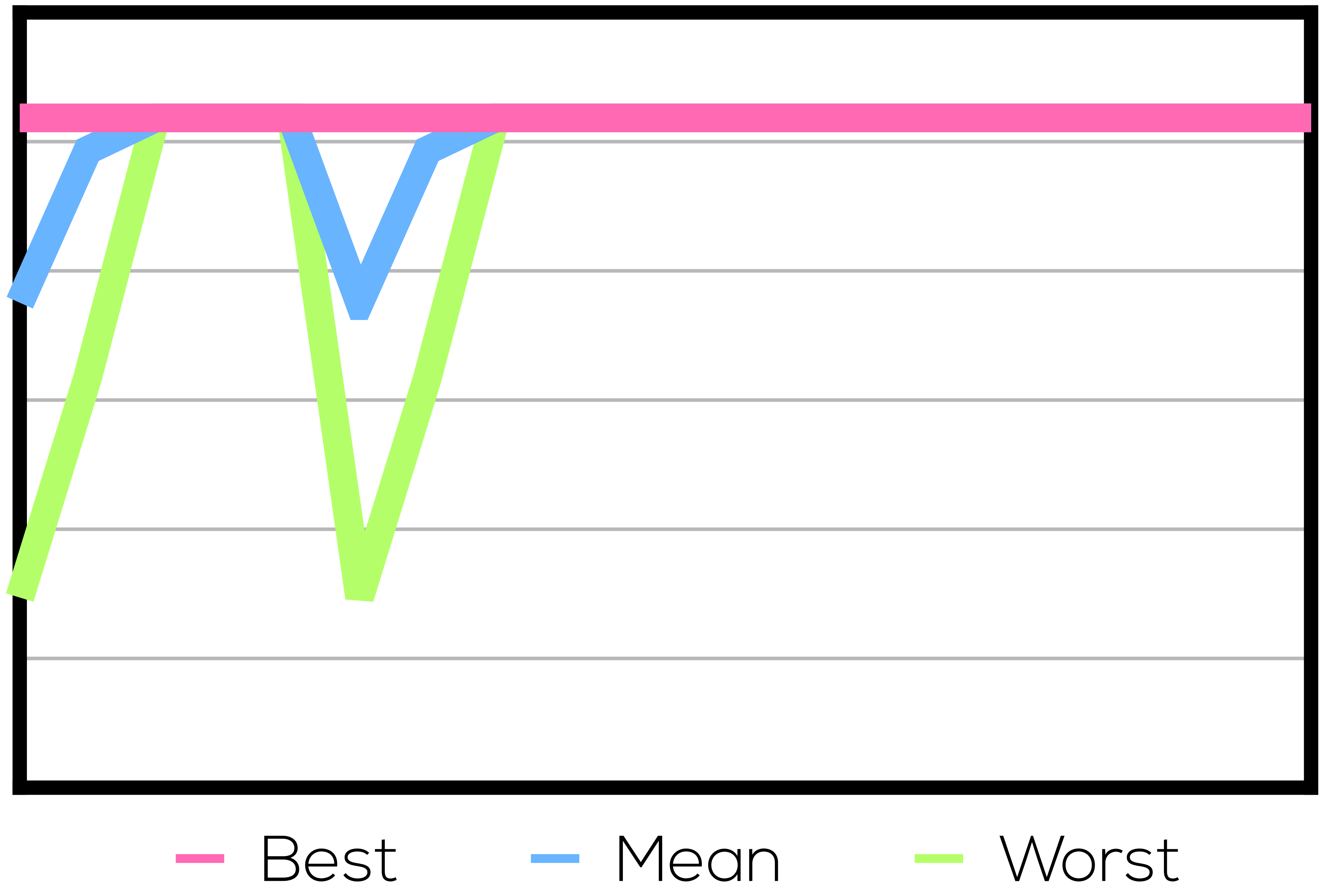
1	1	1	1	0	1	0	0

Comparison

Single Point Crossover



Uniform Crossover

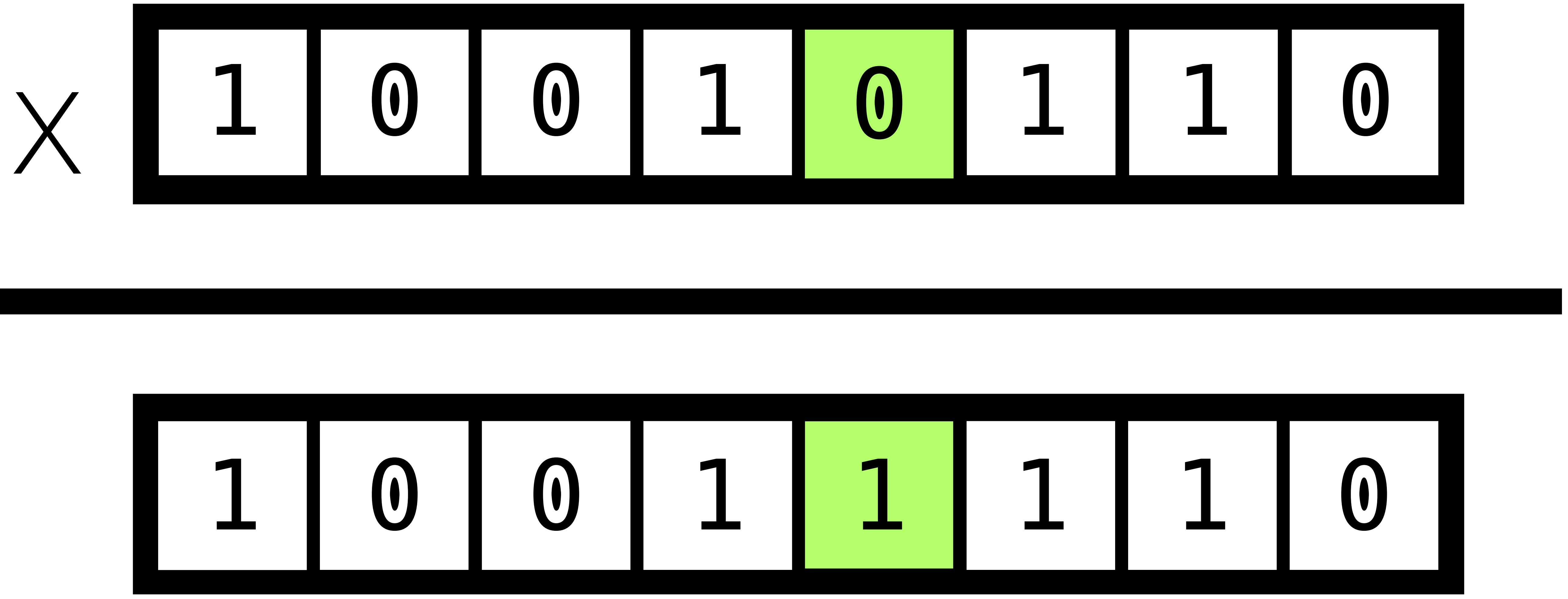


```
num_items=64, population_size=10, iterations=15, elitism=0, seed="Knapsack01"
```

Measuring Performance

- Solution fitness
- Clock time to converge
- Iterations to converge
- Number of solutions explored
- Inheritability

Point Mutation



X	1	0	0	1	0	1	1	0
	1	0	0	1	1	1	1	0

Solution

0	0	0	0	1	0	0	1

1	0	1	0	0	0	0	0

0	0	1	1	1	0	0	0

0	0	1	1	0	1	1	0

Solution

Weight: 15 lbs
Value: $30

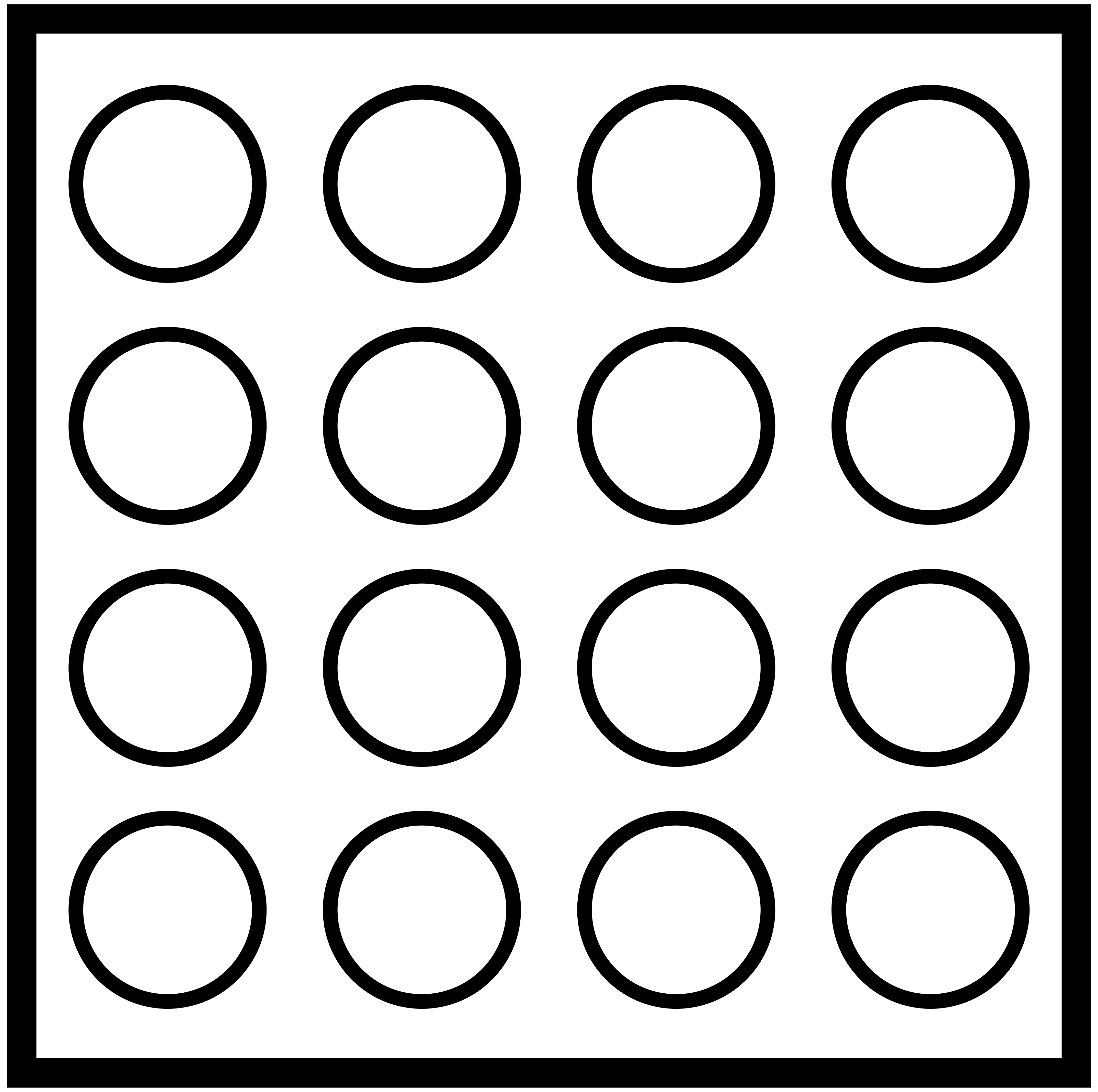
0	0	1	1	0	1	1	0
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
10 lbs	4 lbs	5 lbs	2 lbs	8 lbs	2 lbs	6 lbs	7 lbs
$11	$8	$6	$12	$5	$8	$4	$13

Seating Chart Problem

What seating arrangement will:

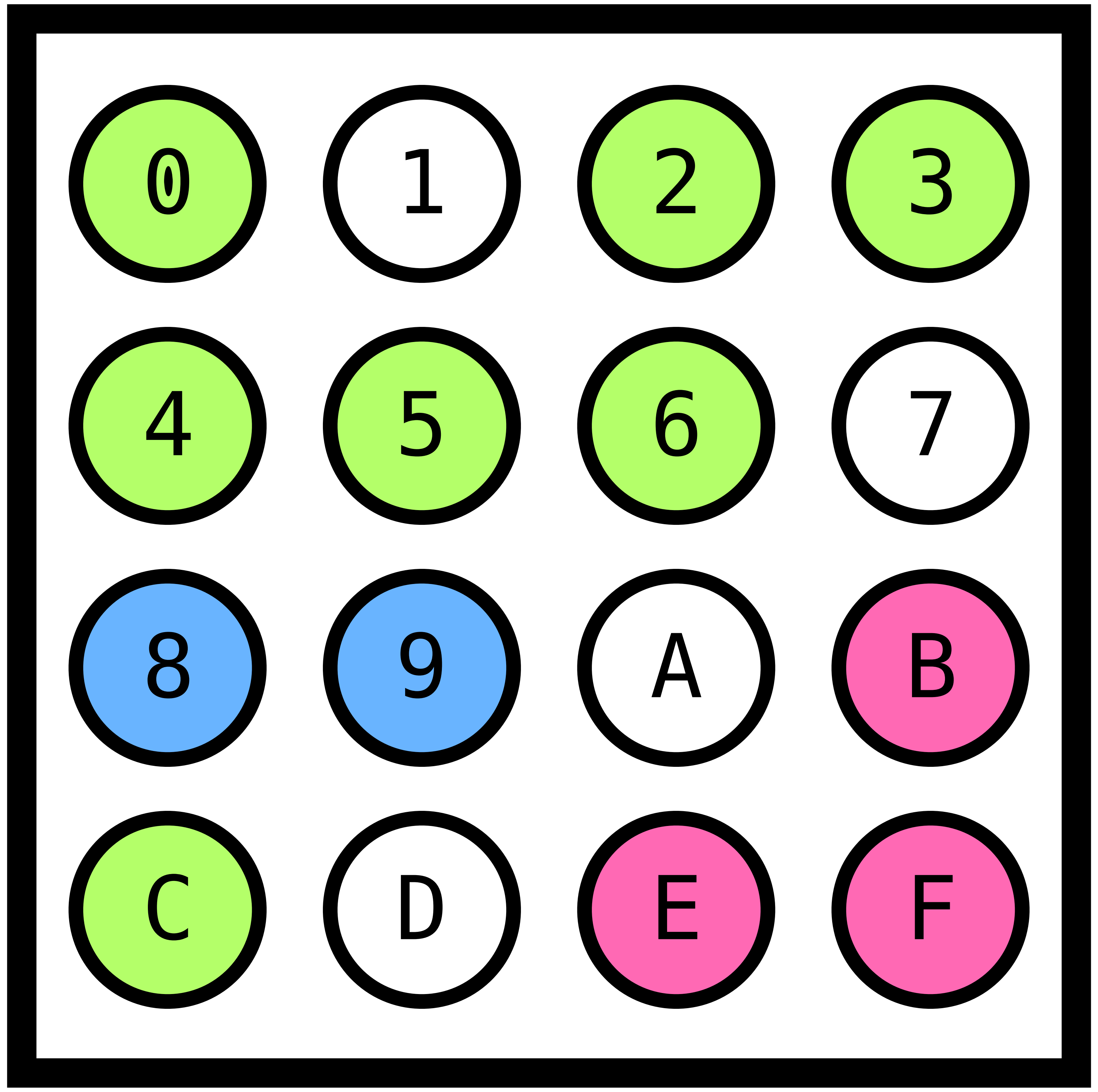
1. Seat people of the same job together?

2. Separate loud jobs from quiet jobs?



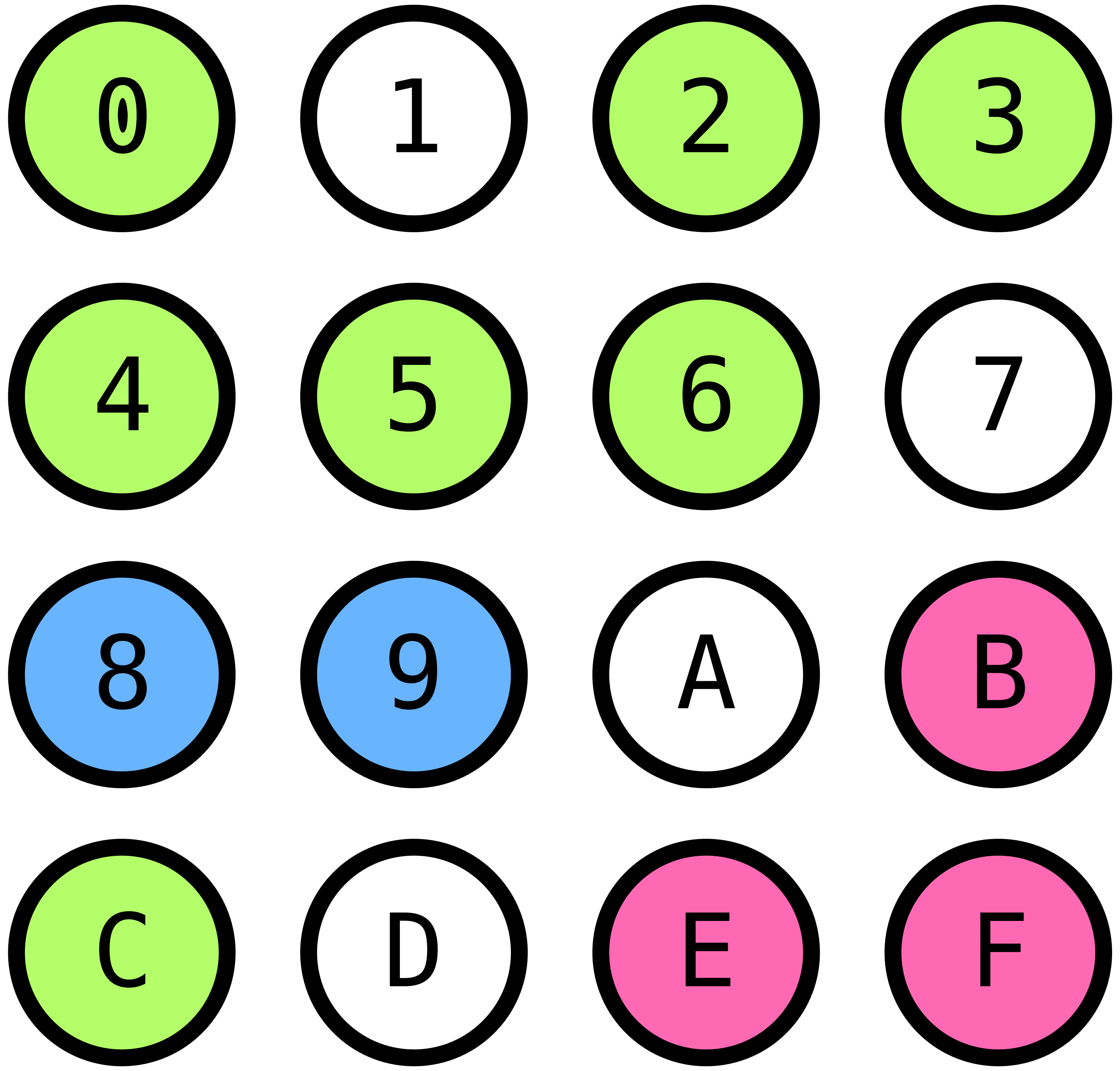
Seating Chart Problem

ID	Name	Role	Seat
0	Sandra Rodgers	Engineer	0
1	Kirk Spence	Engineer	3
2	Terry Burton	Engineer	2
3	Cindy Hill	Engineer	4
4	Steve Thompson	Engineer	6
5	Laura Juel	Engineer	5
6	Emily Cook	Engineer	C
7	Daniel Barrett	Manager	8
8	Cedric White	Manager	9
9	Emily Grimaud	Sales	E
10	Joseph Johnson	Sales	B
11	Raymond Seery	Sales	F

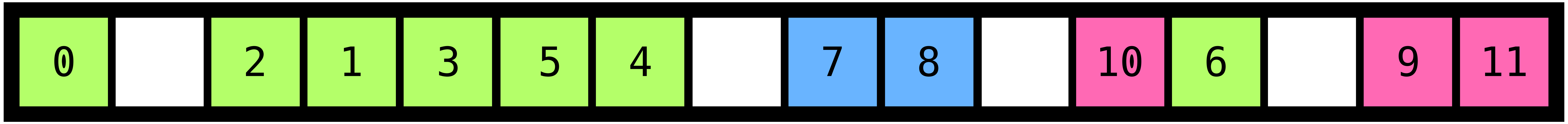


Permutation Encoding

ID	Name	Role	Seat
0	Sandra Rodgers	Engineer	0
1	Kirk Spence	Engineer	3
2	Terry Burton	Engineer	2
3	Cindy Hill	Engineer	4
4	Steve Thompson	Engineer	6
5	Laura Juel	Engineer	5
6	Emily Cook	Engineer	C
7	Daniel Barrett	Manager	8
8	Cedric White	Manager	9
9	Emily Grimaud	Sales	E
10	Joseph Johnson	Sales	B
11	Raymond Seery	Sales	F

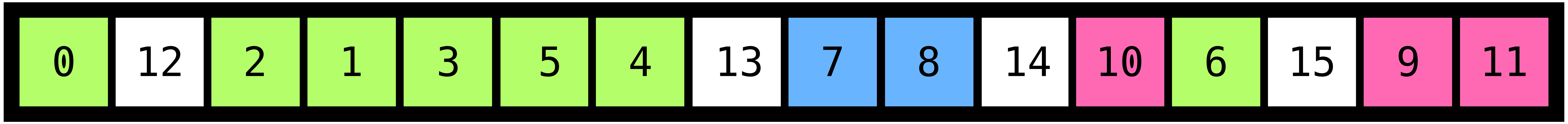


Permutation Encoding



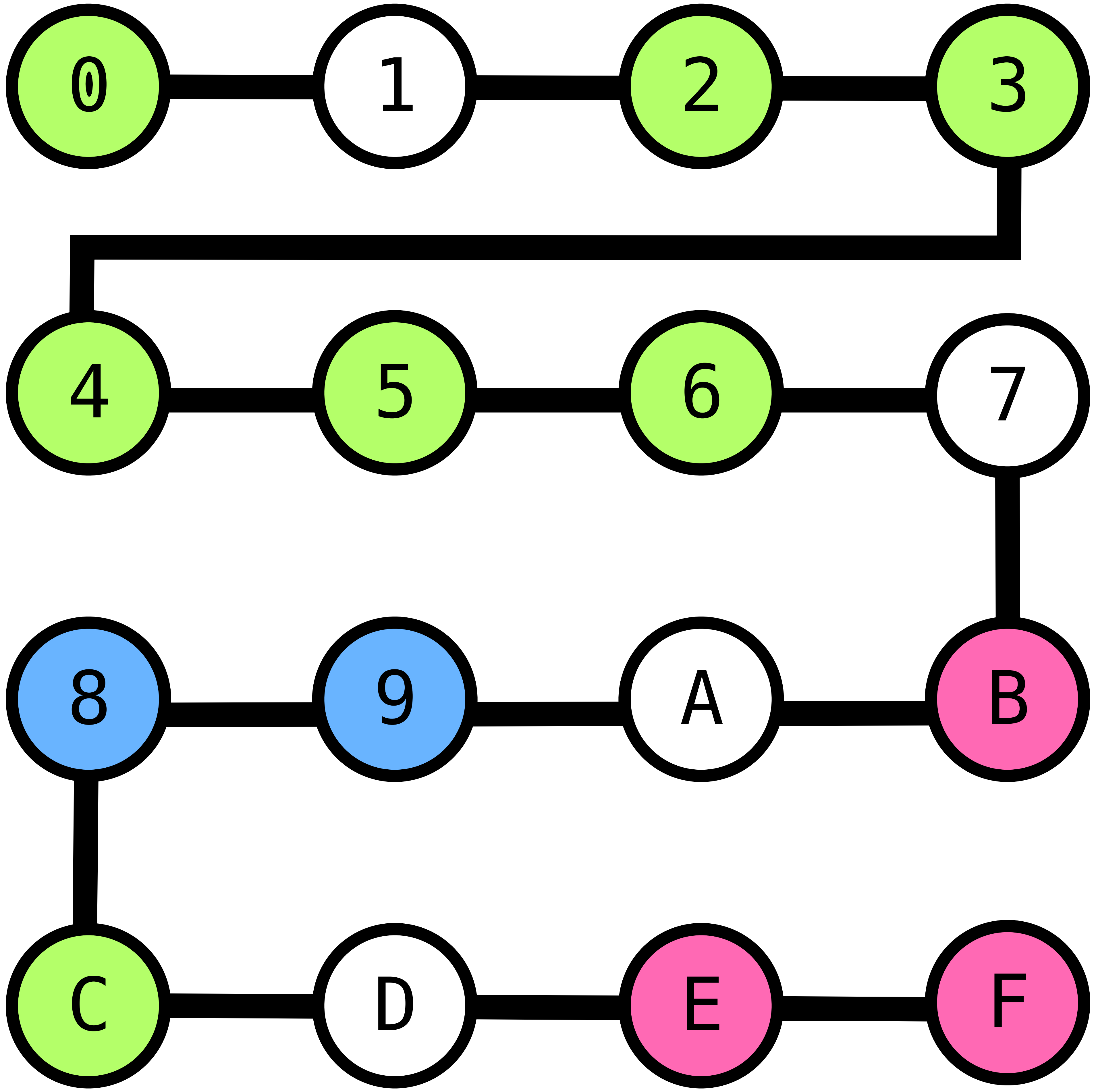
0		2	1	3	5	4		7	8		10	6		9	11

Permutation Encoding

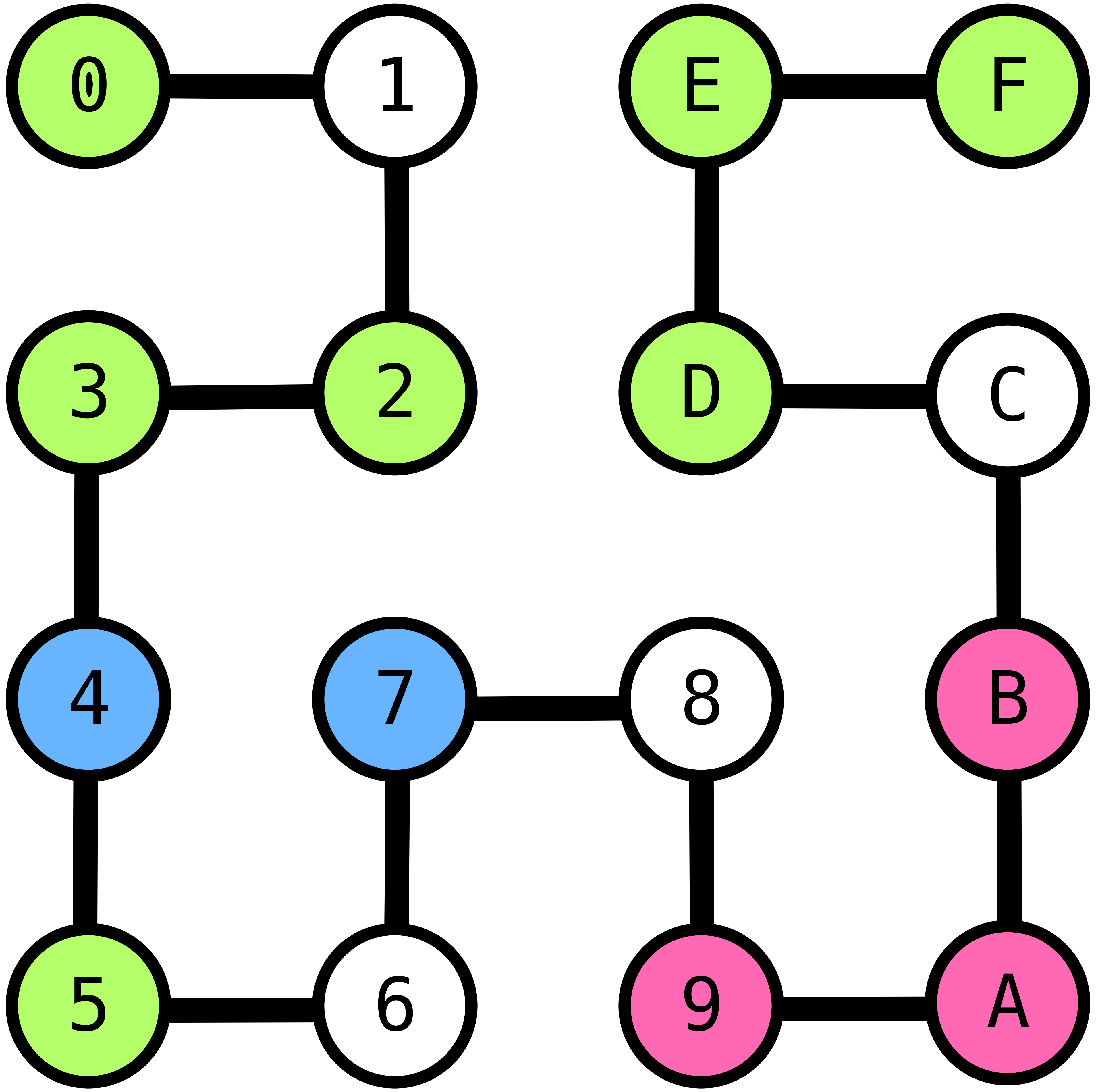


0	12	2	1	3	5	4	13	7	8	14	10	6	15	9	11

Naïve Vectorization

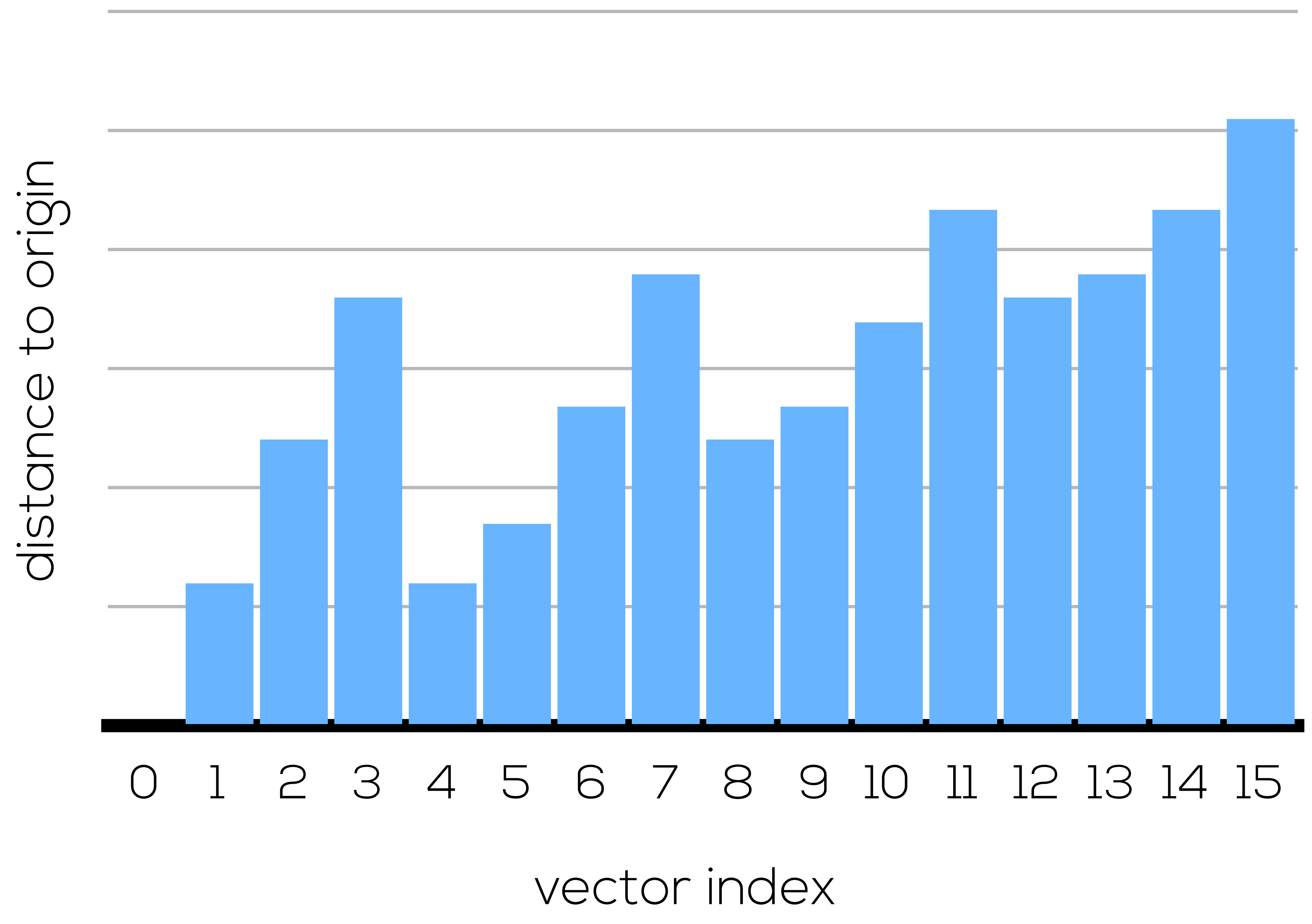


Hilbert Vectorization

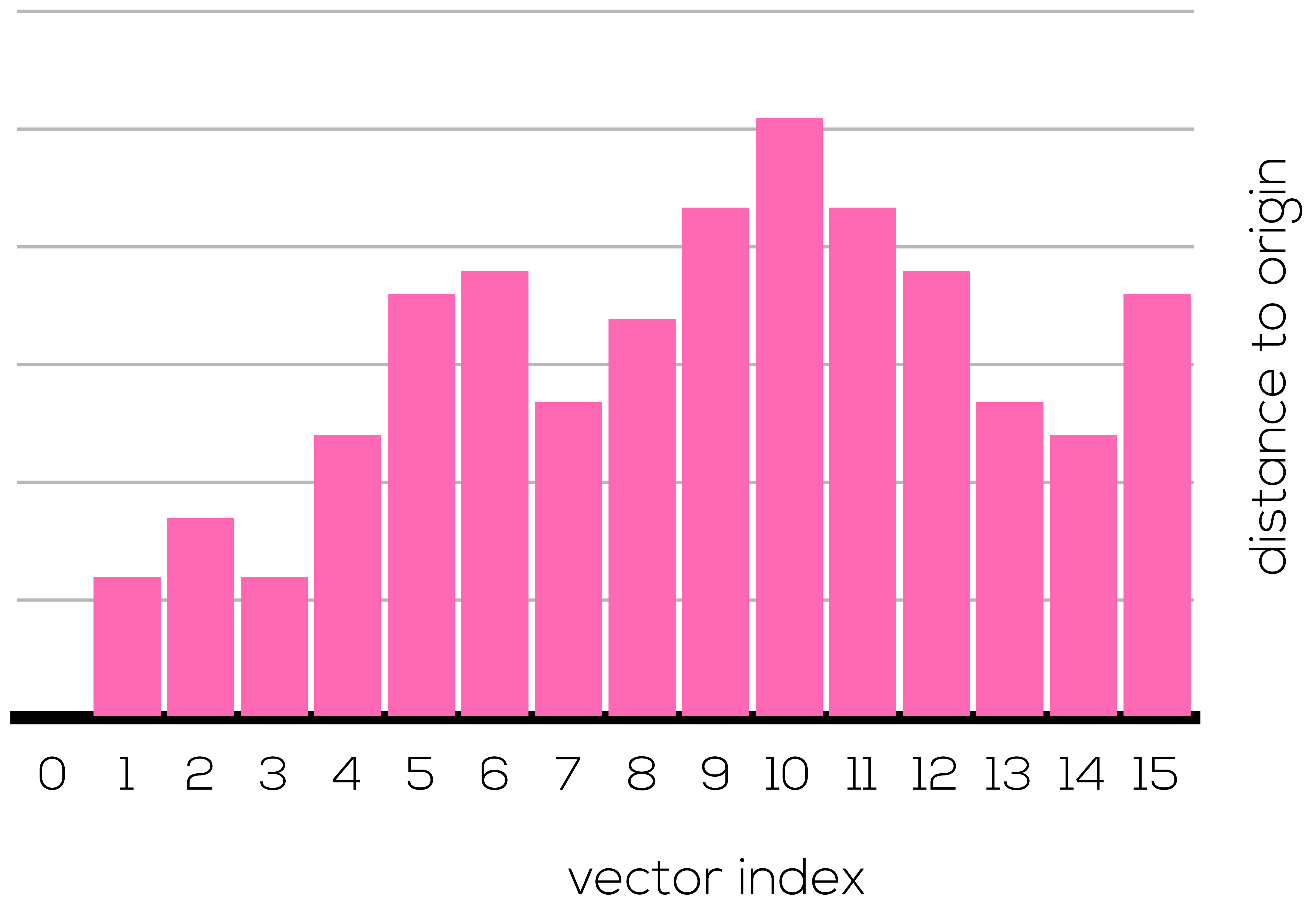


Preserving Locality

Naïve Vectorization

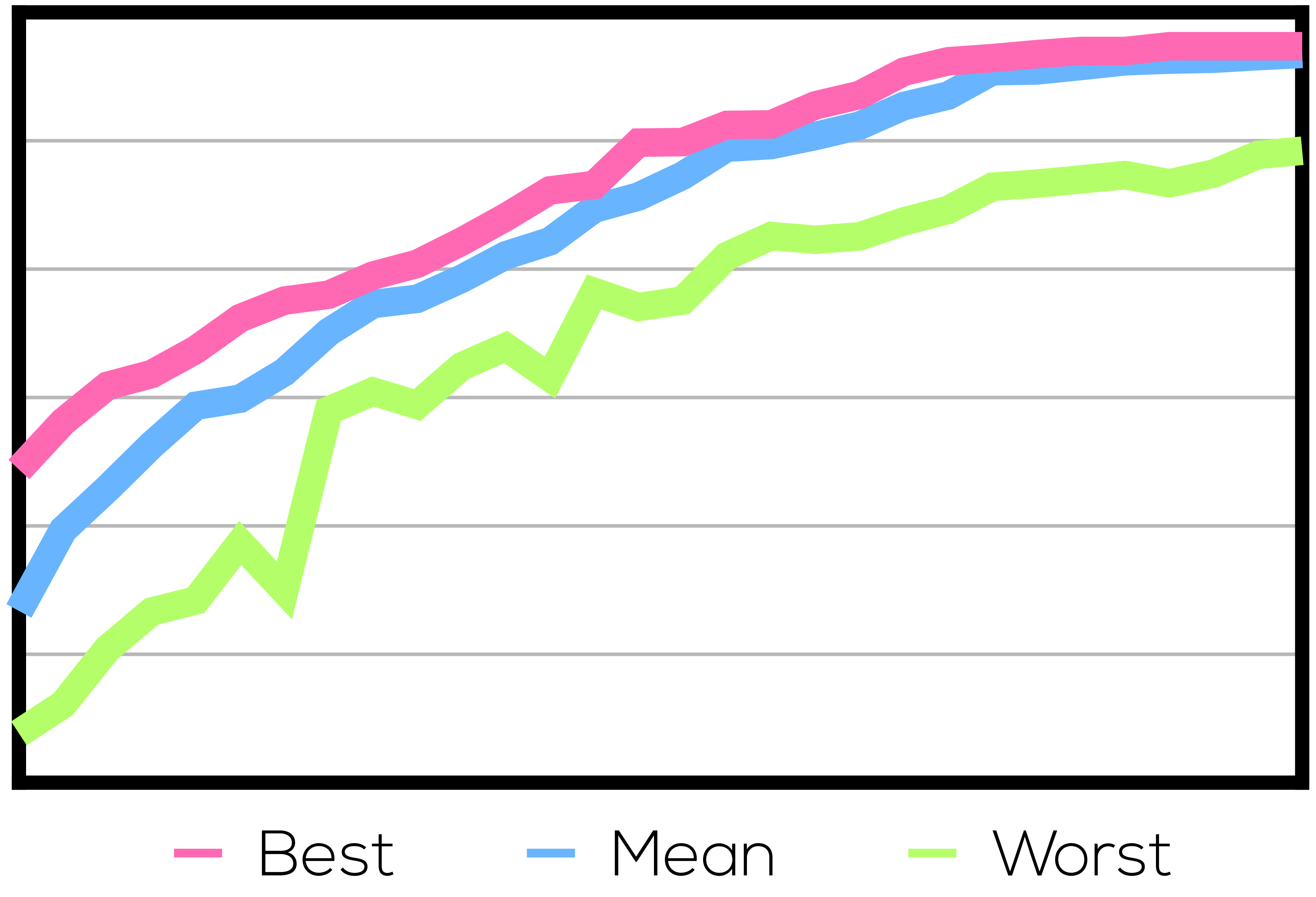


Hilbert Vectorization

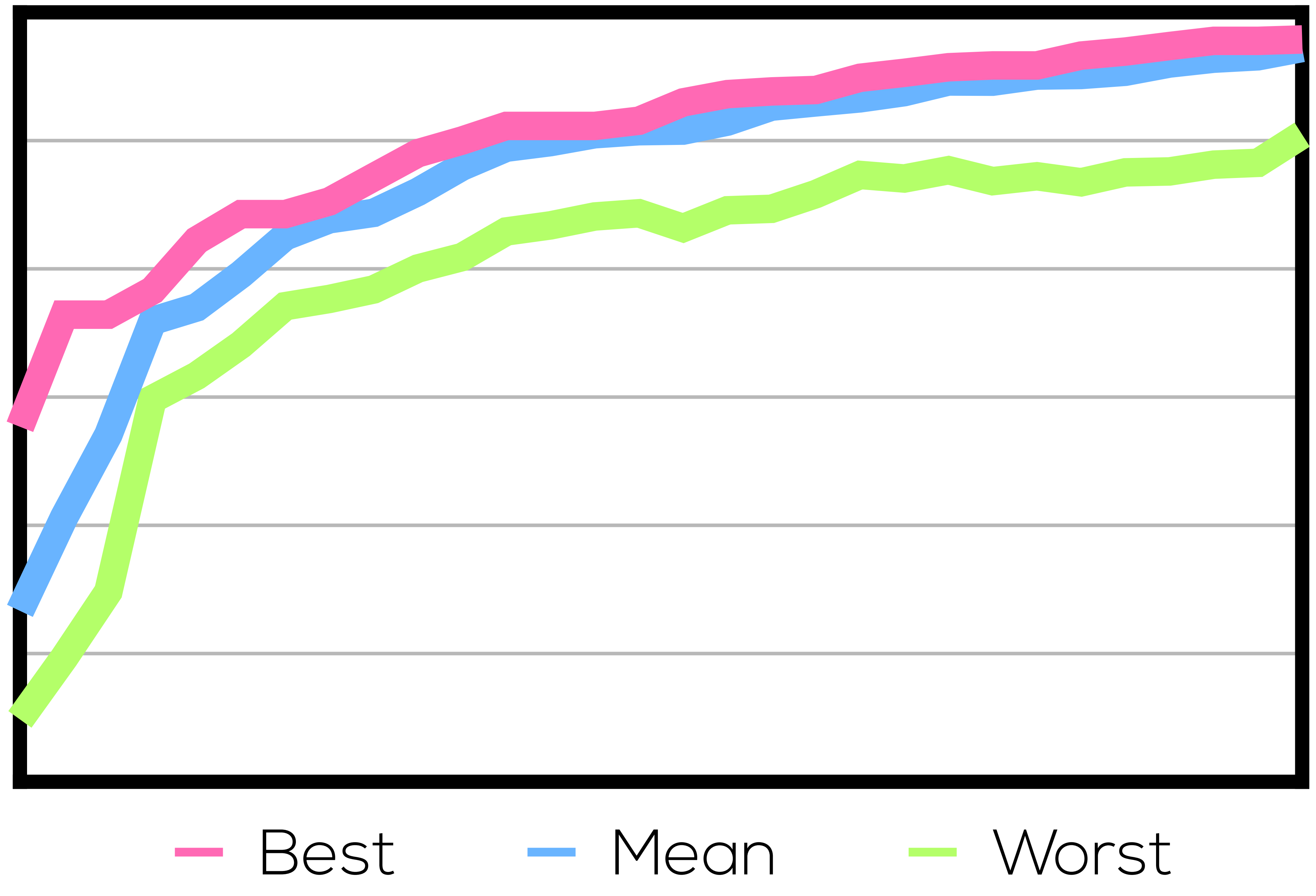


Performance Comparison

Naïve Vectorization

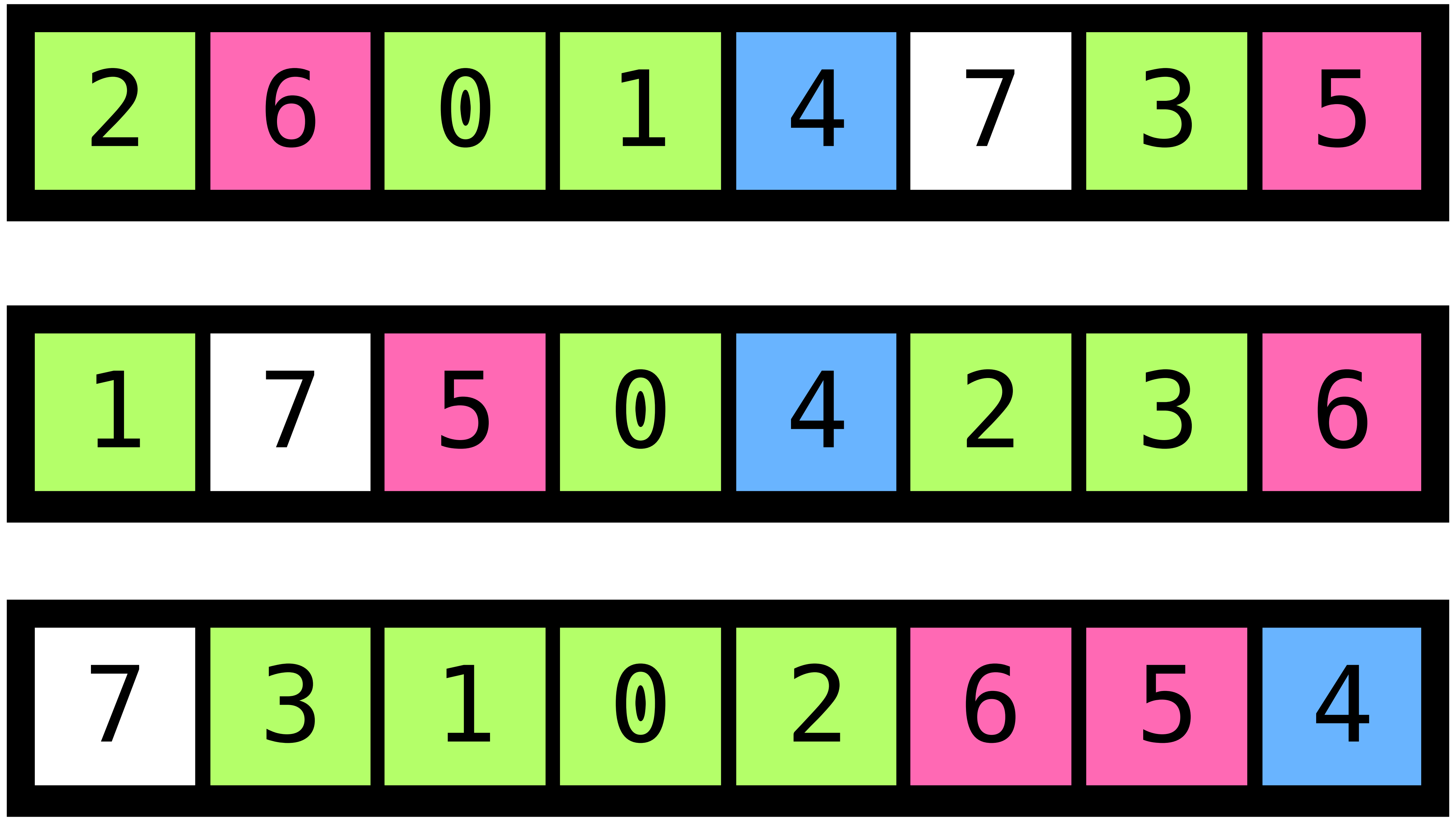


Hilbert Vectorization



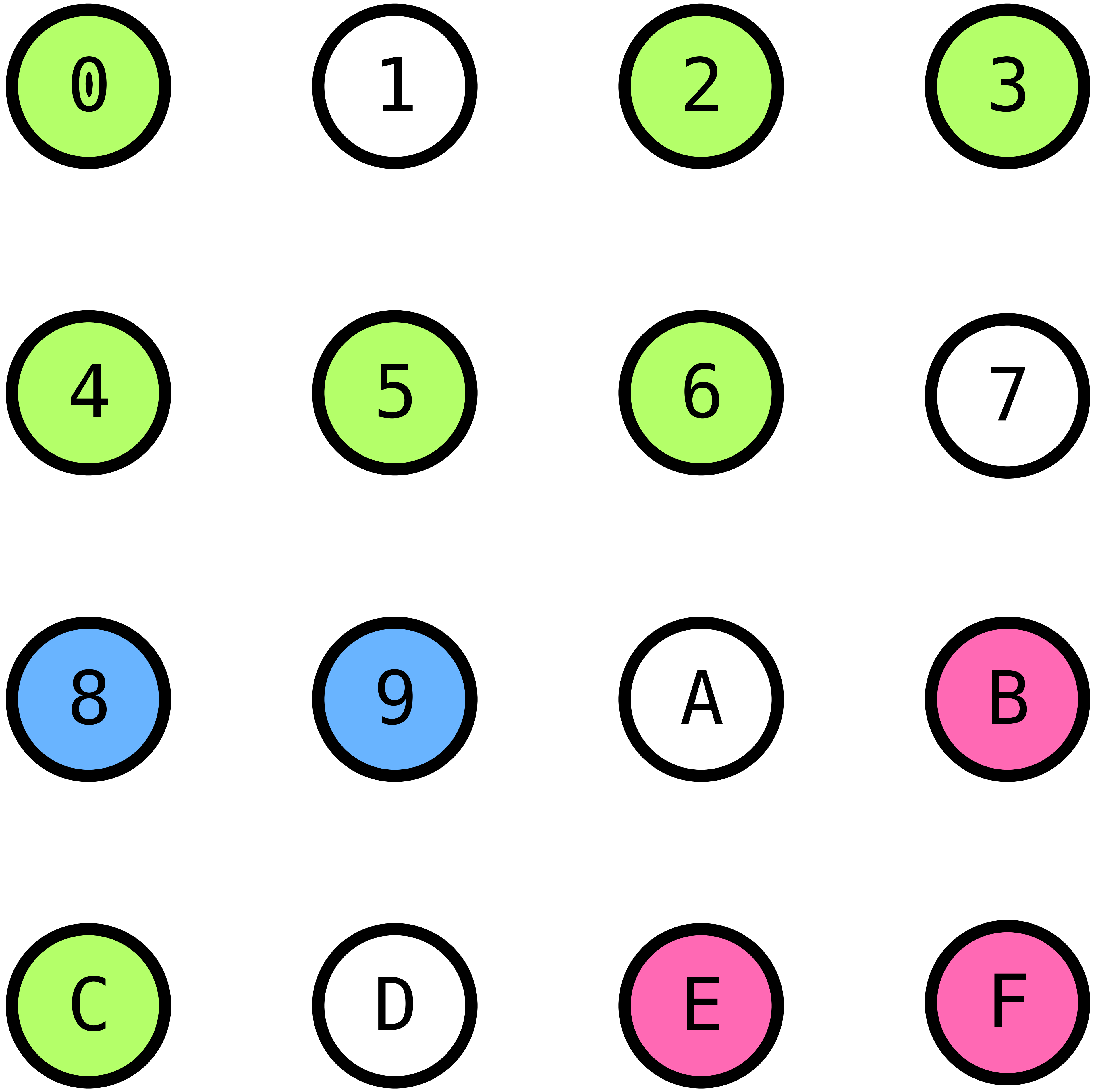
```
map_size=7x7, population_size=200, iterations=30, elitism=0, seed="SeatingChart", crossover=PMX
```

Initialize the Population



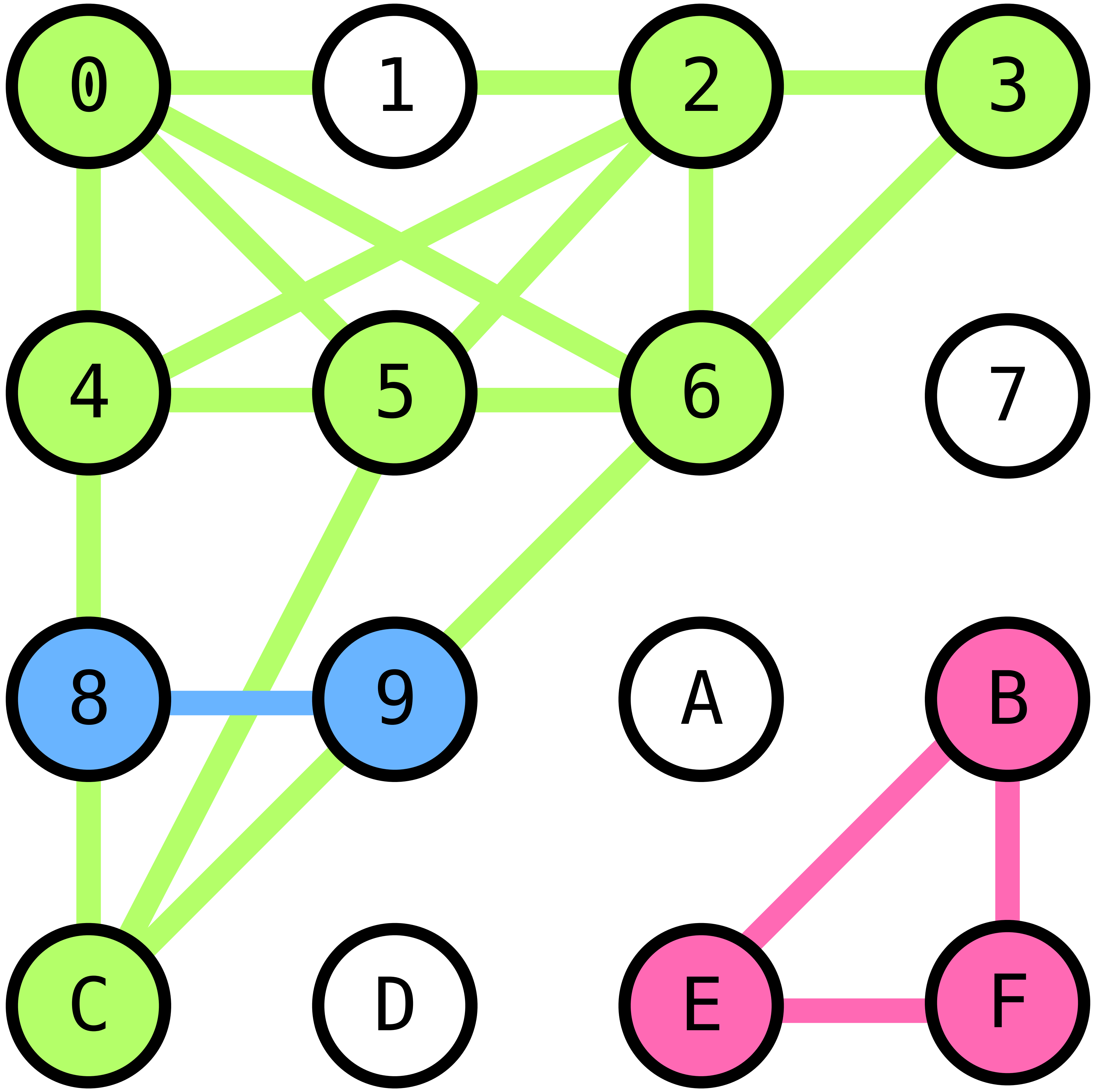
2	6	0	1	4	7	3	5
1	7	5	0	4	2	3	6
7	3	1	0	2	6	5	4

Fitness Function



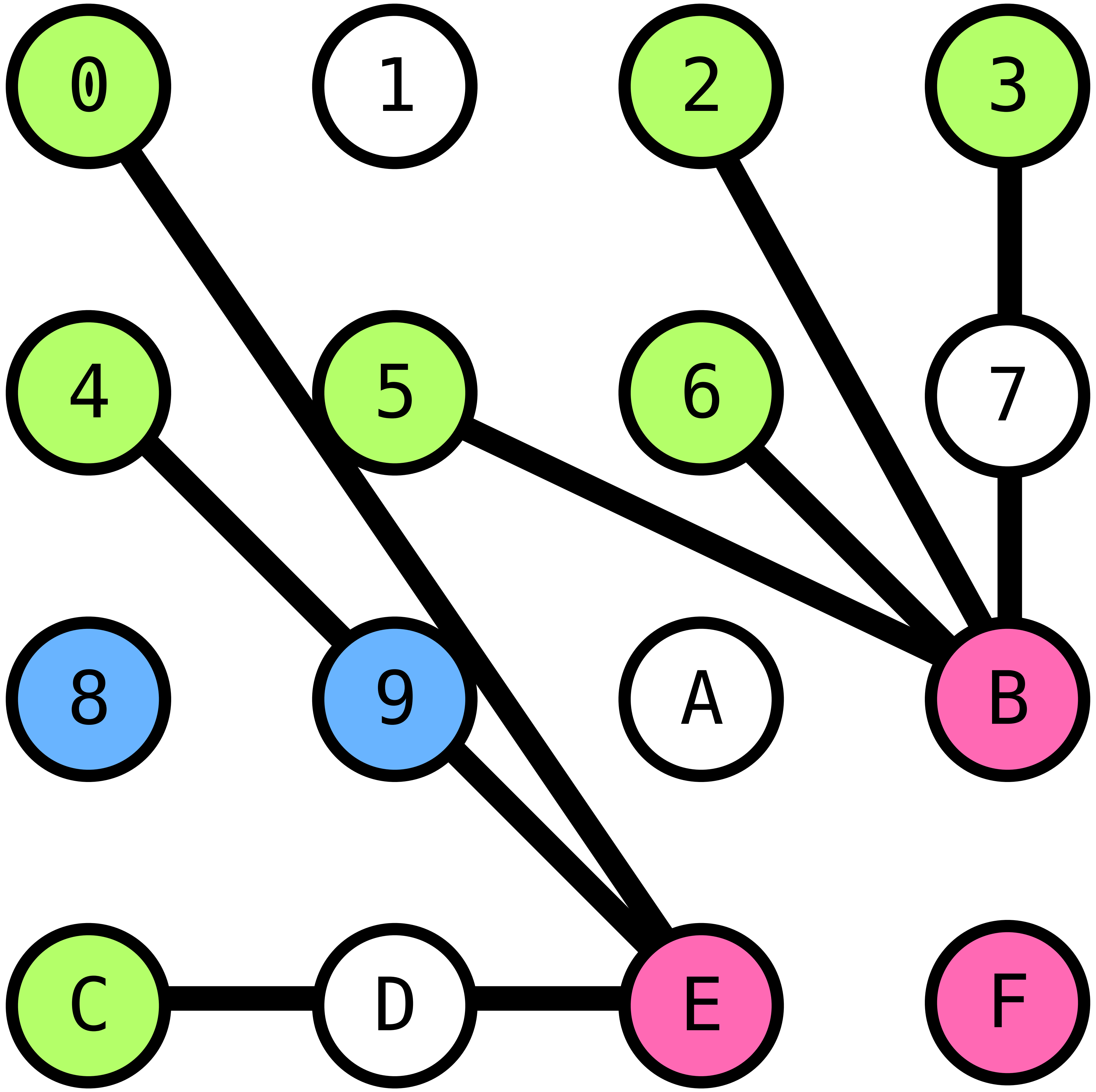
Fitness Function

44.03
3.41
1.00
48.44



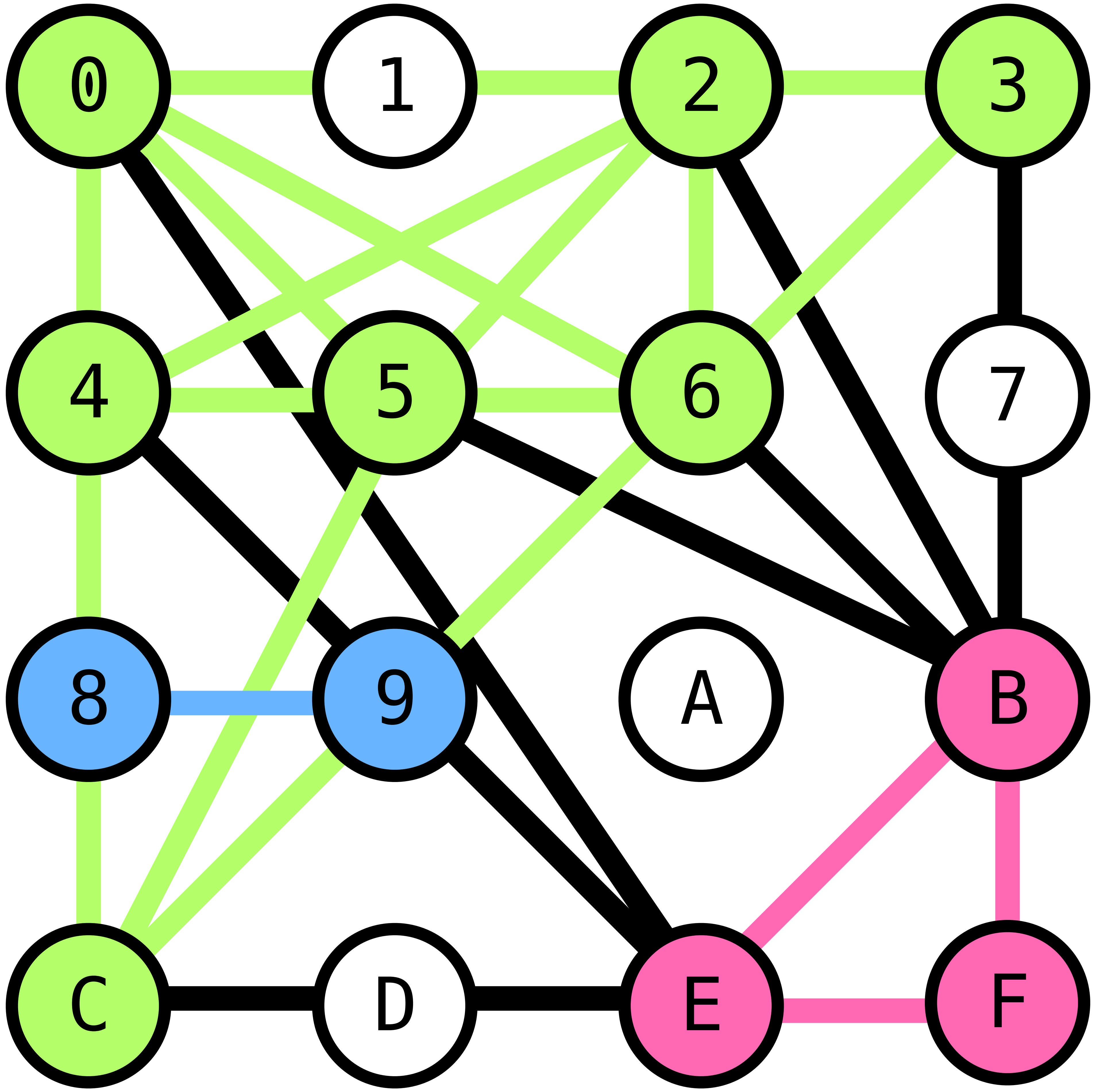
Fitness Function

30.38



Fitness Function

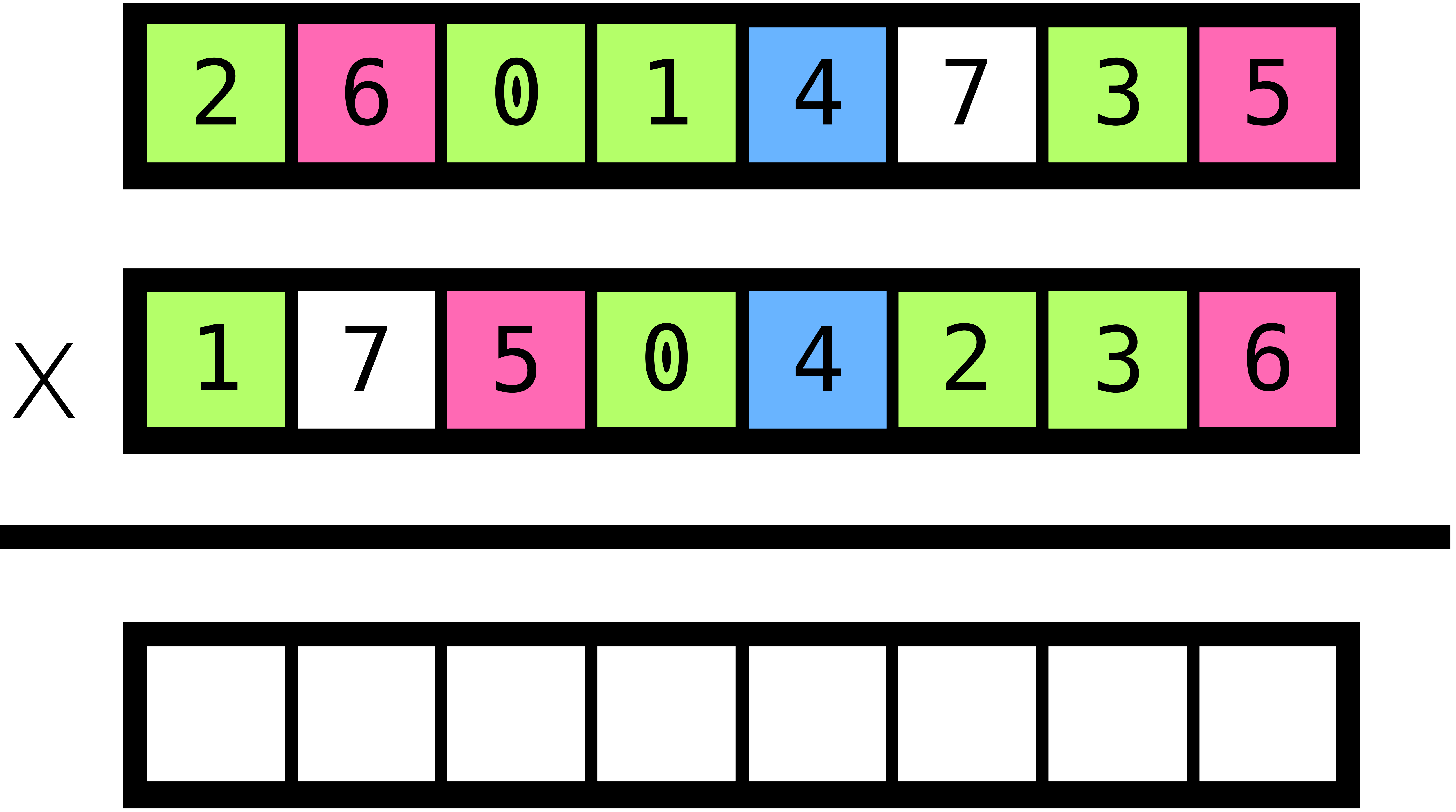
```
 48.44
-30.38
 18.06
```



Fitness Function

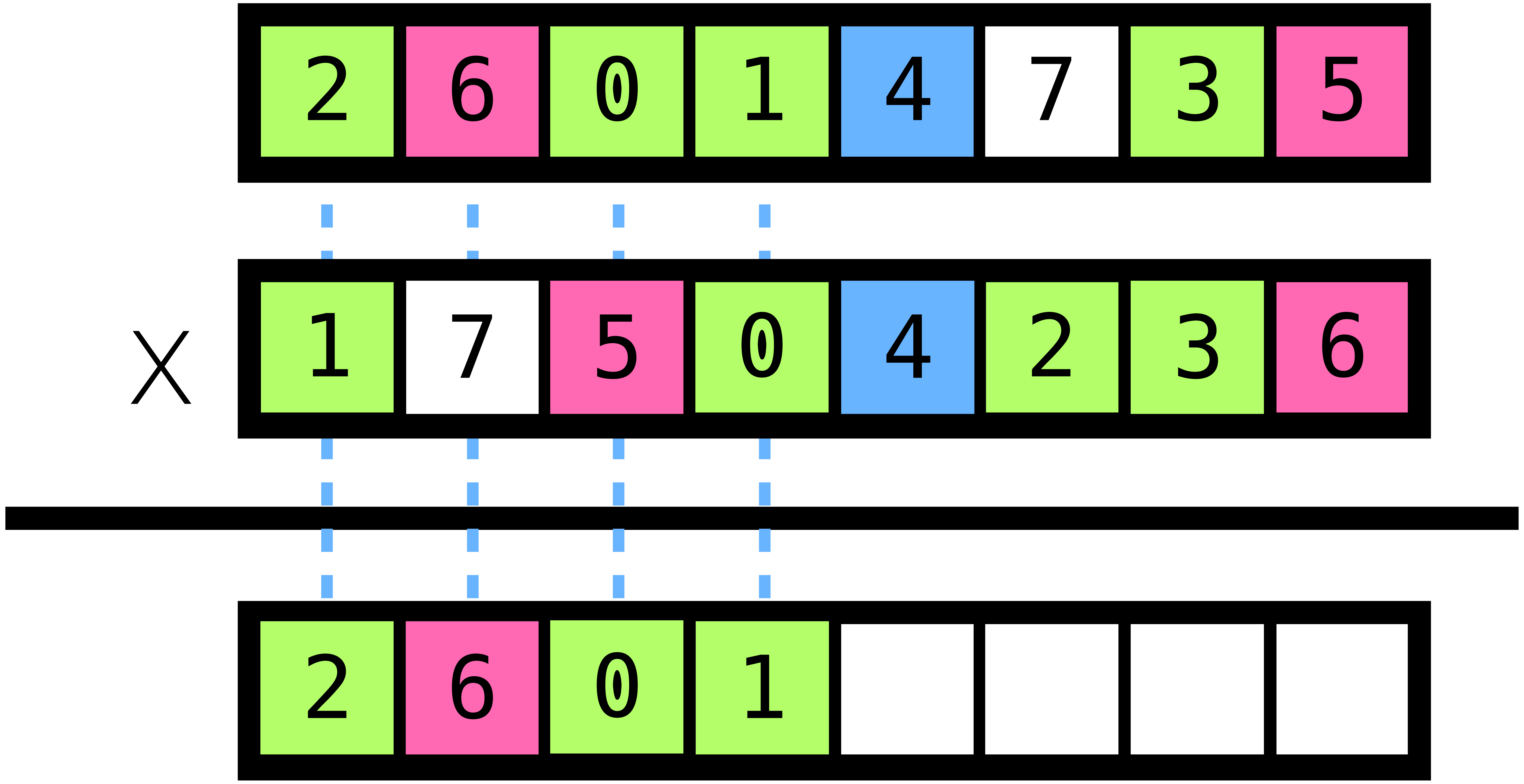
```
def score(self, chromosome):
    seats_role = self.seats_by_role(chromosome)

    # Tally attractive score
    attraction = 0.0
    for role, coords in seats_role.iteritems():
        attraction += spatial.total_edge_length(coords)

    # Tally repulsive score
    repulsion = 0.0
    for role, repulsors in self.repulsion.iteritems():
       …

    return self.max_distance - attraction + repulsion
```

Ordered Crossover (OX)



	2	6	0	1	4	7	3	5
X	1	7	5	0	4	2	3	6

Ordered Crossover (OX)

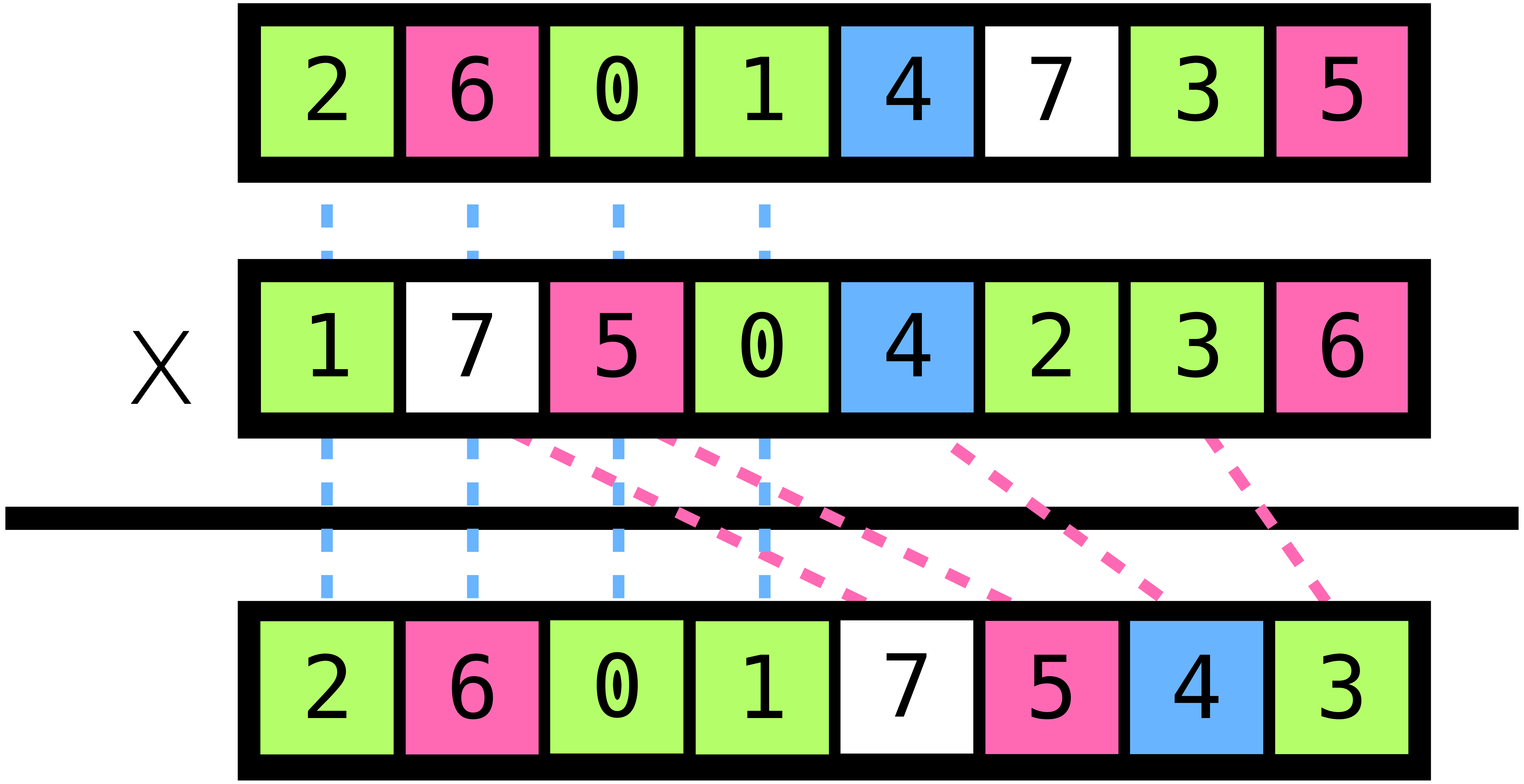


2 6 0 1 4 7 3 5

X 1 7 5 0 4 2 3 6

2 6 0 1

Ordered Crossover (OX)

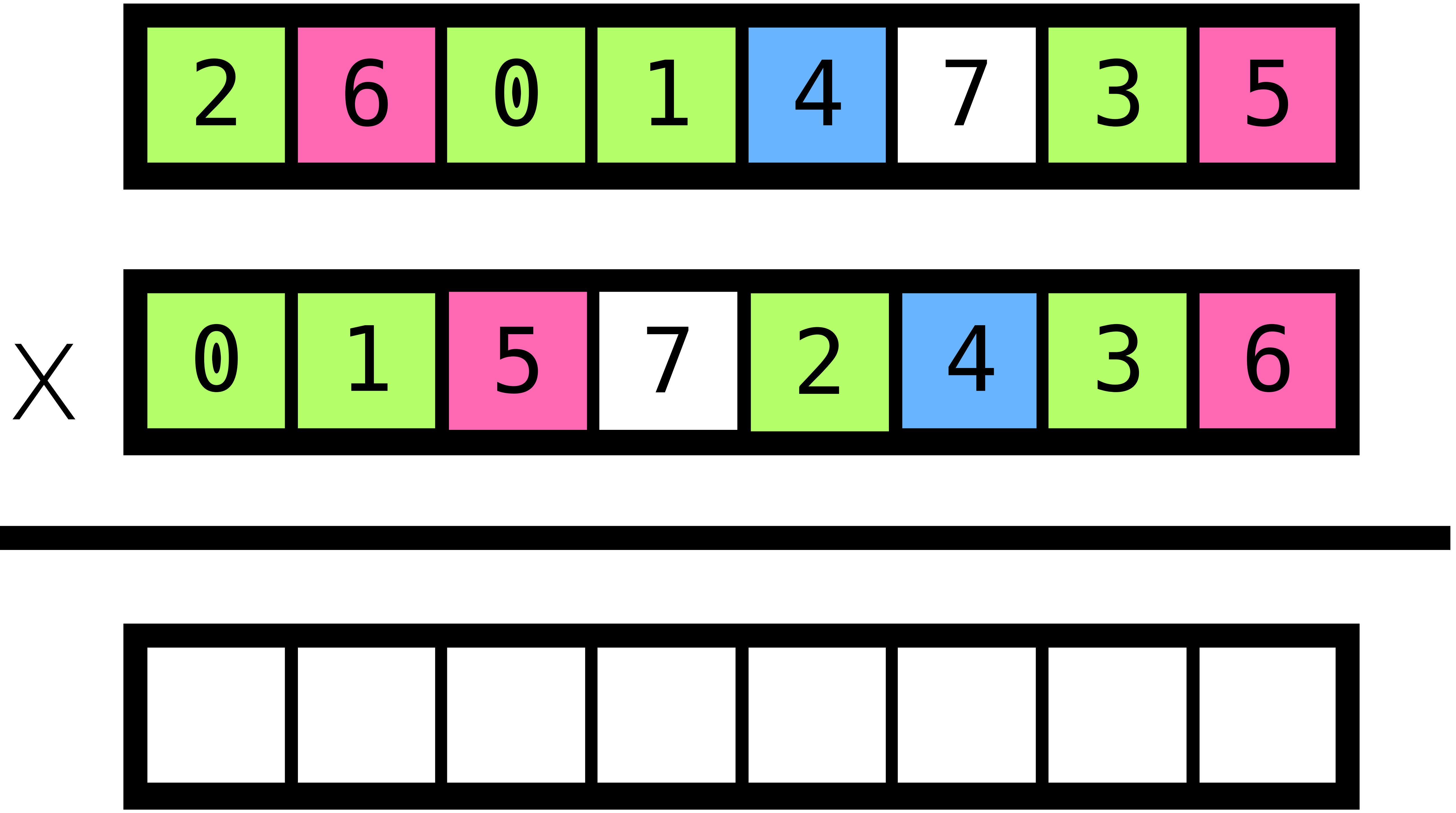


2 6 0 1 4 7 3 5

X 1 7 5 0 4 2 3 6

2 6 0 1 7 5 4 3

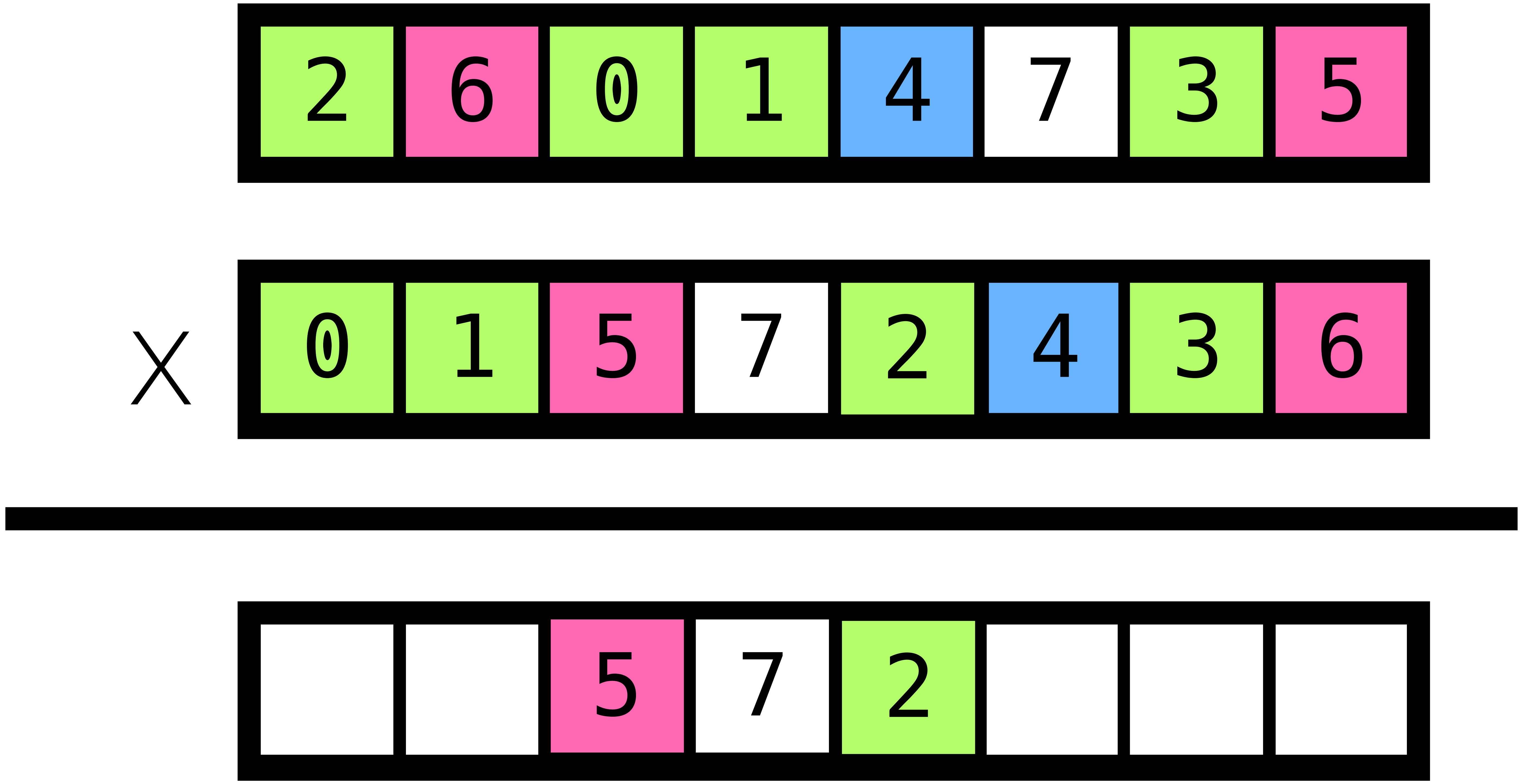
Partially Matched Crossover (PMX)



2 6 0 1 4 7 3 5

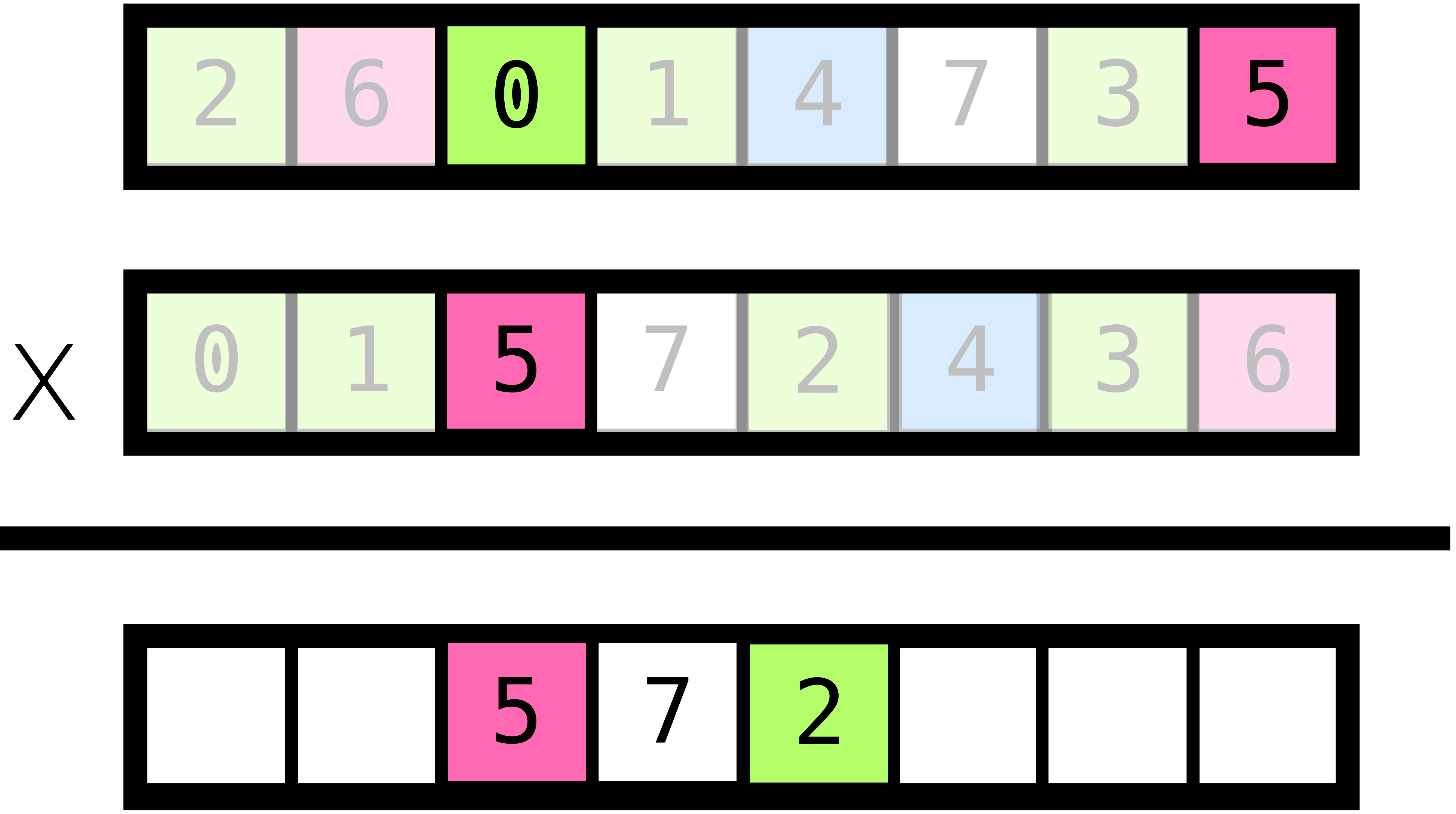
X 0 1 5 7 2 4 3 6

Partially Matched Crossover (PMX)



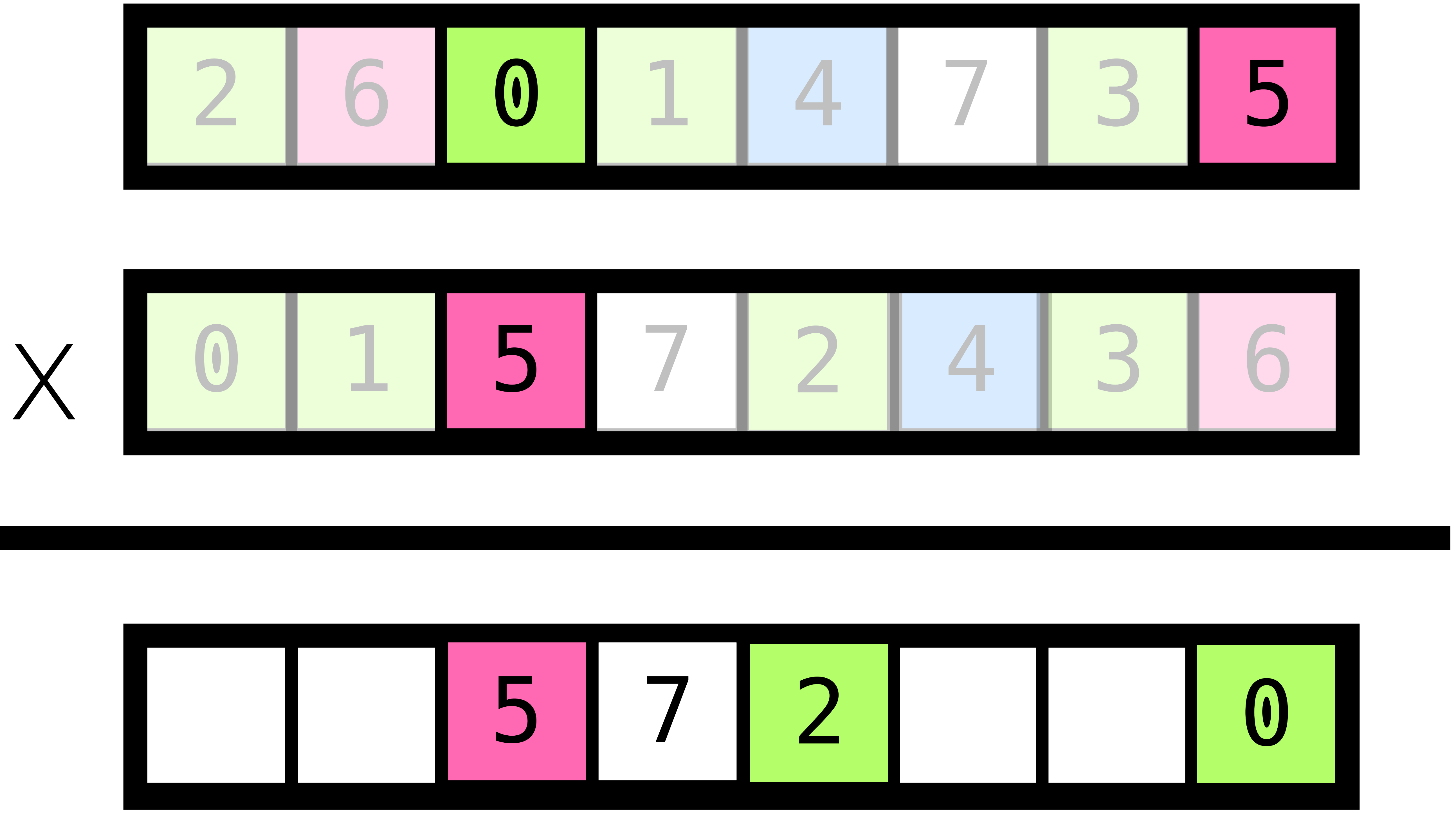
	2	6	0	1	4	7	3	5
X	0	1	5	7	2	4	3	6
			5	7	2			

Partially Matched Crossover (PMX)



	2	6	0	1	4	7	3	5
X	0	1	5	7	2	4	3	6
			5	7	2			

Partially Matched Crossover (PMX)

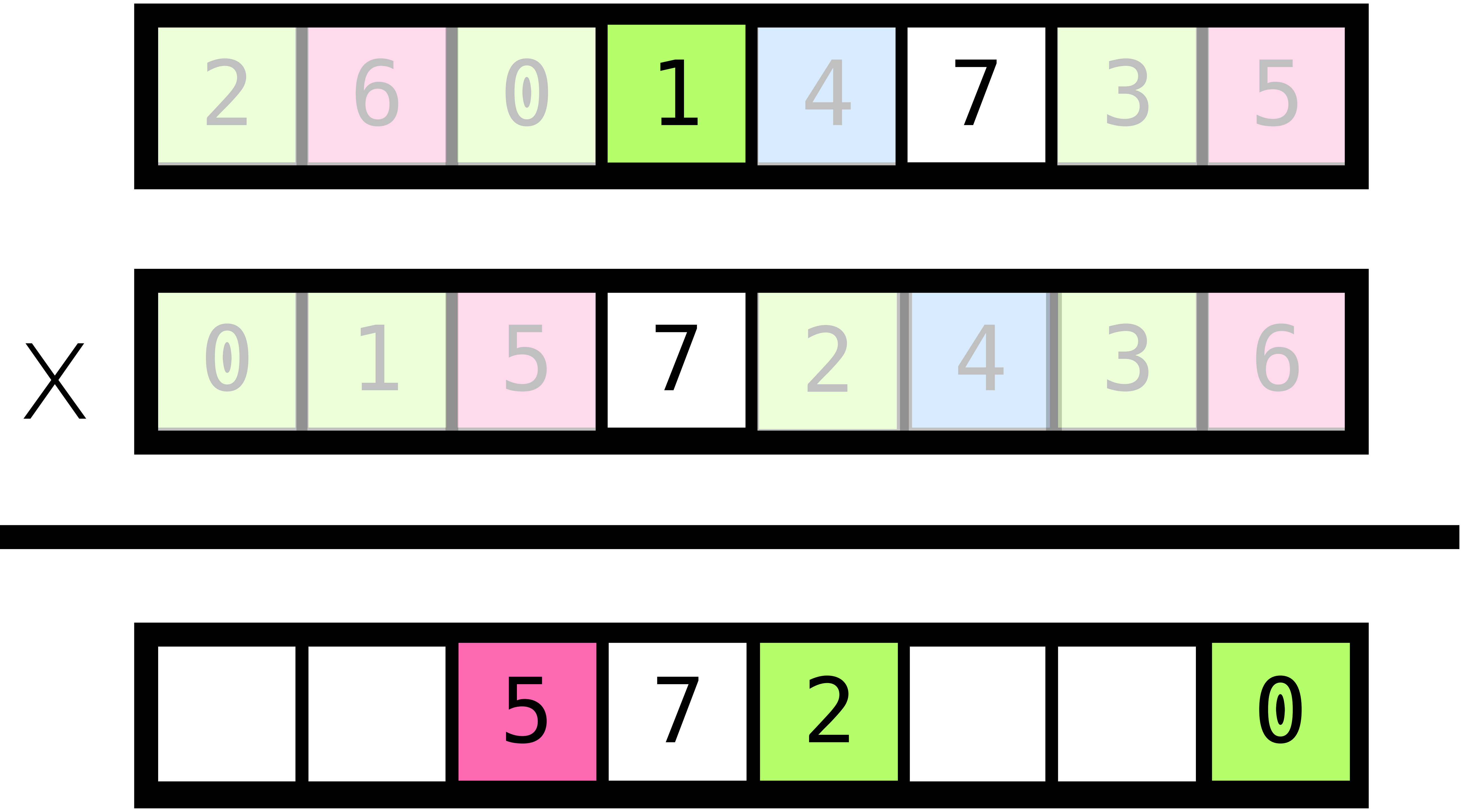


2 6 0 1 4 7 3 5

X 0 1 5 7 2 4 3 6

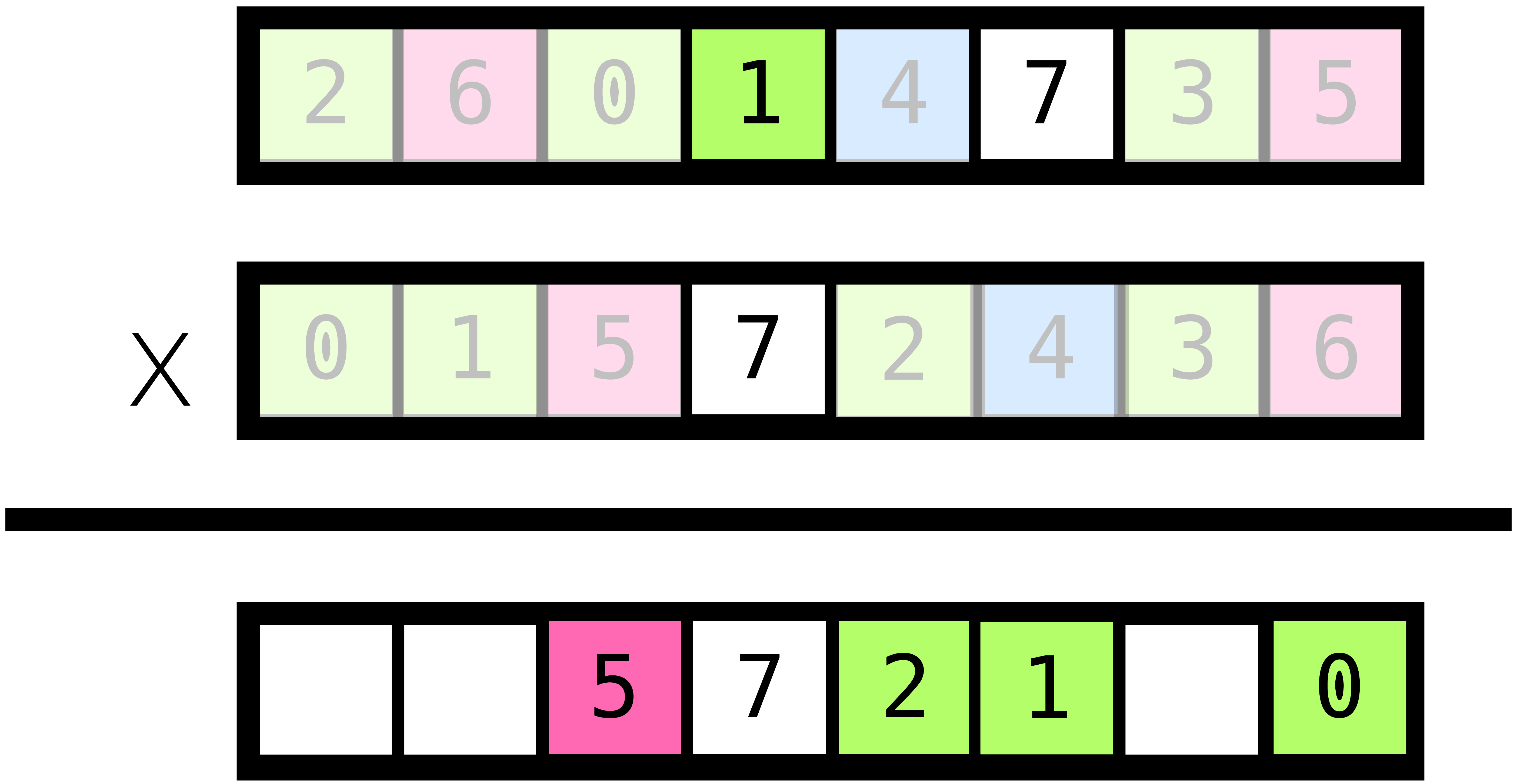
5 7 2 0

Partially Matched Crossover (PMX)



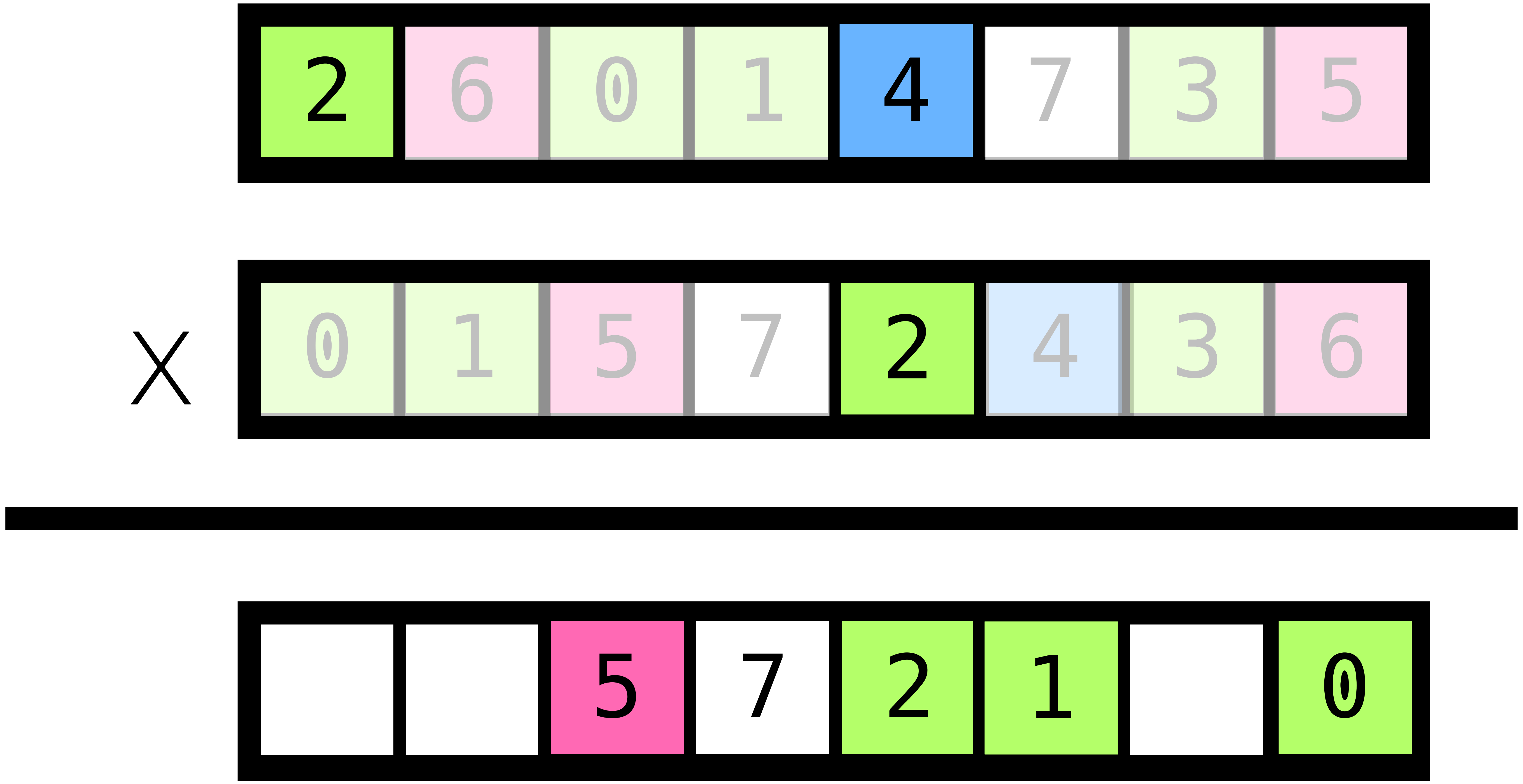
	2	6	0	1	4	7	3	5
X	0	1	5	7	2	4	3	6
			5	7	2			0

Partially Matched Crossover (PMX)



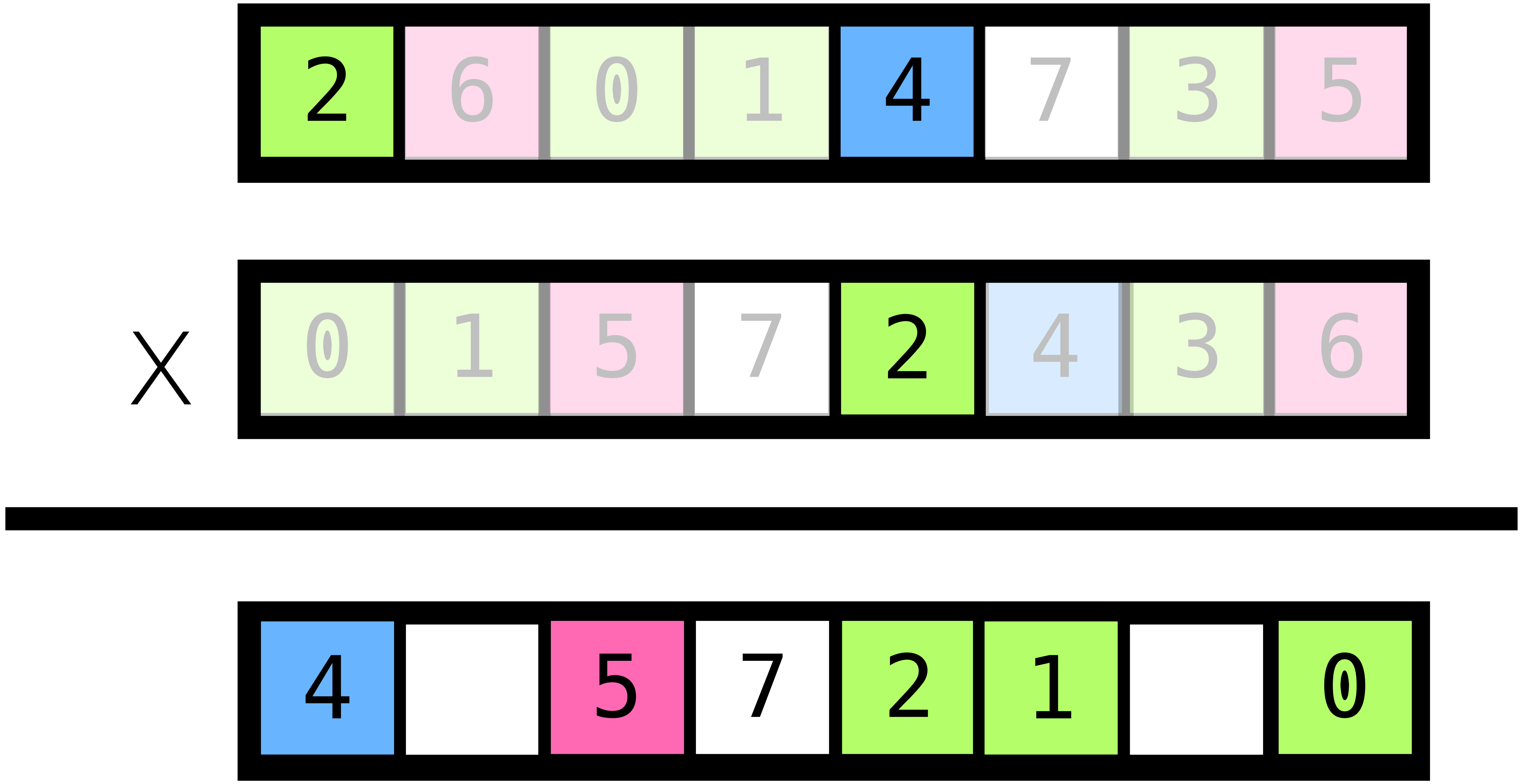
	2	6	0	1	4	7	3	5
X	0	1	5	7	2	4	3	6
			5	7	2	1		0

Partially Matched Crossover (PMX)



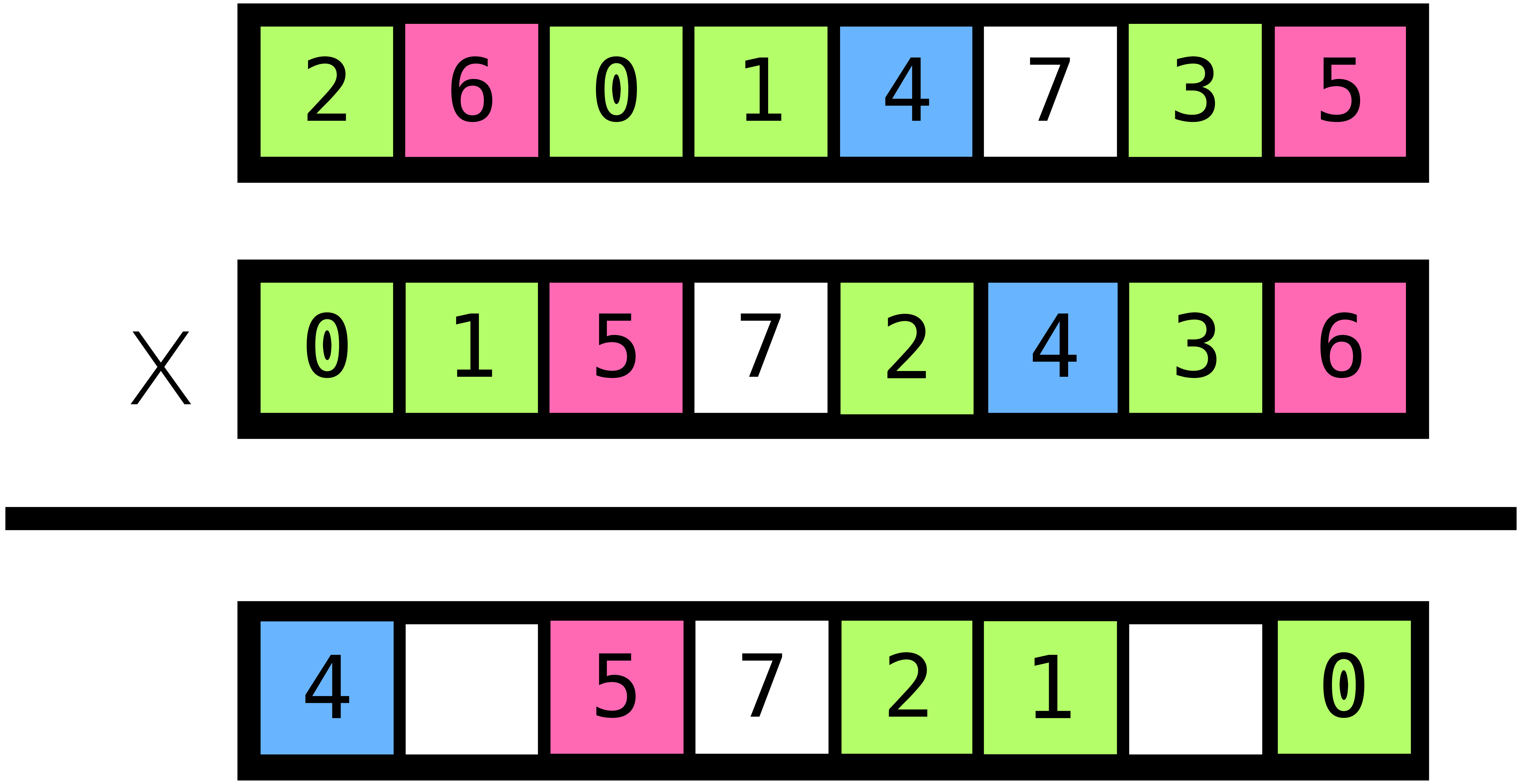
	2	6	0	1	4	7	3	5
X	0	1	5	7	2	4	3	6
			5	7	2	1		0

Partially Matched Crossover (PMX)



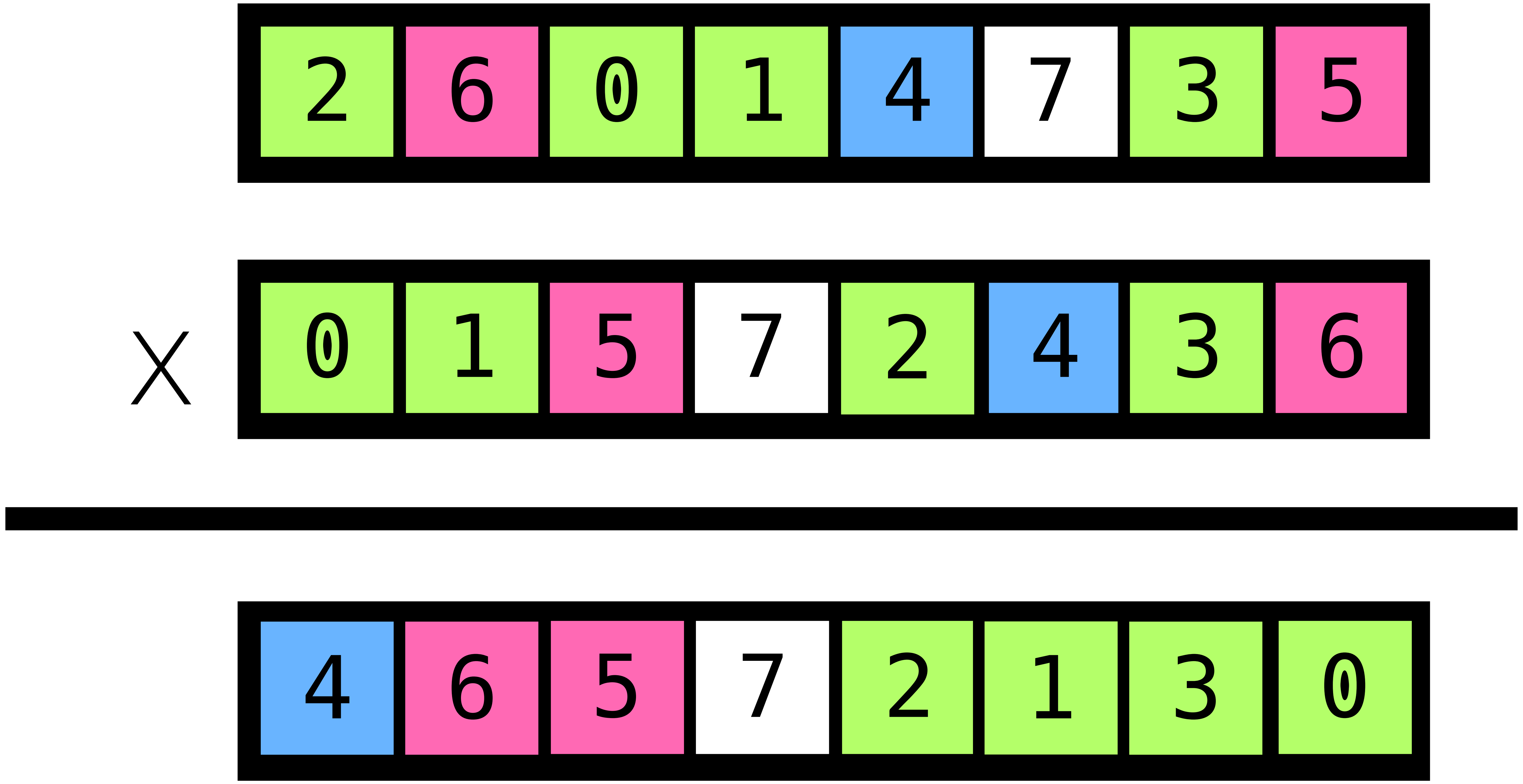
2	6	0	1	4	7	3	5
0	1	5	7	2	4	3	6
4		5	7	2	1		0

Partially Matched Crossover (PMX)



	2	6	0	1	4	7	3	5
X	0	1	5	7	2	4	3	6
	4		5	7	2	1		0

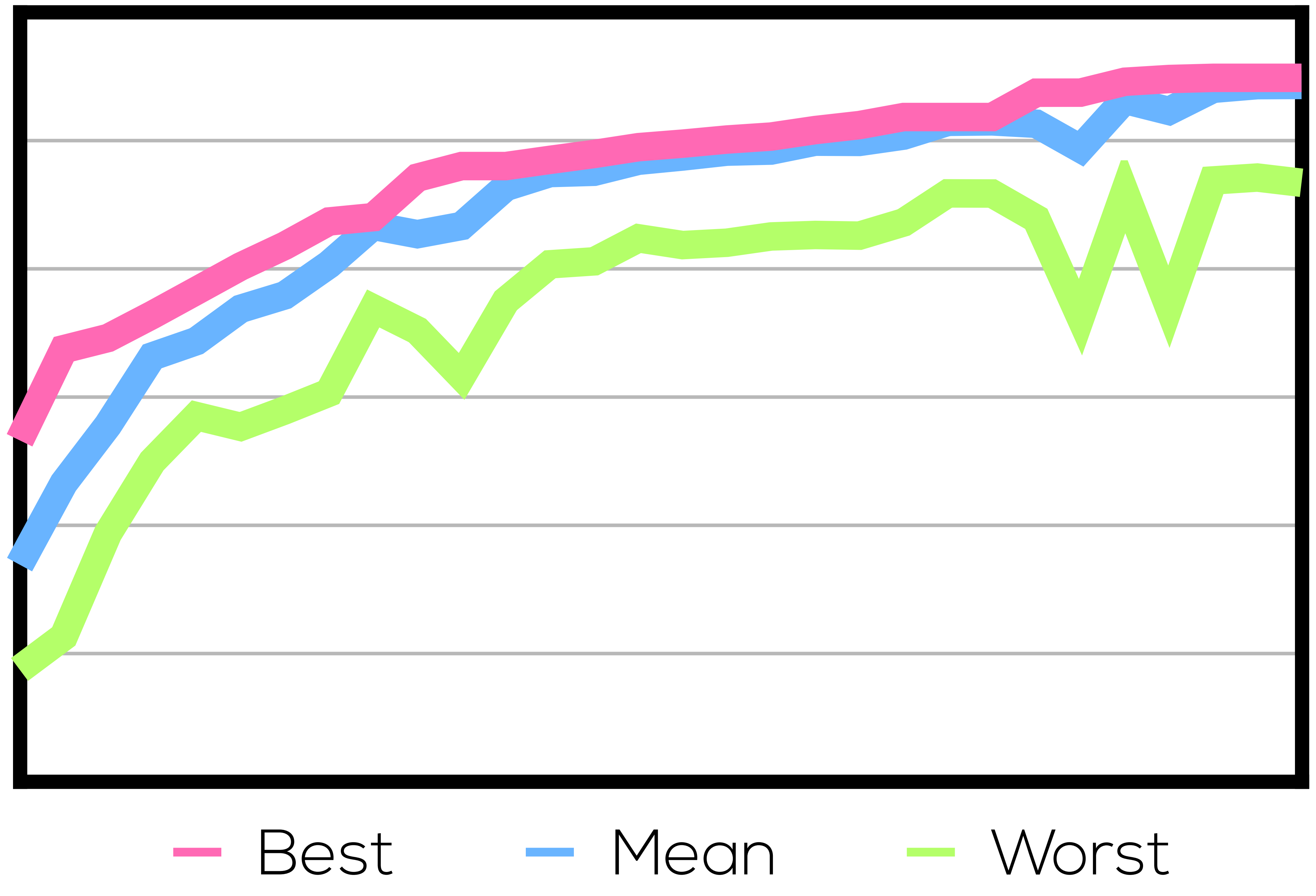
Partially Matched Crossover (PMX)



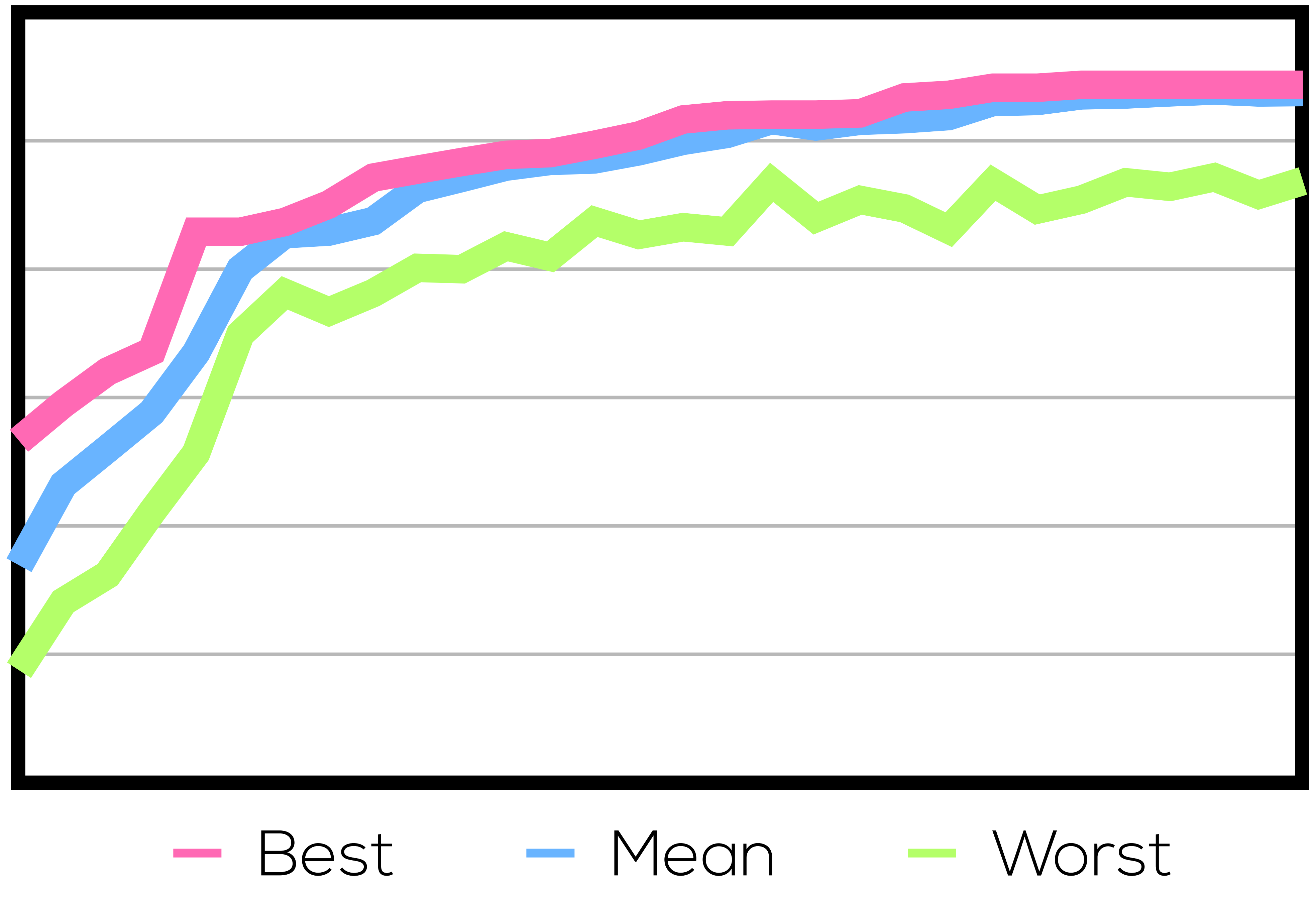
	2	6	0	1	4	7	3	5
X	0	1	5	7	2	4	3	6
	4	6	5	7	2	1	3	0

Performance Comparison

Ordered

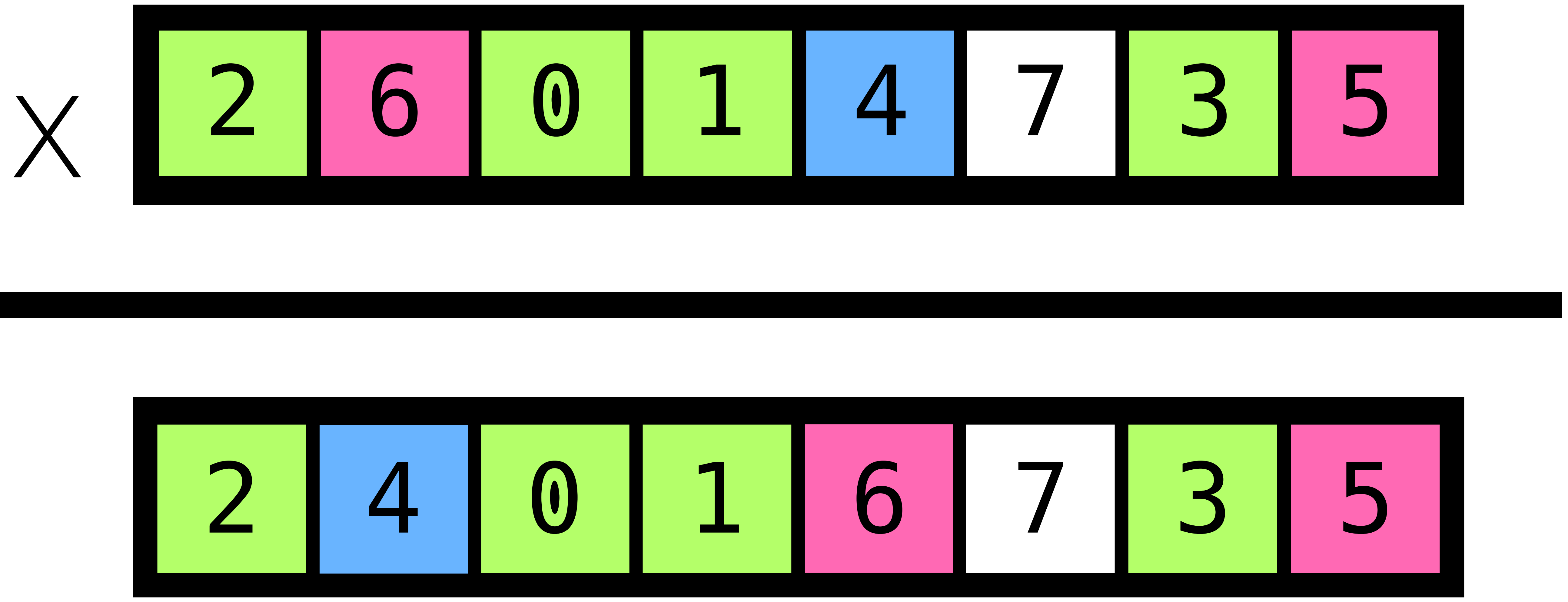


Partially Matched



```
map_size=7x7, population_size=200, iterations=30, elitism=0, seed=“SeatingChart”, map_type=naive
```

Swap Mutation

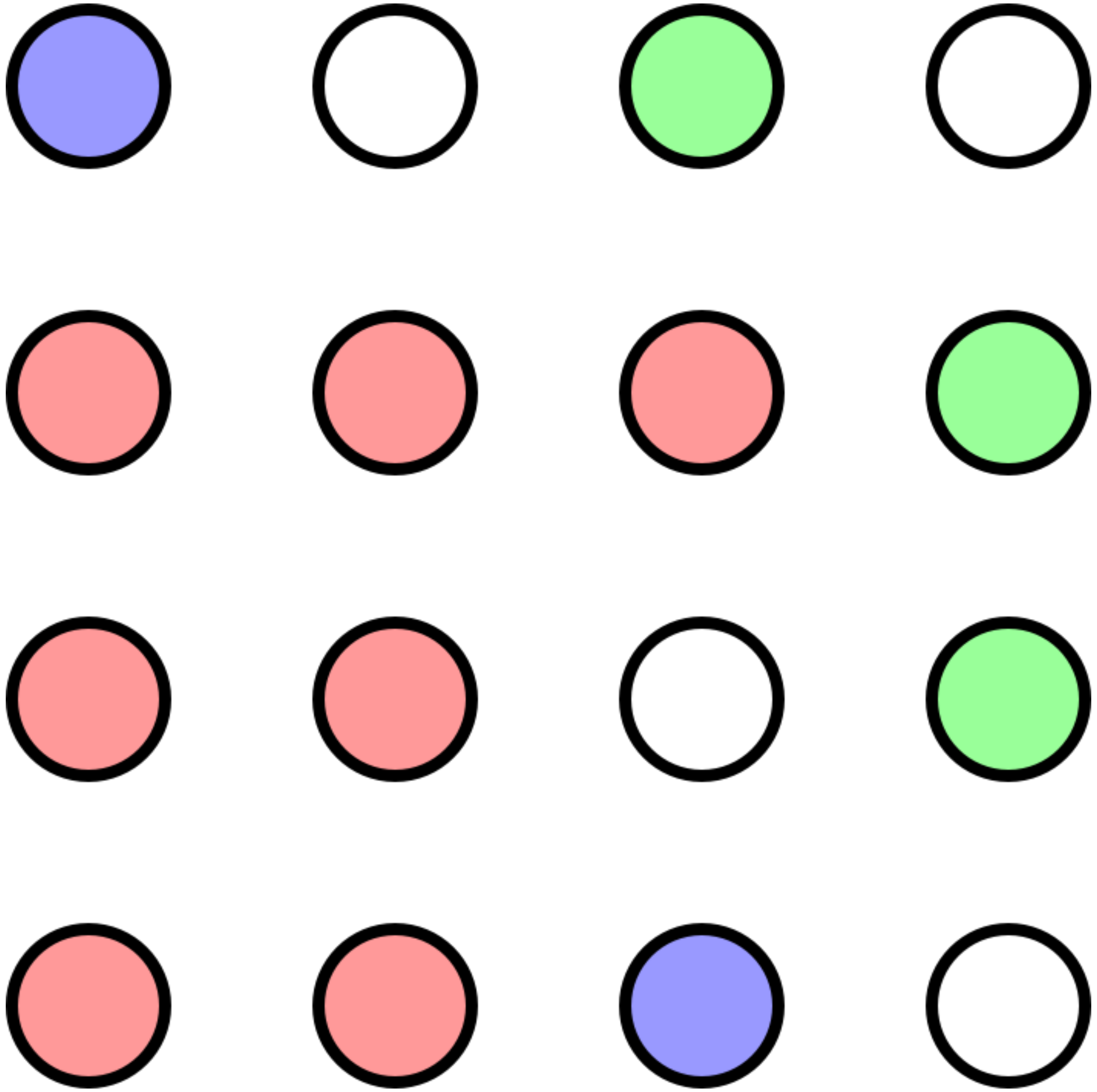


X

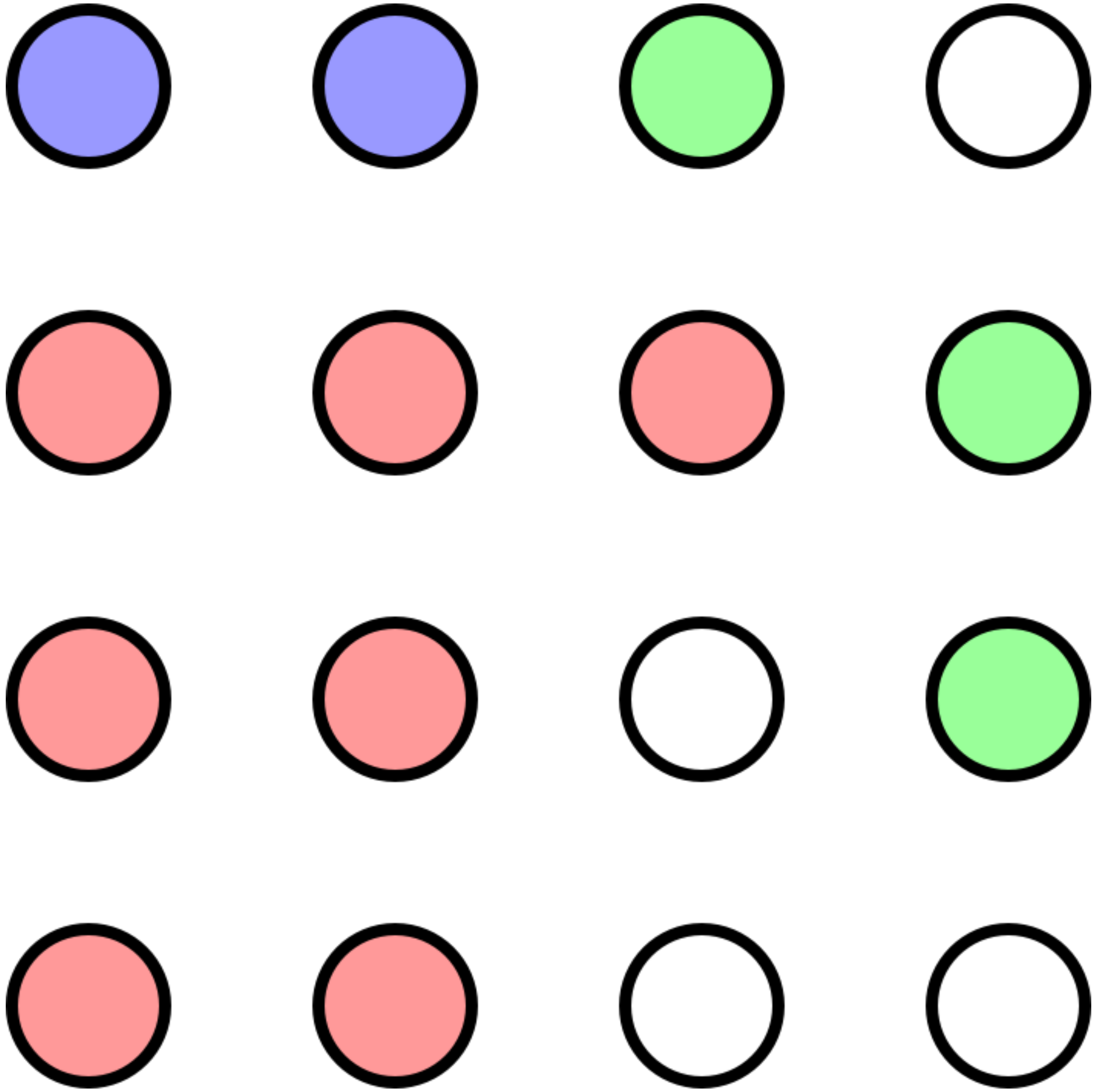
2	6	0	1	4	7	3	5

2	4	0	1	6	7	3	5

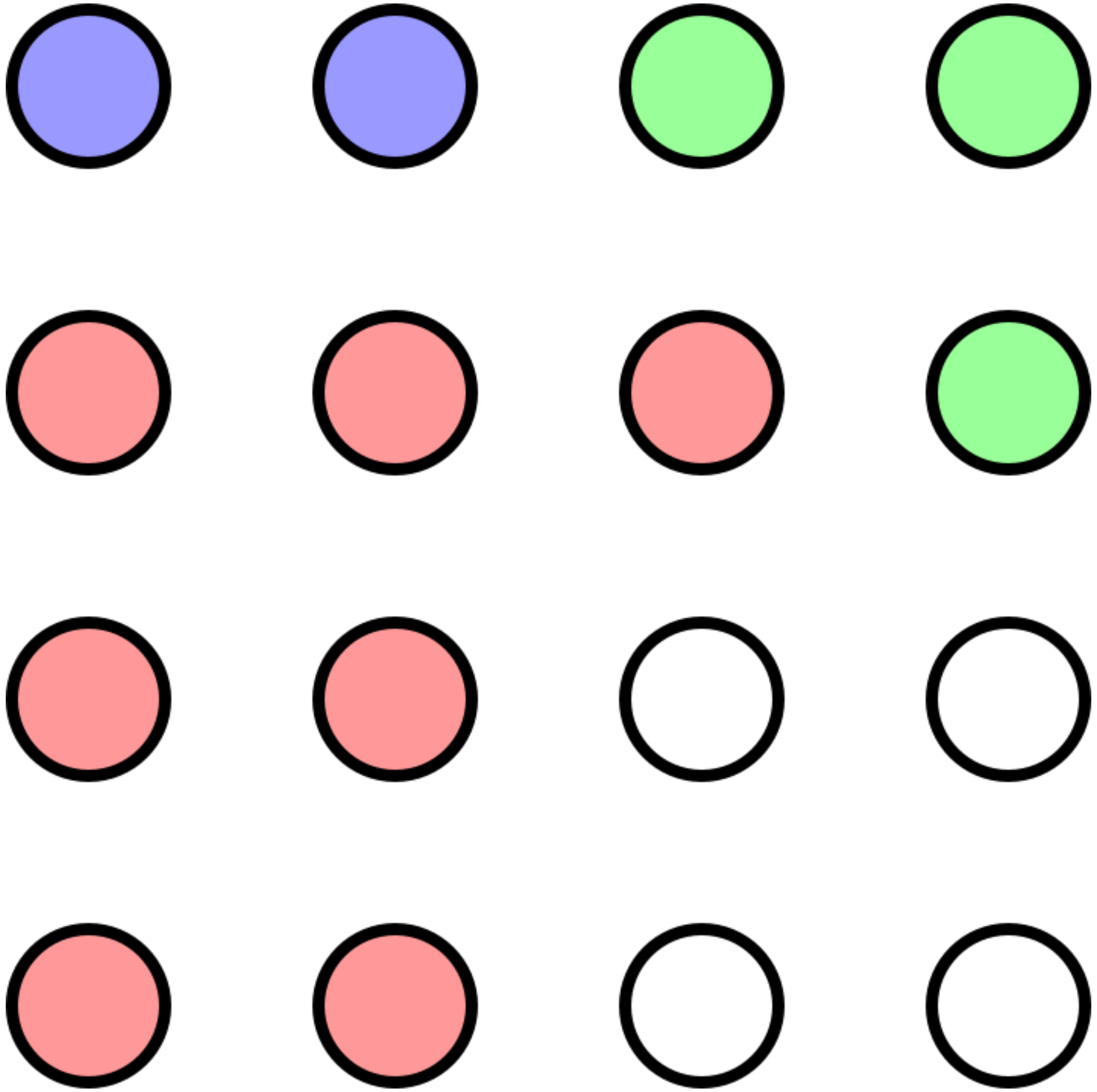
Solution



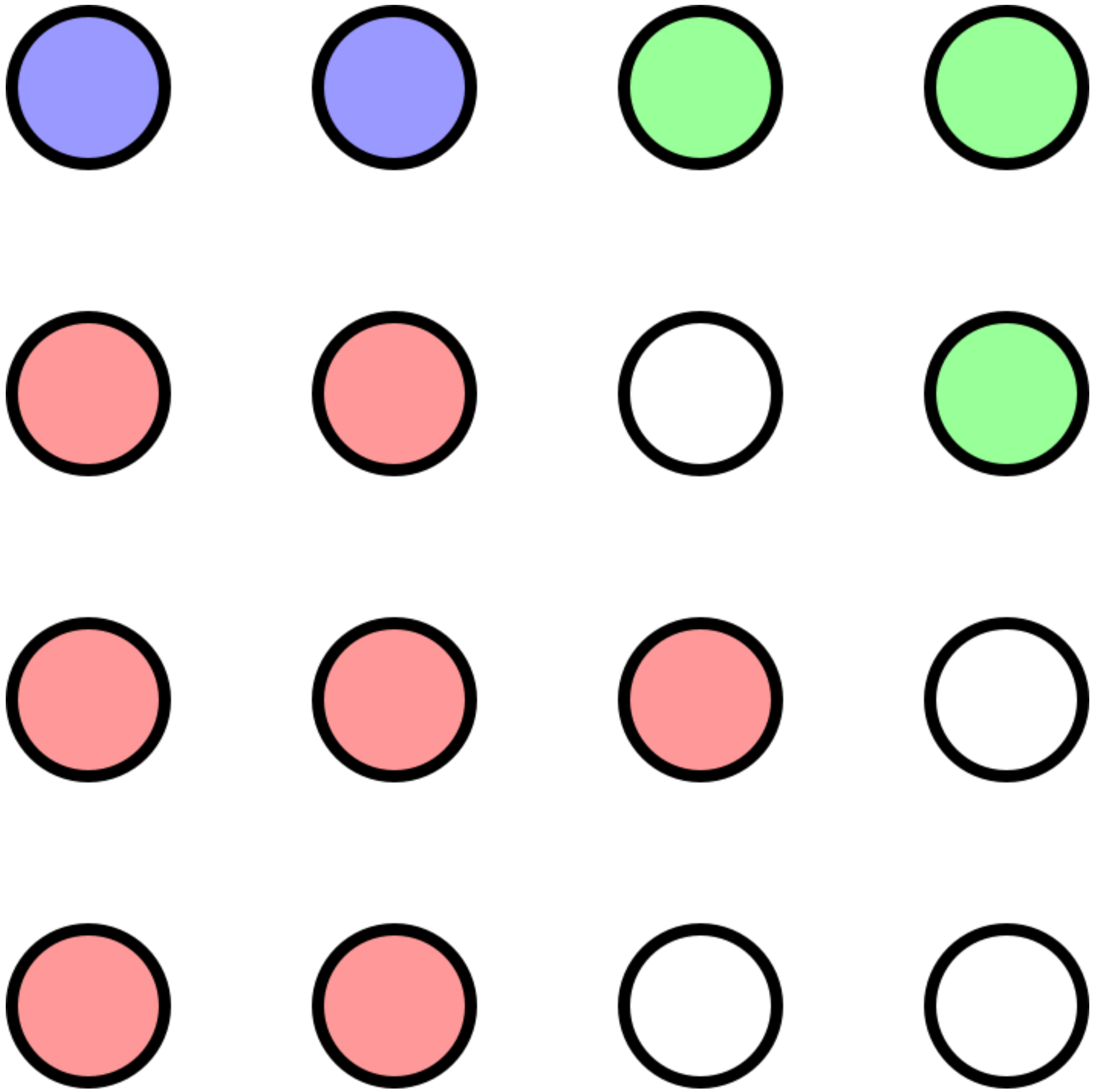
Solution



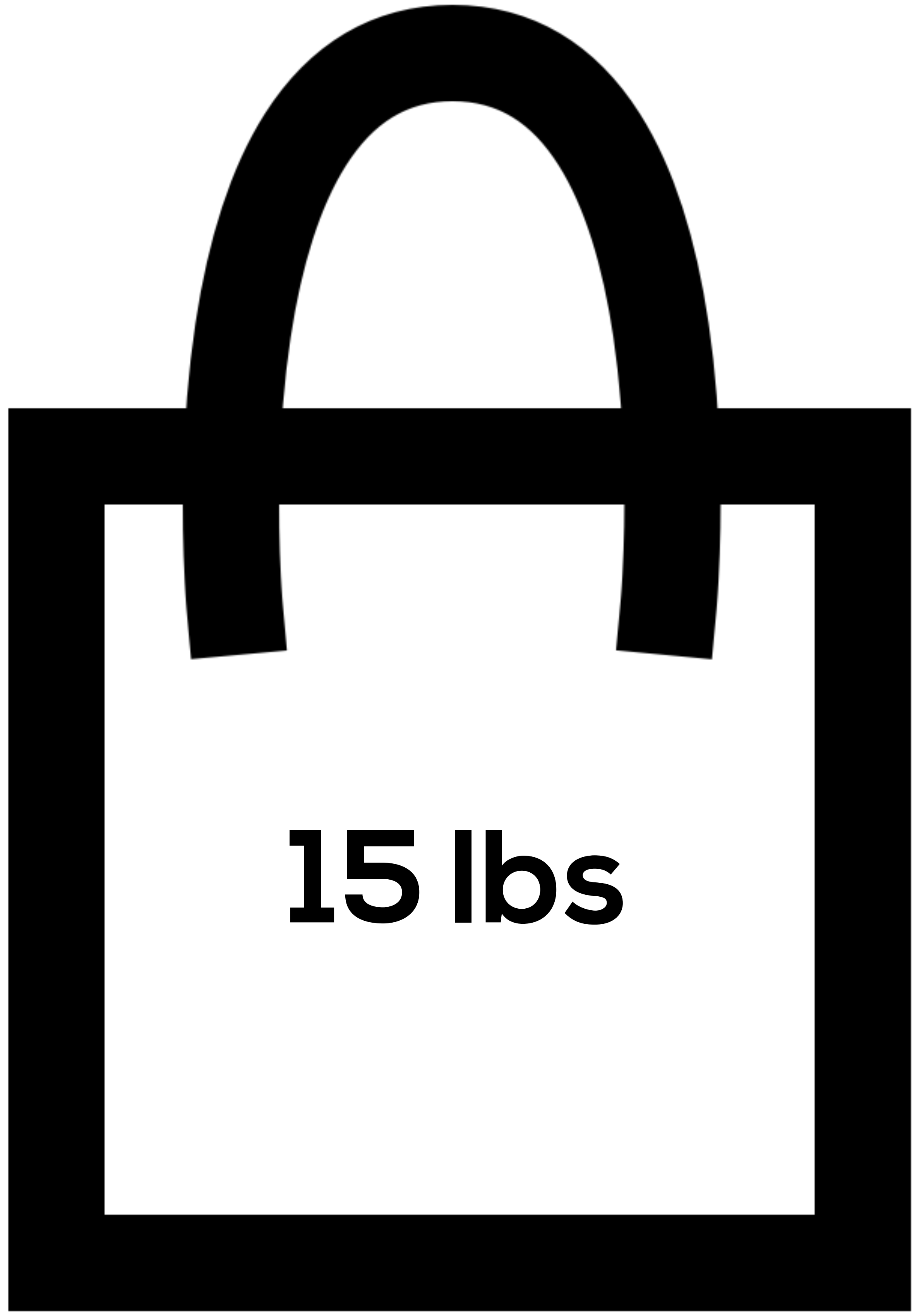
Solution



Solution



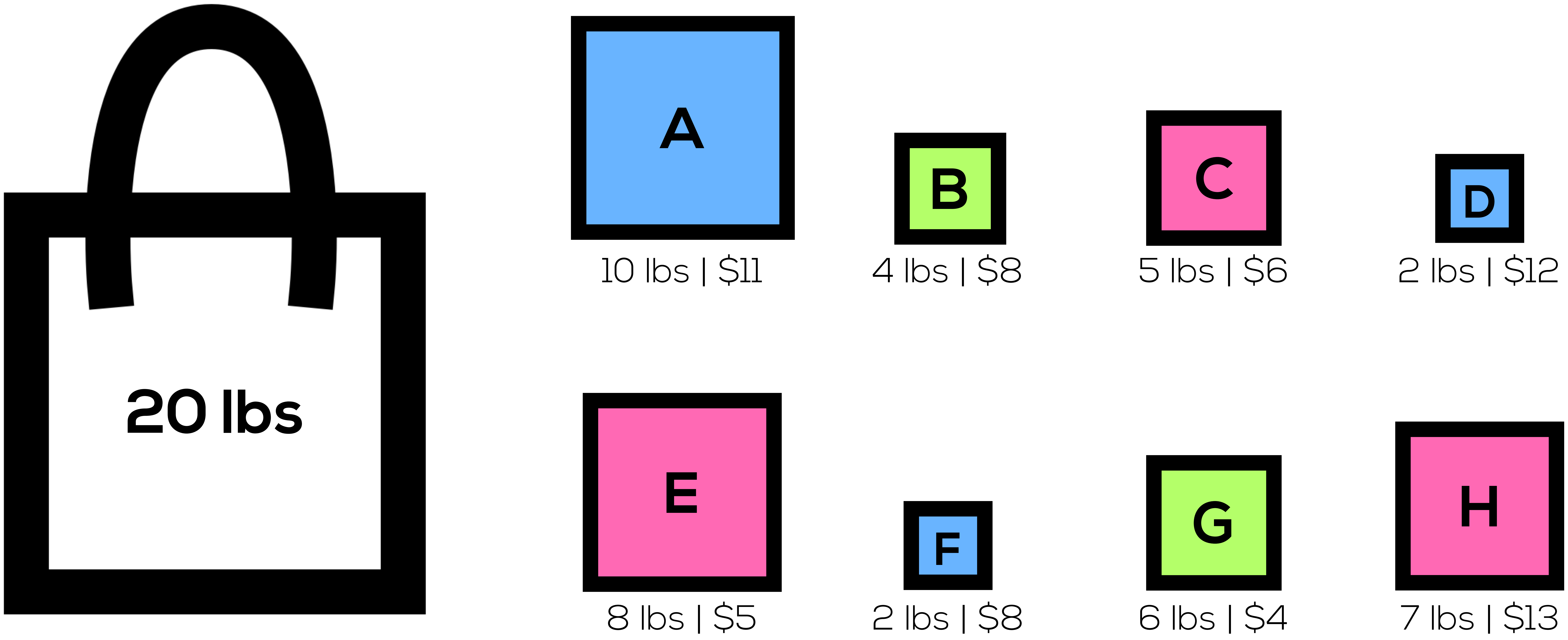
Unbounded Knapsack Problem



Given a set of items, each with a quantified weight and value ($), what combination of those items **in any amounts** will:

1. Fit inside a knapsack capable of carrying a fixed amount of weight?
2. Maximize the value of the knapsack's payload?

Unbounded Knapsack Problem



20 lbs

A

10 lbs | $11

B

4 lbs | $8

C

5 lbs | $6

D

2 lbs | $12

E

8 lbs | $5

F

2 lbs | $8

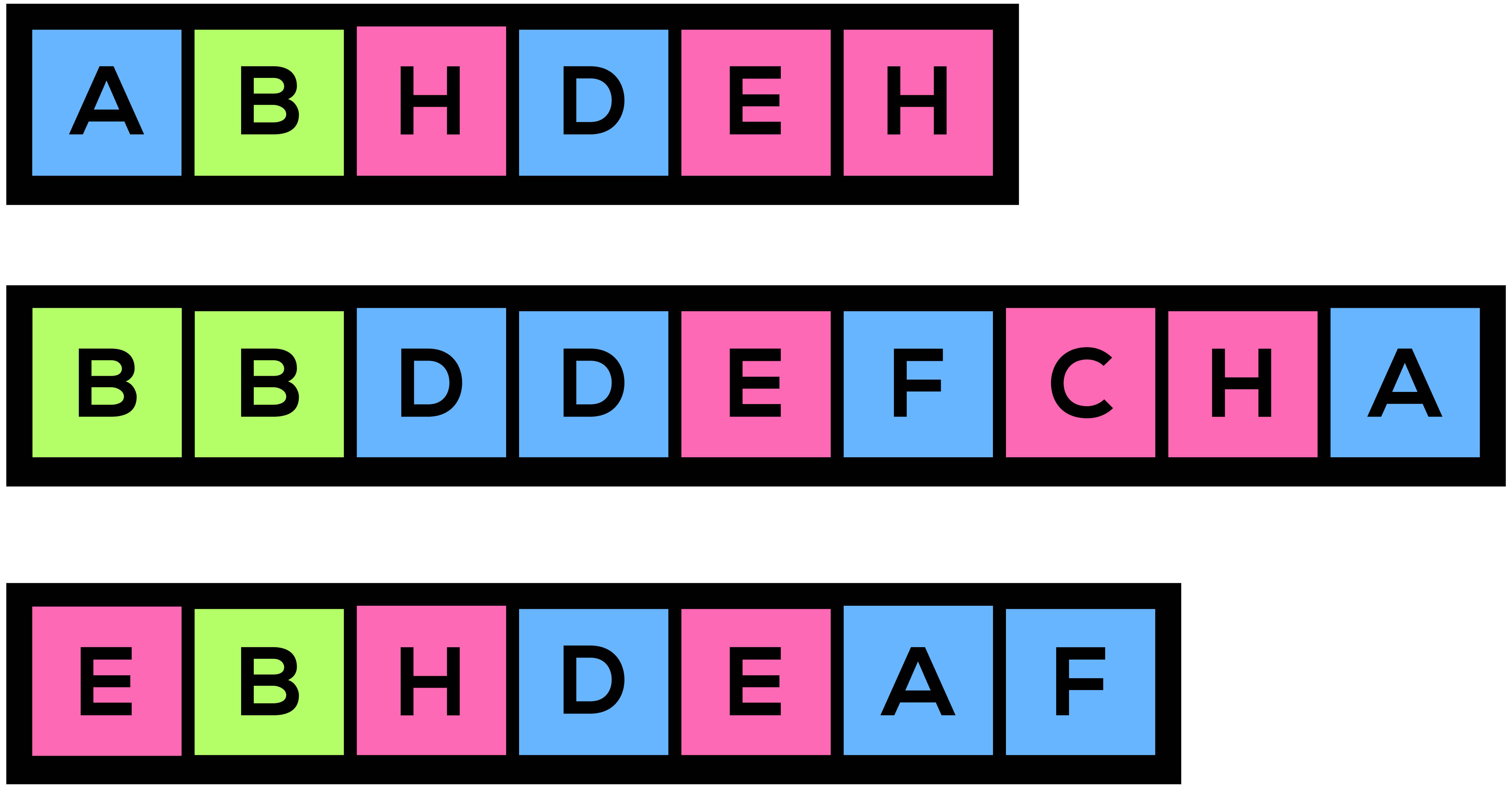
G

6 lbs | $4

H

7 lbs | $13

List Encoding



A B H D E H

B B D D E F C H A

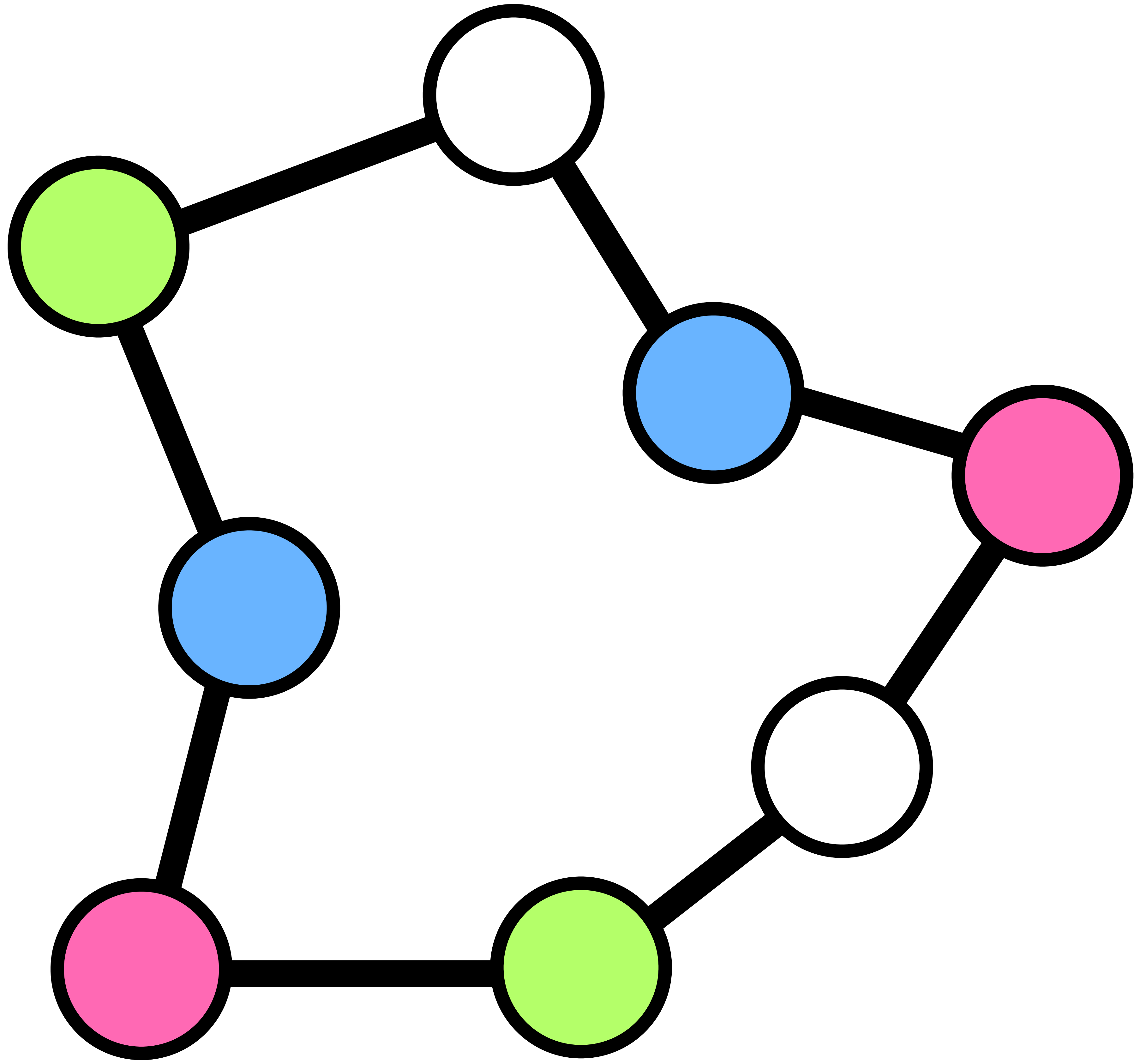
E B H D E A F

Unbounded Knapsack Problem

Encoding	List
Crossover	Cut and Splice / Single Point
Selection	Proportionate
Mutation	Point

Traveling Salesman Problem

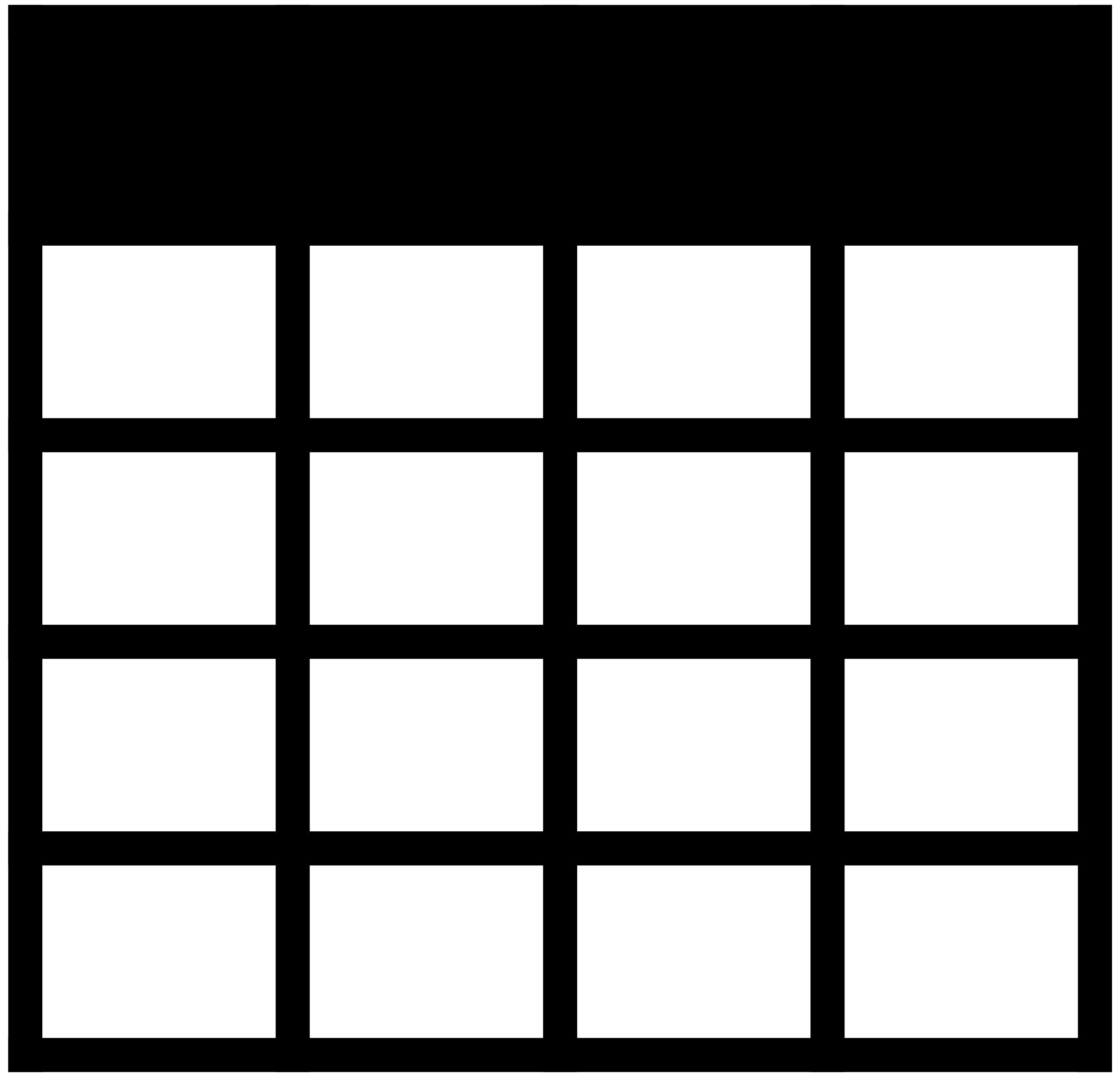
Minimize the total distance required to visit all stops along a route.



Traveling Salesman Problem

Encoding	Permutation
Crossover	Ordered or Edge Recombination
Selection	Proportionate
Mutation	Swap

Nurse Scheduling Problem



7 days / week
3 shifts / day

- There **must** always be n nurses scheduled.
- A nurse **cannot** be scheduled while on PTO.
- A nurse **cannot** be scheduled for more than two consecutive shifts.
- A nurse **should** have at least two shifts off between scheduled shifts.
- A nurse’s shift preferences **should** be respected.

Nurse Scheduling Problem

Name	Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Grace	1							
Constance	3		PTO	PTO				
Ronald	2							
Robyn	2						PTO	PTO
Judy	1							
Amy	1, 2							
Brad	3							
Valarie	1				PTO			
James	2							
Thelma	2, 3	PTO						
...	...	...	...	...	...	...	...	...

Nurse Scheduling Problem

Encoding	Binary (with constraints)
Crossover	Single Point Crossover
Selection	Proportionate
Mutation	Point

Encoding Constraints

	Grace						Constance							…	
	1	0	0	1	0	…	0	0	1	1	0	1	…	…	…
Shift	1	2	3	1	2	…	1	2	3	1	2	3	…	…	…
Day	Mon			Tues			Mon			Thurs				…	

When to Evolve

Use a GA on problems when:

- There may not be an exact solution
- Search spaces are large
- The fitness landscape is complex
- The best solutions lie on a Pareto front

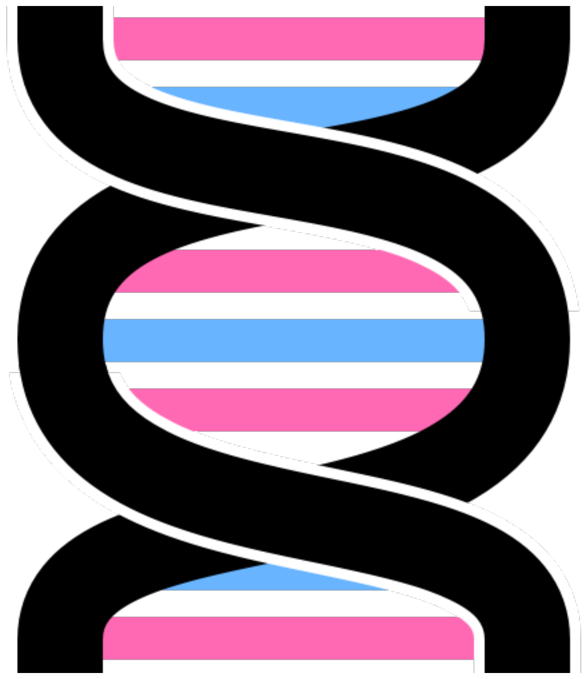
Genetic Algorithms

- Finds better solutions
- Global search
- Easily parallelized
- Broader applications

vs

Simulated Annealing

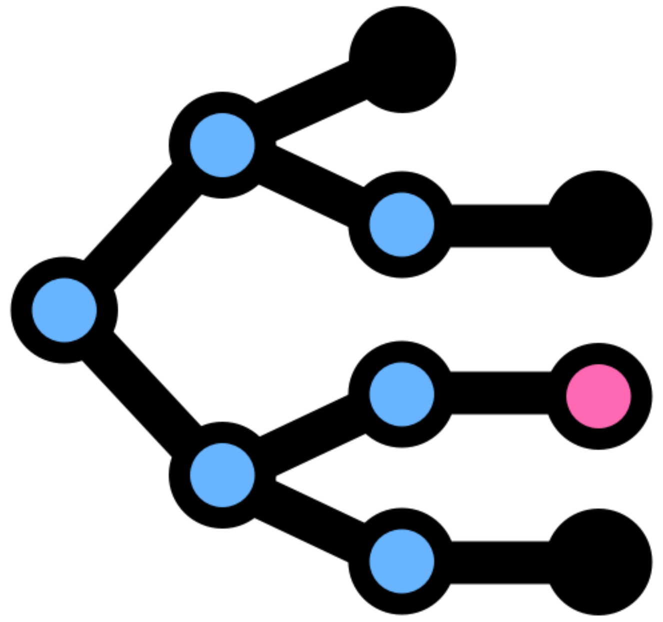
- Finds solutions faster
- Local search



Genetic Algorithms

- Approximate solutions
- Better able to cope with large, complex problems

VS



Branch & Bound

- Best solution, guaranteed
- Limited to small problems
  - O(n log n)

Genetic Algorithms

- Discrete optimization
- Wins at complex fitness landscapes
- No calculus required =)

vs

Gradient Descent

- Continuous optimization
- Wins at speed

GENETIC ALGORITHMS

- Optimization
- Classification (GBML)
- Human Comparable Design

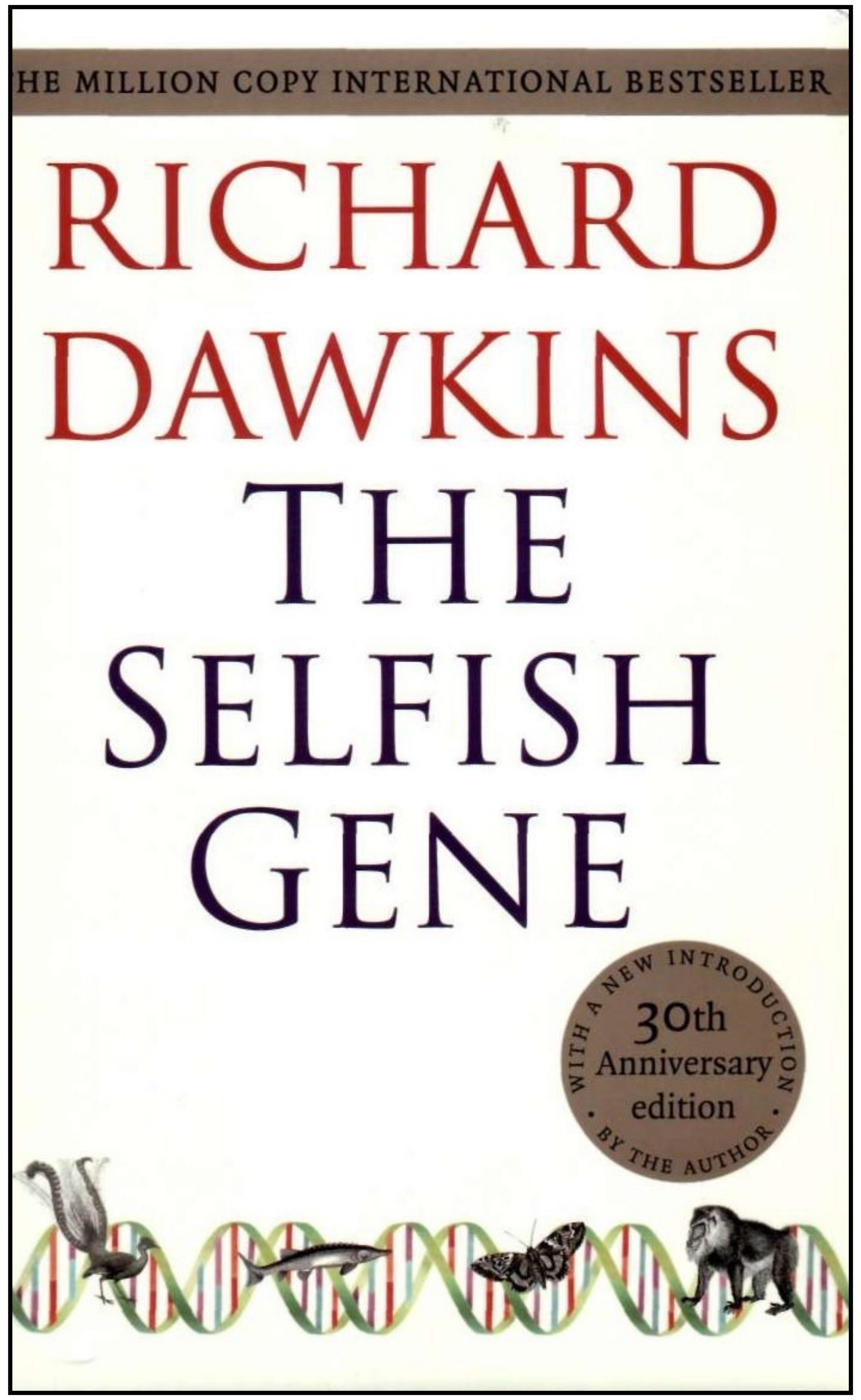
VS

NEURAL NETWORKS

- Classification†
- Pattern recognition

† Also for screenplay creation

Further Reading



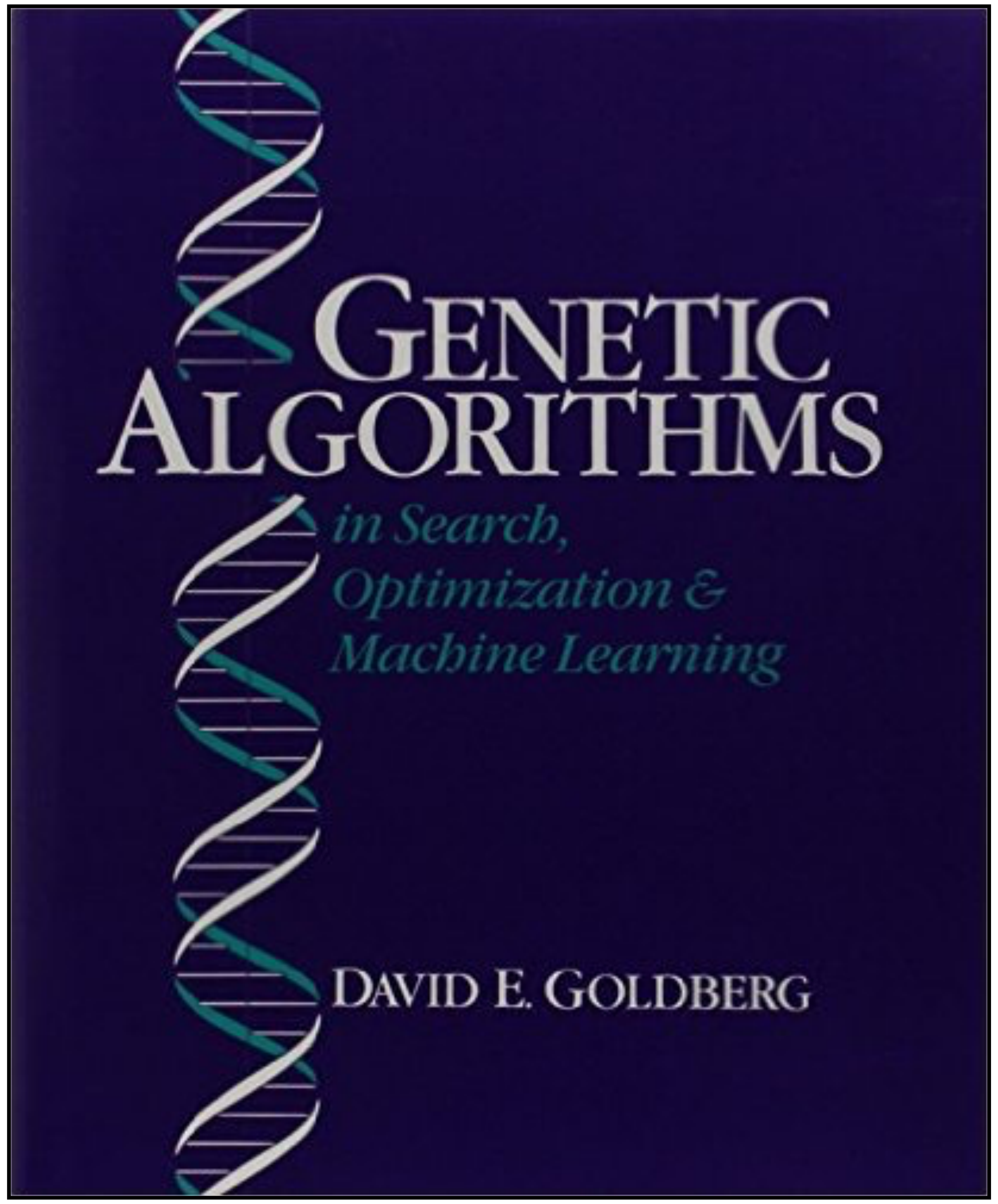
HE MILLION COPY INTERNATIONAL BESTSELLER

RICHARD DAWKINS

THE SELFISH GENE

WITH A NEW INTRODUCTION BY THE AUTHOR

30th Anniversary edition



GENETIC ALGORITHMS

in Search, Optimization & Machine Learning

DAVID E. GOLDBERG

Special Thanks To:

Brook Graham
Dr. Wes Kerr
Kyle Putnam
Paul Sipe
Jennifer Wadella

Life Finds a Way...