CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

Metaheuristic Optimization

Genetic Algorithms

Dr. Diarmuid Grimes

Assignment 1: Due November 8th

- 2 parts
  - NP-Completeness (20%) – Very similar to lab
  - GA (80%) – Will be given code outline so will just need to implement some functions such as crossover and mutation operators, then run the code to generate results, discuss the results and interpret why
- Submit pdf through turnitin (one time submission) – Note Academic Integrity section!
- Submit code separately – Note Academic Integrity section!
  - Random seed must be set to student id so results can be reproduced
- Rules for which formula F, which set of clauses, which set of problem instances you present results on.
  - Example: Student R012345654 Diarmuid Grimes
  - Choose formula F for students with id ending in 4, choose set of clauses for students whose first name is in the range A-I, choose set of problem instances for students whose surname is in the range A-I

Mutation vs Crossover

From Talbi book (Pg. 221):

"A large mutation probability will disrupt a given individual and the search is more likely random. Generally, small values are recommended for the mutation probability (pm ∈ [0.001, 0.01]). Usually, the mutation probability is initialized to 1/k where k is the number of decision variables. Hence, on average, only one variable is mutated"

TSP -- Representation

TSP – Matrix representation

- Binary n x n matrix M represents ordering information, where m_{ij}=1 if and only if city i precedes city j

e.g. the tour 1-5-2-6-3-4-1 is represented as

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

TSP - Matrix representation

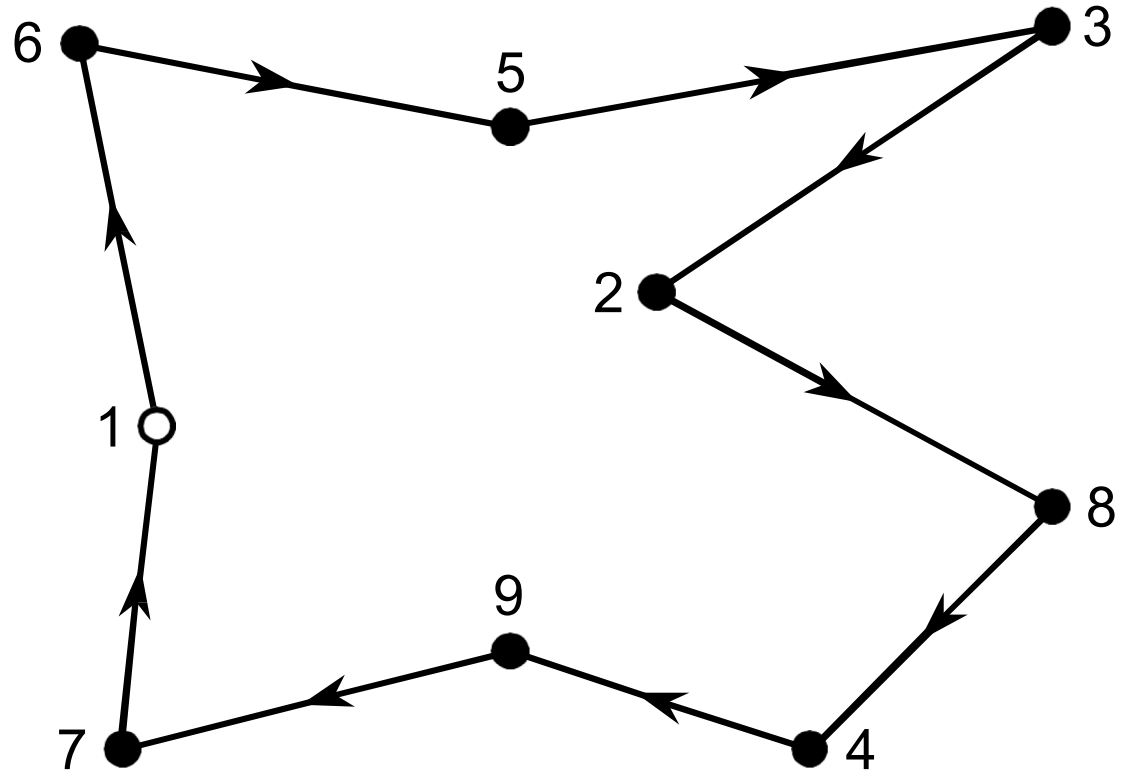
- Binary n x n matrix M represents ordering information, where m_{ij}=1 if and only if city i p

e.g. the tour 1 ented as

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

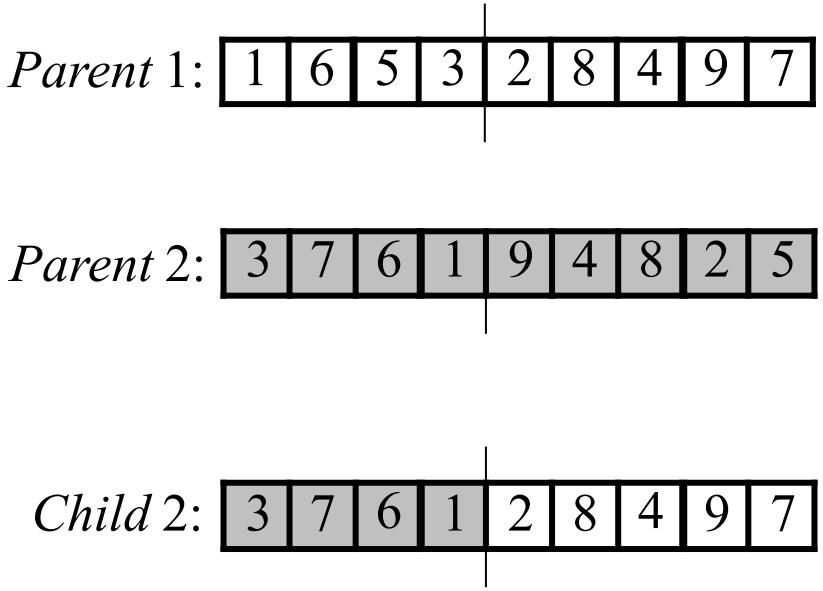
Not very practical due to memory requirements

Traveling Salesman Problem -- TSP



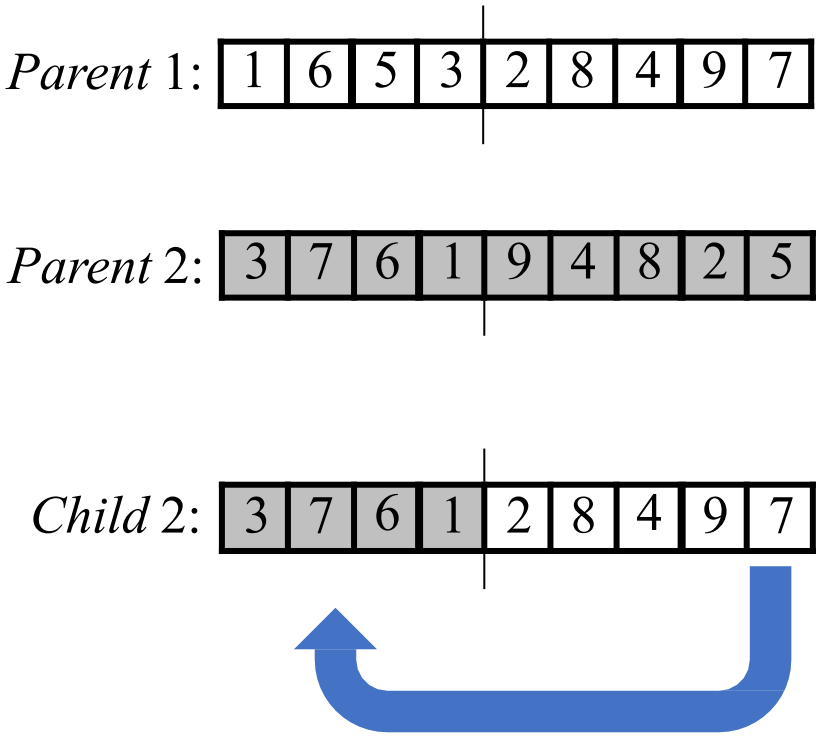
1	6	5	3	2	8	4	9	7

How does the crossover operator work?



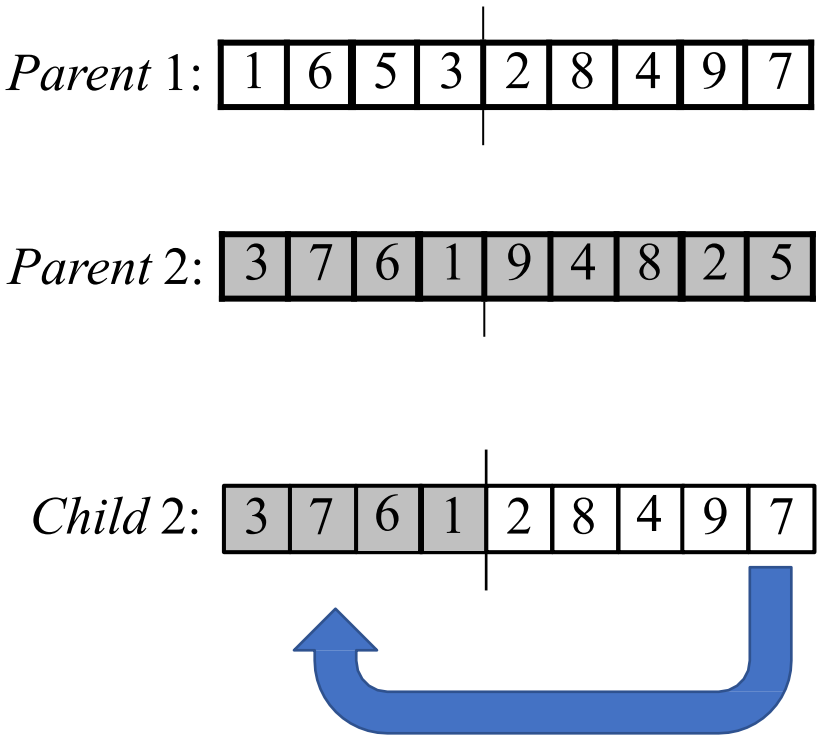
Is this OK?

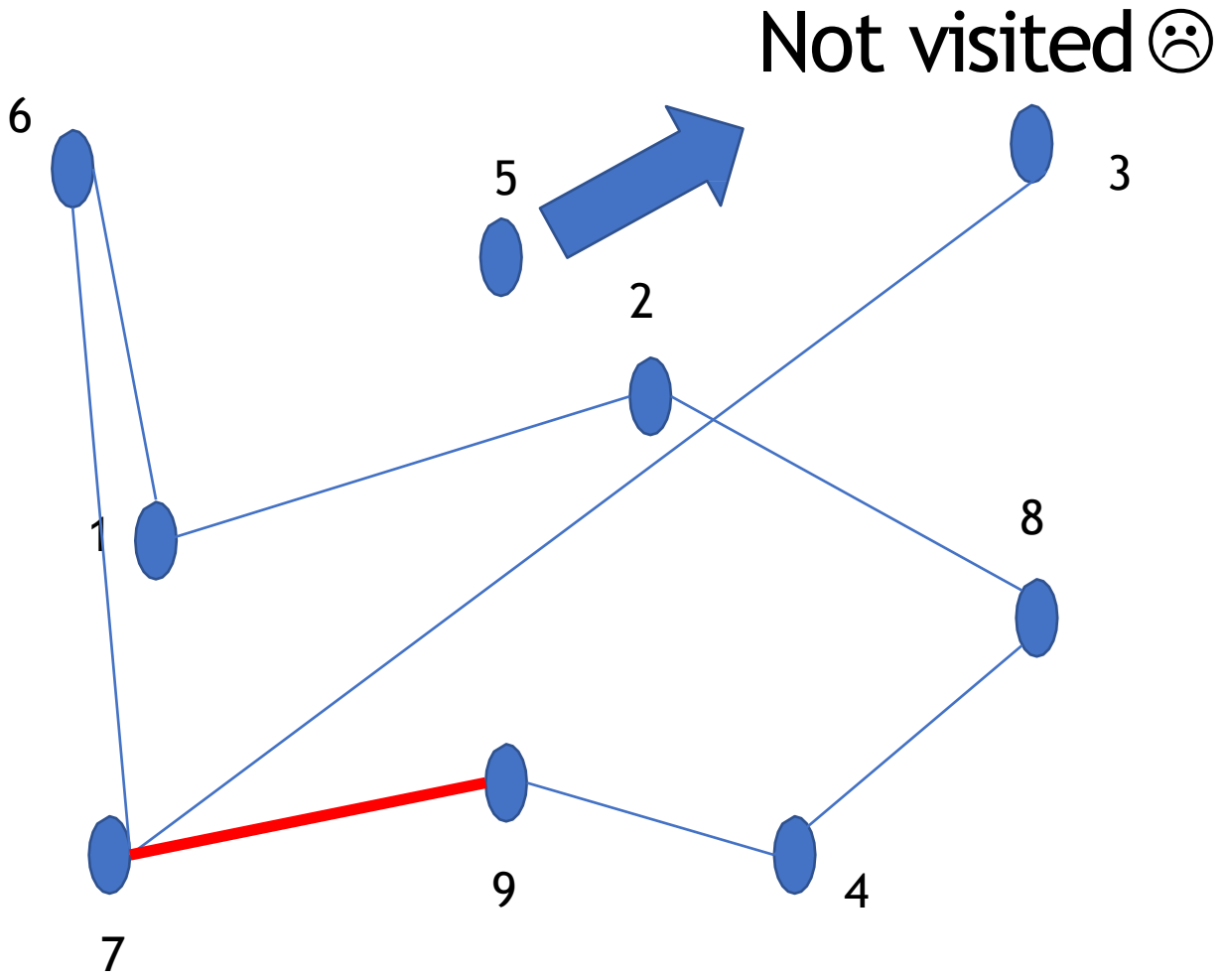
How does the crossover operator work?



One point crossover

How does the crossover operator work?

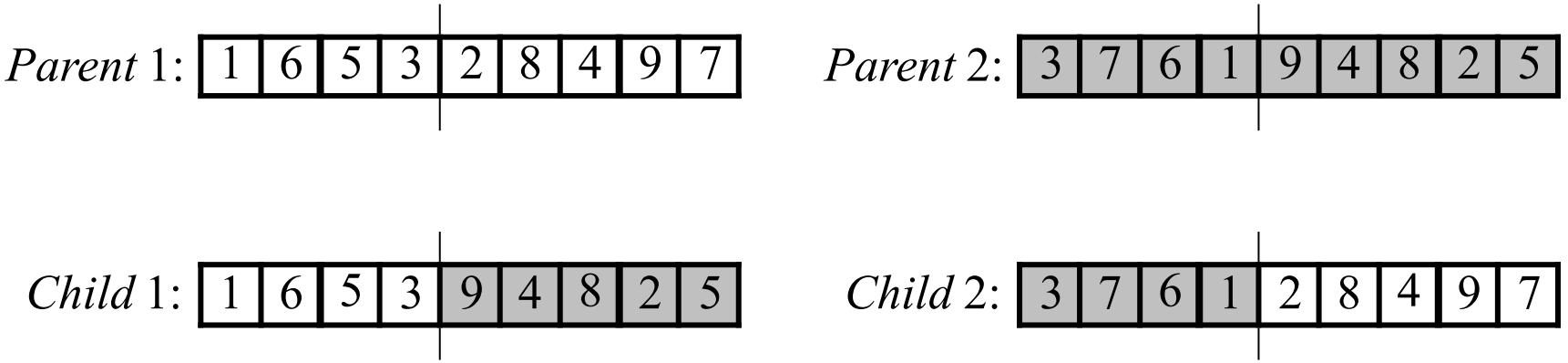




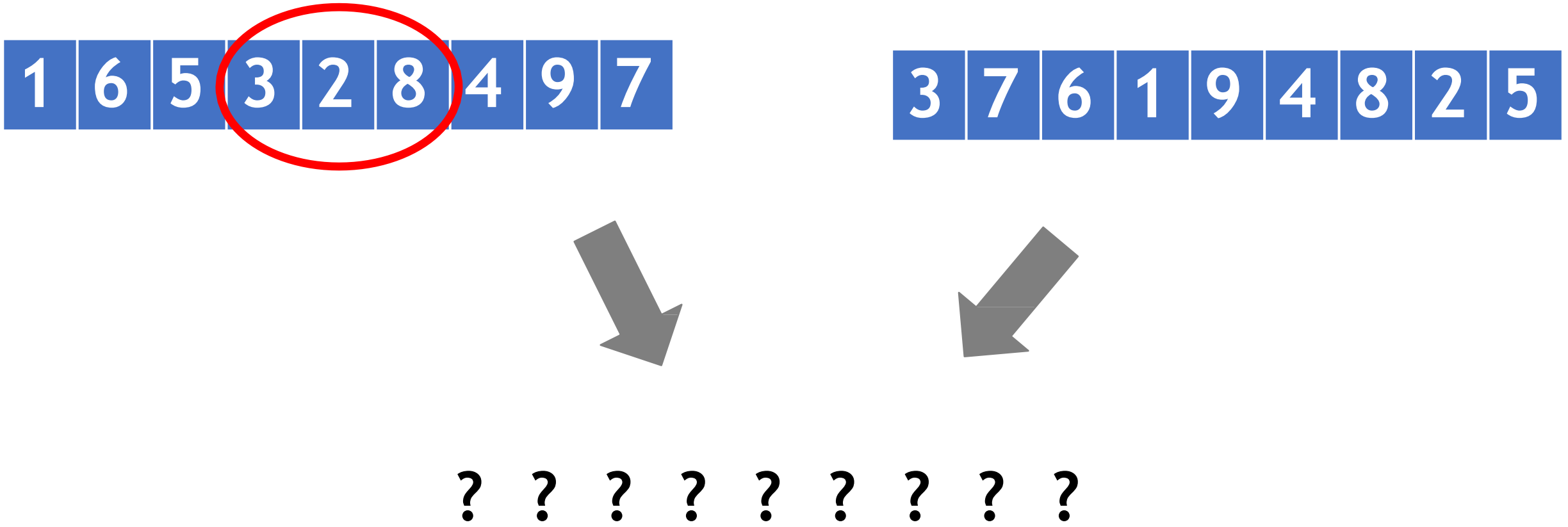
How does the crossover operator work?

The crossover operator in its classical form cannot be directly applied to the TSP.

A simple exchange of parts between parents would produce illegal routes containing duplicates and omissions - some cities would be visited twice while some others would not be visited at all.

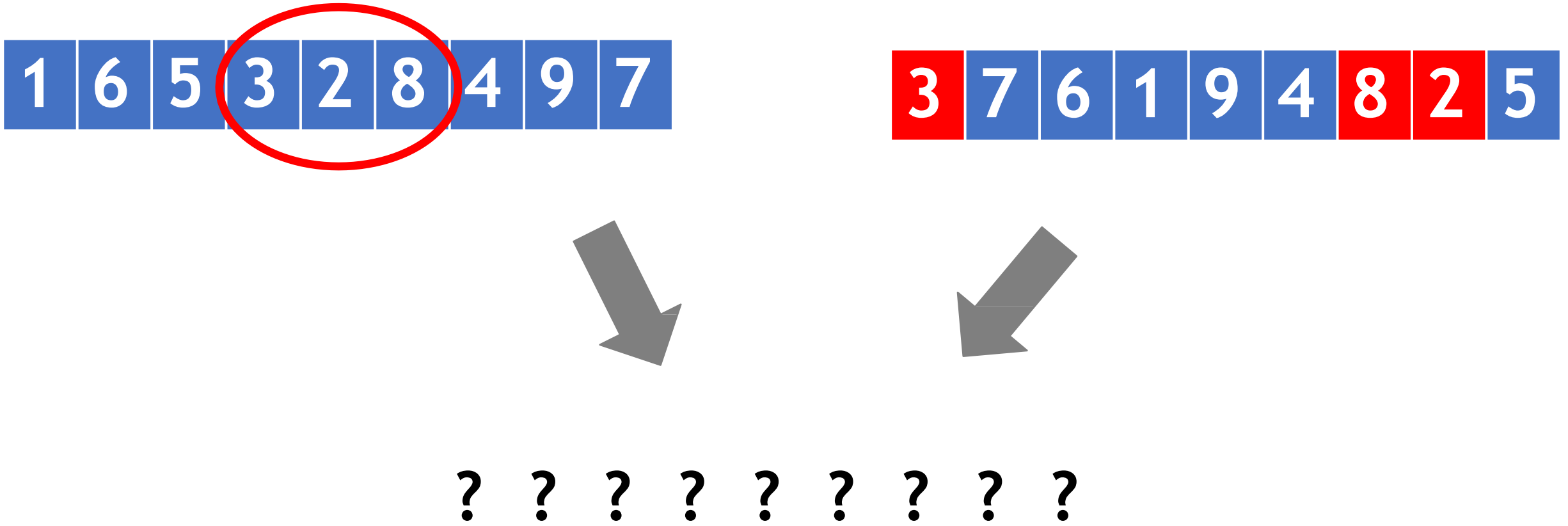


How does the order-1 (permutation) crossover operator work?



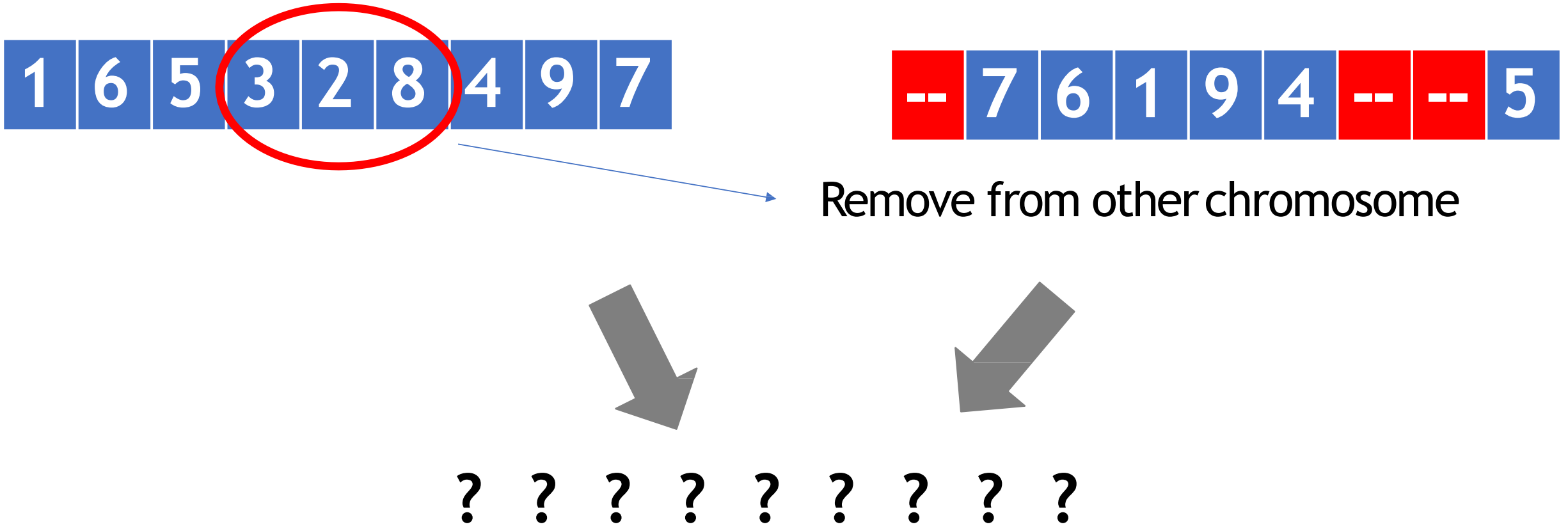
Order 1 crossover

How does the order-1 (permutation) crossover operator work?



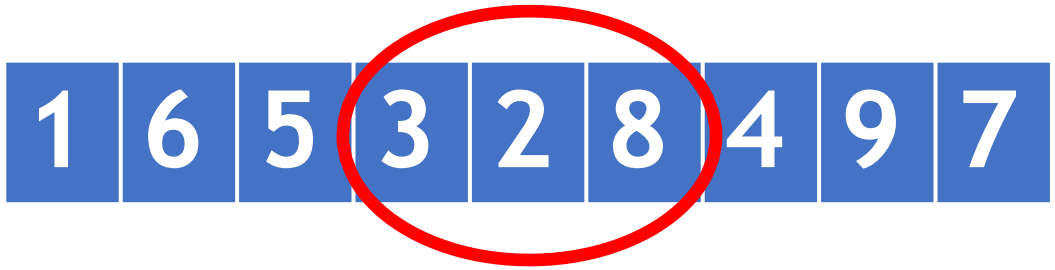
Order 1 crossover

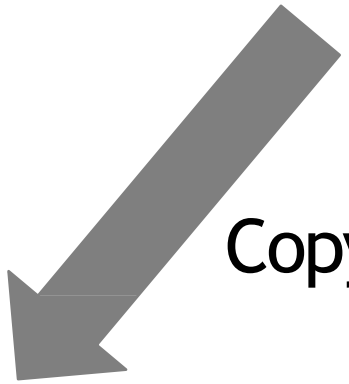
How does the order-1 (permutation) crossover operator work?



Order 1 crossover

How does the crossover operator work?



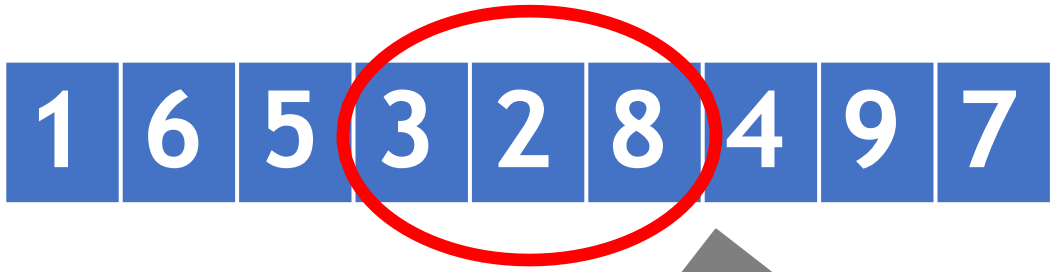


Copy the current tour in the child

7 6 1 9 4 5 ? ? ?

Order 1 crossover

How does the crossover operator work?



We can also change the order

Add selected sequence in our new child

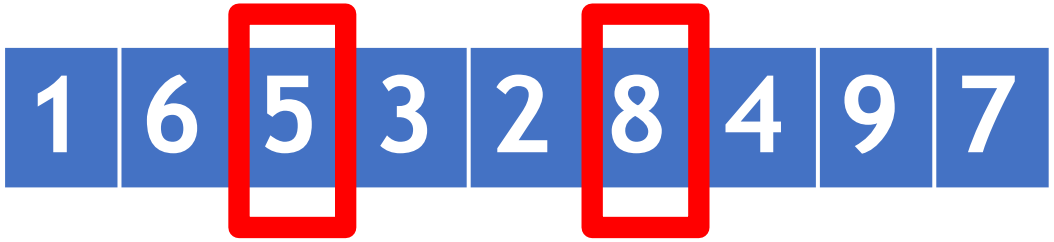
7 6 1 9 4 5 3 2 8

Order 1 crossover

How do mutation operators work for permutation problems?

- Can’t just take one value and replace it with a random value
- Number of possible mutation operators for permutation problems, e.g. **reciprocal exchange** and **inversion**
- The **reciprocal exchange** operator simply swaps two random selected cities in the chromosome
- The **inversion operator** selects two random points along the chromosome string and reverses the order of the cities between these two points

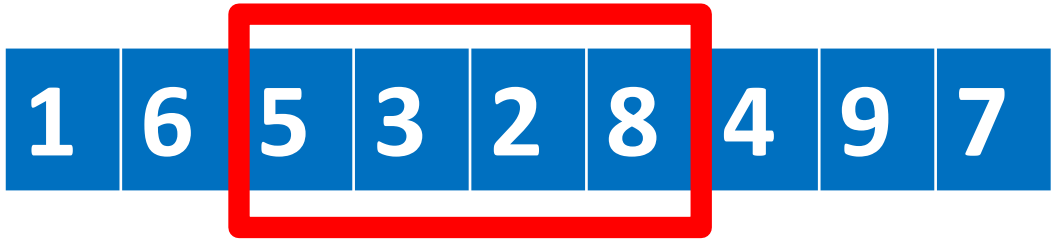
How does the mutation operator work?



Reciprocal Exchange

1 6 8 3 2 5 4 9 7

How does the mutation operator work?

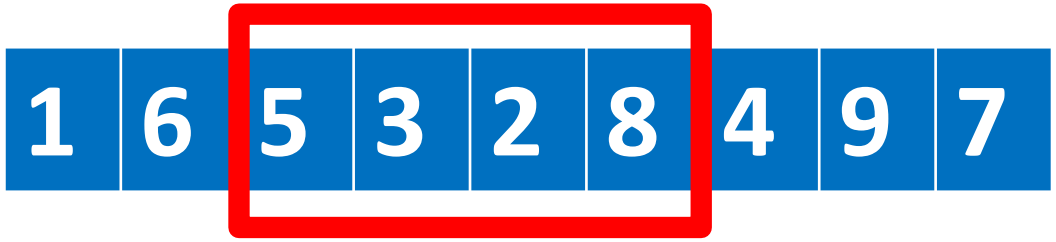


Inversion Operator

1 6 8 2 3 5 4 9 7

Alternative: Scramble Mutation

- Shuffles the elements between the selected locations



Scramble Mutation

1 6 2 5 8 3 4 9 7

More Crossover Operators

Permutation problems (e.g., TSP)

Uniform order-based crossover (TSP)

- Randomly select positions that will no longer change. The remaining positions are filled with the missing elements copied in the order they appear in the other parent
- Genes between cuts are copied to the children
  - Starting from the second crossover site, copy the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear

Also known as Permutation Encoding – Order Crossover (OX)

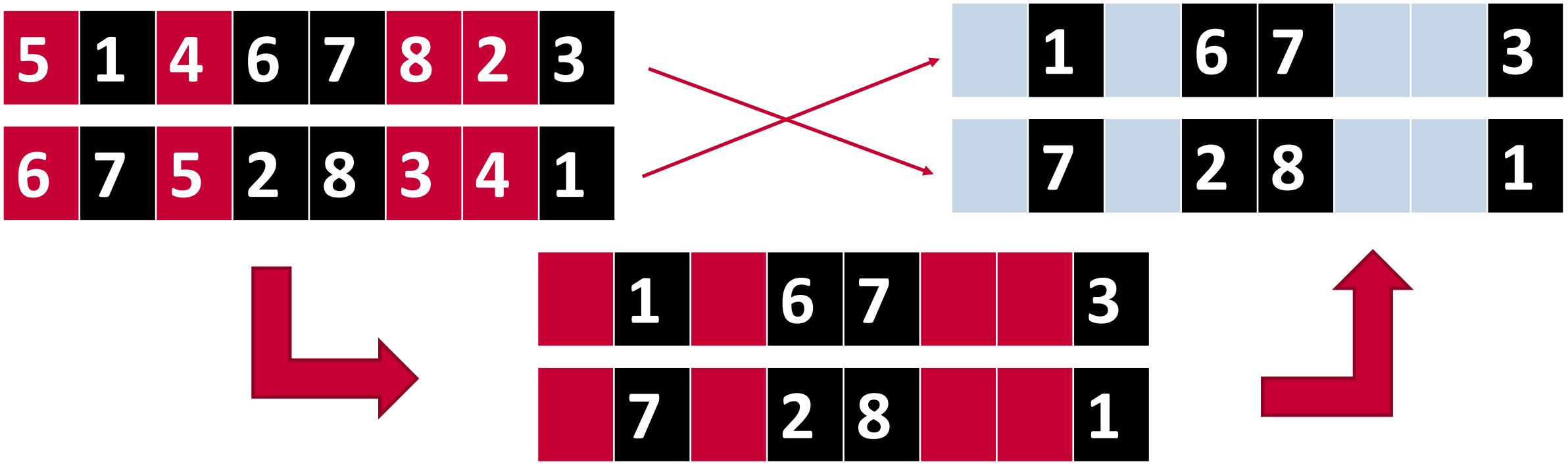
Uniform order-based crossover (TSP)

- Randomly select positions that will no longer change. The remaining positions are filled with the missing elements copied in the order they appear in the other parent

5	1	4	6	7	8	2	3
6	7	5	2	8	3	4	1

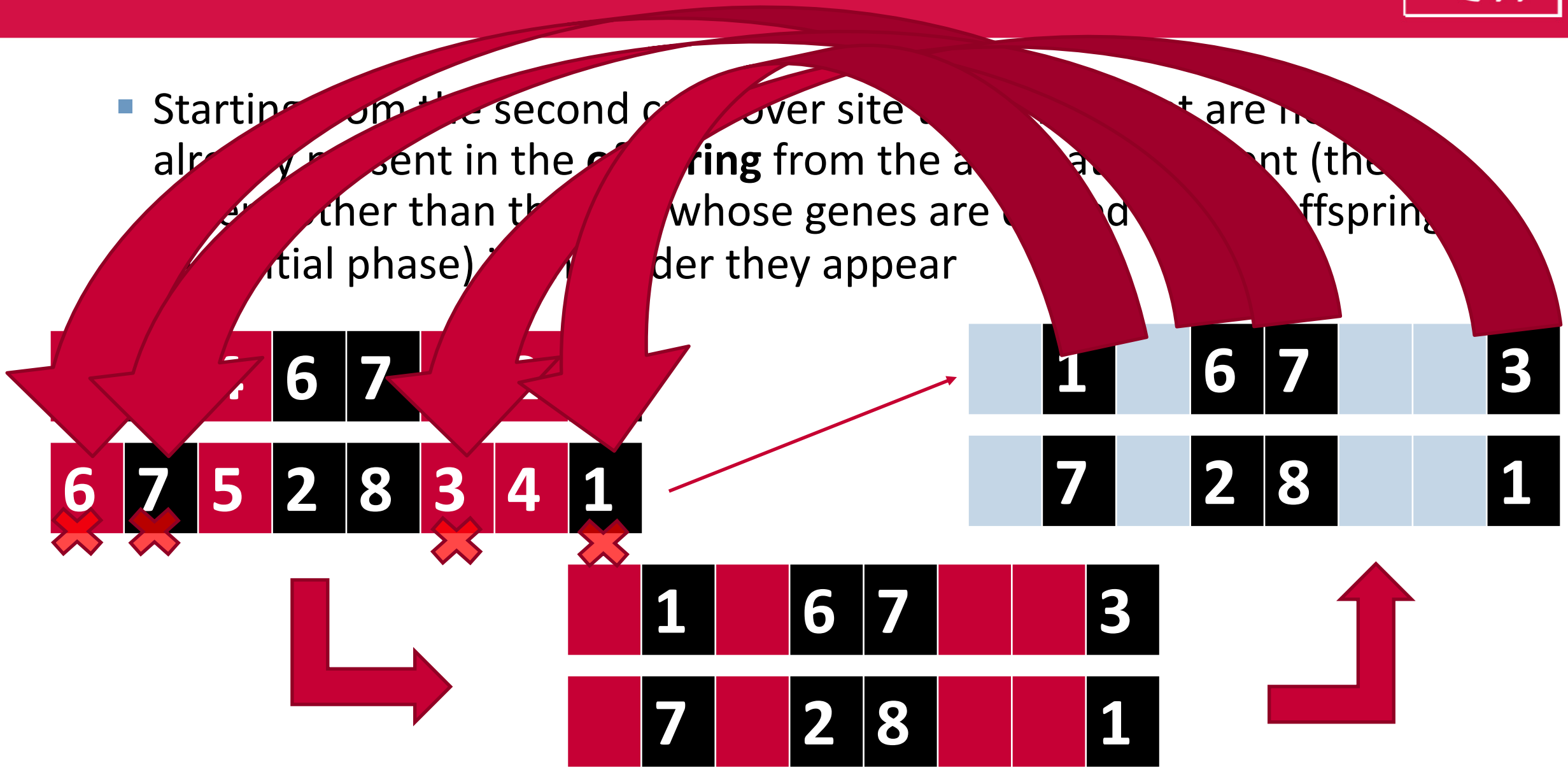
Uniform order-based crossover (TSP)

- Starting from the second crossover site the genes that are not already present in the **offspring** from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear



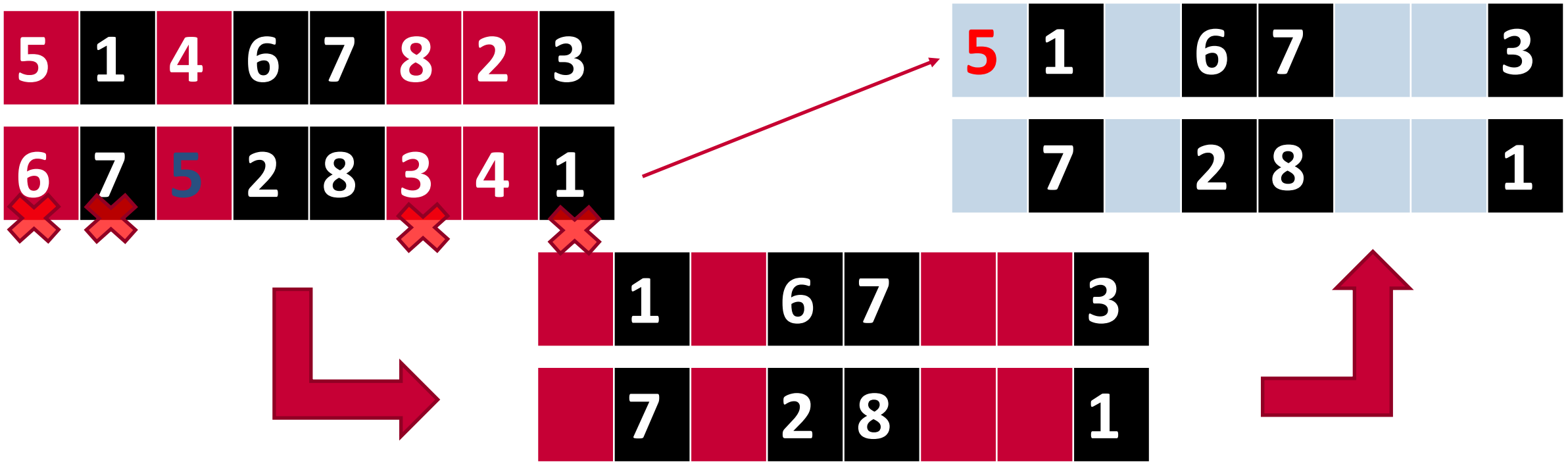
Uniform order-based crossover (TSP)

- Starting from the second crossover site, genes are removed if they are already present in the **offspring** from the alternate parent (the parent other than the one whose genes are being copied to the offspring in the initial phase) in the order they appear



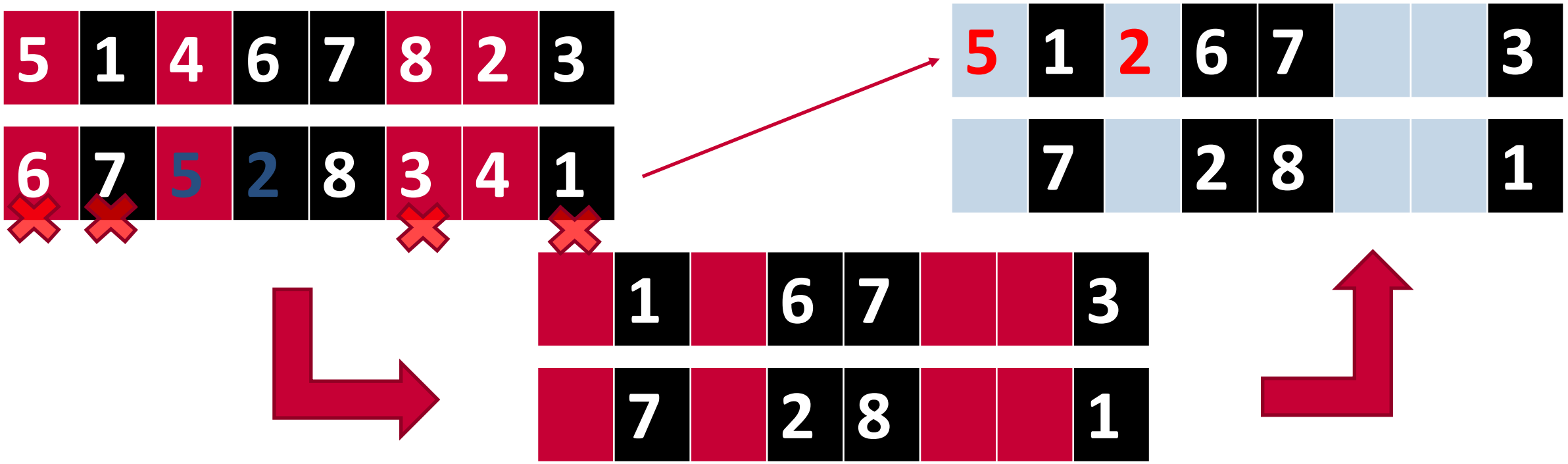
Uniform order-based crossover (TSP)

- Starting from the second crossover site the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear



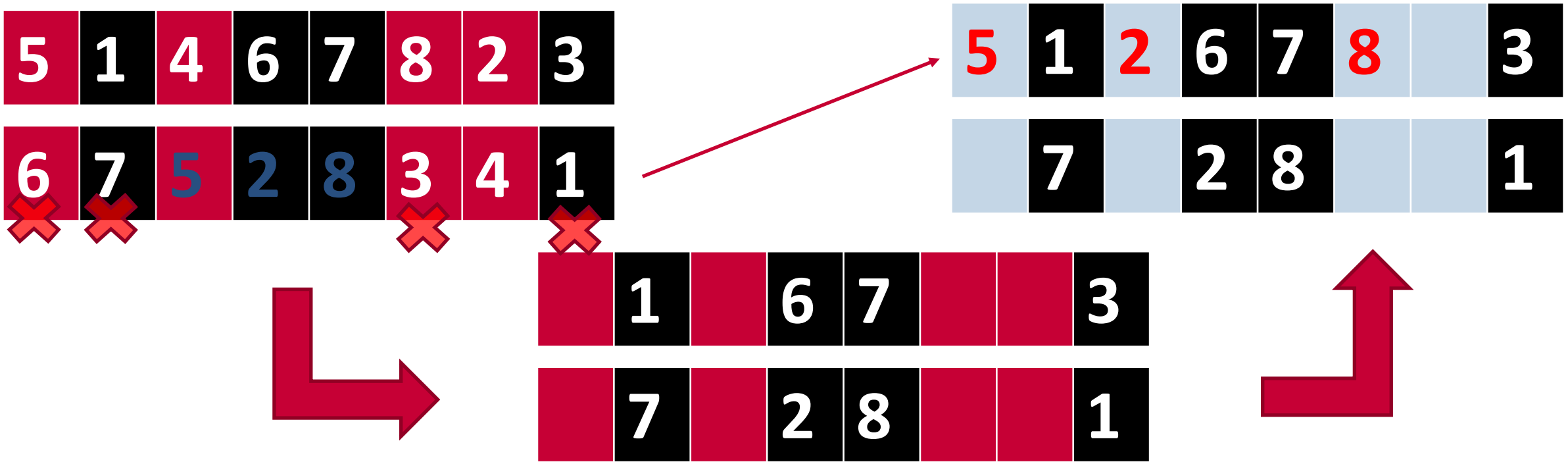
Uniform order-based crossover (TSP)

- Starting from the second crossover site the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear



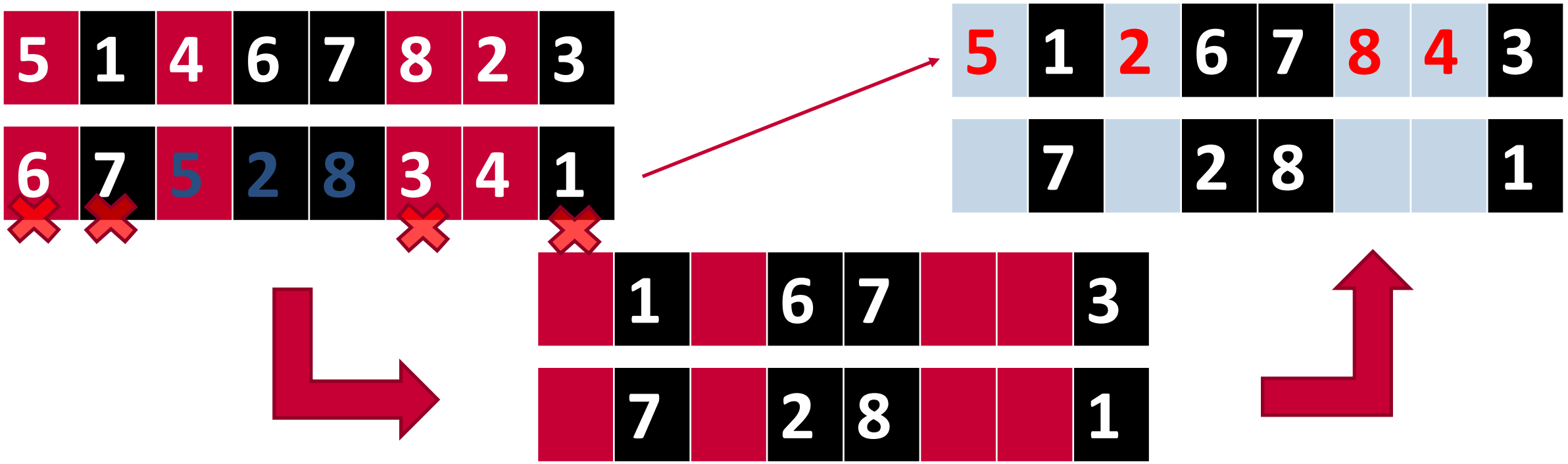
Uniform order-based crossover (TSP)

- Starting from the second crossover site the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear



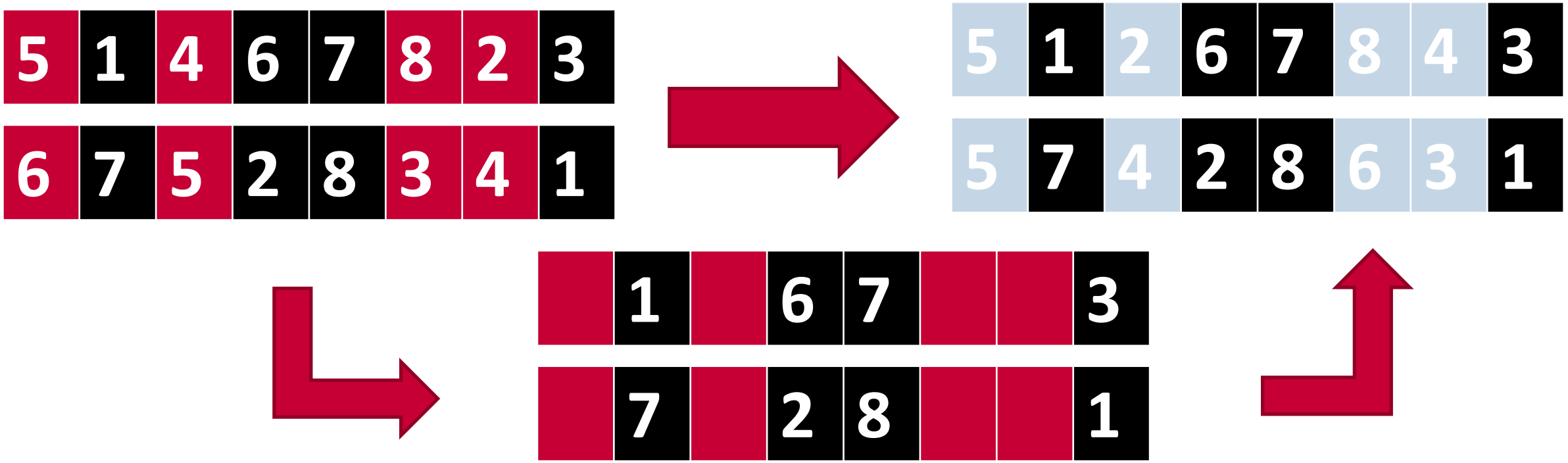
Uniform order-based crossover (TSP)

- Starting from the second crossover site the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear



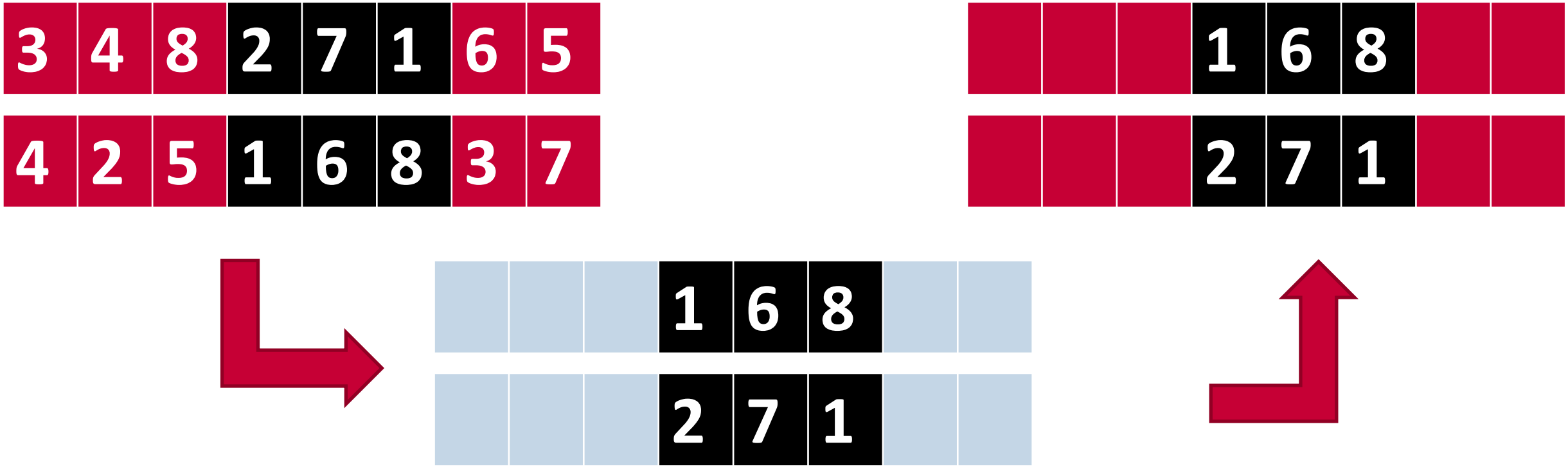
Uniform order-based crossover (TSP)

- Starting from the second crossover site the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear

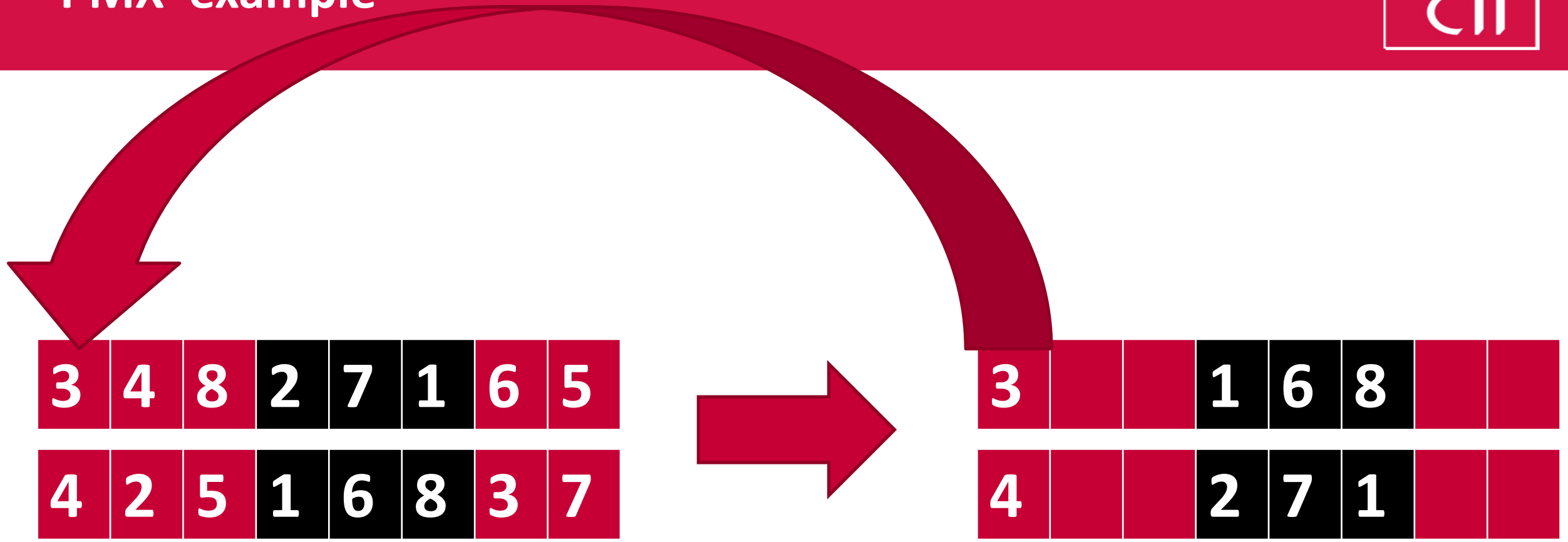


Partially Mapped Crossover (PMX)

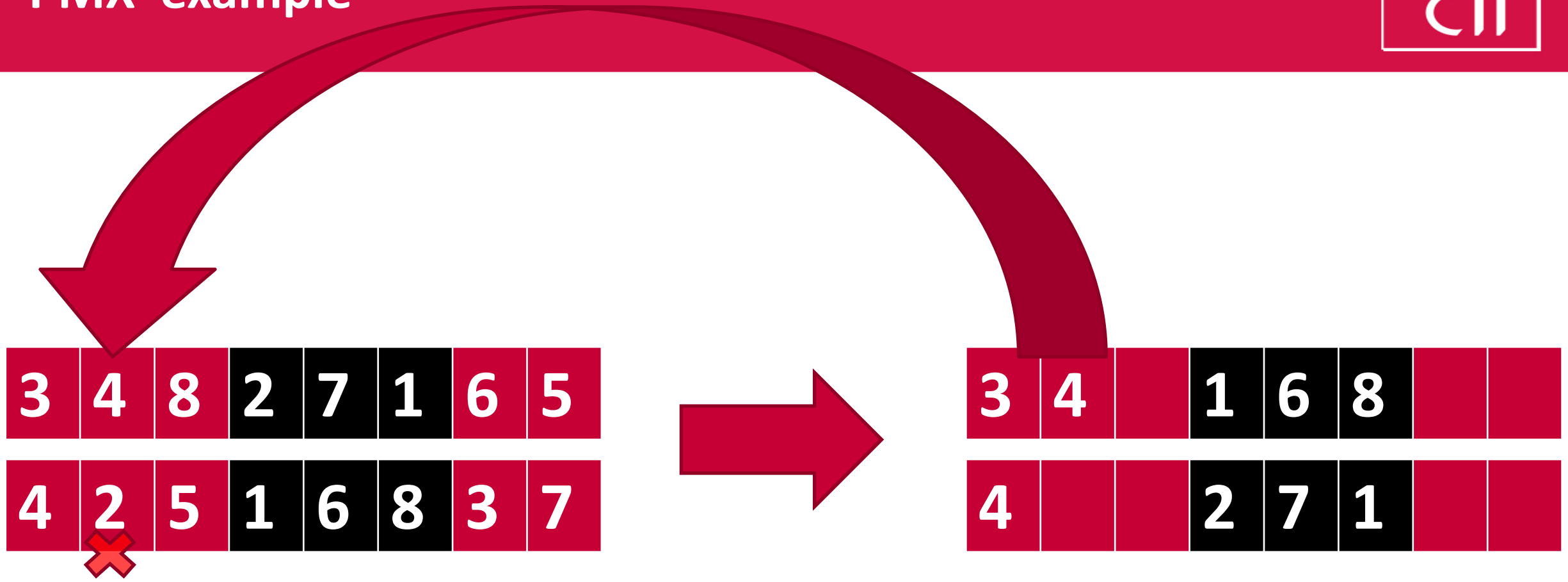
PMX example



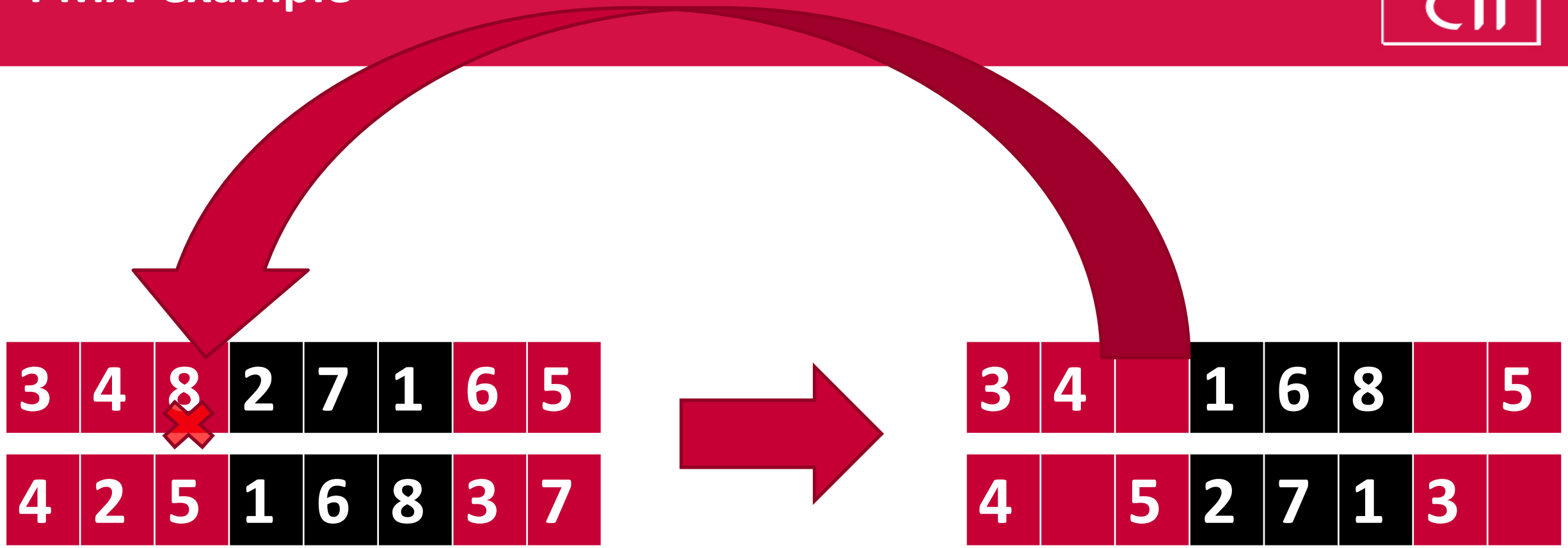
PMX example



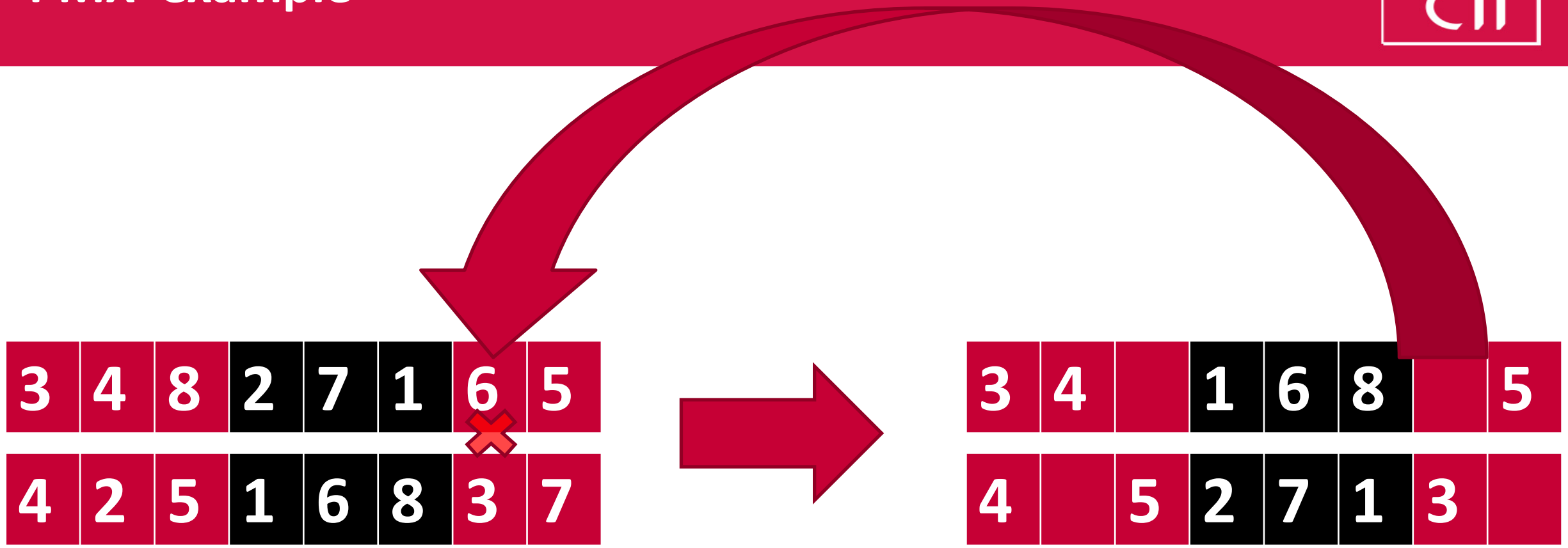
PMX example



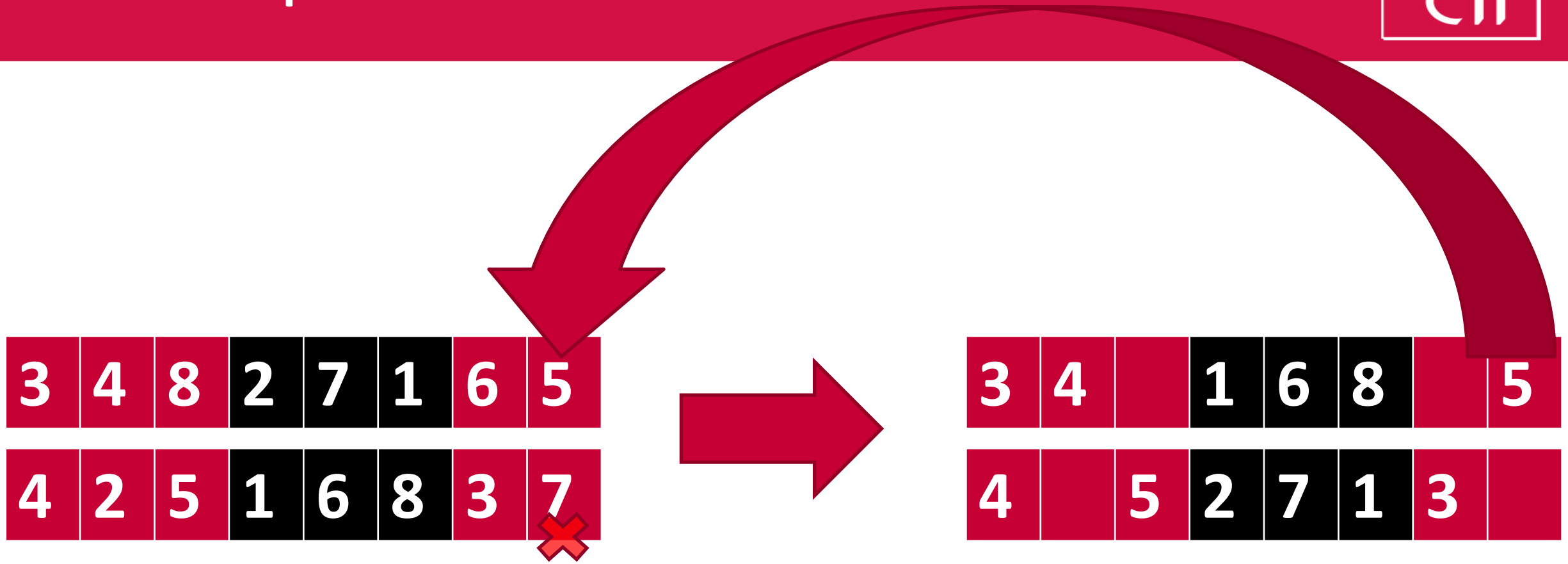
PMX example



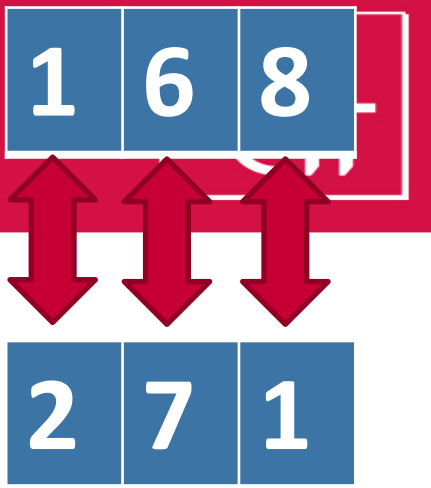
PMX example



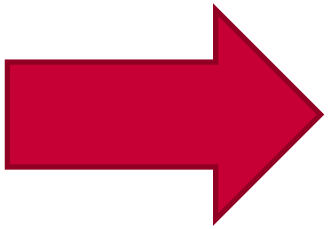
PMX example



PMX example

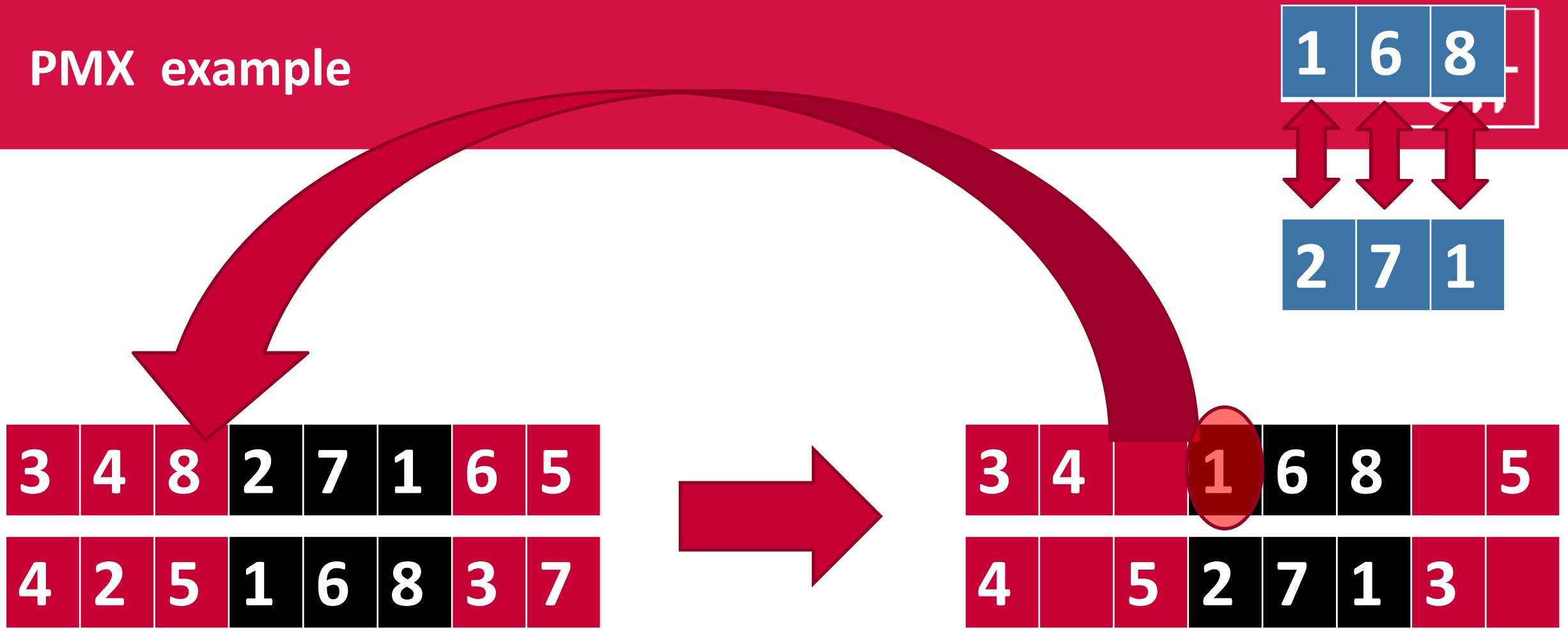


3	4	8	2	7	1	6	5
4	2	5	1	6	8	3	7



3	4		1	6	8		5
4		5	2	7	1	3	

PMX example



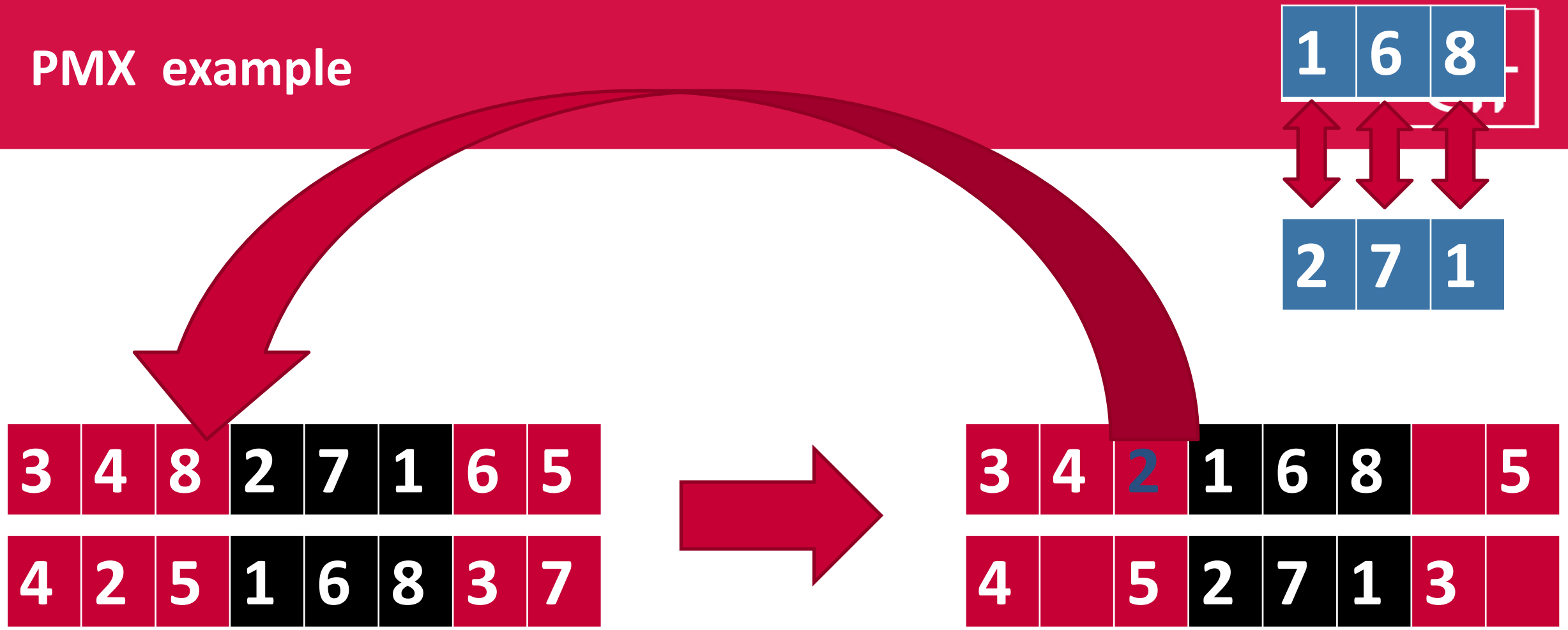
8 ⟷ 1

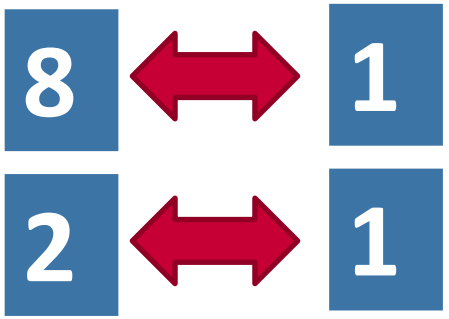
2 ⟷ 1

1 is already in the offspring ☹

2 occupies this location

PMX example

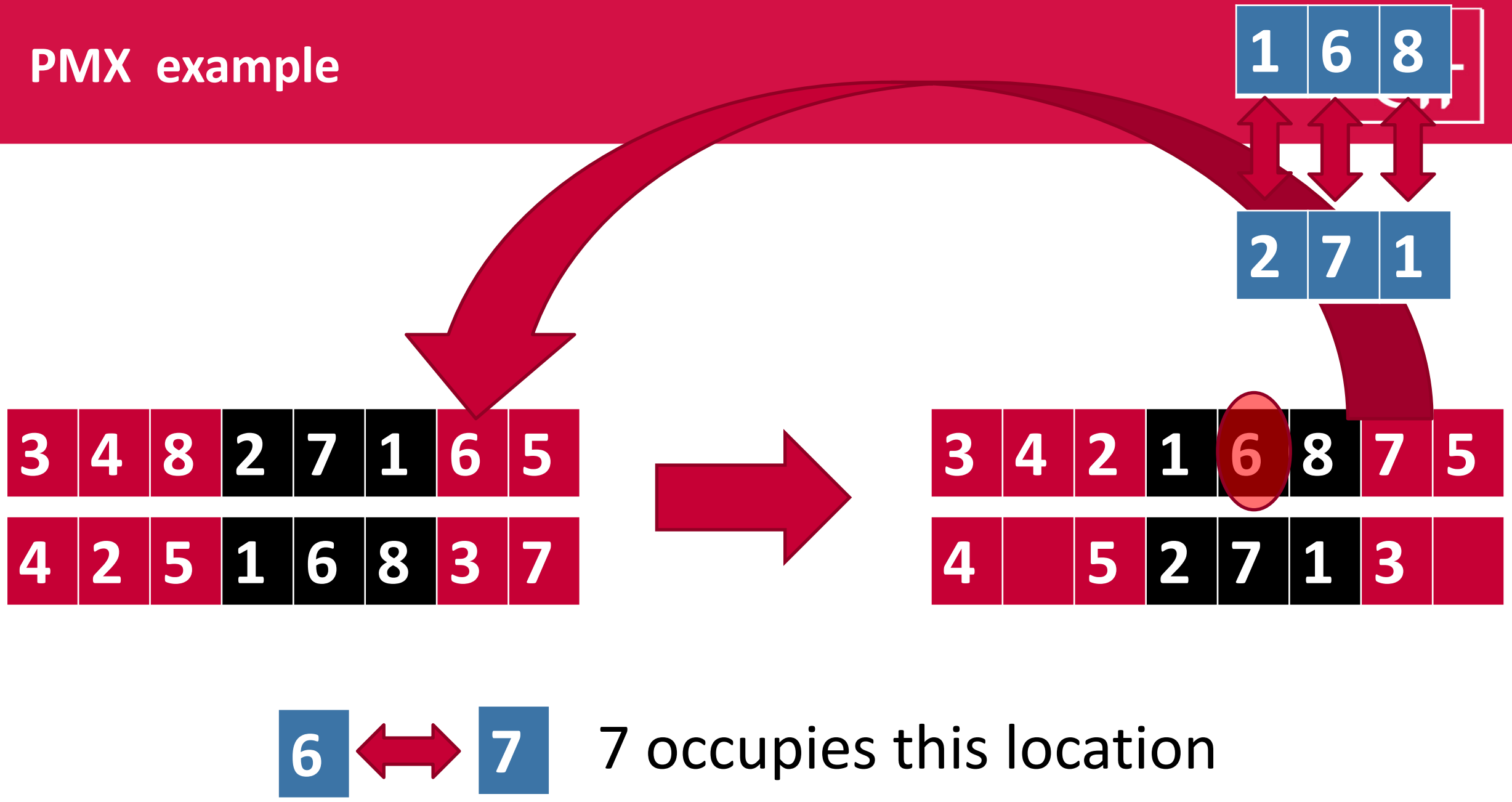




1 is already in the offspring ☹

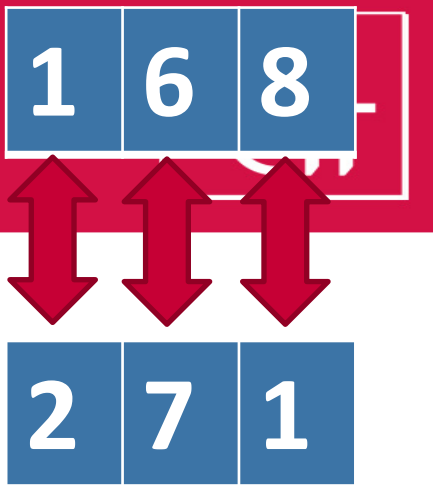
2 occupies this location

PMX example

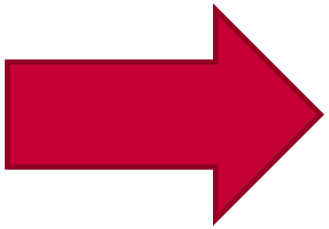


6 ⟷ 7 7 occupies this location

PMX example



3	4	8	2	7	1	6	5
4	2	5	1	6	8	3	7



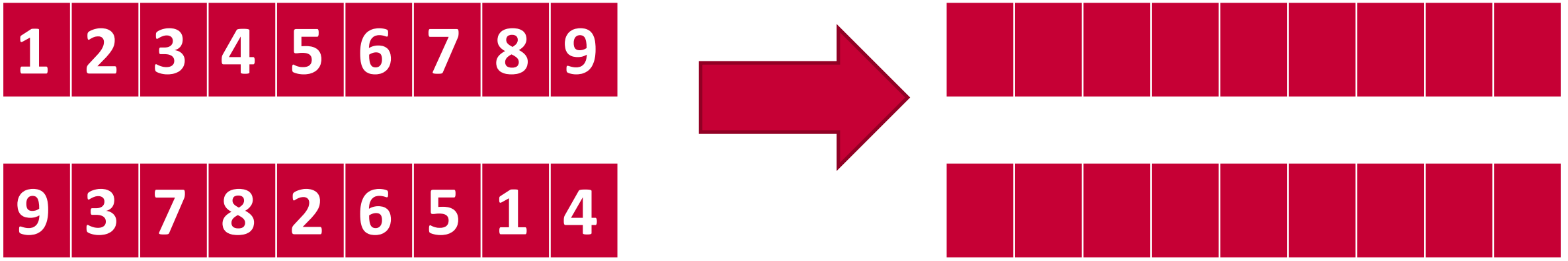
3	4	2	1	6	8	7	5
4	8	5	2	7	1	3	6

Similarly, we complete the second offspring

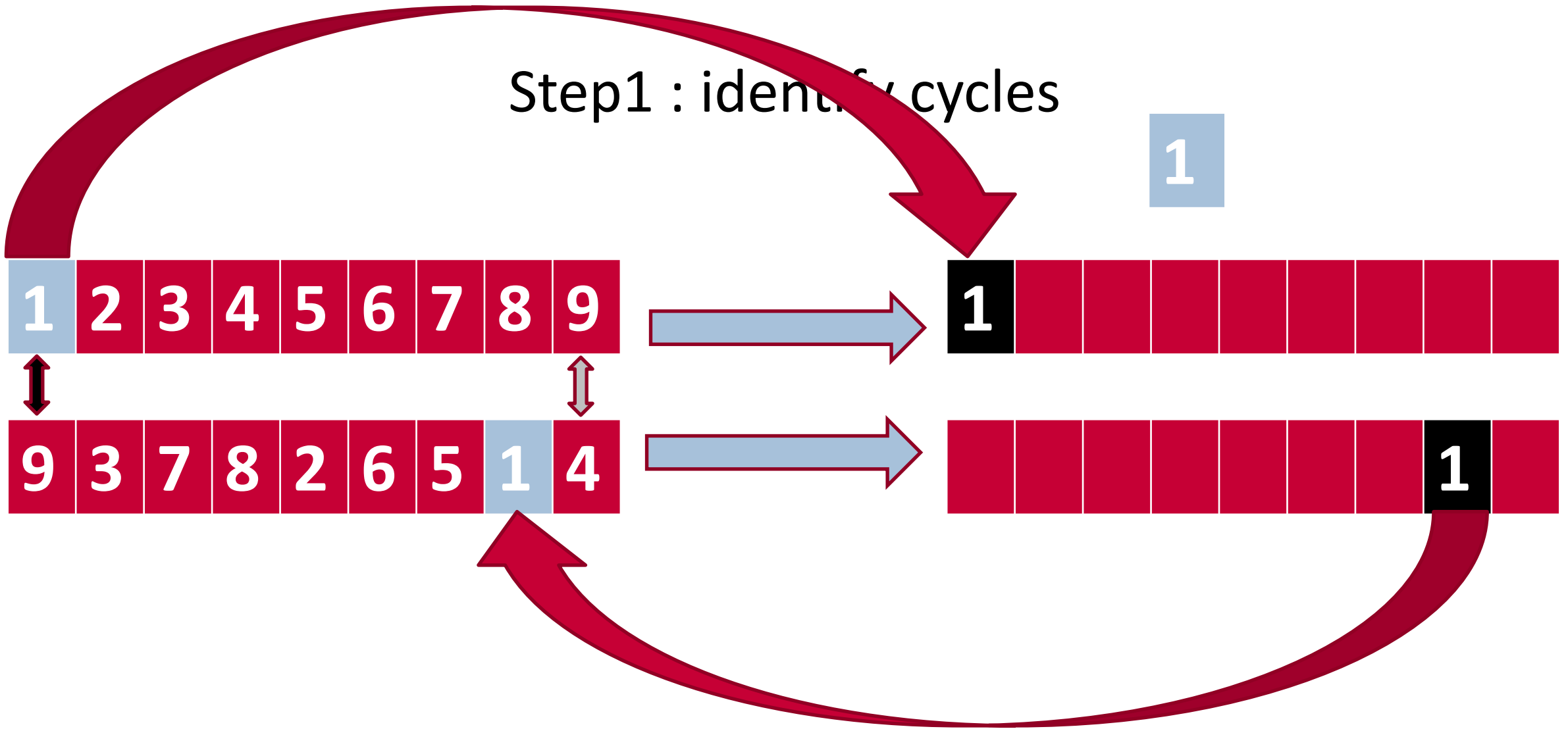
Cycle Crossover

Cycle Crossover

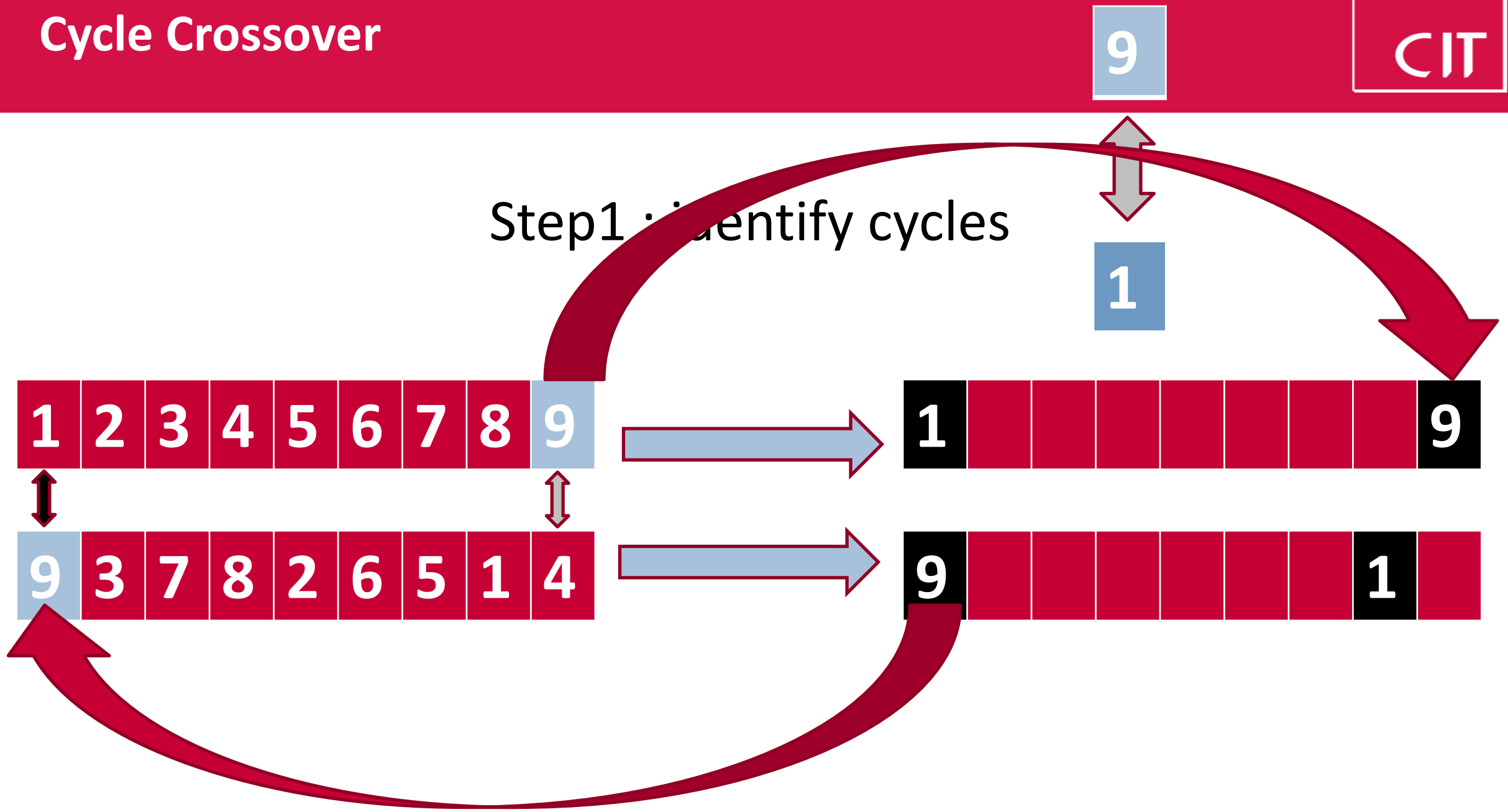
Step1 : identify cycles



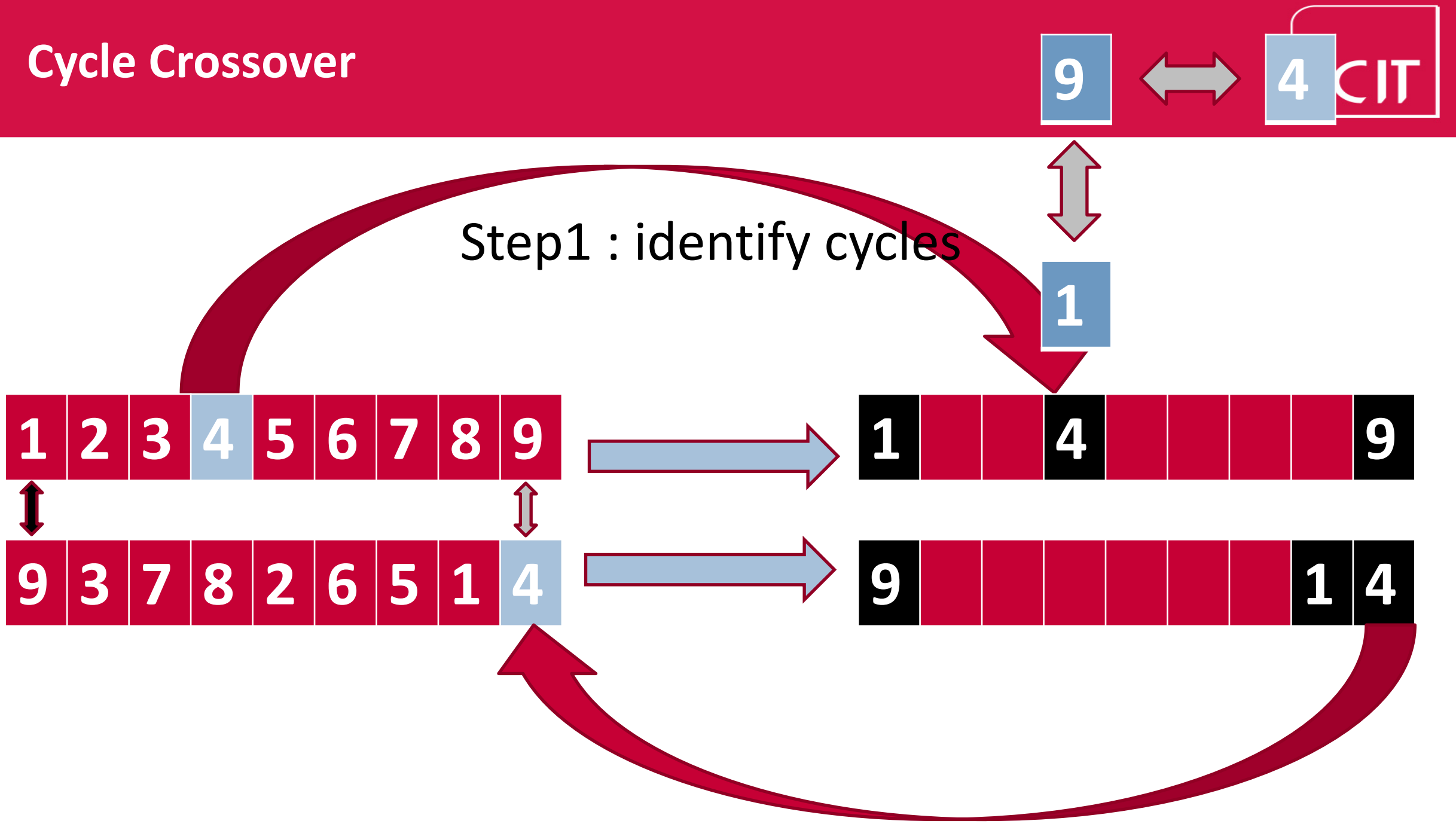
Cycle Crossover



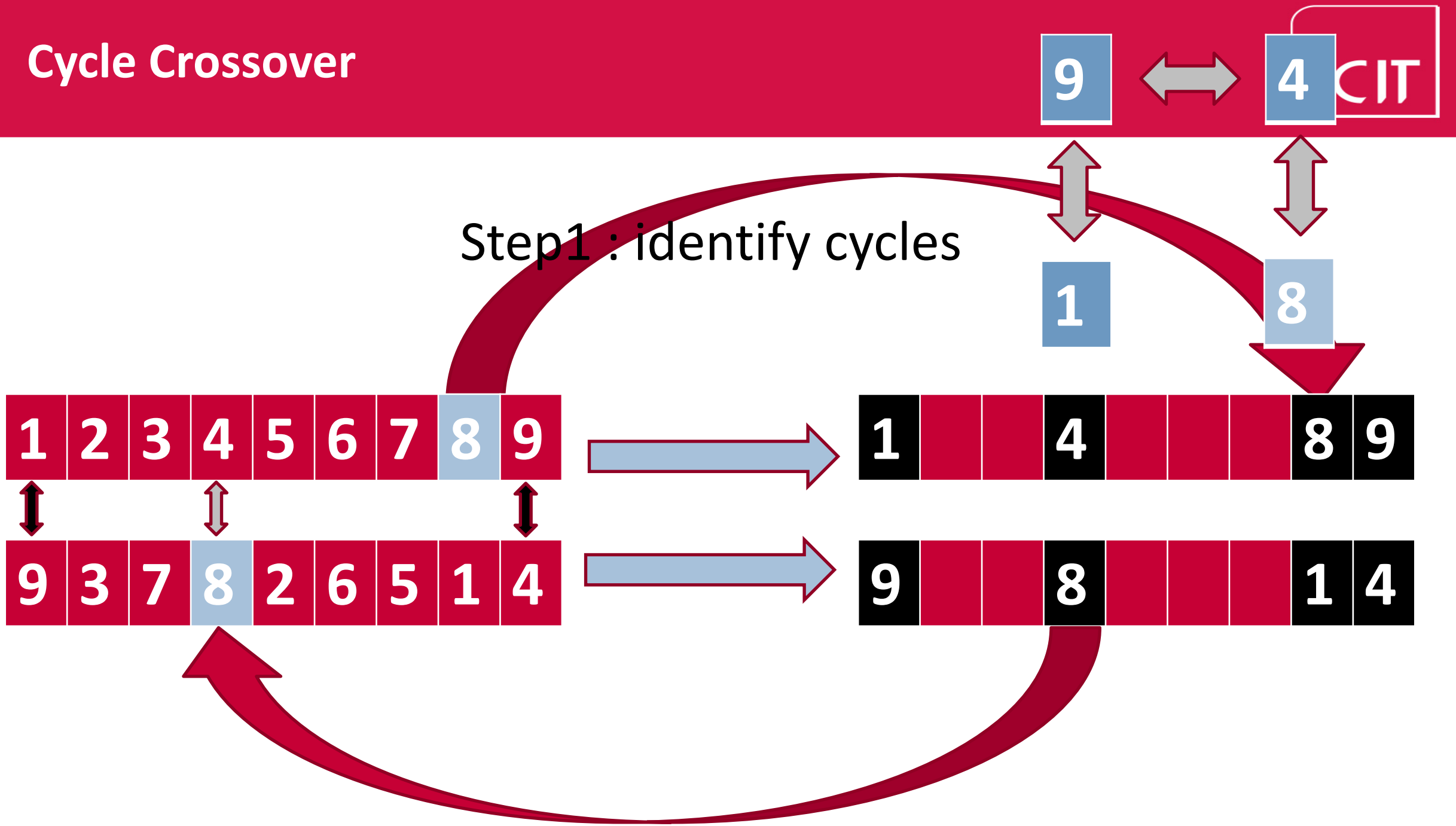
Cycle Crossover



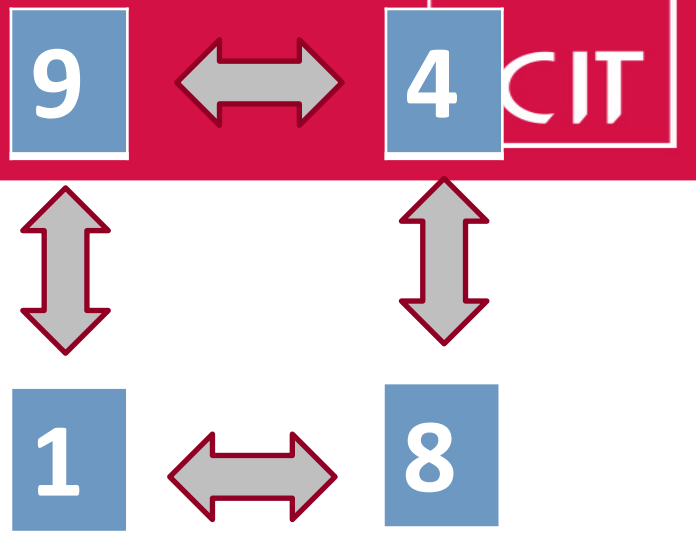
Cycle Crossover



Cycle Crossover

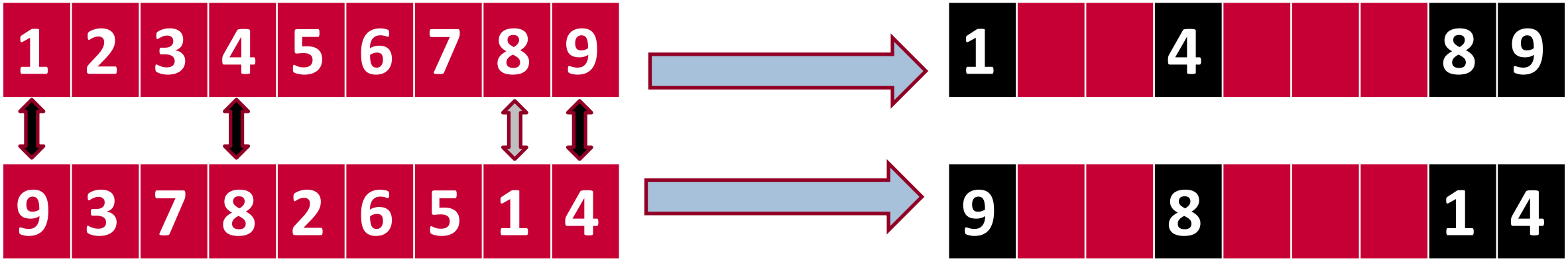


Cycle Crossover



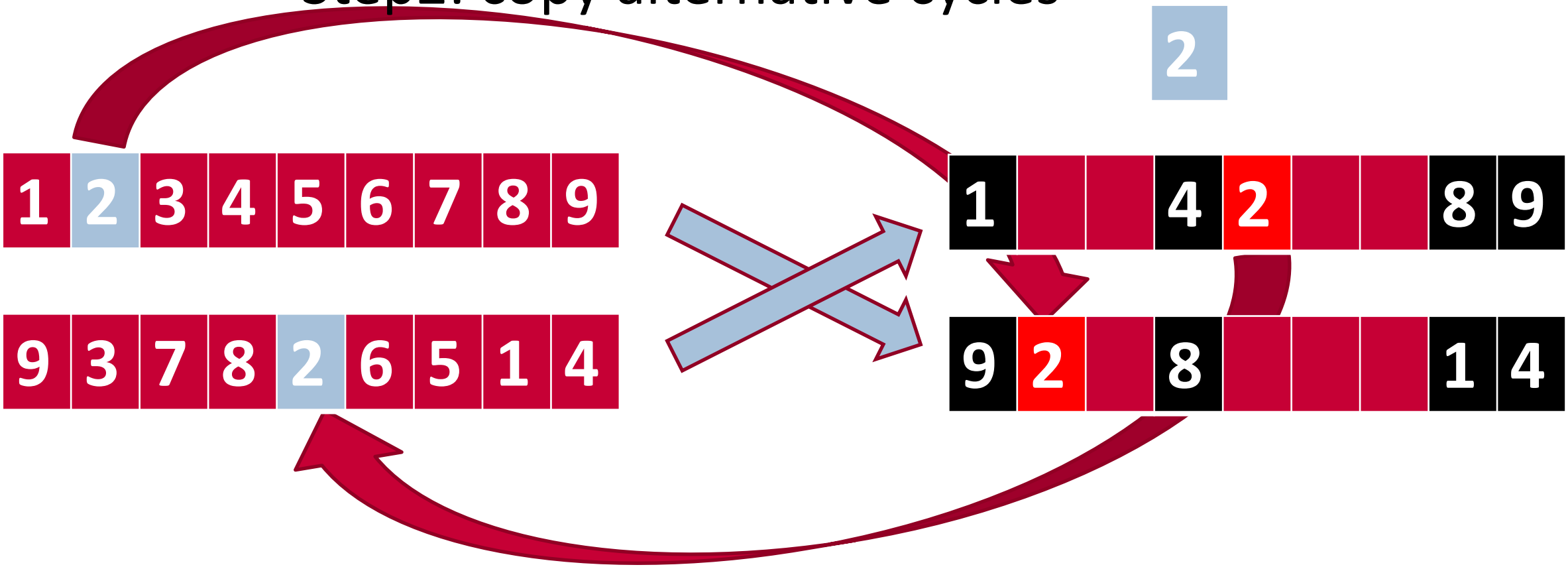
Step1 : identify cycles

End of cycle

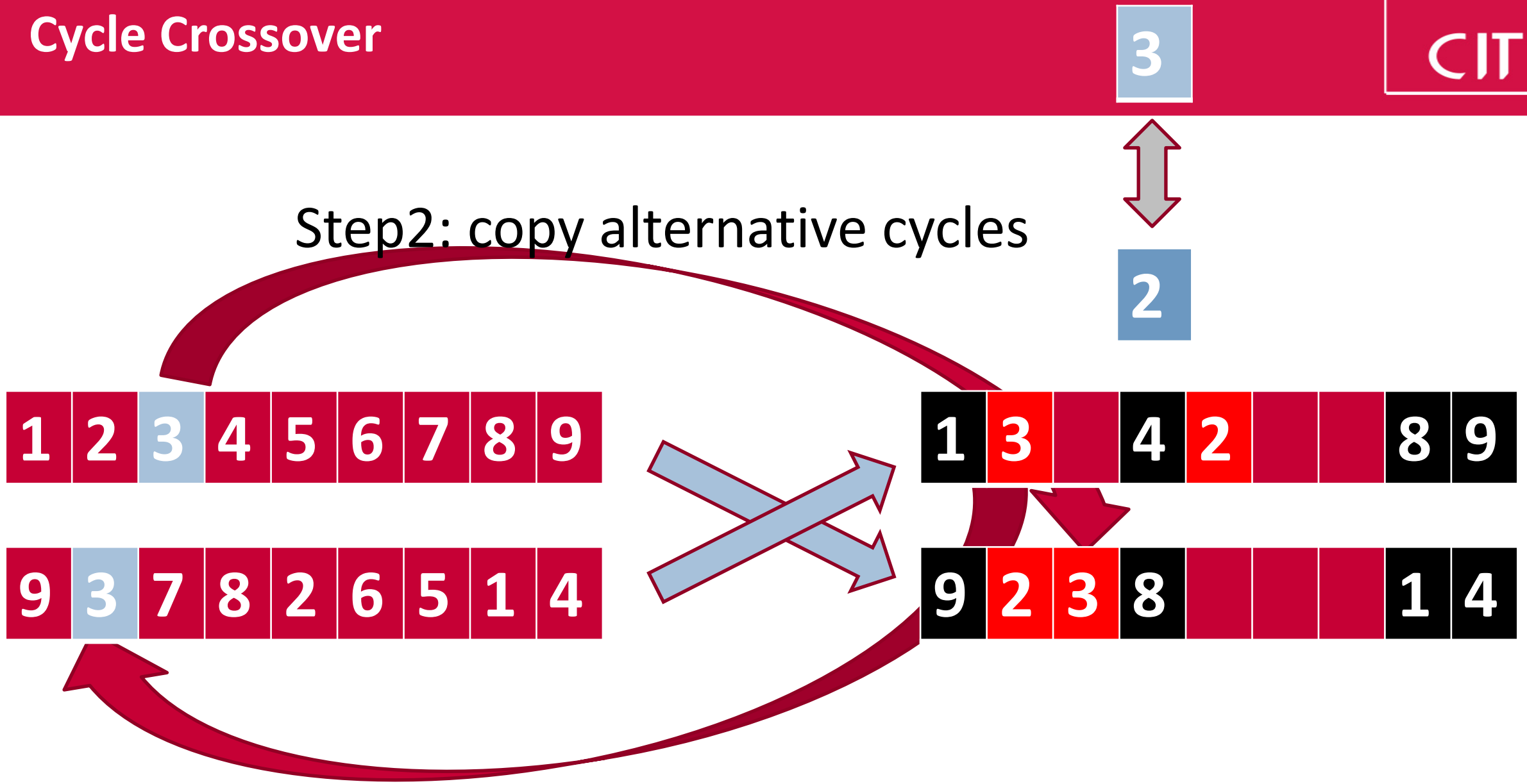


Cycle Crossover

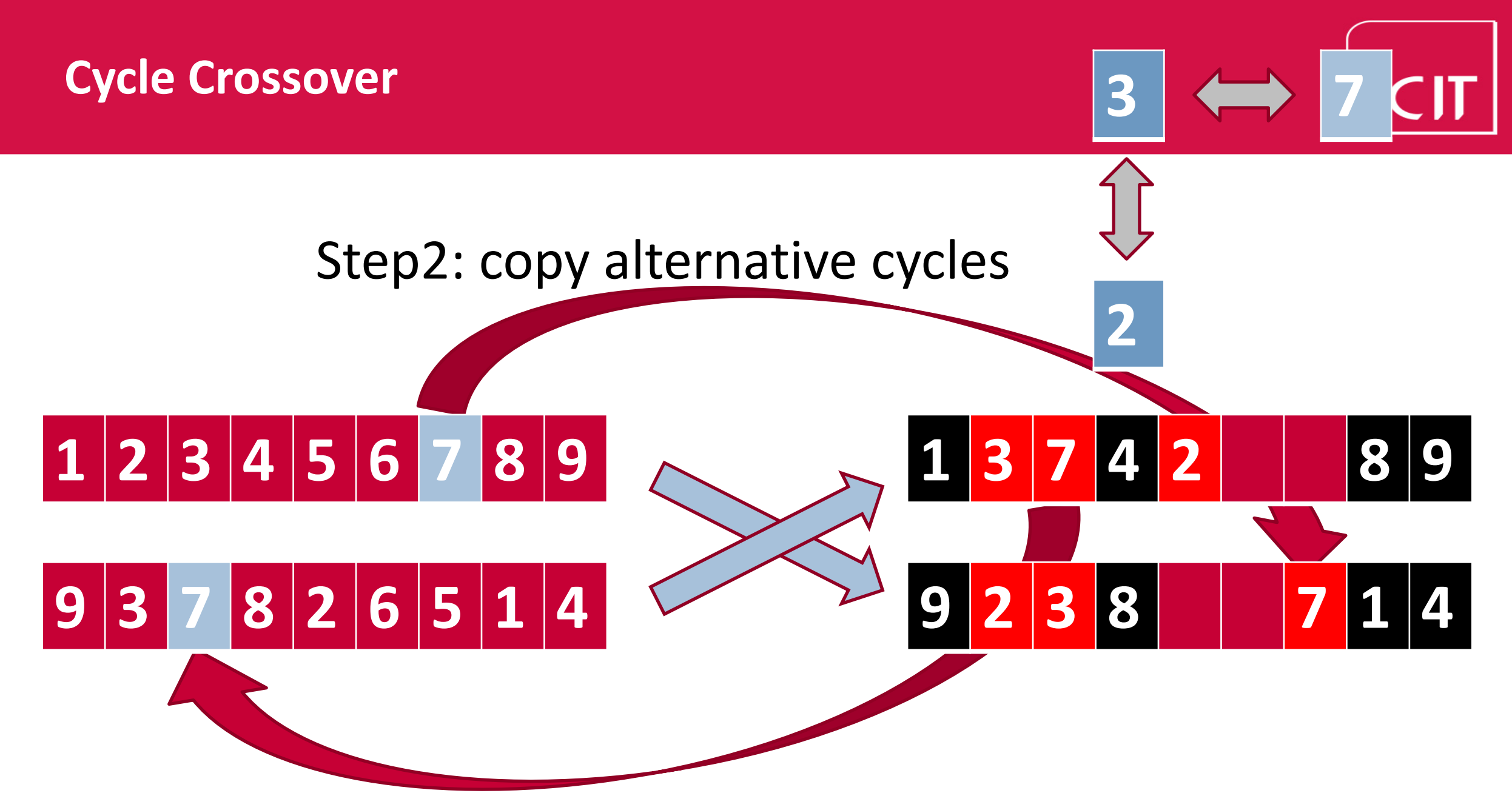
Step2: copy alternative cycles



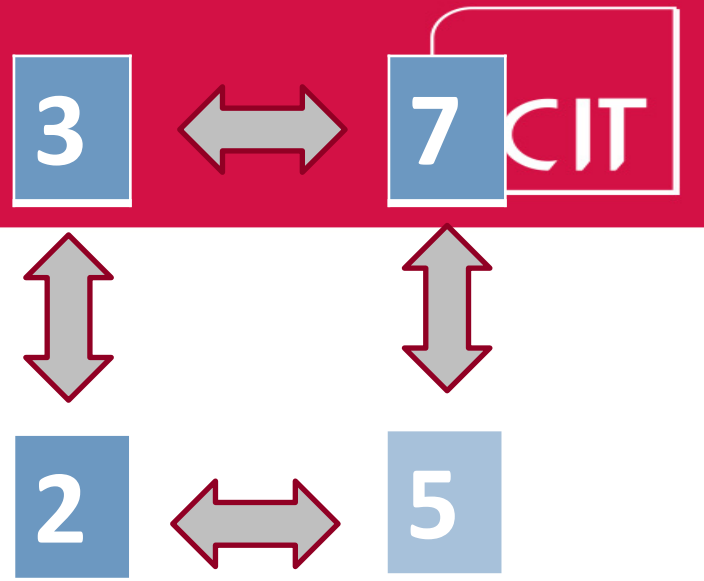
Cycle Crossover



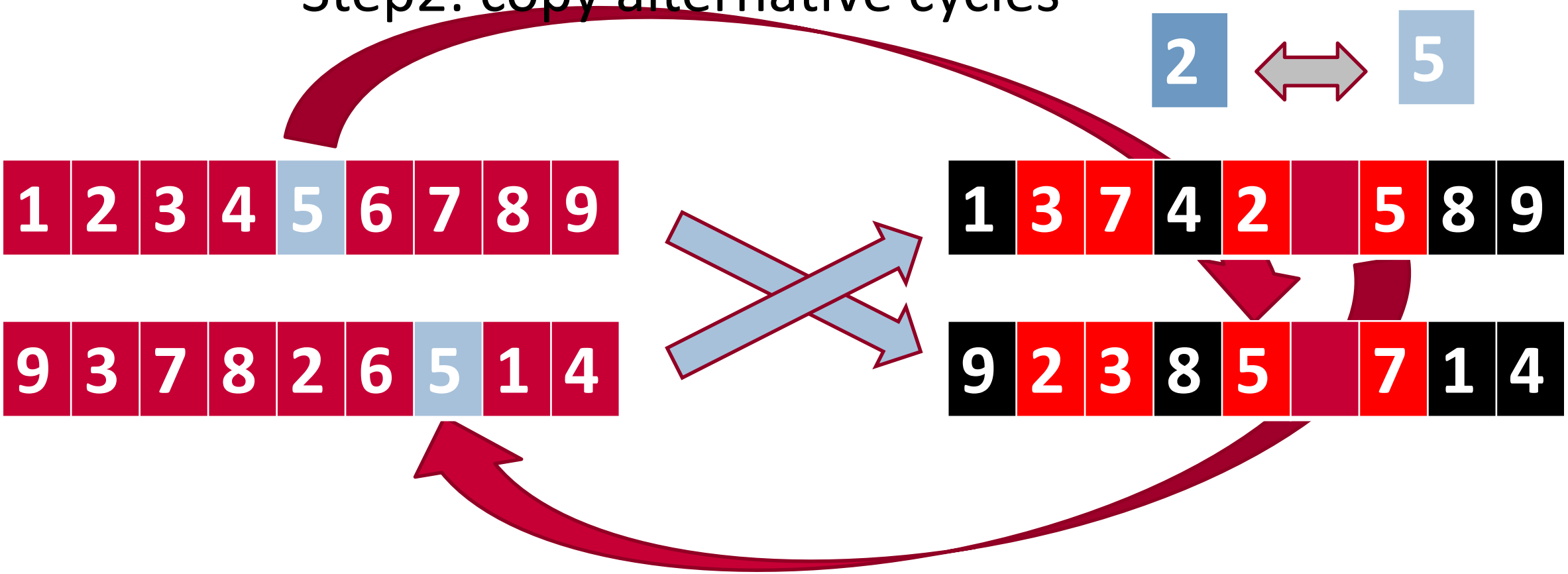
Cycle Crossover



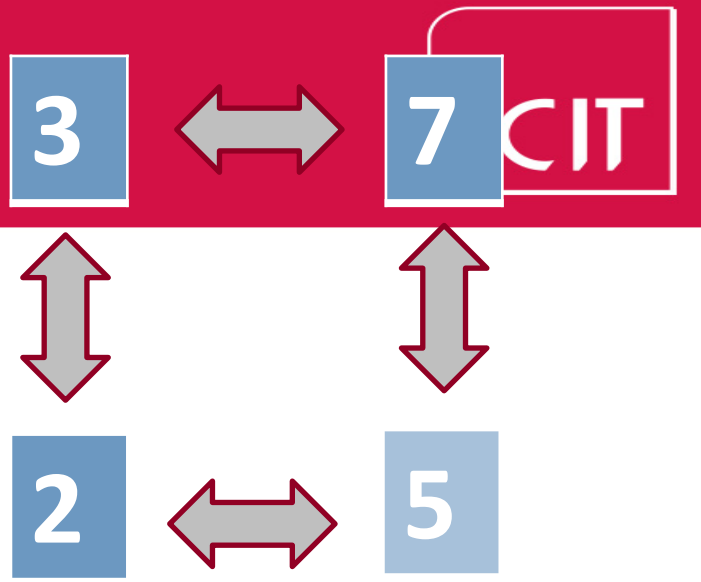
Cycle Crossover



Step2: copy alternative cycles

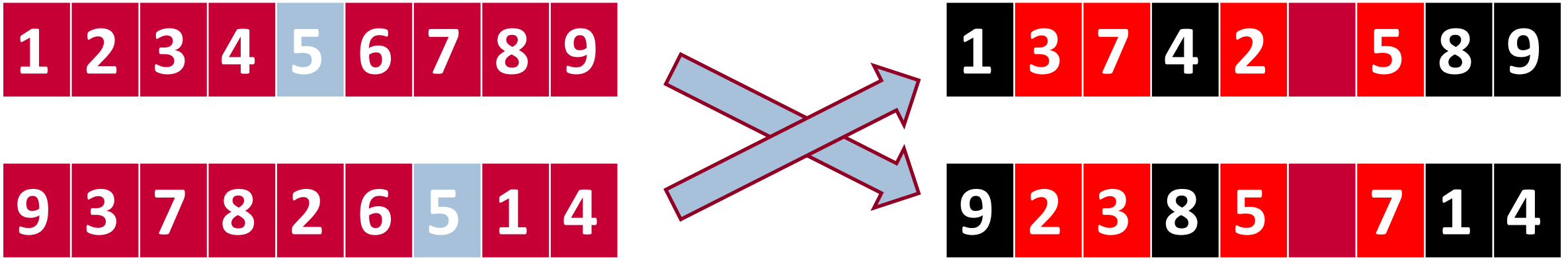


Cycle Crossover

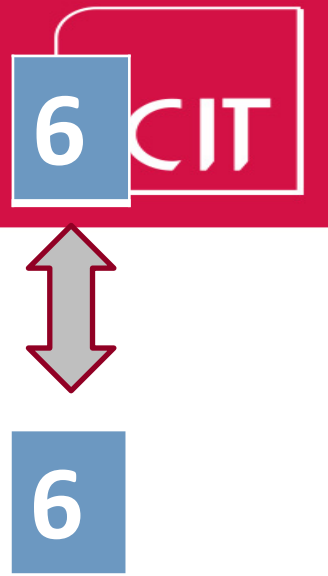


Step2: copy alternative cycles

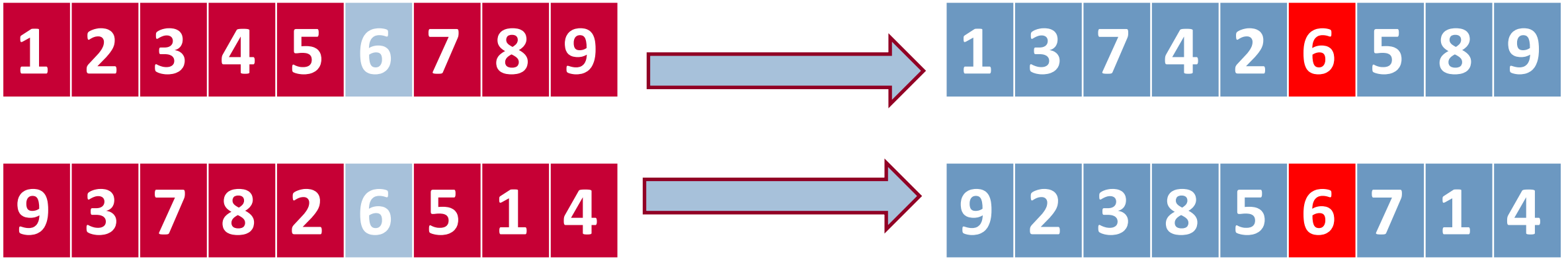
End of cycle



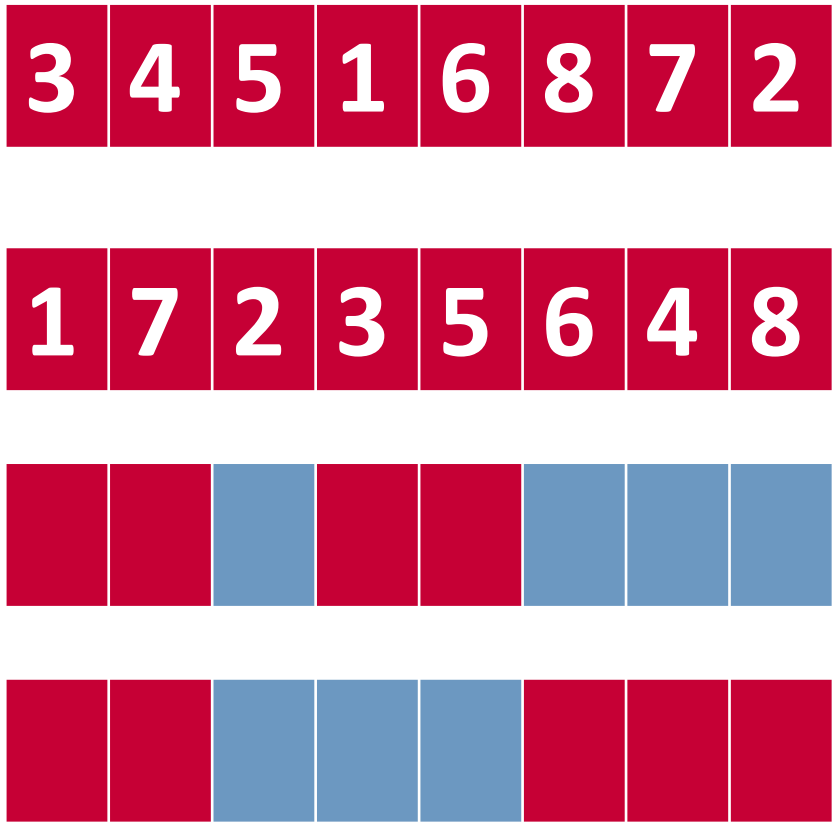
Cycle Crossover



copy alternative cycles



Exercise: Create children using Uniform, PMX, and Cycle



Exercise


3 4 5 1 6 8 7 2


1 7 2 3 5 6 4 8


1 3 5 6 4 8 7 2


3 5 2 1 7 6 4 8


1 4 2 3 5 8 7 6


3 7 5 1 6 2 4 8


3 7 5 1 6 8 4 2


1 4 2 3 5 6 7 8

Components of a GA

- Initialization
  - Usually at random, but can use prior knowledge
- **Selection**
  - **Give preference to better solutions**
- Variation
  - Create new solutions through crossover and mutation
- Replacement
  - Combine original population with newly created solutions

Commonly used operators

- **Selection methods**
  - **Resource used for slides: http://www.geatbx.com/docu/algindex-02.html**
- Replacement methods
- Variation operators

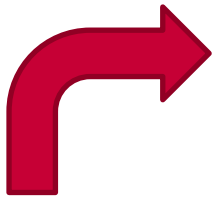
Mating Pool

- In the reproduction step, a mating pool of size ***S*** is created

- Apply crossover operation in the regeneration step (from candidates in the mating pool)

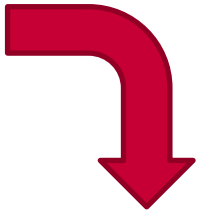
Mating Pool

Copy of the generation



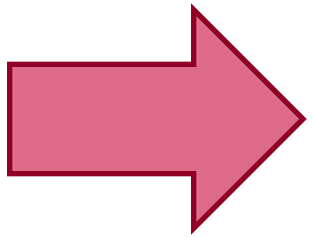
	Mating Pool
1	
2	
N-1	
N	

New generation after
A. Selection
B. Crossover
C. Mutation



	Generation at time t
1	
2	
N-1	
N	

Current Generation



	Generation at time t+1
1	
2	
N-1	
N	

New Generation

Selection

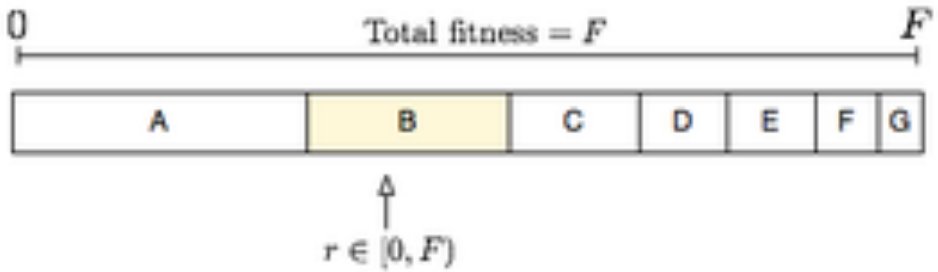
- Various methods can be used
- Main idea is to make copies of solutions that perform better at the expense of solutions that perform worse
- Two major classes of methods
  - Proportionate
  - Ordinal

Proportionate-based methods

- Probability of selecting a solution is proportional to its fitness value
- Probability Pselect(X) of selecting X from the population P(t) is

$$p_{select}(X) = \frac{f(X)}{\sum_{Y \in P(t)} f(Y)}$$

- Fitness is assumed to be positive

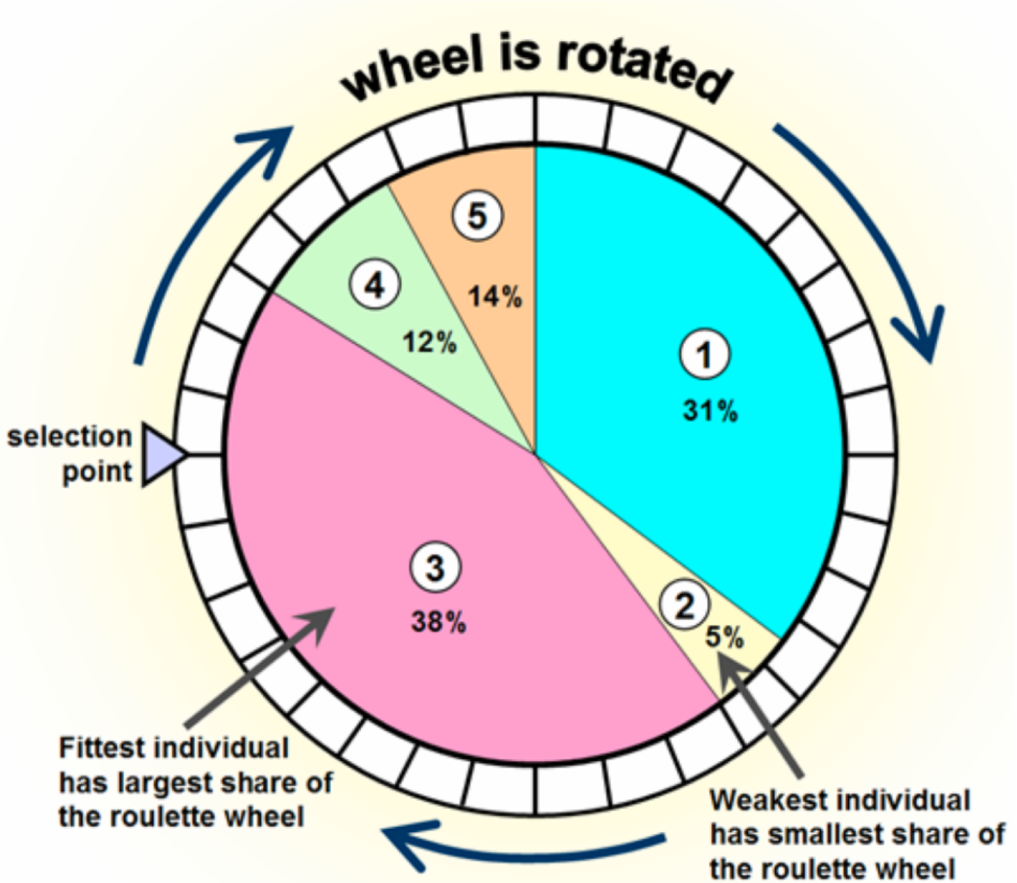


Source: Wikipedia

Proportionate-based methods

- We will look at two methods
  - Roulette Wheel
  - Stochastic Universal Sampling (SUS)

Roulette Wheel



Roulette Wheel

- Fitness of an individual corresponds to a slice of the roulette
- Implementation: Fitnesses are mapped to contiguous segments of a line
- Each individual's segment is equal in size to its fitness
- A uniform random number is generated and the individual whose segment spans the random number is selected
- Spin the roulette N times in order to select N individuals

Roulette Wheel: An Example

No.	chromosome	fitness	fraction of total
1	0100010001	6.82	0.31
2	1100101001	1.11	0.05
3	1100111001	8.48	0.38
4	0101011111	2.57	0.12
5	1100100100	3.08	0.14
	Totals:	22.05	1.00

Roulette Wheel: An Example

Number of Individual	1	2	3	4	5	6	7	8	9	10	11
Fitness	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0

Maximise objective function

Total is 11

So 2/11, 1.8/11 …..

$$p_{select}(X) = \frac{f(X)}{\sum_{Y \in P(t)} f(Y)}$$

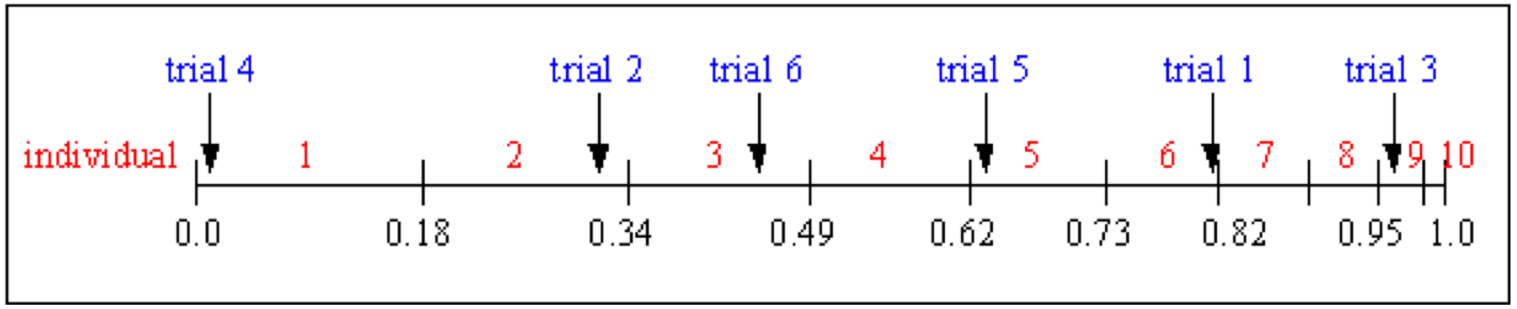
N.B. Don't need to sort them according to their fitness values, just done in this example to make it easier to see!

Roulette Wheel: An Example

Number of Individual	1	2	3	4	5	6	7	8	9	10	11
Fitness	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
Selection Probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0

Maximise objective function

- Sample of 6 random numbers: 0.81, 0.32, 0.96, 0.01, 0.65, 0.42



After selection:
1, 2, 3, 5, 6, 9

Roulette Wheel: Minimisation?

Need to transform the fitness!

Number of Individual	1	2	3	4	5	6	7	8	9	10	11
Fitness	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.5	0.3	0.1	0.0
Fitness transformed	0	0.2	0.4	0.6	0.8	1.0	1.2	1.5	1.7	1.9	2.0

- Subtract maximum fitness from original fitness values
- Alternative used is 1/fitness
  - Issue?

- Negative numbers?
  - If negative values, add |min| value to all values first

Roulette Wheel: Minimisation Example

Number of Individual	1	2	3	4	Total
Fitness	5	12	5	3	25

- 1/fitness_i?
- Max – fitness_i?

Roulette Wheel: Minimisation Example

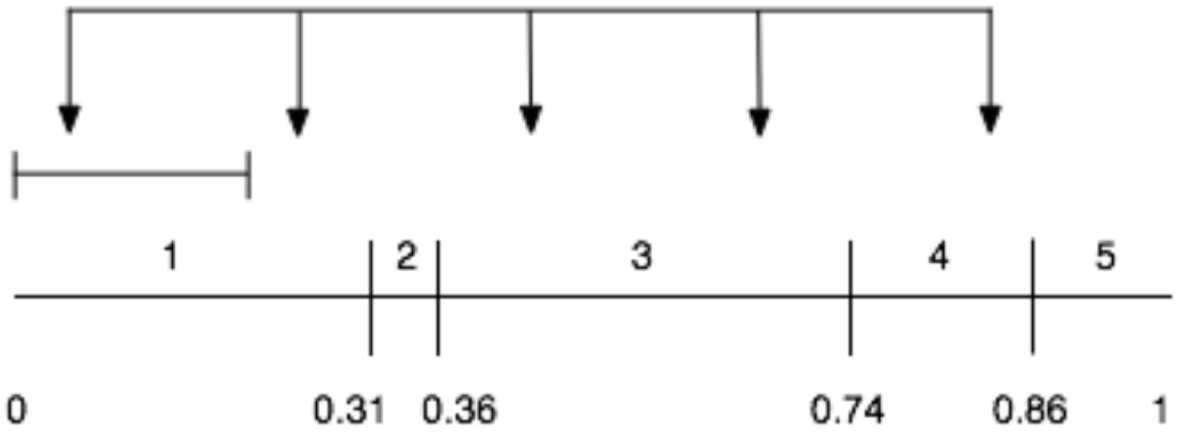
Number of Individual	1	2	3	4	Total
Fitness	5	12	5	3	25
1/fitness	0.2	0.08	0.2	0.33	0.81
Max-fitness	7	0	7	9	23

- But Max- will result in 0 for worst solution(s) so it/they will never be chosen!
  - Add token amount, e.g. Max+1 – fitness_i

Stochastic Universal Sampling (SUS)

Choose N parents:

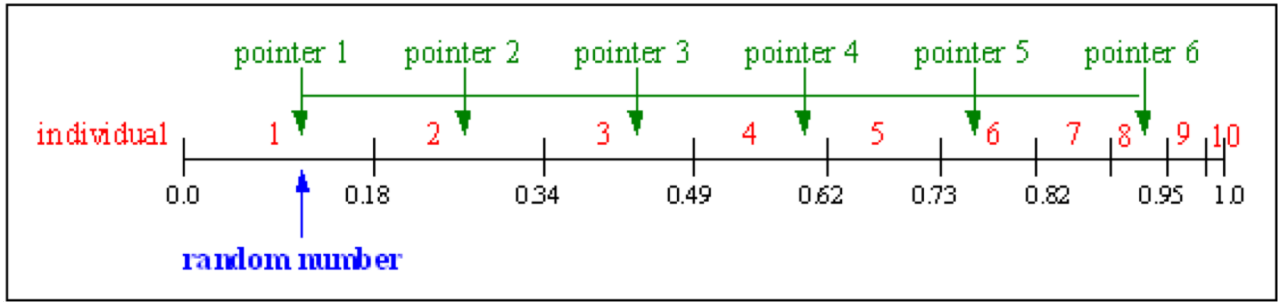
- Make N equally spaced marks
- Generate a uniform random number in [0, 1/N]. That number is the position of the 1st mark.
- (Equivalent to spinning the equally spaced marks on top of the roulette. Just need to spin it once)



Stochastic Universal Sampling (SUS)

For 6 individuals to be selected:

- The distance between the pointers is 1/6 = 0.167
- Sample of 1 number in the range [0, 0.167]



- After Selection the mating population consists of the individuals:
  - 1, 2, 3, 4, 6, 8
- Stochastic universal sampling ensures a selection of offspring which is closer to what is deserved then roulette wheel selection.

Rank based selection

- Attempt to remove problems of Fitness Proportional Selection by basing selection probabilities on relative rather than absolute fitness
- Rank population according to fitness and then base selection probabilities on rank where fittest has rank P and worst has rank 1
- This imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time

Ordinal-base methods

- Probability of selecting an individual depends on its relative order (or ranking) compared with other members

- Two commonly used methods:

  - Tournament selection

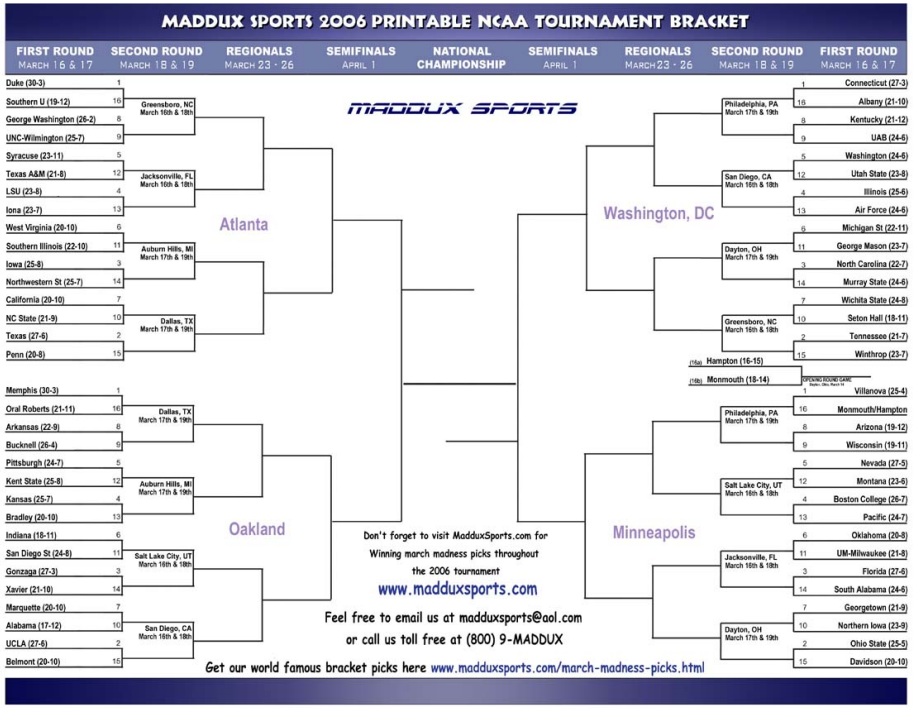
  - Truncation selection

Tournament Selection

- Pick s individuals and keep the best one
- s is a parameter
- Two variations:
  - With replacement
  - Without replacement

Example: binary tournament selection

- s=2
- Pick a pair of individuals from the population
- The winner survives, the loser dies
- Repeat N times



With and without replacement

- Imagine that we are picking individuals from a bag
- With replacement, after picking s individuals, we put them back in the bag
- Without replacement, we do not put them back in the bag
- What's the difference?
  - Without replacement has less variance (just like SUS has less variance compared to the roulette wheel)

Truncation selection

- Selects a given percentage *t* of the best individuals in the population

- t is a parameter
  - E.g. t=0.25 means selecting the top 25% individuals

- Make multiple copies of individuals, as needed.
  - With t=0.25 we would make 4 copies of each, in order to select N individuals

Proportionate vs. ordinal-based methods

- Fitness proportionate-based methods have 2 major drawbacks:
  - Super individual can take over the population too quickly
  - Search tends to stall when all the individuals have about the same fitness
- Ordinal-based methods are often preferred
  - Enable a sustained pressure towards better solutions
  - Can control selection pressure easily

Classification of Selection Schemas