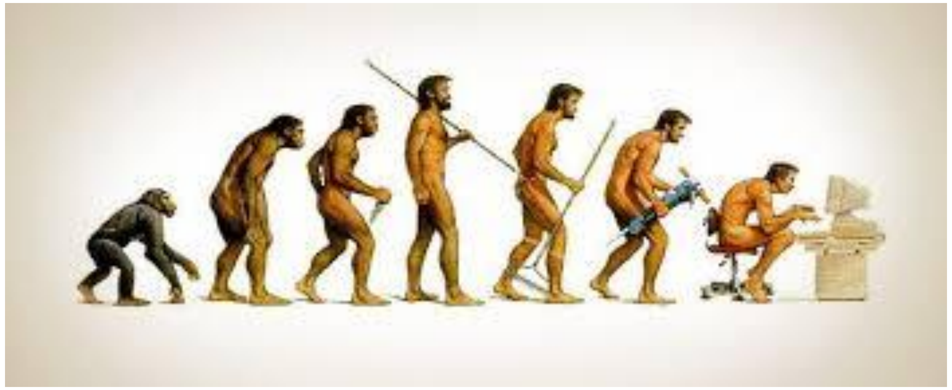
Genetic Algorithm

Local Search and GA

- **Local search** methods involve making small changes to potential solutions to a problem until an optimal solution is identified.
- Genetic Algorithm (GA) is a search-based **optimization** technique based on the principles of **Natural Genetics** and **Natural Selection**.
- It finds the **Optimal Solution** based in **survival of the fittest**.
- Optimization is the process of making something better i.e. to get the “best” output value.

Evolutionary Computing



- **Natural systems** as guiding metaphors

- Charles Darwinian Evolution – 1859

- **Theory of natural selection**
  - It proposes that the plants and animals that exist today are the result of millions of years of adaptation to the demands of the environment

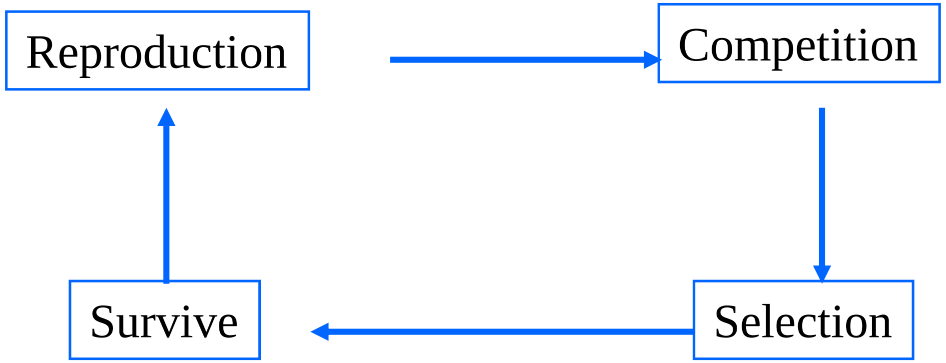
- Over time, the entire population of the ecosystem is said to evolve to contain organisms that, **on average, are fitter for environment** than those of previous generations of the population

Genetic Algorithms

- The concepts of **GAs** are directly derived from **natural evolution**.
- **GAs** emulate ideas from **genetics** and **natural selection** and can **search potentially large spaces**
- Based on: **survival of the most fittest individual**
- Two key steps: *reproduction, survive*

Genetic Algorithm

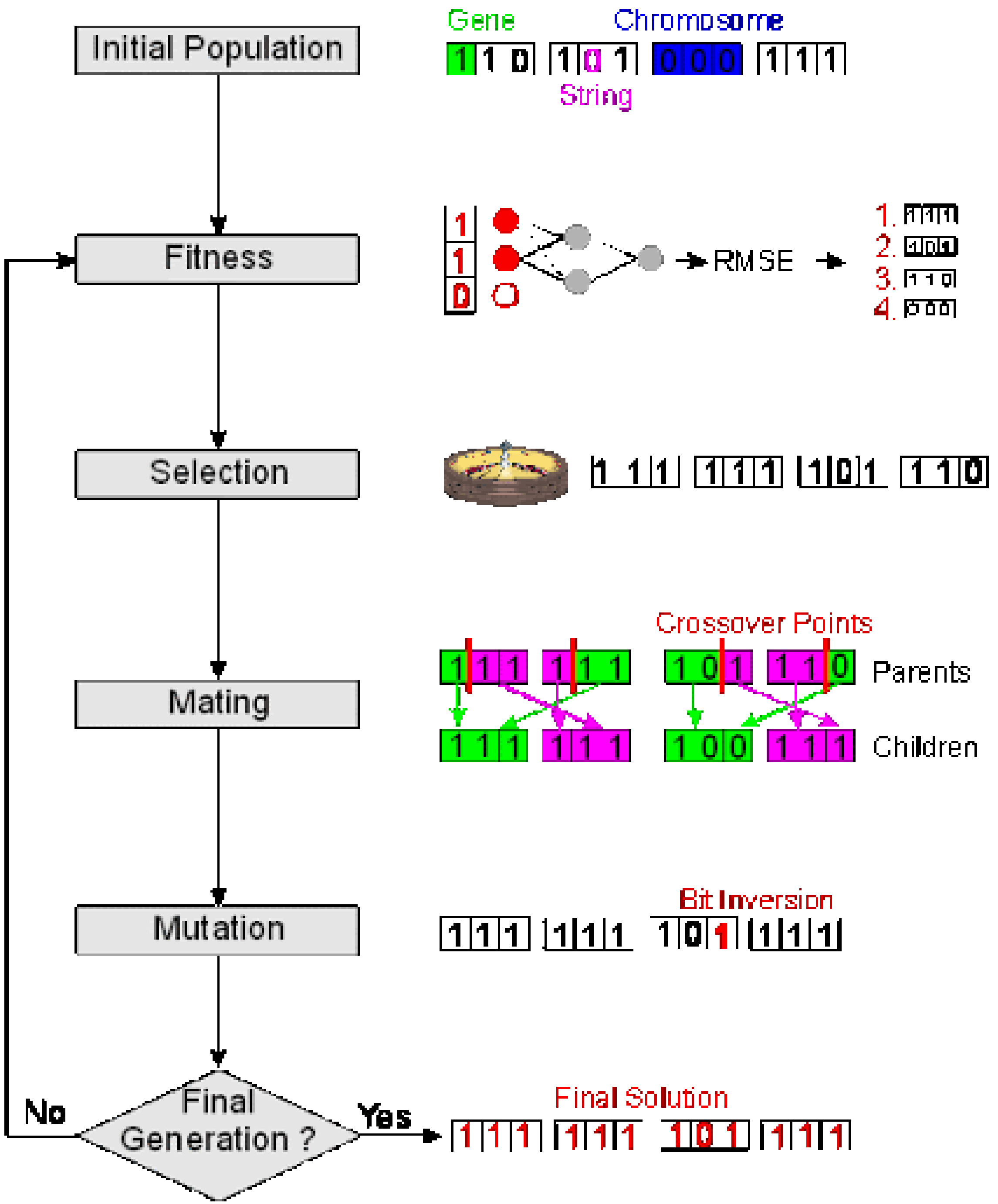
- Based on Darwinian Paradigm



- Intrinsically a robust search and optimization mechanism

Genetic Algorithm

- The process for running a genetic algorithm is as follows.
  1. Generate a random population of chromosomes (this is the first generation).
  2. If termination criteria are satisfied, stop. Otherwise, continue with step 3.
  3. Determine the fitness of each chromosome.
  4. Apply crossover and mutation to selected chromosomes from the current generation to generate a new population of chromosomes—the next generation.
  5. Return to step 2.



Initial Population
Gene
Chromosome
String
Fitness
RMSE
Selection
Crossover Points
Parents
Children
Mating
Mutation
Bit Inversion
Final Generation ?
No
Yes
Final Solution

Genetic Algorithms

- Before we can **apply Genetic Algorithm to a problem**, we need to answer:
  - How can an **individual be represented**?
  - What is the **fitness function**?
  - How are **individuals selected**?
  - How do **individuals are generated**?

Representation of States (Solution)

- States as **sequence of strings**
- Each state or individual is represented as a string over finite alphabet $\{0,1\}$. It is also called **chromosome** which contains **genes**.

genes

Solution: 607 $\xrightarrow{\text{Encoding}}$ $\boxed{1001011111}$

Chromosome: Binary String

Recall that every state is a solution in local search

Reproduction: Building New States

- After representing <u>States</u> (Solutions).
- Build a **population of random solutions**.
- Let them **reproduce** using **genetic operations**.
  - At each generation: apply ”<u>survival of the fittest</u>”
  - <u>Hopefully better and better solutions</u> evolve over time
  - The <u>best solutions are more likely to survive and more likely to produce even better solutions</u>

Evaluation and Selection

- We then see how good the solutions are, using an <u>evaluation function</u> (recall f(n) in infomed search)
  - Often **it is a heuristic**, especially if it is computationally expensive to do a complete evaluation
  - The final population can then be evaluated more deeply to decide on the best solution

Survival of the Fittest

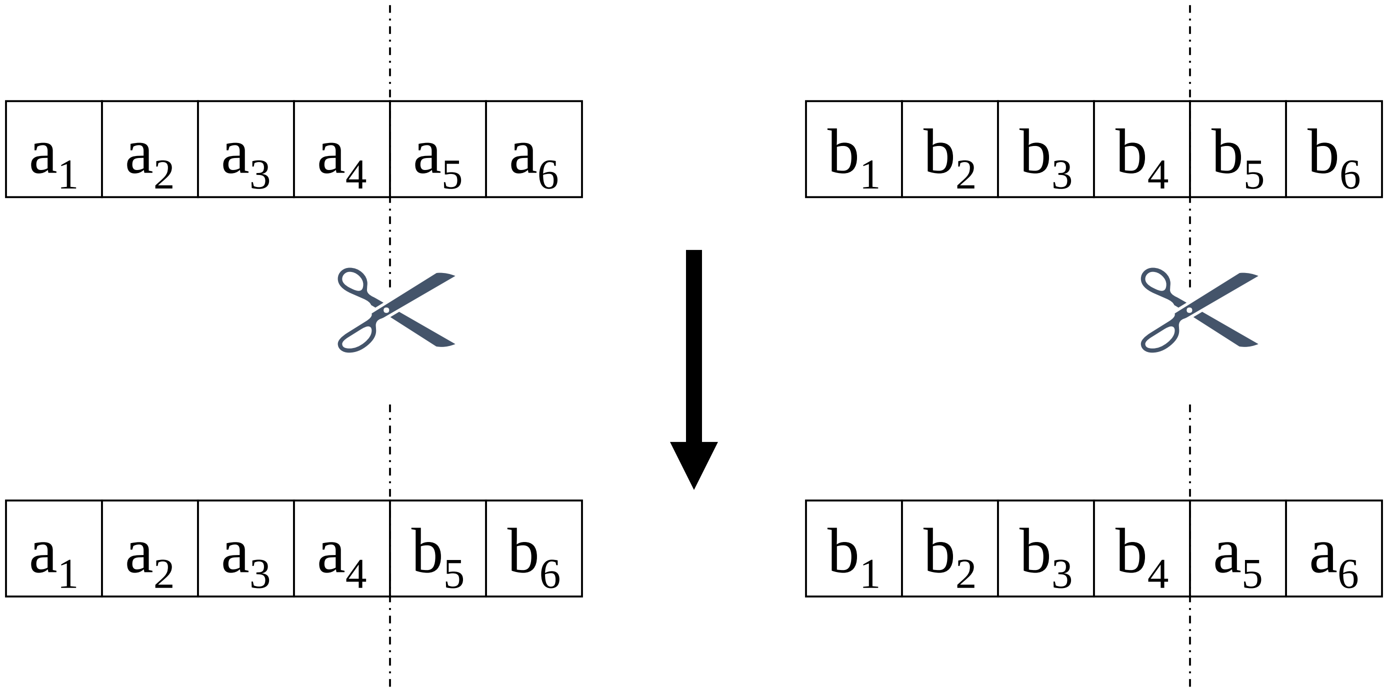
- Select the surviving population
- Likelihood of survival is related in some way to your score on the fitness function
  - The most common technique is roulette wheel selection
- Note we always keep the **best solution** so far
- Remember: Its local search

Genetic Operators

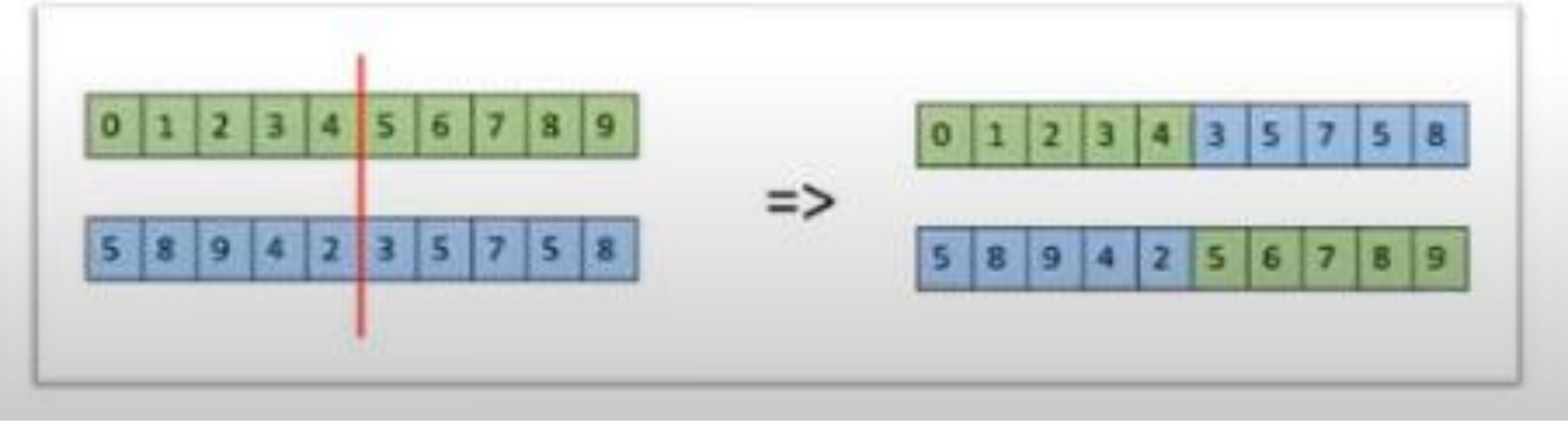
- These operators mimic what happens to our genetic material when we reproduce
  - Crossover
  - Mutation

Crossover Operator

- Cut two solutions at a random point and switch the respective parts
- Typically a value of 0.7 for crossover probability gives good results.

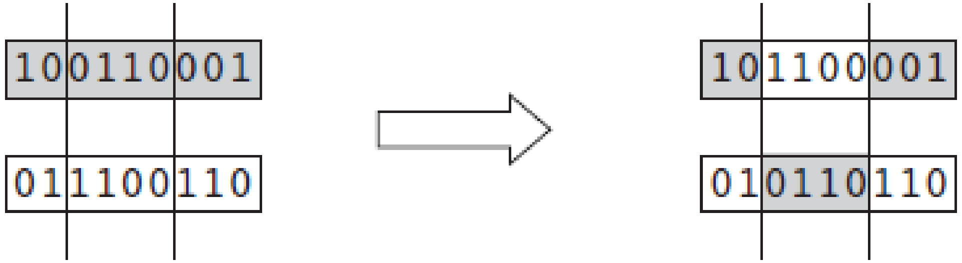


Crossover



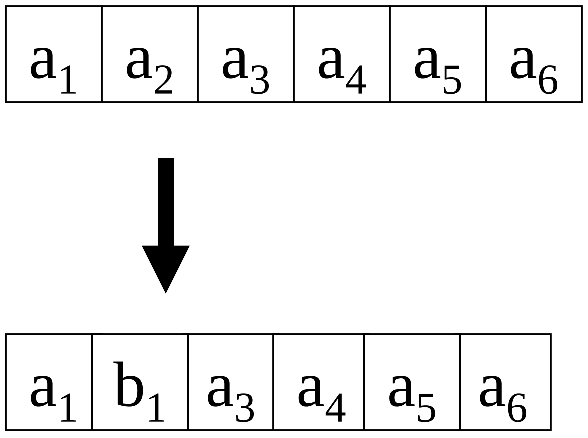
Crossover operator

- The crossover operator is applied to two chromosomes of the same length as follows:
  - 1. Select a random crossover point.
  - 2. Break each chromosome into two parts, splitting at the crossover point.
  - 3. Recombine the broken chromosomes by combining the front of one with the back of the other, and vice versa, to produce two new chromosomes.
- For example, consider the following two chromosomes:
  - 110100110001001
  - 010101000111101



Mutation operator

- Randomly **change one bit** in the solution
  - Mutation is a **unary operator** (i.e., applied to just one argument—a single gene)
- Occasional mutation makes the method much less sensitive to the original population and also allows ”new” solutions to emerge



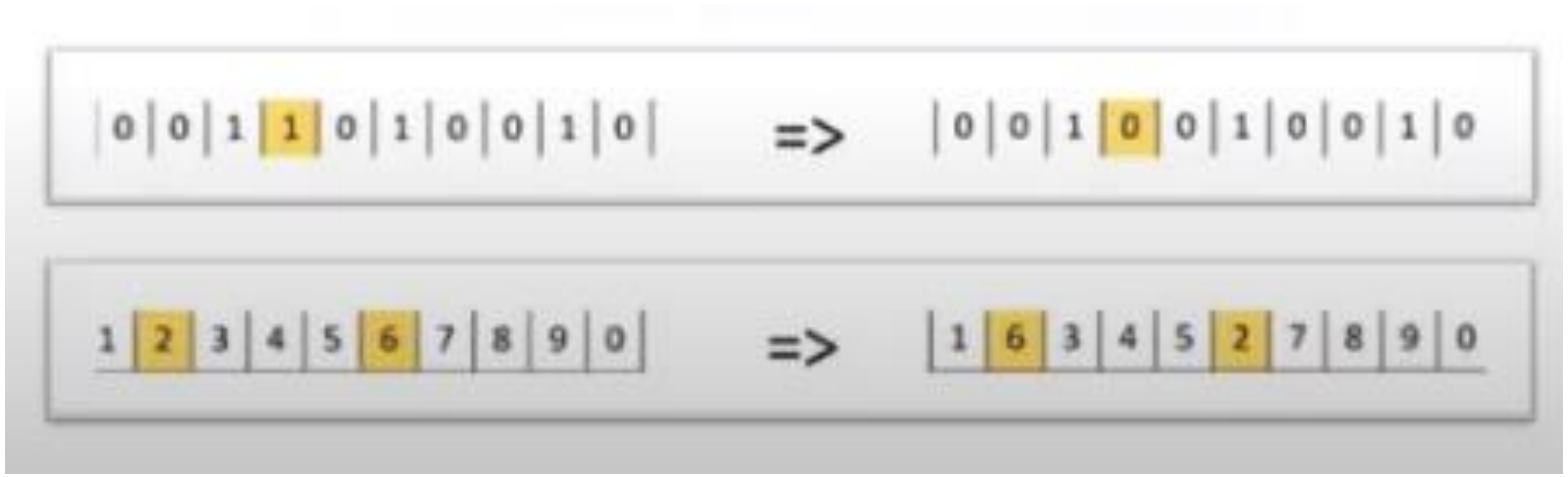
Mutation of a2 = b1

010101110001001

$\Downarrow$

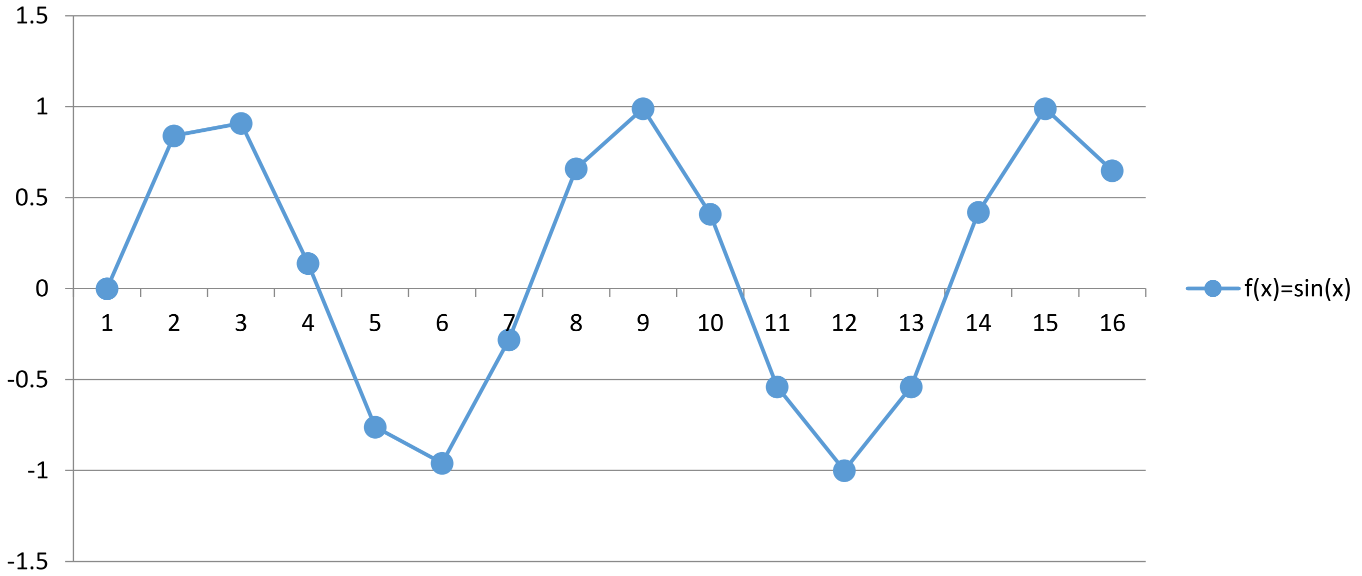
010101110101001

Mutation



Fitness Function (Optimization)

- Fitness Function for a mathematic function
- Ex. attempt to maximize the function:
  - $f(x) = \sin(x)$ in range $0 \leq x \leq 15$



Fitness Function (Optimization)

- **Using population size of 4 chromosomes**
- <u>First Generation:</u> Generate a random population:

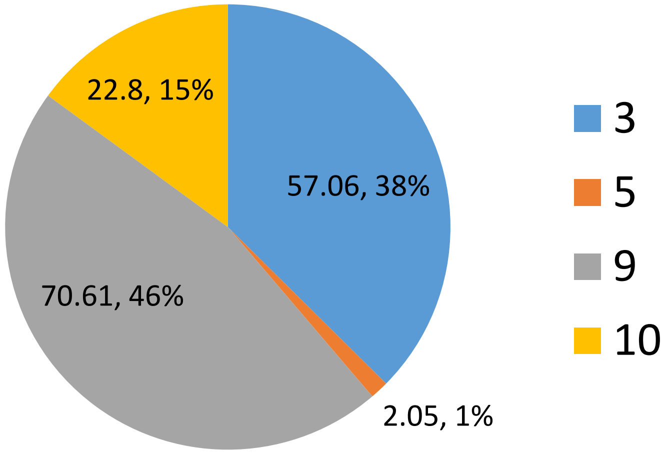
c1 = 1001	(9)	c3 = 1010	(10)
c2 = 0011	(3)	c4 = 0101	(5)

- To calculate **<u>fitness</u>** of a chromosome, we calculate $f(x)$ for its decimal value
- Assign fitness as a numeric value from <u>0 to 100</u>
  - 0 is the least fit
  - 100 is the most fit.
- $f(x)$ generates real numbers between -1 and 1.
- Assign a fitness score of 100 to $f(x) = 1$ and fitness of 0 to $f(x) = -1$
- Fitness of 50 will be assigned to $f(x) = 0$

Fitness Function (Optimization)

Fitness of *x*, $f'(x)$:

- $f'(x) = 50(f(x) + 1)$
- $f'(x) = 50(\sin(x) + 1)$



x	f(x)=sin(x)	f'(x) = 50(f(x)+1)
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Fitness Function (Optimization)

- The range of real numbers **from 0 to 100** is divided up between the chromosomes proportionally to each chromosome's fitness.
- In first generation:
  - c1 has 46.3% of the range (i.e., from 0 to 46.3)
  - c2 37.4% of the range (i.e., from 46.4 to 83.7)
  - …

Table 14.1 Generation 1

Chromosome	Genes	Integer value	$f(x)$	Fitness $f'(x)$	Fitness ratio
c1	1001	9	0.41	70.61	46.3%
c2	0011	3	0.14	57.06	37.4%
c3	1010	10	−0.54	22.80	14.9%
c4	0101	5	−0.96	2.05	1.34%

Roulette-wheel selection

- A random number is generated between $0 - 100$
- This number will fall in the range of one of the chromosomes (suppose c1)
  - c1 chromosome has been selected for reproduction
- The next random number is used to select second chromosome's (suppose c2)
- Hence, **fitter chromosomes will tend to produce more offspring than less fit chromosomes**.

- **Important:** Method does not stop less fit chromosomes from reproducing, to ensure that populations do not **stagnate**, by constantly breeding from the same parents.

Next Generation

- Need 4 random numbers for next generation
- Assume that:
- First random number is 56.7 (c2 is chosen)
- Second random number is 38.2 (c1 is chosen)
- Third random number is 20 (c1 is chosen)
- Fourth random number is 85 (c3 is chosen)

Next Generation cont.

- <u>Combine first two</u> to produce two new offspring:
  - Crossover point

c1 = 1001 (9)	c2 = 0011 (3)
c5 = 1011 (11)	**c6 = 0001 (1)**

- <u>Combine last two</u> to produce two new offspring:
  - Crossover point

c1 = 1001 (9)	c3 = 1010 (10)
c8 = 1011 (11)	**c7 = 1000 (8)**

Second generation: c5 to c8

- c4 did not have a chance to reproduce (its genes will be lost)
- **Fittest chromosome in the first generation (c1), able to reproduce twice**
- Passing on its highly fit genes to all members of the next generation

Table 14.2 Generation 2

Chromosome	Genes	Integer value	$f(x)$	Fitness $f'(x)$	Fitness ratio
c5	1011	11	−1	0	0%
c6	0001	1	0.84	92.07	48.1%
c7	1000	8	0.99	99.47	51.9%
c8	1011	11	−1	0	0%

Extremely fit

Termination criteria

- Termination criteria would probably determine that c7 is the most fit (local optimum) and the algorithm will be stopped.

Example: $f(x) = x^2$

- **Problem: Maximize the function $f(x) = x^2$ where x varies between 1 and 31.**

- Step 1:
  - Code the decision variable of the problem a binary unsigned integer of length 5.
  - To start we select an initial population of size 4 by tossing a fair coin times.
  - Fitness function is defined by $f(x) = x^2$

Example: $f(x) = x^2$

String No.	Initial Pop.	x-value	$f(x) = x^2$
1	0 1 1 0 1	13	169
2	1 1 0 0 0	24	576
3	0 1 0 0 0	8	64
4	1 0 0 1 1	19	361

String No.	x-value	$f(x) = x^2$	P-select $\frac{f_i}{\sum f}$	Expected Count $\frac{f_i}{avg}$	Roulette Wheel
1	13	169	0.14	0.58	1
2	24	576	0.49	1.97	2
3	8	64	0.06	0.22	0
4	19	361	0.31	1.23	1

$$f(x) = x^2$$

- The best strings get more copies, while the weak ones just die off.
- After selection, crossover takes place
  - Strings combined randomly using one point cross over
  - Apply one bit mutation

String No.	x-value	One Point Cross Over	New Children
1	13	0 1 1 0 | 1	1 1 0 0 1
2	24	1 1 0 0 | 0	0 1 1 0 0
2	24	1 1 | 0 0 0	1 0 0 0 0
4	19	1 0 | 0 1 1	1 1 0 1 1

Apply Mutation	New Population
1 1 **0** 0 1	1 1 1 0 1
0 1 1 0 **0**	0 1 1 0 1
1 0 **0** 0 0	1 0 1 0 0
1 1 0 **1** 1	1 1 0 0 1

$$f(x) = x^2$$

- New Population

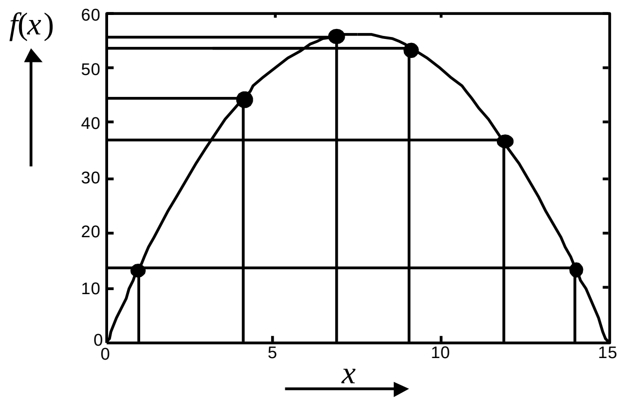
String No.	Initial Pop.	x-value	$f(x) = x^2$
1	1 1 1 0 1	29	841
2	0 1 1 0 1	13	169
3	1 0 1 0 0	20	400
4	1 1 0 0 1	25	625

Practice Question

- Find the maximum value of function

$$f(x) = -x^2 + 15x$$

  - Represent problem using *chromosomes* built from four *genes*:
  - Initial random population of size $N = 6$:



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

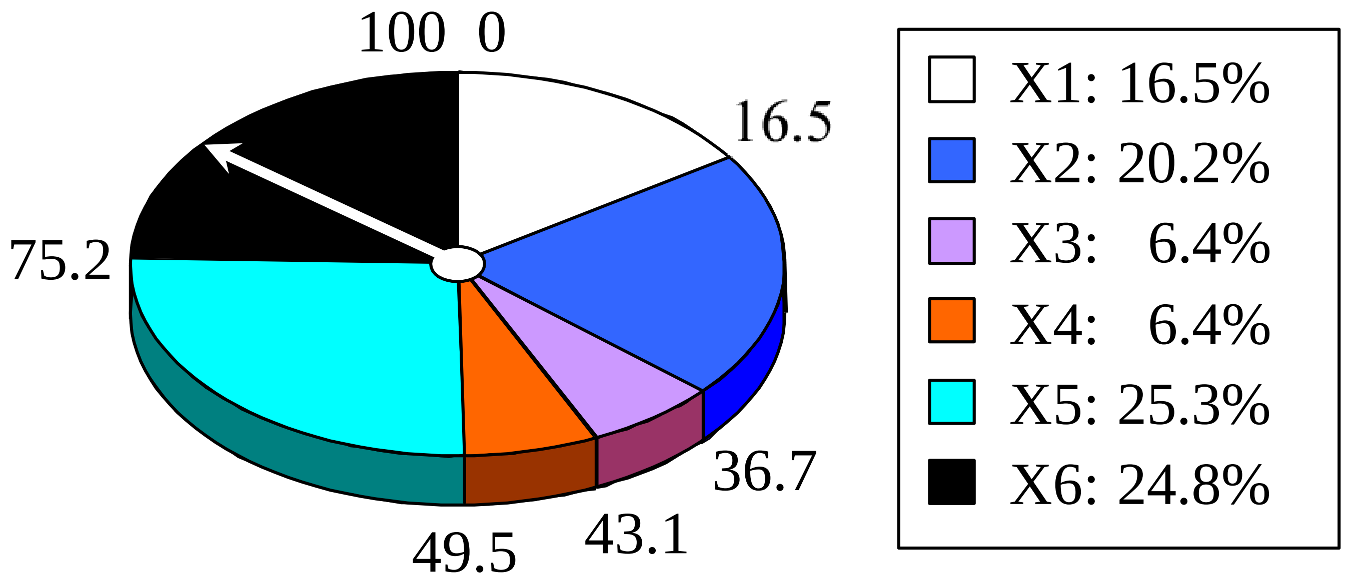
Practice Question

- Determine chromosome *fitness* for each chromosome:

Chromosome label	*Chromosome string*	*Decoded integer*	*Chromosome 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

Practice Question

- Use fitness ratios to determine which chromosomes are selected for *crossover* and *mutation* operations:



Practice Question

- Assume we have the following function
$$f(x) = x^3 - 60x^2 + 900x + 100$$
where x is constrained to $[0 - 31]$. We wish to maximize $f(x)$ (the optimal is $x = 10$). Using a binary representation we can represent x using five binary digits.
- Given the following four chromosomes give the values for x and $f(x)$.

Chromosome	Binary String
P_1	11100
P_2	01111
P_3	10111
P_4	00100

- If P_3 and P_2 are chosen as parents and we apply one point crossover show the resulting children, C_1 and C_2. Use a crossover point of 1 (where 0 is to the very left of the chromosome)
- Do the same using P_4 and P_2 with a crossover point of 2 and create C_3 and C_4
- Calculate the value of x and f(x) for C_1 to C_4.
- Assume the initial population was $x = \{17, 21, 4, 28\}$. Using one-point crossover, what is the probability of finding the optimal solution? Explain your reasons.

Practice Question – SOLUTION

- **Given the following four chromosomes give the values for x and $f(x)$.**

Chromosome	Binary String	x	f(x)
P_1	11100	28	212
P_2	01111	15	3475
P_3	10111	23	1227
P_4	00100	4	2804

Practice Question – SOLUTION

- **If P_3 and P_2 are chosen as parents and we apply one point crossover show the resulting children, C_1 and C_2. Use a crossover point of 1 (where 0 is to the very left of the chromosome)**
- **Calculate the value of x and f(x) for C_1 to C_4.**

Chromosome	Binary String	x	f(x)
C_1	11111	31	131
C_2	00111	7	3803
C_3	00111	7	3803
C_4	01100	12	3998

GA Example - Crossover

P_3	1	0	1	1	1
P_2	0	1	1	1	1
C_1	1	1	1	1	1
C_2	0	0	1	1	1

P_4	0	0	1	0	0
P_2	0	1	1	1	1
C_3	0	0	1	1	1
C_4	0	1	1	0	0

GA Example - After First Round of Breeding

- The average evaluation has risen P_2, was the strongest individual in the initial population. It was chosen both times but we have lost it from the current population
- We have a value of x=7 in the population which is the closest value to 10 we have found

Chromosome	Binary String	x	f(x)
P_1	11111	31	131
P_2	00111	7	3803
P_3	00111	7	3803
P_4	01100	12	3998
	TOTAL		11735
	AVERAGE		2933.75