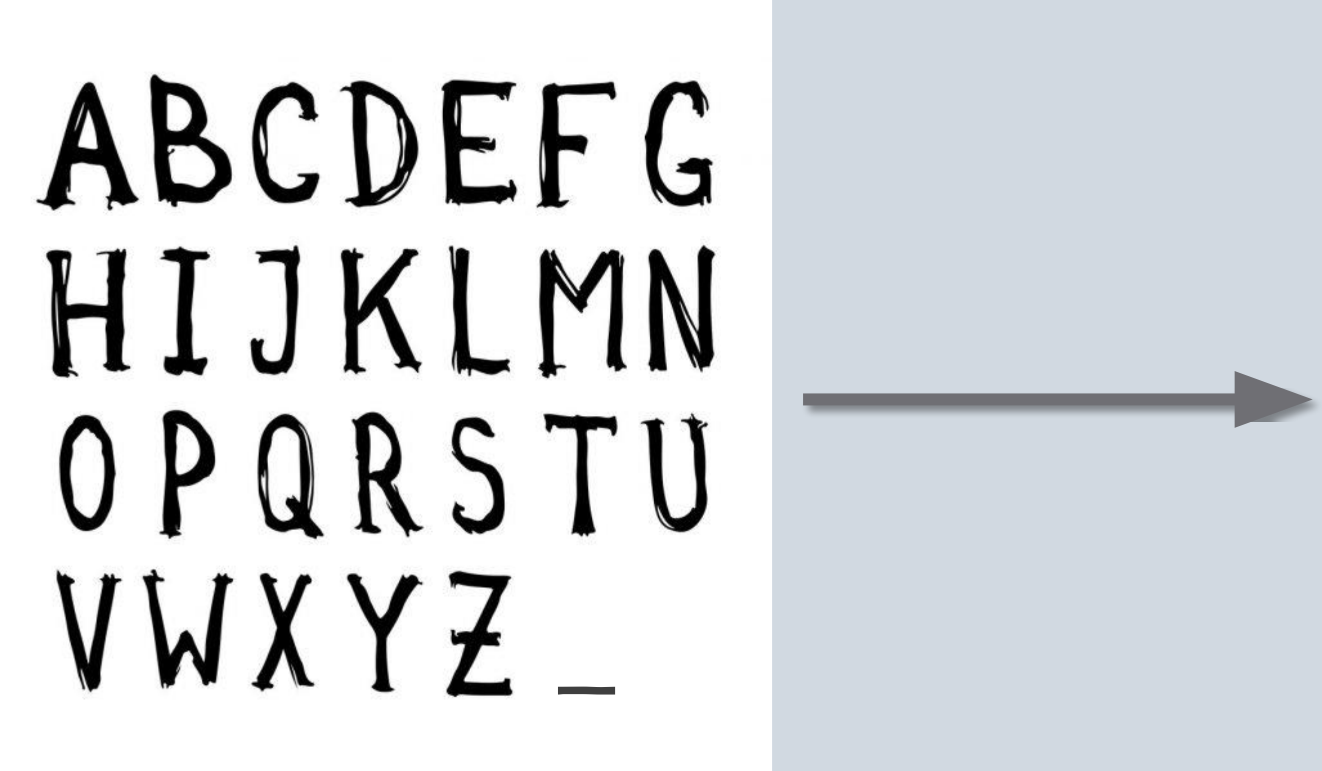
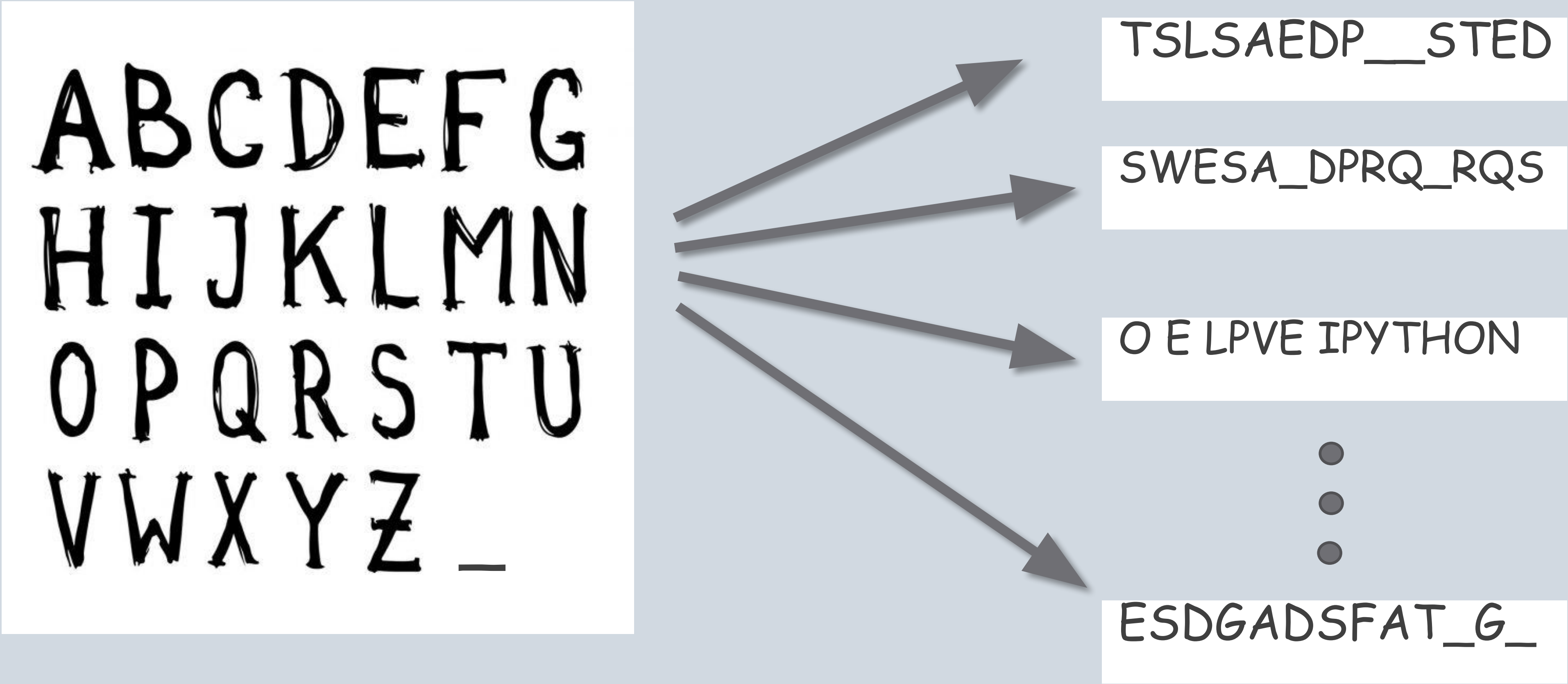
Genetic Algorithm

Let us look at a problem



I_LOVE_PYTHON

First step towards genetic algorithm (Generate Population)



TSLSAEDP__STED

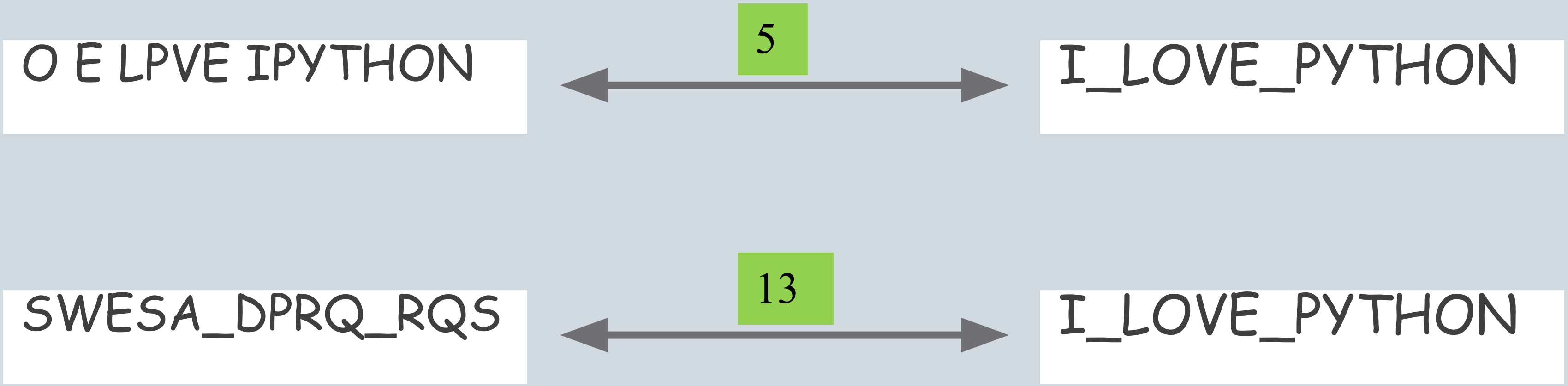
SWESA_DPRQ_RQS

O E LPVE IPYTHON

ESDGADSFAT_G_

FITNESS FUNCTION

A Fitness Score is given to each individual which **shows the ability of an individual to “compete”**. The individual having optimal fitness score (or near optimal) are sought.



O E LPVE IPYTHON

5

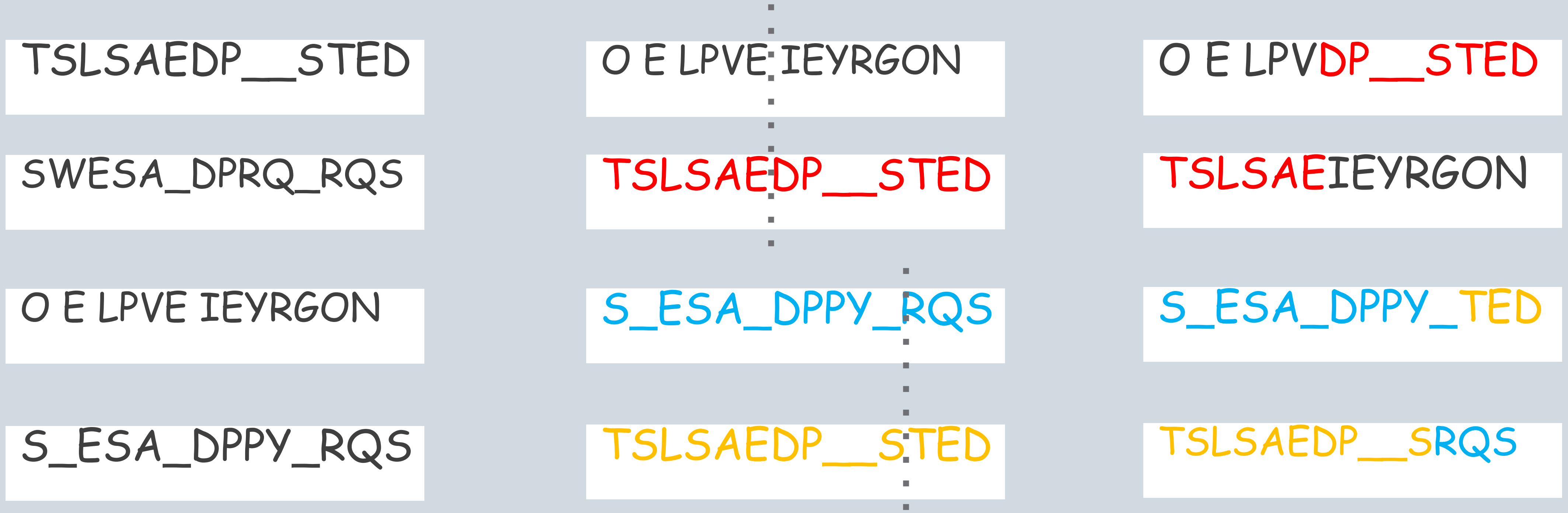
I_LOVE_PYTHON

SWESA_DPRQ_RQS

13

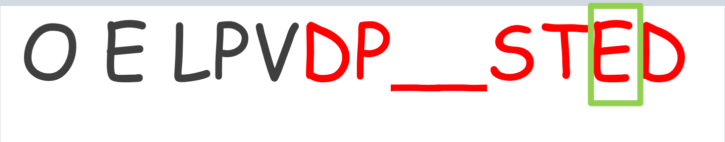
I_LOVE_PYTHON

PARENT SELECTION AND CROSSOVER

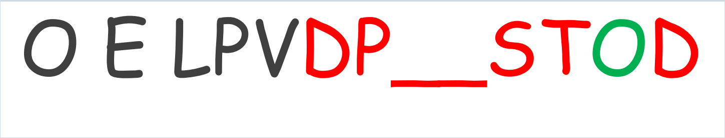


TSLSAEDP__STED	O E LPVE IEYRGON	O E LPVDP__STED
SWESA_DPRQ_RQS	TSLSAEDP__STED	TSLSAEIEYRGON
O E LPVE IEYRGON	S_ESA_DPPY_RQS	S_ESA_DPPY_TED
S_ESA_DPPY_RQS	TSLSAEDP__STED	TSLSAEDP__SRQS

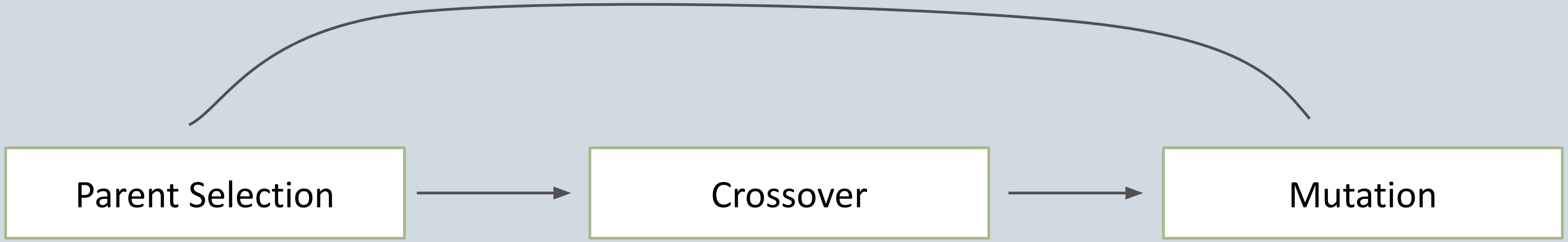
MUTATION



O E LPVDP__STED



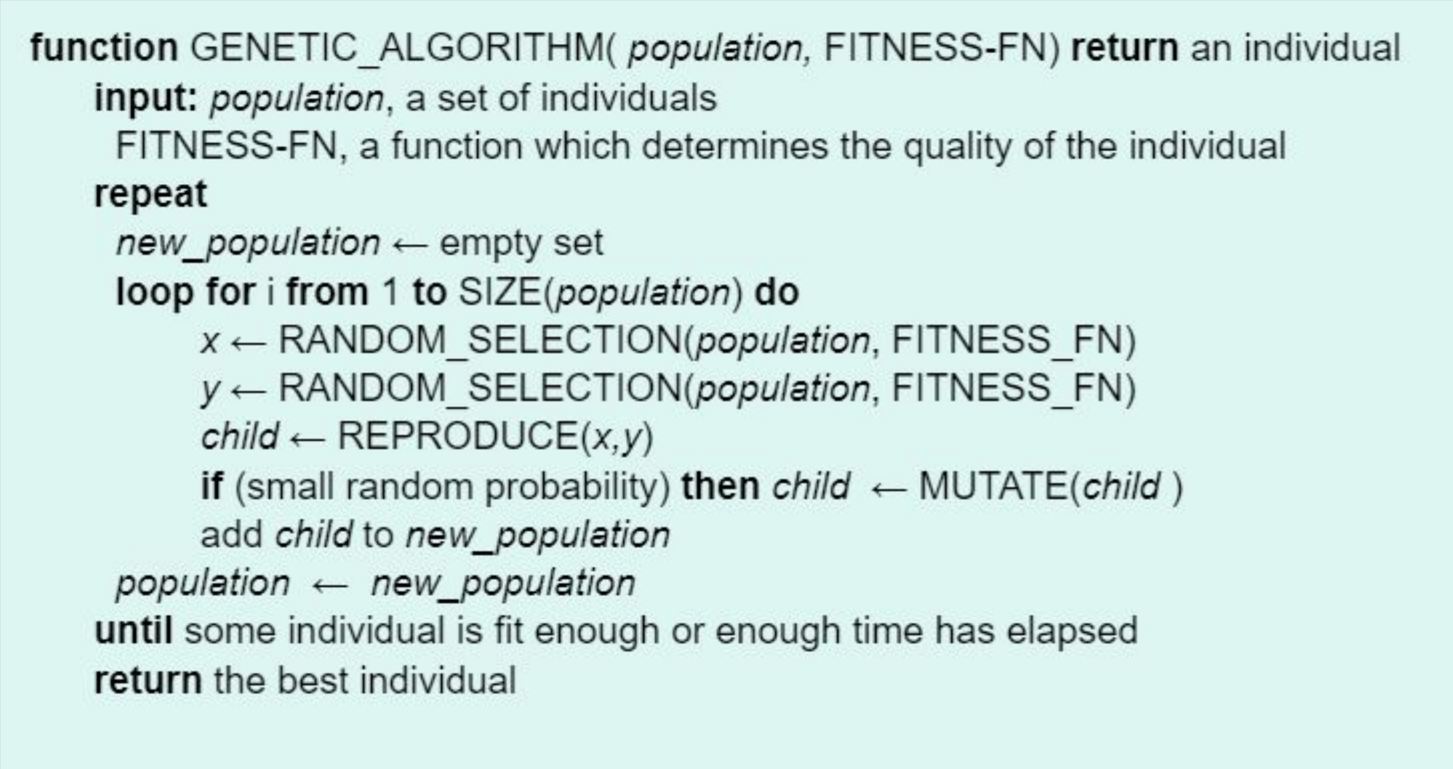
O E LPVDP__STOD


Parent Selection

Crossover

Mutation

Algorithm



```
function GENETIC_ALGORITHM( population, FITNESS-FN) return an individual
    input: population, a set of individuals
     FITNESS-FN, a function which determines the quality of the individual
    repeat
     new_population ← empty set
     loop for i from 1 to SIZE(population) do
          x ← RANDOM_SELECTION(population, FITNESS_FN)
          y ← RANDOM_SELECTION(population, FITNESS_FN)
          child ← REPRODUCE(x,y)
          if (small random probability) then child  ← MUTATE(child )
          add child to new_population
     population  ←  new_population
    until some individual is fit enough or enough time has elapsed
    return the best individual
```