5/27/2022

intelligent systems

Genetic Algorithms

Team members :-

1- Ahmed Abd Elrahiem Mohamed
2- Omar Gamal Fathi
3- Mostafa Hussien Amin
4- Ahmed Mohamed Abd Elhamid

Dr. Abdel-Rahman Hedar

Table of Contents :

Introduction

The project problem and importance:

Get the phrase automatically Structure Using genetic algorithms

an exploration into the possibility of automatically acquiring the phrase structure of a language.

The system automatically acquires a grammar of scored context-free rules, where each rule is binary branching. Two sources of distributional information are used to acquire and score the rules.

Save time and get the best results.

Summary to genetic algorithms

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

Methodology

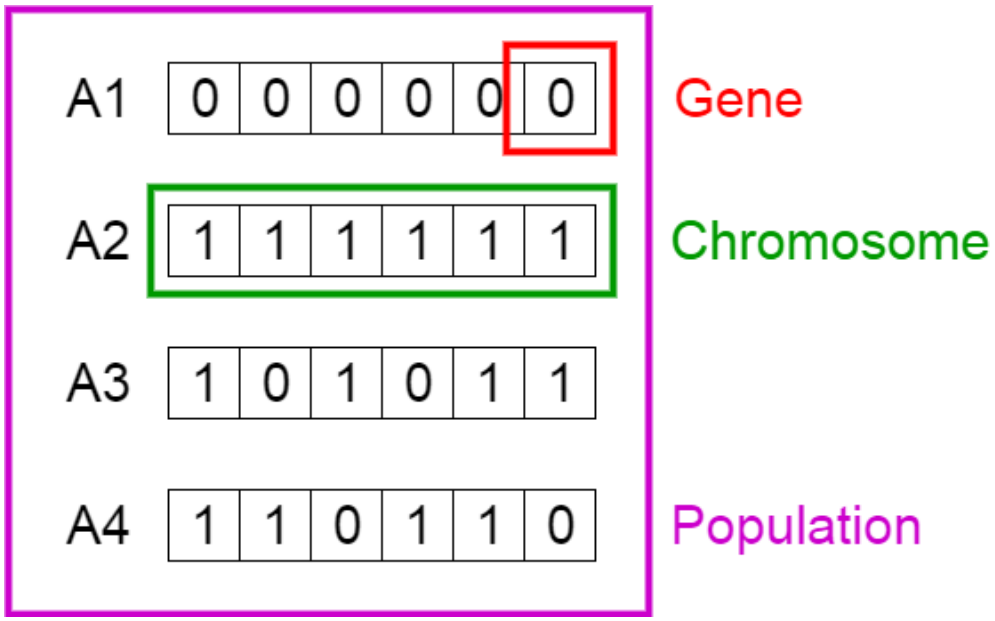
The Genetic algorithms and analyze their steps

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

Initial Population

The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem you want to solve.

An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution).



Fitness Function

The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

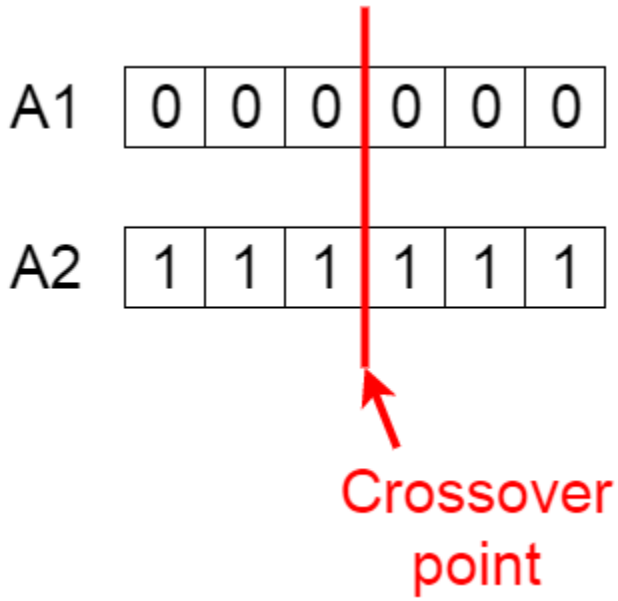
Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation.

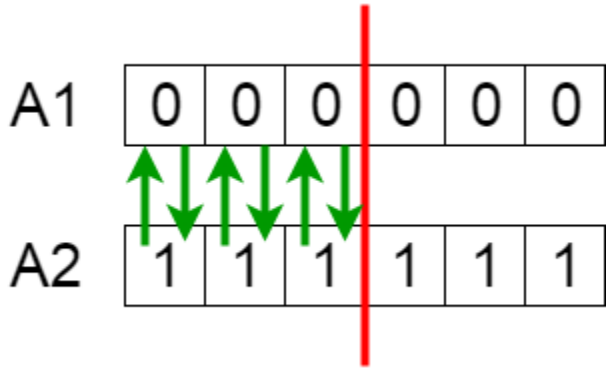
Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

Crossover

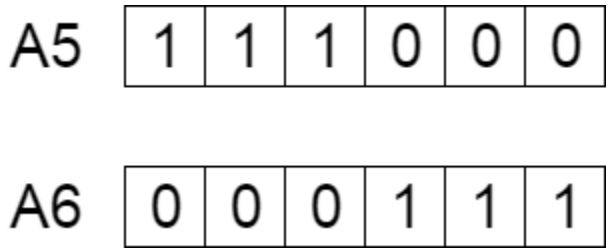
Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.



Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

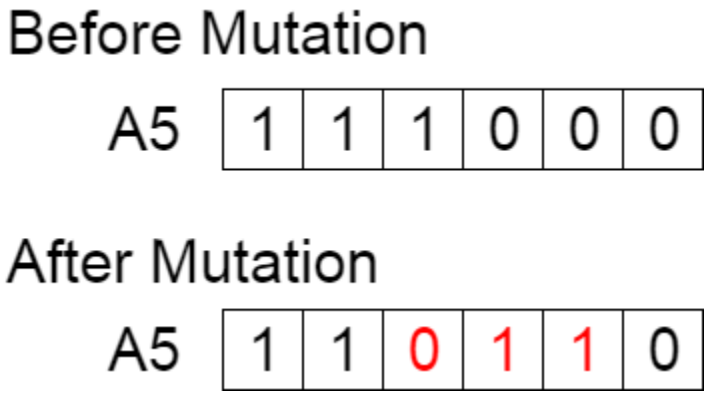


The new offspring are added to the population.



Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.



Mutation occurs to maintain diversity within the population and prevent premature convergence.

Pseudocode

```
START
Generate the initial population
Compute fitness
REPEAT
Selection
Crossover
Mutation
Compute fitness
UNTIL population has converged
STOP
```

Experimental Simulation

- **The programming languages and environments used in the project**
  - Python programming language used to implement the algorithm code
  - Idle shell the environment to run python.
- **The details of programming the primary function and its procedures used to implement the introduced**

  PASSWORD = 'This is target phrase'
  MUTATE_PROBABILITY = 1

  - **def fitness(password, test_word):**
    #Fitness is calculated based on number of letter in test_word that matches with password.
  - **def generate_word(password):**

    #Here random sentence is generated and returned.
  - **def generate_population(population_size, password):**

    #Population of random sentences are generated.

- **def generate_mating_pool(population, population_size, test_word):**

  #Pool of individuals is made based on their fitness score.. if fitness of an individual is 10%, 10 such individual is added to the mating pool.

- **def create_child(individual1, individual2):**

  #A child individual is created by choosing randomly from individual1 or individual2.

  # If random value from (0 to 100) is more than 50, choose i'th element from individual1.

  # If random value from (0 to 100) is less than 50, choose i'th element from individual2.

  # This converts array to a string.

- **def create_children(mating_pool, population_size):**

  #Two individuals are selected from mating pool randomly and a new population is generated.

- **def mutate_word(word, mutate_probability):**

  #A word is mutated based on mutate_probability

- **def mutate_population(population, mutate_probability):**

  #Each individuals in a population argument is subjected to mutate_word function where new mutated word is generated.

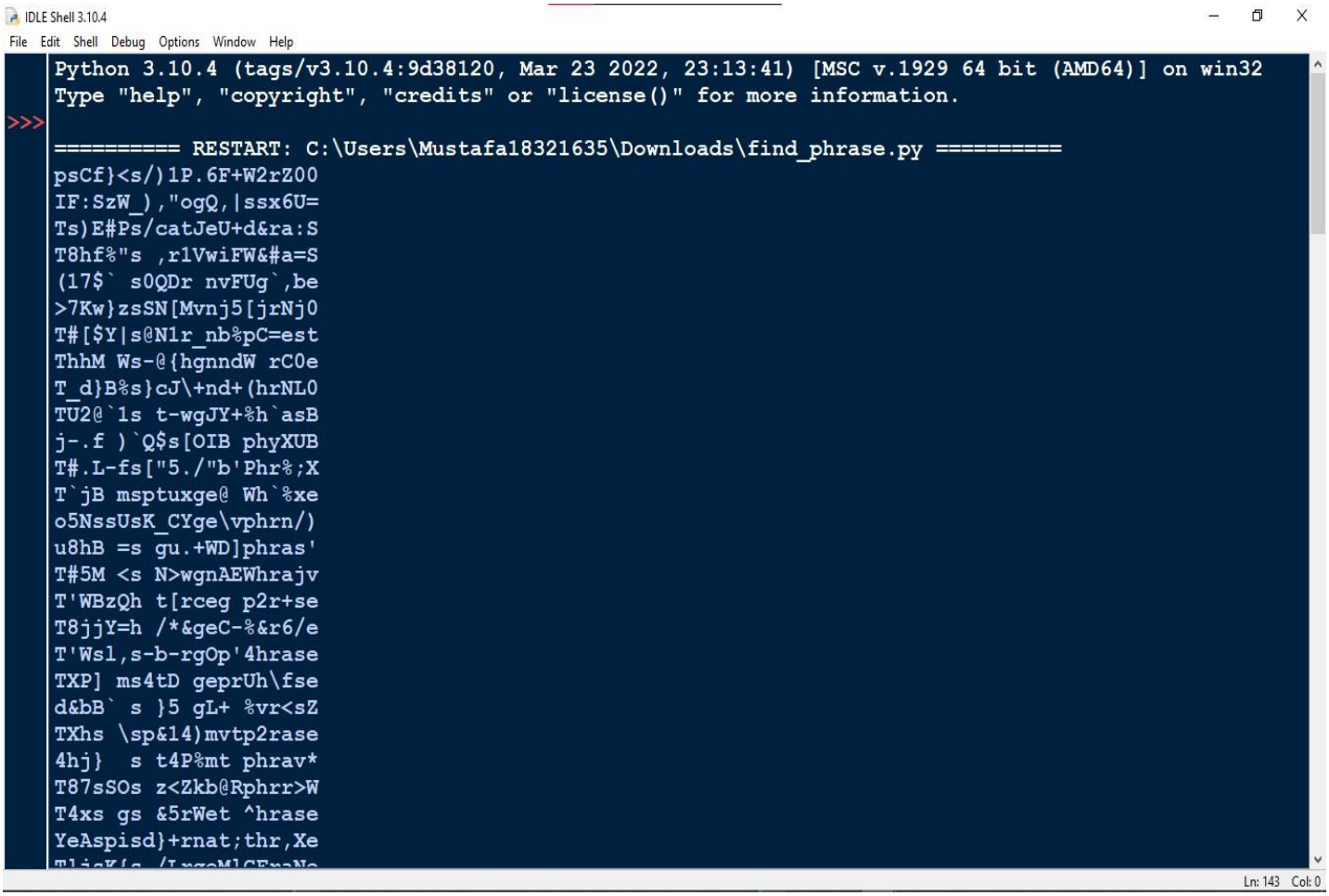
- **def check_fitness(population, password):**

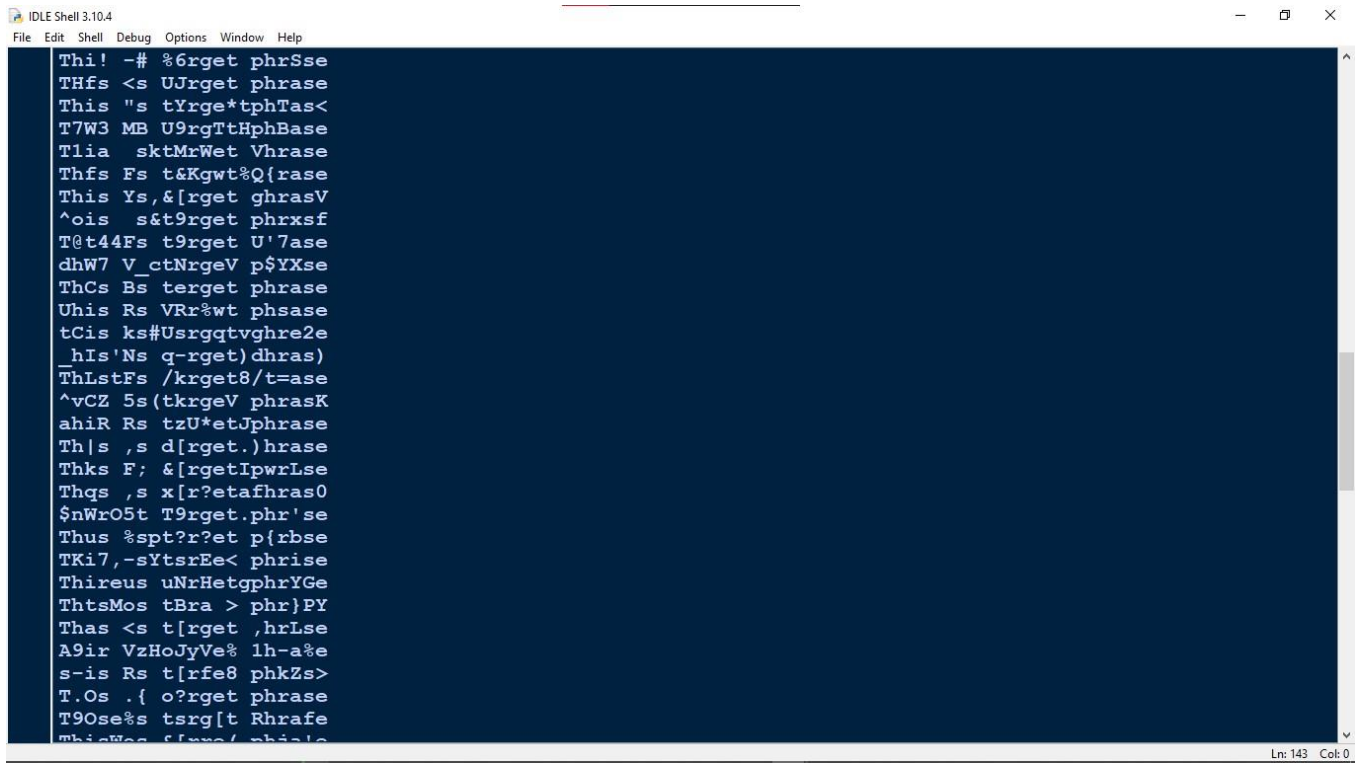
  #Whole population is checked if its individuals are equal to password.

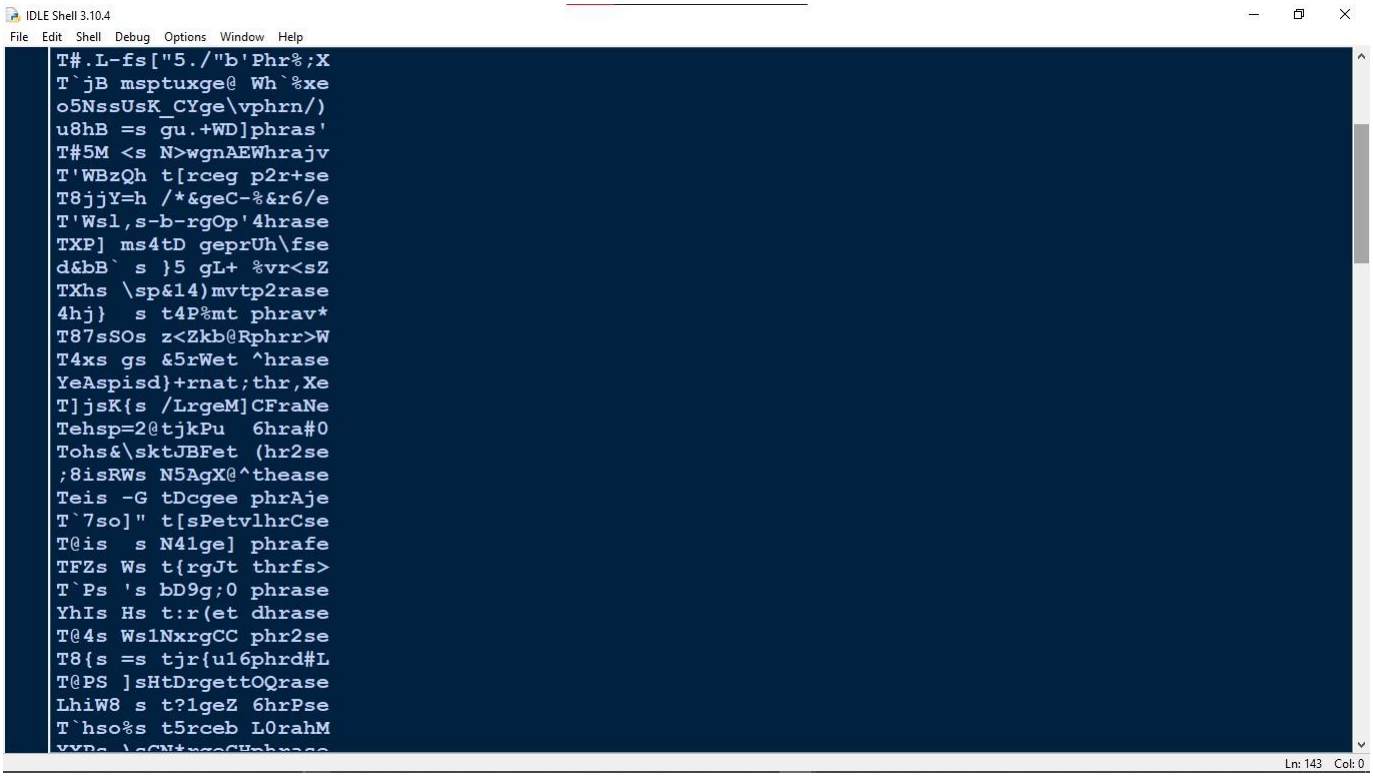
  # Random population generation

  # Loop until individuals match with password.

- **The test cases used to test the programmed codes**

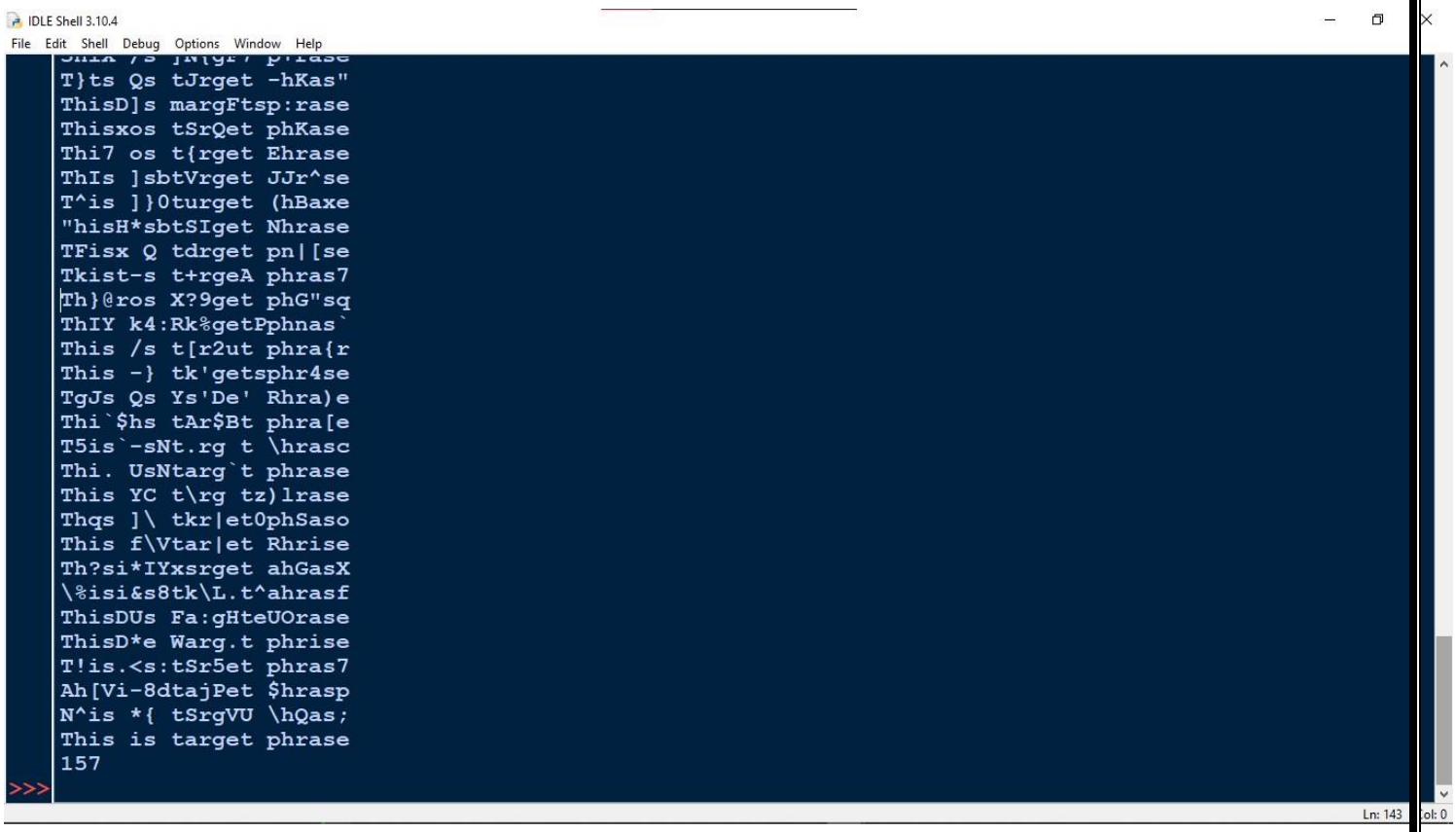



IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Thi! -# %6rget phrSse
THfs <s UJrget phrase
This "s tYrge*tphTas<
T7W3 MB U9rgTtHphBase
T1ia sktMrWet Vhrase
Thfs Fs t&Kgwt%Q{rase
This Ys,&[rget ghrasV
^ois s&t9rget phrxsf
T@t44Fs t9rget U'7ase
dhW7 V_ctNrgeV p$YXse
ThCs Bs terget phrase
Uhis Rs VRr%wt phsase
tCis ks#Usrgqtvghre2e
_hIs'Ns q-rget)dhras)
ThLstFs /krget8/t=ase
^vCZ 5s(tkrgeV phrasK
ahiR Rs tzU*etJphrase
Th|s ,s d[rget.)hrase
Thks F; &[rgetIpwrLse
Thqs ,s x[r?etafhras0
$nWrO5t T9rget.phr'se
Thus %spt?r?et p{rbse
TKi7,-sYtsrEe< phrise
Thireus uNrHetgphrYGe
ThtsMos tBra > phr}PY
Thas <s t[rget ,hrLse
A9ir VzHoJyVe% 1h-a%e
s-is Rs t[rfe8 phkZs>
T.Os .{ o?rget phrase
T9Ose%s tsrg[t Rhrafe
Ln: 143 Col: 0


IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
T#.L-fs["5./"b'Phr%;X
T`jB msptuxge@ Wh`%xe
o5NssUsK_CYge\vphrn/)
u8hB =s gu.+WD]phras'
T#5M <s N>wgnAEWhrajv
T'WBzQh t[rceg p2r+se
T8jjY=h /*&geC-%&r6/e
T'Wsl,s-b-rgOp'4hrase
TXP] ms4tD geprUh\fse
d&bB` s }5 gL+ %vr<sZ
TXhs \sp&14)mvtp2rase
4hj} s t4P%mt phrav*
T87sSOs z<Zkb@Rphrr>W
T4xs gs &5rWet ^hrase
YeAspisd}+rnat;thr,Xe
T]jsK{s /LrgeM]CFraNe
Tehsp=2@tjkPu 6hra#0
Tohs&\sktJBFet (hr2se
;8isRWs N5AgX@^thease
Teis -G tDcgee phrAje
T`7so]" t[sPetvlhrCse
T@is s N41ge] phrafe
TFZs Ws t{rgJt thrfs>
T`Ps 's bD9g;0 phrase
YhIs Hs t:r(et dhrase
T@4s Ws1NxrgCC phr2se
T8{s =s tjr{u16phrd#L
T@PS]sHtDrgettOQrase
LhiW8 s t?1geZ 6hrPse
T`hso%s t5rceb L0rahM
Ln: 143 Col: 0

Results and Technical Discussion

- **The main program results and outputs**



- **Evaluation experimental procedure and analysis of results**

- **Algorithms achieve the best results in the fastest time, it's perform operations in a few seconds and reach the correct sentence.**

Conclusions:

Genetic algorithms get the phrase automatically with a structure in very little time and the best results.

References:

- **Wikipedia**
- **Slide**
- **Google Scholar**
- https://apps.dtic.mil/sti/citations/ADA460382?fbclid=IwAR2kJC5SuXqilS-Djs1SeMVGUkZpMMiS0Sh_lUmaQENJelLHDA2j79M4hHw
- https://aclanthology.org/P08-1010.pdf?fbclid=IwAR2cinqoODoQwcHf2HjEw8LZCq04y3SE8aPTZeQrSw8x-N15Fzfw7dAWtcc

Appendix:

https://github.com/mustafa7ussien/find_phrase?fbclid=IwAR3nuhA5_wcTusppyL3bPm9W2SRiQPruCNgjNeVISiIj_XUoF37lEkO-PJ8