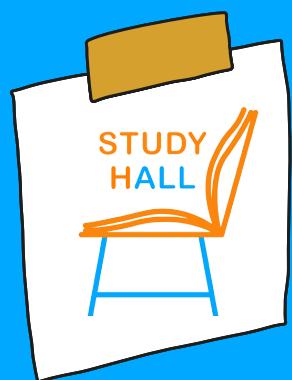


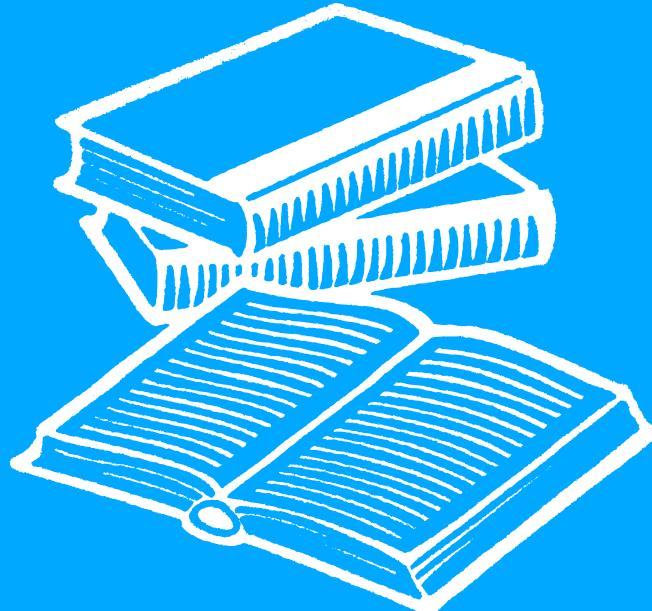
STUDY-HALL

PROGETTO DI INTELLIGENZA
ARTIFICIALE
A.A. 2022/2023

STUDIA DOVE, QUANDO E QUANTO VUOI.



COSA TRATTEREMO



1

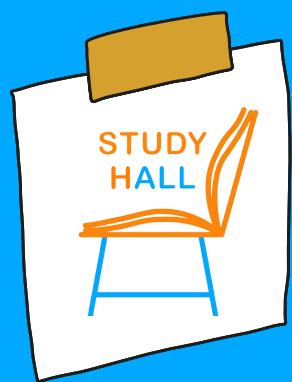
ESIGENZE ED
OBIETTIVI DEL
PROGETTO

2

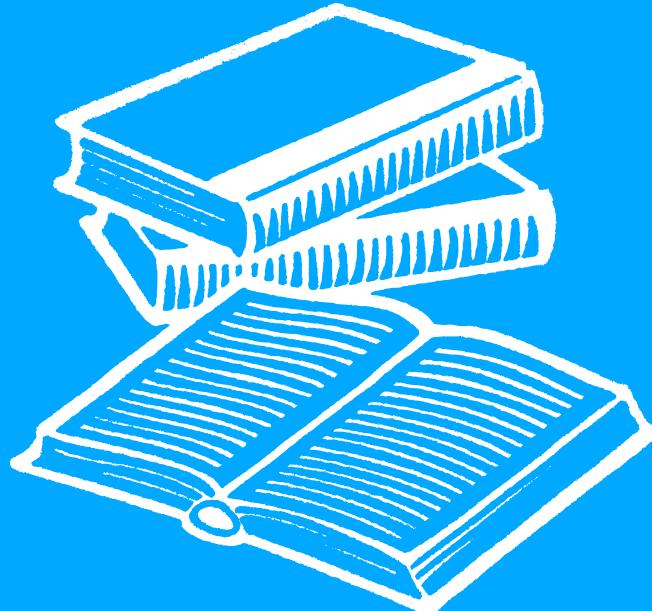
PROGETTAZIONE
DELL'ALGORITMO

3

CONSIDERAZIONI E
CONCLUSIONI



COSA TRATTEREMO



1

ESIGENZE ED
OBIETTIVI DEL
PROGETTO

2

PROGETTAZIONE
DELL'ALGORITMO

3

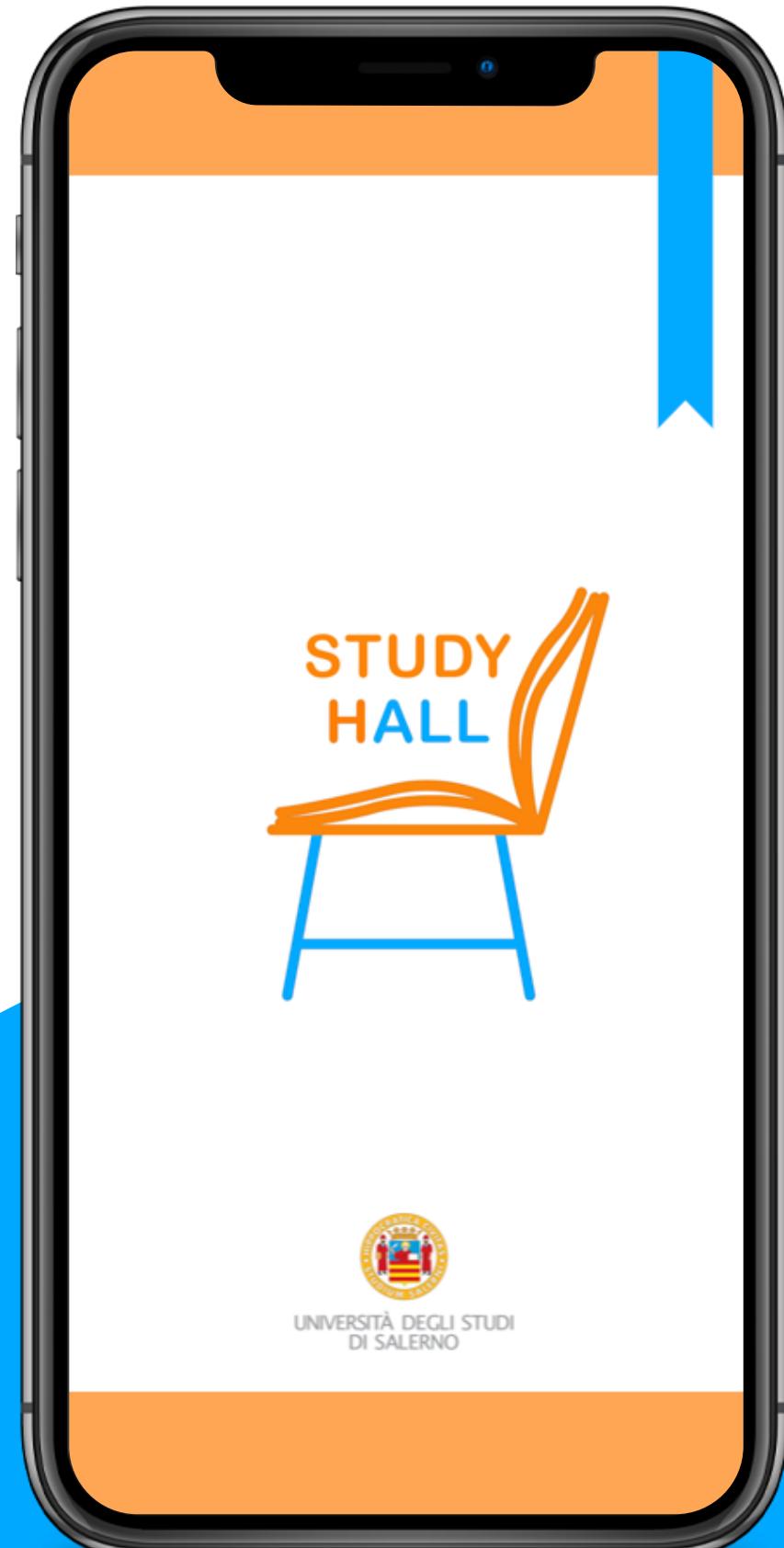
CONSIDERAZIONI E
CONCLUSIONI

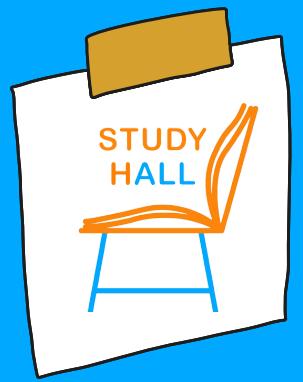


INTRODUZIONE

HAI BISOGNO DI UN **POSTO TRANQUILLO** IN
CUI STUDIARE MA LE AULE STUDIO SONO
SEMPRE PIENE?

STUDY HALL TI PERMETTE DI ASSICURARE
UN POSTO PER STUDIARE IN OGNI AULA
STUDIO DI UNISA





LASCIA CHE STUDY HALL
PRENOTI PER TE I GIORNI
MIGLIORI PER POTER STUDIARE
IN TRANQUILLITÀ

DOVE

SCEGLI L'EDIFICIO
DELLA TUA FACOLTÀ

QUANDO

SCEGLI I GIORNI PIÙ
ADATTI PER STUDIARE

QUANTO

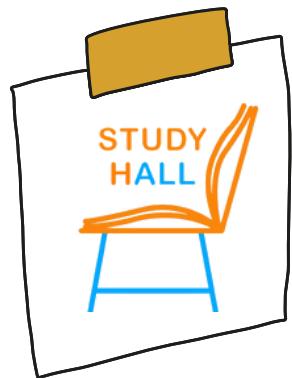
SCEGLI QUANTE ORE
VORRESTI STUDIARE



OBIETTIVI

DARE LA POSSIBILITÀ AGLI STUDENTI DI SCEGLIERE
GIORNI, FASCE ORARIE E IL NUMERO DI ORE CHE
VOGLIONO DEDICARE ALLO STUDIO

LASCIARE QUANTI PIÙ POSTI LIBERI NEL MINOR
NUMERO POSSIBILE DI AULE STUDIO IN PREVISIONE
DI INCENTIVI PRENOTAZIONI FUTURE.



SPECIFICHE P.E.A.S

PERFORMANCE

ENVIRONMENT

ACTUATORS

SENSORS

VIENE VALUTATO SE L'AGENTE RIESCE AD ALLOCARE LE PRENOTAZIONI IN MODO TALE CHE RISPETTINO LE RICHIESTE DELLO STUDENTE

TUTTE LE COMBINAZIONI DI PRENOTAZIONI DATI GLI INPUT FORNITI DAGLI UTENTI

SLOT (FASCE ORARIE) DI PRENOTAZIONE TRA CUI SCEGLIERE NELL'ARCO DELLA SETTIMANA

INSIEME DEI DATI FORNITI IN INPUT FINALIZZATI AL RAGGIUNGIMENTO DELL'OBBIETTIVO



CARATTERISTICHE DELL'AMBIENTE

**COMPLETAMENTE
OSSERVABILE**

DETERMINISTICO

STATICO

DISCRETO

SINGOLO



COSA TRATTEREMO

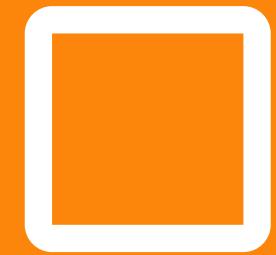


1



ESIGENZE ED
OBIETTIVI DEL
PROGETTO

2



PROGETTAZIONE
DELL'ALGORITMO

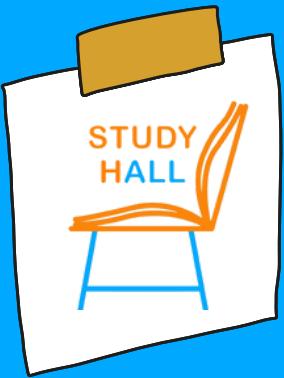
3



CONSIDERAZIONI E
CONCLUSIONI

**COME RISOLVIAMO IL
PROBLEMA?**

ALGORITMI GENETICI

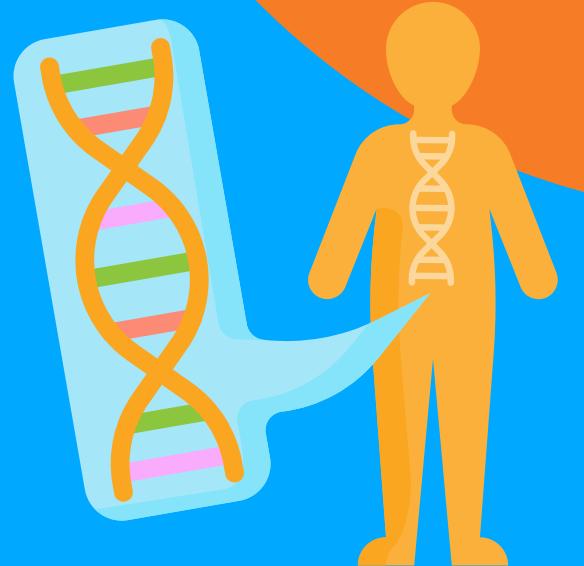


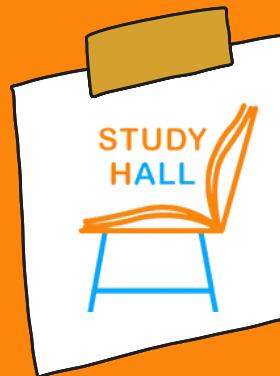
ALGORITMI GENETICI (GA)

COSTITUISCONO UNA PROCEDURA AD ALTO LIVELLO ISPIRATA ALLA GENETICA

Un GA evolve una **popolazione** di individui (soluzioni candidate) producendo di volta in volta soluzioni sempre **migliori** rispetto ad una funzione obiettivo, fino a raggiungere l'**ottimo** o un'altra condizione di terminazione.

**COSA SONO GLI
ALGORITMI GENETICI**



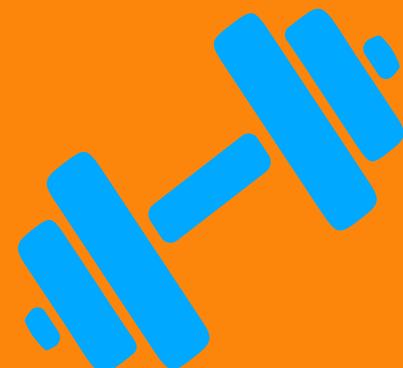


FUNZIONE DI FITNESS

COS'È

**FUNZIONE IN GRADO DI ASSOCIARE UN
VALORE AD OGNI SOLUZIONE ED È
FONDAMENTALE PER LA DEFINIZIONE DI
UN GA.**

**MISURA IL LIVELLO DI ADEGUATEZZA
DEGLI INDIVIDUI**



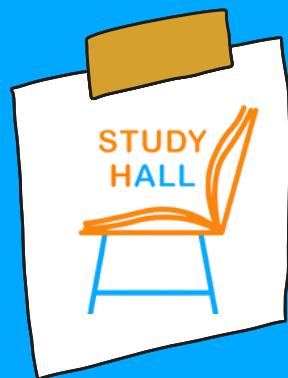
$$f(y) = x_a - y_a$$

E PER STUDY HALL?

**MASSIMIZZA IL NUMERO DI ORE ASSEGNAME
AD UNO STUDENTE RISPETTO A QUELLE
RICHIESTE DA ESSO**

$$f(a_i) = \begin{cases} 1, & foa = fos \\ 0, & \text{altrimenti} \end{cases}$$

**CERCA DI FAR COINCIDERE LE FASCE
ORARIE ASSEGNAME DALL'AGENTE CON
QUELLE DELLO STUDENTE**



FUNZIONE DI FITNESS

MA ABBIAMO DUE FUNZIONI DI FITNESS,
COME FACCIAMO CONFRONTI TRA LE DUE?

PREFERENCE SORTING!

TECNICA CHE PERMETTE DI FORNIRE UN
ORDINAMENTO TOTALE TRA GLI INDIVIDUI

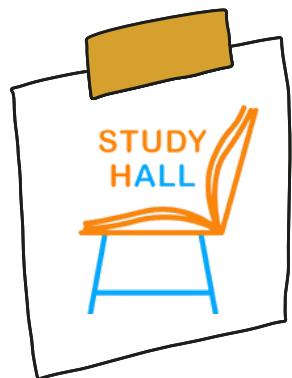
QUAL È LA MIGLIORE?

$$f(a_i) = \begin{cases} 1, & foa = fos \\ 0, & \text{altrimenti} \end{cases}$$

*foa = fos
altrimenti*

FOA: FASCE ORARIE ASSEGNAME DALL'AGENTE

FOS: FASCE ORARIE DELLO STUDENTE



PARAMETRI DEL GA

INIZIALIZZAZIONE

LA POPOLAZIONE INIZIALE SARÀ
COMPLETAMENTE CASUAL

RAPPRESENTAZIONE INDIVIDUI

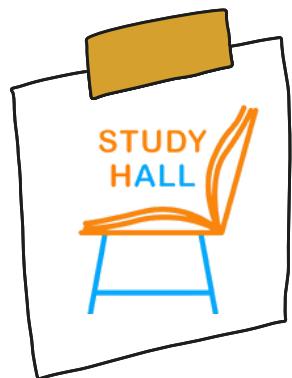
OGNI GENE SARÀ UNA SINGRA DEL TIPO
"ABC, AD, ALM, DFC"..."
DOVE LE LETTERE INDICANO GLI STUDENTI E OGNI
SOTTOSTRINGA TRA "," INDICA UNA FASCIA ORARIA

SIZE POPOLAZIONE

LA SIZE DELLA POPOLAZIONE
SARÀ 4

SIZE MATING POOL

LA SIZE DEL MATING POOL
SARÀ 4



PARAMETRI DEL GA

ALGORITMO DI SELEZIONE

È STATO IMPLEMENTATO L'ALGORITMO
"ROULETTE WHEEL SELECTION"

ALGORITMO DI CROSSOVER

SONO STATI IMPLEMENTATI
"SINGLE POINT CROSSOVER"
"TWO POINT CROSSOVER"

ALGORITMO DI MUTAZIONE

È STATO IMPLEMENTATO L'ALGORITMO
"INVERSION MUTATION"

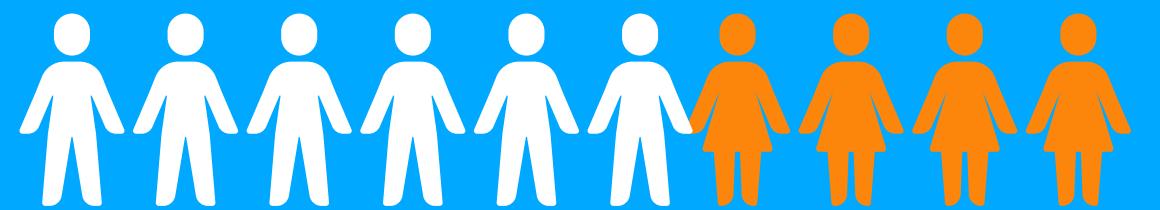
STOPPING CONDITION

SONO STATE SCELTE LE CONDIZIONI
"ASSENZA DI MIGLIORAMENTI"
"TEMPO DI ESECUZIONE"



ALGORITMO DI SELEZIONE

UNA COPIA DI ALCUNI INDIVIDUI
NELLA SUCCESSIVA GENERAZIONE;
SI CERCANO SOLUZIONI SEMPRE
MIGLIORI



ROULETTE WHEEL SELECTION

GLI INDIVIDUI RICEVONO UNA
PROBABILITÀ DI SELEZIONE PARI AL
VALORE DELLA LORO FITNESS RELATIVA
ALL'INTERA POPOLAZIONE



ALGORITMO DI CROSSOVER

TWO POINT CROSSOVER

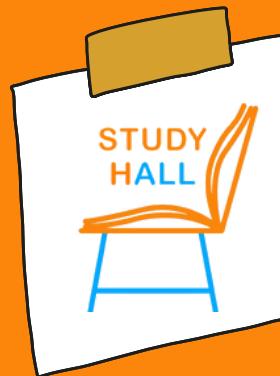
VA A CONSIDERARE DUE PUNTI E SI PROCEDE
ALLA GENERAZIONE DI DUE FIGLI TRAMITE
SCAMBIO DI CROMOSOMI

SINGLE POINT CROSSOVER

SI SELEZIONA UN PUNTO DEL PATRIMONIO GENETICO
DEI GENITORI E SI PROCEDE ALLA GENERAZIONE DI
DUE FIGLI TRAMITE SCAMBIO DI CROMOSOMI

L'ACCOPPIAMENTO DI INDIVIDUI
(PARENTS) PER CREARNE DI NUOVI
(OFFSPRINGS), DA AGGIUNGERE ALLA
NUOVA GENERAZIONE

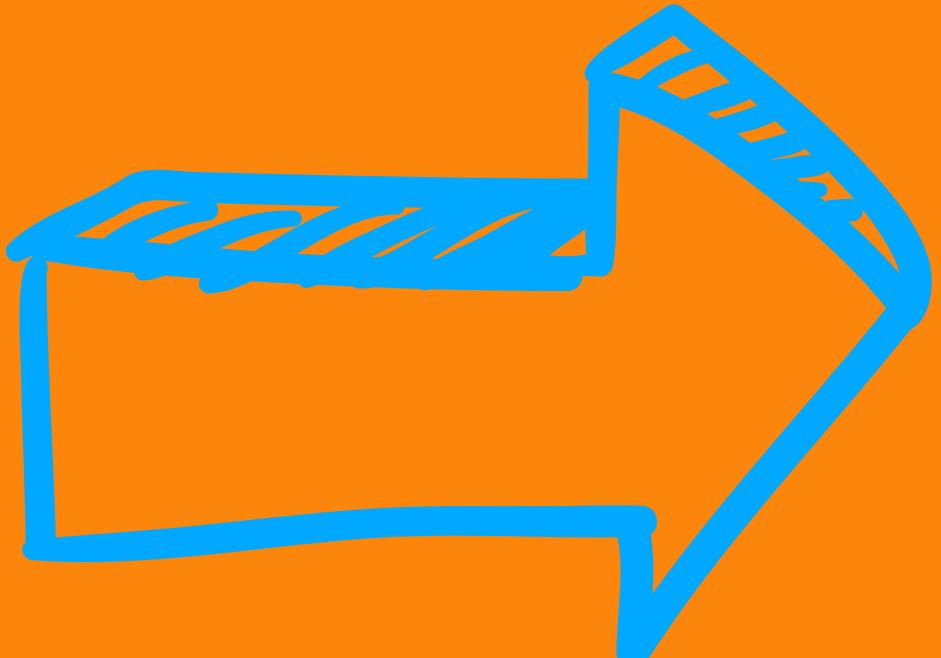
SPERIMENTAZIONE EMPIRICA QUALE SCEGLIAMO?



ALGORITMO DI CROSSOVER

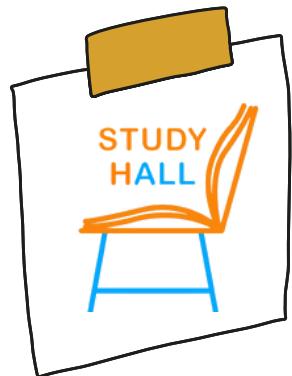
SINGLE POINT CROSSOVER

SI SELEZIONA UN PUNTO DEL PATRIMONIO GENETICO
DEI GENITORI E SI PROCEDE ALLA GENERAZIONE DI
DUE FIGLI TRAMITE SCAMBIO DI CROMOSOMI



L'ALGORITMO SINGLE POINT
RIESCE A FORNIRE SOLUZIONI
MIGLIORI DEL TWO POINT!



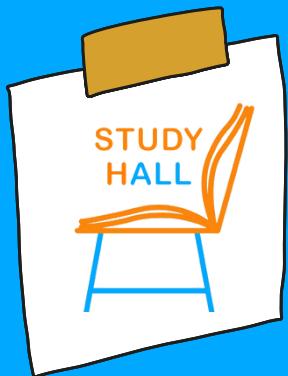


ALGORITMO DI MUTAZIONE

**MODIFICA CASUALE DI
ALCUNE PARTI DEL CORREDO
GENETICO**



INVERSION MUTATION
**SI SCEGLIE UN SUBSET DI GENI IN
MODO CASUALE E LO SI RIBALTA.**



COSA TRATTEREMO



1

ESIGENZE ED
OBIETTIVI DEL
PROGETTO

2

PROGETTAZIONE
DELL'ALGORITMO

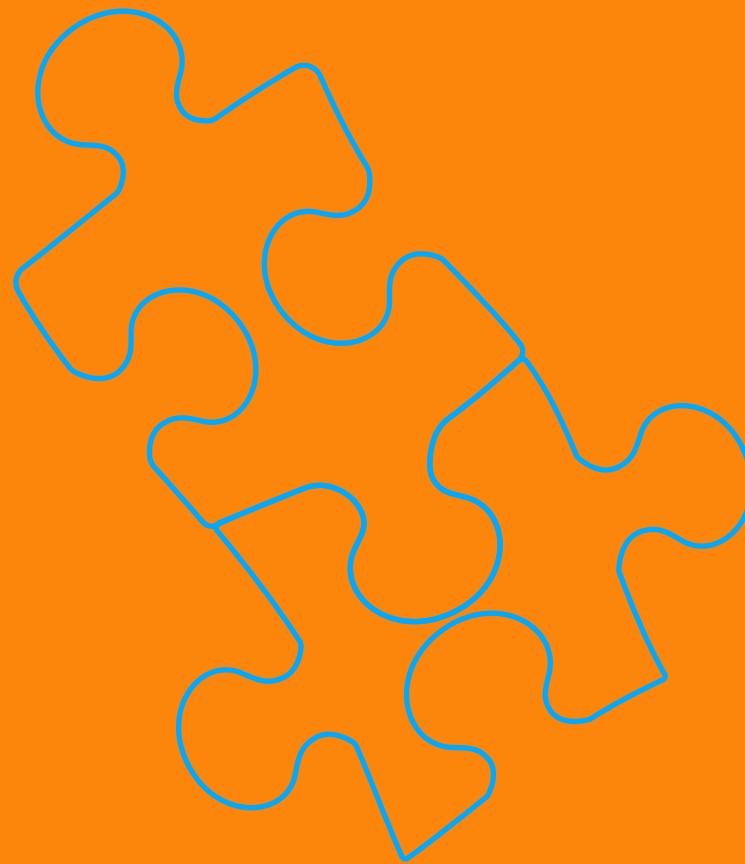
3

CONSIDERAZIONI E
CONCLUSIONI



CONCLUSIONI

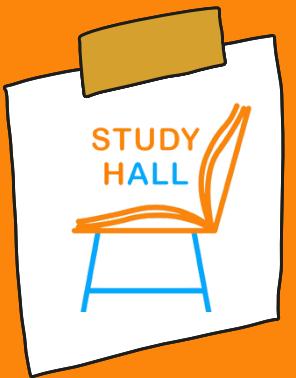
IL SINGLE POINT, PER IL
NOSTRO ALGORITMO,
FUNZIONA MEGLIO DEL TWO
POINT



L'ALGORITMO FORNISCE SOLUZIONI DI
GRAN LUNGA MIGLIORI CON
L'AUMENTARE DEL TEMPO DI
ESECUZIONE

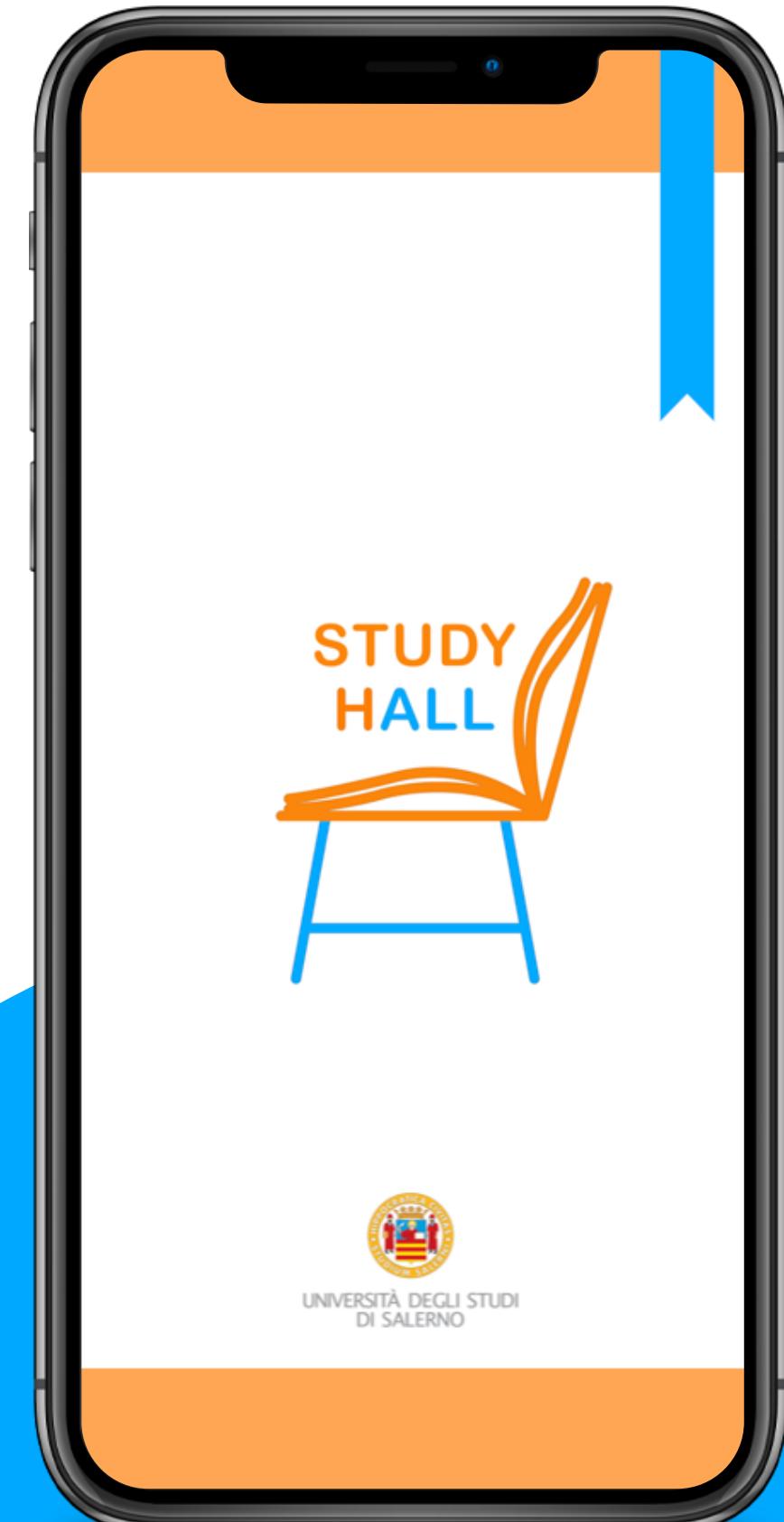
L'ALGORITMO FUNZIONA MEGLIO
ELIMINANDO LA PRIMA FUNZIONE DI
FITNESS; QUESTA VERRÀ COMUNQUE
USATA COME VALUTAZIONE
DELL'OUTPUT FINALE

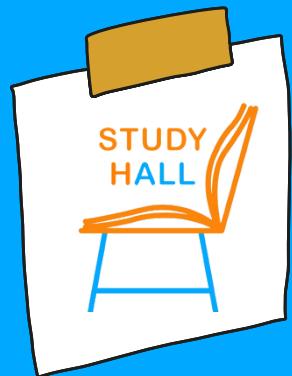




CONCLUSIONI

**IN CONCLUSIONE, POSSIAMO DIRE
CHE È STATA UNA *BELLA*
ESPERIENZA POTER APPLICARE
QUESTI ARGOMENTI AD UN
PROGETTO VERO E PROPRIO.**





MEMBRI DEL TEAM



**MARIA IMMACOLATA
COLELLA**

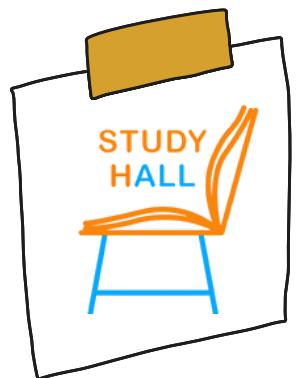


MICHELE MATTIELLO



CARLO SORRENTINO

GRAZIE!



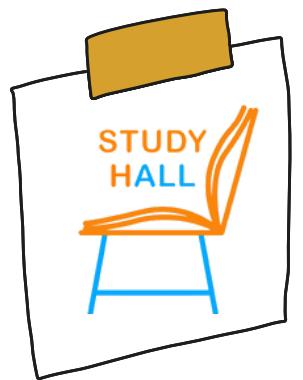
INPUT UTENTE

Studente	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Ore totali
A	1110	0001	1001	0000	1100	10
B	0101	0000	1111	0011	1100	14
C	1111	1111	1111	1111	0000	22

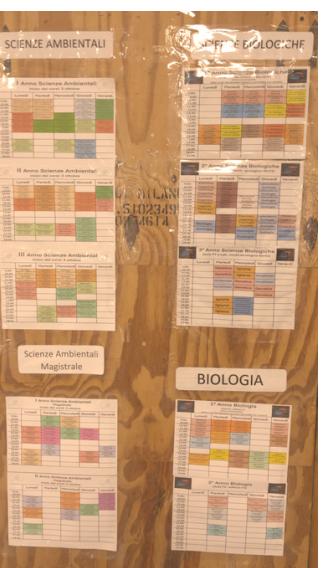
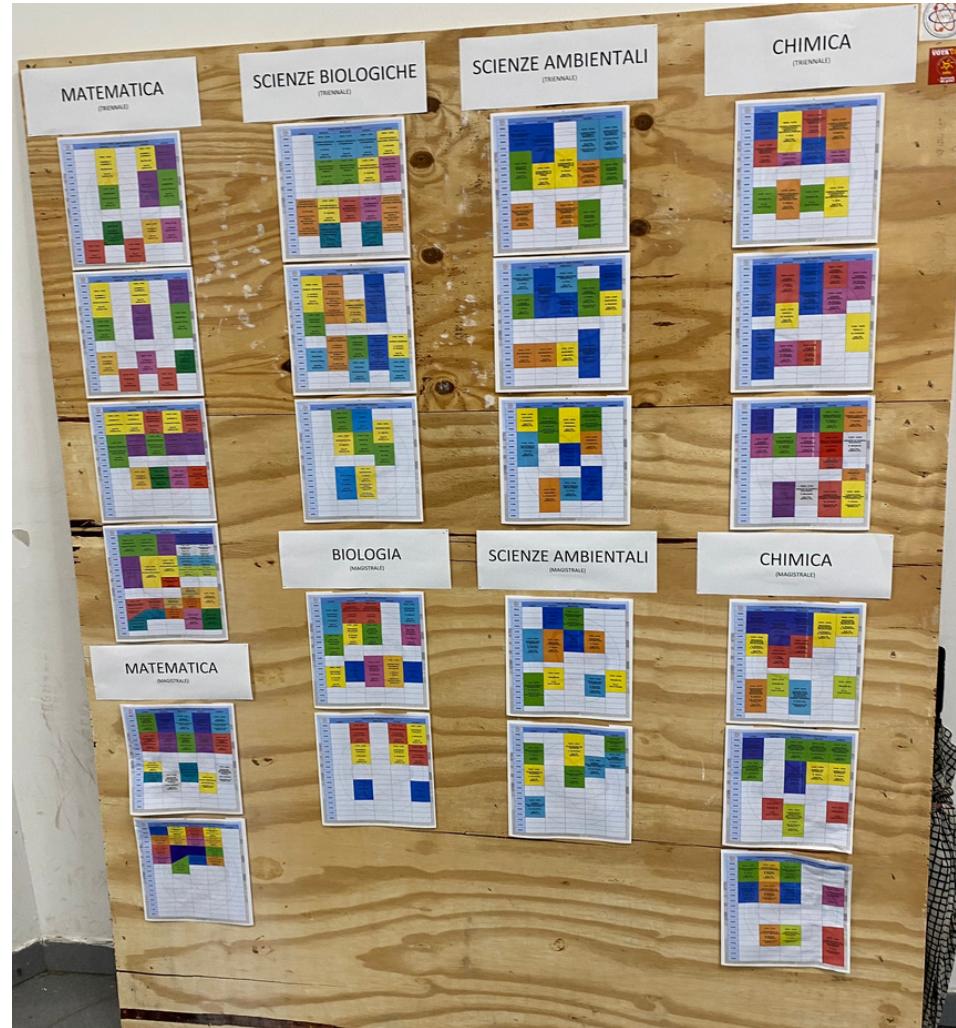
LETTERE: CODIFICA DEGLI STUDENTI, LETTERE MAIUSCOLE E MINUSCOLE E CIFRE

SEQUENZA 4 BIT: 1: DISPONIBILITÀ, 0: NON DISPONIBILITÀ

ORE TOTALI: NUMERO DI ORE CHE LO STUDENTE VUOLE DEDICARE ALLO STUDIO



INPUT UTENTE - LE FONTI



Studente	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì
A	1110	0001	1110	0001	0001
B	1100	0000	0001	0000	0001
C	0001	0011	1001	1111	0011
D	0001	0001	0001	0001	1111
E	0001	1110	0001	1110	0001
F	0001	0001	1000	0001	1100
G	0010	0001	0001	1000	0011
H	0000	0001	0001	0011	0001
I	0011	0001	0001	0001	0001
J	1111	0001	0001	1001	1111
K	0011	0101	0001	0001	1111
L	0011	1001	0010	0000	0000
M	0000	0001	0001	0000	0000
N	0000	0010	0001	0010	0001
O	0001	0001	0001	1101	0001
P	1000	0001	0011	1010	1101
Q	0010	0001	0001	0111	0101
R	0011	0011	0011	0011	0011
S	1111	0001	1111	0011	0001
T	1001	1111	0011	1011	0011
U	0011	1101	0000	0011	0001
V	0011	0001	0001	1111	1000
W	0001	0011	0011	1111	0001
X	0001	0011	0001	1111	0001
Y	1100	0101	0001	0100	1000
Z	0001	0011	1000	0000	0001
a	1100	0000	0001	0000	0001
b	1111	0000	0000	1001	1111
c	1111	0001	1111	0011	0001
d	0011	1101	0000	0011	0001



OUTPUT AGENTE

AC, ABC, AC, BC,C,C,C,AC,ABC, BC, BC, ABC, C,C, BC, BC, AB, AB,,,

LETTERE:

CODIFICA DEGLI STUDENTI, LETTERE MAIUSCOLE,
MINUSCOLE E CIFRE

VIRGOLE:

SEPARAZIONE DELLE FASCE ORARIE, OGNUNA
COMPRESA TRA LE VIRGOLE



INIZIALIZZAZIONE POPOLAZIONE

**IL PRIMO METODO GENERA
RANDOMICAMENTE LE ASSEGNAZIONI
DEGLI STUDENTI PER OGNI FASCIA ORARIA.**

**IL SECONDO METODO COMPONE LE
ASSEGNAZIONI PER UN'AULA STUDIO.**

IL TERZO METODO COMPONE UN GENE.

**IL QUARTO METODO CREA LA
POPOLAZIONE INZIALE.**



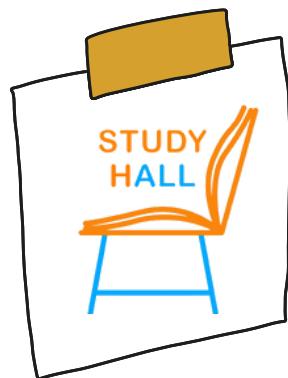
```
import random

def generateOutputTime(studenti: list):
    time = ""
    for i in studenti:
        value = random.uniform(0, 1)
        value = i if value > 0.5 else ''
        time += value
    return time

def generateOutputStudyroom(studenti: list, timeSlot):
    output = ""
    for i in range(timeSlot):
        output+=generateOutputTime(studenti)+","
    return output

def generateGene(studenti: list, numberStudyroom, timeSlot):
    output = ""
    for i in range(numberStudyroom):
        output+=generateOutputStudyroom(studenti, timeSlot)
    return output[0:len(output)-1]

def generatePopulation(studenti: list, sizePopulation: int, timeSlot: int, numberStudyroom: int):
    output = []
    for i in range(sizePopulation):
        output.append(generateGene(studenti, numberStudyroom, timeSlot))
    return output
```



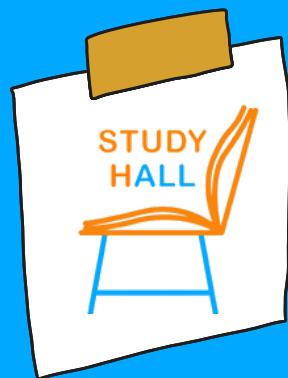
FUNZIONE DI FITNESS

$$f(y) = x_a - y_a$$

**SCORRE OGNI GENE A CUI ESSA VIENE APPLICATA E VERIFICA
QUANTE ORE L'AGENTE HA ASSEGNATO AD UNO STUDENTE
RISPETTO ALLE ORE CHE QUEST'ULTIMO HA INSERITO IN INPUT
INIZIALMENTE**



```
def fitnessQuantita(gene: str, quantita: list, studenti: list): #range [0, 1]
    output = 0
    for j in range(len(studenti)):
        i = 0
        for c in gene:
            if c == studenti[j]:
                i+=1
        output += i-int(quantita[j]) if int(quantita[j]) > i else int(quantita[j]) - i
    if output < 0:
        output = 1 / (-1 * output)
    if output == 0:
        output = 1
    return output
```

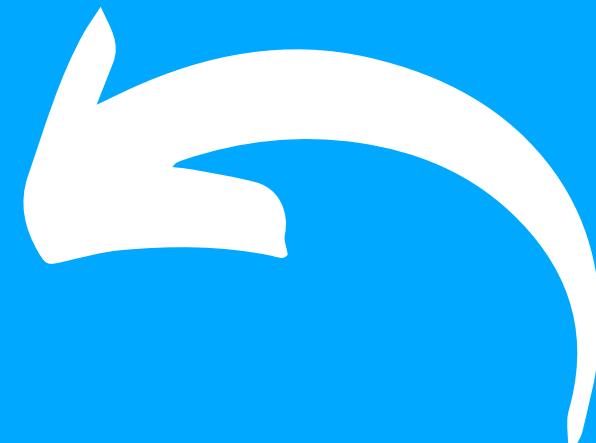


FUNZIONE DI FITNESS

```
def fitnessFasceOrarie(gene: str, fasceStudenti: list, studenti: list, studyroom: int):
    output = 0
    separateGene = gene.split(',')
    studyroomList = []
    n = int(len(separateGene) / studyroom)
    for i in range(studyroom):
        studyroomList.append(separateGene[i*n: (i+1)*n])
    for k in range(len(studenti)):
        for i in range(len(studyroomList[0])):
            count = 0
            for j in range(len(studyroomList)):
                if studenti[k] in studyroomList[j][i]:
                    count += 1
            if (count == 1 and fasceStudenti[k][i] == 1):
                output += 5
            elif count == 0 and fasceStudenti[k][i] == 0:
                output += 3
            elif count == 1 and fasceStudenti[k][i] == 0:
                output += 1
            elif fasceStudenti[k][i] == 1 and count == 0:
                output += 2
            elif output - count > 0:
                output -= count
    return output
```

$$f(a_i) = \begin{cases} 1, \\ 0, \end{cases}$$

*foa = fos
altrimenti*



**DIVIDE IL GENE IN TRE PARTI,
OGNUNA PER OGNI AULA STUDIO.
UNA VOLTA OTTENUTE LE TRE
STRINGHE (UNA PER OGNI AULA
STUDIO) E CONTROLLA QUANTE
VOLTE UNO STUDENTE È STATO
ASSEGNATO AD UN'AULA STUDIO E IN
QUALE FASCIA ORARIA**

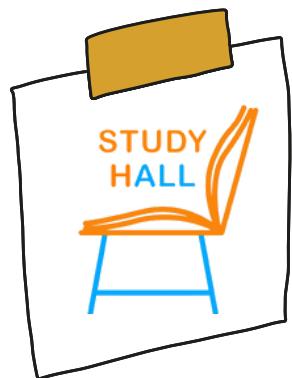


ROULETTE WHEEL SELECTION

L'ALGORITMO ROULETTE WHEEL SELECTION, PER OGNI FUNZIONE DI FITNESS ASSEGNA UNA PROBABILITÀ DI SELEZIONE AD OGNI GENE DELLA POPOLAZIONE E, TRAMITE LA LIBRERIA NUMPY, SCEGLIE I GENI MIGLIORI DA PORTARE AVANTI

```
import numpy as np
def rouletteWheelSelection(resultsFitnessFasceOrarie, resultsFitnessQuantita, population):
    newPopulation = []
    totallyFasceOrarieValue = sum(resultsFitnessFasceOrarie)
    #if totallyFasceOrarieValue > 0:
    probabilitiesFasceOrarie = [value/totallyFasceOrarieValue for value in resultsFitnessFasceOrarie]
    for i in range(len(population)):
        newPopulation.append(np.random.choice(population, p=probabilitiesFasceOrarie))
    #else:
    #    totallyQuantitaValue = sum(resultsFitnessQuantita)
    #    probabilitiesQuantita = [value/totallyQuantitaValue for value in resultsFitnessQuantita]
    #    for i in range(len(population)):
    #        newPopulation.append(np.random.choice(population, p=probabilitiesQuantita))
    return newPopulation
```



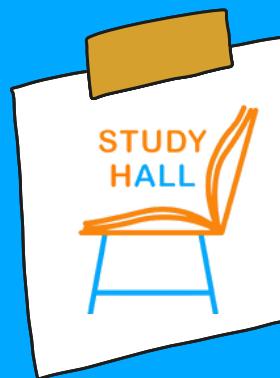


SINGLE POINT CROSSOVER

L'ALGORITMO SINGLE POINT CROSSOVER SPLITTA UN GENE PER OGNI EVENIENZA DEL CARATTERE ',', SUCCESSIVAMENTE OTTERREMO UNA LISTA, IN CUI OGNI CELLA INDICA UNA FASCIA ORARIA; INFINE VIENE DIVISA LA LISTA A METÀ ED EFFETTUIAMO IL CROSSOVER COL SECONDO GENE



```
def singlePoint(population: list):
    part = (len(population[0].split(',')) - 1)/2
    firstCut = int(part)
    newPopulation = []
    for i in range(0, len(population), 2):
        firstGene = population[i].split(',')
        secondGene = population[i + 1].split(',')
        newFirstGene = firstGene[0:firstCut] + firstGene[firstCut:]
        newSecondGene = secondGene[0: firstCut] + secondGene[firstCut:]
        newPopulation.append(','.join(newFirstGene))
        newPopulation.append(','.join(newSecondGene))
    return newPopulation
```



TWO POINT CROSSOVER



L'ALGORITMO TWO POINT CROSSOVER SPLITTA UN UN GENE PER OGNI EVENIENZA DEL CARATTERE "", SUCCESSIVAMENTE OTTERREMO UNA LISTA, IN CUI OGNI CELLA INDICA UNA FASCIA ORARIA; INFINE VIENE DIVISA LA LISTA IN DUE PUNTI ED EFFETTUIAMO IL CROSSOVER COL SECONDO GENE

```
def twoPoint(population: list):
    part = (len(population[0].split(',')) - 1)/3
    firstCut = int(part)
    secondCut = int(2 * part)
    newPopulation = []
    for i in range(0, len(population), 2):
        firstGene = population[i].split(',')
        secondGene = population[i + 1].split(',')
        newFirstGene = firstGene[0:firstCut] + secondGene[firstCut: secondCut] + firstGene[secondCut:]
        newSecondGene = secondGene[0: firstCut] + firstGene[firstCut: secondCut] + secondGene[secondCut:]
        newPopulation.append(','.join(newFirstGene))
        newPopulation.append(','.join(newSecondGene))
    return newPopulation
```



INVERSION MUTATION

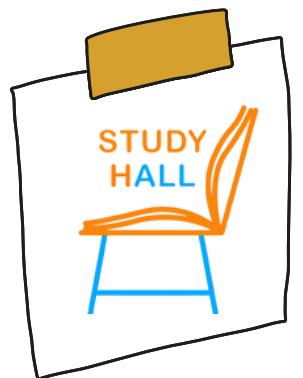
L'ALGORITMO INVERSION MUTATION GENERA RANDOMICAMENTE DUE INDICI DEL GENE CREANDONE UN SUBSET, LO INVERTE E LO RIMETTE NELLA SUA POSIZIONE ALL'INTERNO DEL GENE



```
from random import randint

def inversion(population: list):
    newPopulation = []
    for i in range(len(population)):
        newGene = population[i].split(',')
        while True:
            firstValue = randint(0, len(newGene))
            secondValue = randint(0, len(newGene))
            if firstValue < secondValue:
                break
        part = newGene[firstValue: secondValue]
        part.reverse()
        newPopulation.append(', '.join(newGene[0:firstValue] + part + newGene[secondValue:]))

    return newPopulation
```



OUTPUT

```
ESPLORA RISORSE ... PROBLEMI OUTPUT TERMINALE CONSOLE DI DEBUG zsh - code + ×
```

EDITOR APERTI

- main.py code M
- dataset.csv code X
- mutation.py code
- crossover.py code
- fitnessFunction.py c... M
- selection.py code M
- initialization.py code M

FIAMODULE

- > __pycache__
- < code >
- > __pycache__
- crossover.py
- dataset.csv
- fitnessFunction.py M
- initialization.py M
- main.py M
- mutation.py M
- selection.py M
- > Documentation

```
A-- --- A-- B-- A--  
C-- B-- --- AC--  
B-- B-- C-- AB- C--  
C-- AB- A-- BC- B--  
  
fasce orarie: 1 quantita: 0.01694915254237288  
code git:(main) ✘ python3 main.py  
ABC --- AC- C-- B--  
--- C-- ABC A-- B--  
C-- BC- AB- AB- B--  
C-- A-- C-- A-- BC-  
  
A-- C-- B-- --- ABC  
--- AC- AB- AC- C--  
AC- AC- BC- ABC C--  
ABC --- A-- AB- BC-  
  
--- BC- ABC B-- AB-  
--- --- BC- AC-  
ABC B-- ABC C-- ---  
B-- BC- C-- A-- C--  
  
fasce orarie: 6 quantita: 0.015384615384615385  
--- A-- AB- --- C--  
ABC AB- ABC AB- AC-  
B-- AC- --- BC- ABC  
AB- C-- A-- BC- ---  
  
B-- AB- B-- AB- AC-  
C-- BC- B-- --- A--  
BC- A-- AB- AB- ---  
B-- --- B-- AB- A--  
  
BC- --- BC- ABC ABC  
BC- B-- C-- B-- B--  
AB- BC- C-- ABC AC-  
A-- B-- B-- C-- AB-  
  
fasce orarie: 30 quantita: 0.014705882352941176  
code git:(main) ✘ python3 main.py  
A-- B-- BC- BC- ABC  
AC- AC- AC- ABC ---  
AB- AC- AB- --- ABC  
AC- AC- ABC AB- ---  
  
A-- ABC C-- B-- AC-  
AB- B-- C-- --- ---  
C-- AB- AB- --- B--  
AB- ABC A-- AC- ---
```

Riga 4, colonna 29 Spazi: 4 UTF-8 LF Testo normale ↻ ↻