Alejandro Martinez - 21430

Configuración Inicial:

Define la frase objetivo que el algoritmo debe generar.

```
TARGET_PHRASE = "to be or not to be"
POPULATION_SIZE = 200
MUTATION_RATE = 0.01
ALPHABET = "abcdefghijklmnopqrstuvwxyz "
```

Establece el tamaño máximo de la población.

Configura la tasa de mutación como un porcentaje (0-100%) para controlar la diversidad genética.
Inicialización de la Población: Genera una población inicial de secuencias de caracteres de igual longitud que la frase objetivo. Cada secuencia debe ser creada de manera aleatoria utilizando letras del alfabeto y espacios.

```
population: List[str] = [random_string(len(TARGET_PHRASE)) for i in
range(POPULATION_SIZE)]

generation = 0
best_individual = ''
best_fitness = 0
```

Ejecución del Algoritmo:

Implementa el ciclo principal del algoritmo genético, que incluye las siguientes etapas:

Selección: Evalúa y selecciona individuos de la población en función de su similitud a la frase objetivo.

```
def selection(population: List[str], fitnesses: List[int]):
    total_fitness = sum(fitnesses)
    pick = random.uniform(0, total_fitness)
    current = 0
    for individual, fit in zip(population, fitnesses):
        current += fit
        if current > pick:
            return individual
```

Reproducción: Combina pares de individuos seleccionados para crear descendencia.

```
def crossover(parent1: str, parent2: str):
    crossover_point = random.randint(0, len(parent1) - 1)
```

```
    child = parent1[:crossover_point] + parent2[crossover_point:]
    return child
```

Mutación: Aplica la tasa de mutación para introducir pequeños cambios aleatorios en los genes de algunos individuos.

```
def mutate(individual: str):
    mutated = ''.join(
        (char if random.random() > MUTATION_RATE else random.choice(ALPHABET))
        for char in individual
    )
    return mutated
```

Sustitución: Reemplaza la población anterior con la nueva generación creada.

```
population = new_population
```

Convergencia:

Monitorea la población en cada generación para verificar si alguna secuencia ha alcanzado la frase objetivo. Registra el número de generaciones necesarias para lograr la convergencia.

```
population = new_population
best_individual = max(population, key=fitness)
best_fitness = fitness(best_individual)
```

Visualización:

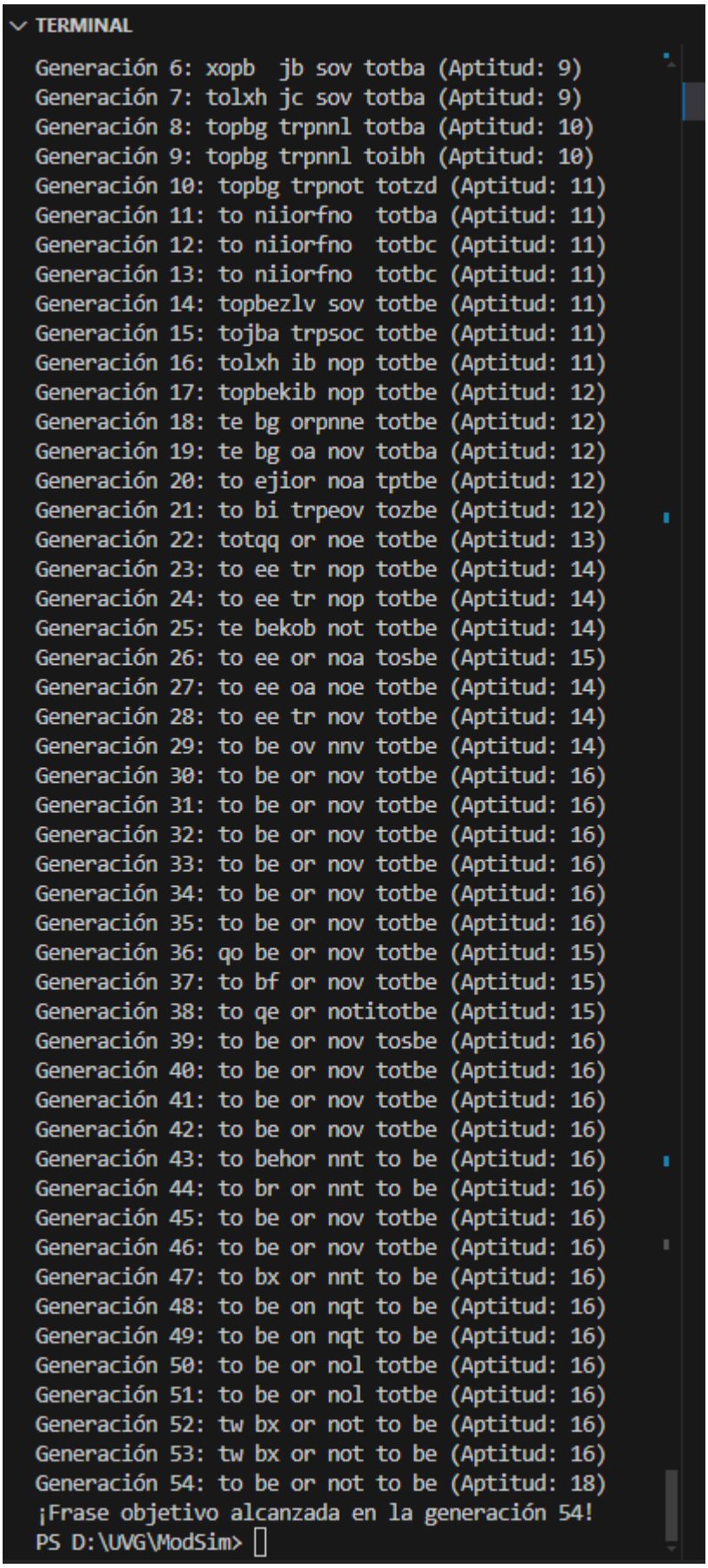
Implementa una función de visualización que muestre el progreso de la población en cada generación. Esto puede incluir la impresión de la secuencia más apta en cada generación.

```
print(f"Generación {generation}: {best_individual} (Aptitud: {best_fitness})")
```

Incluir el número de generaciones requeridas para converger en la frase objetivo.

55

```
POPULATION_SIZE = 200
MUTATION_RATE = 0.01
```



Incluir las características de las funciones de Selección, Reproducción y Mutación.

Selección

```
def selection(population: List[str], fitnesses: List[int]):
    total_fitness = sum(fitnesses)
    pick = random.uniform(0, total_fitness)
    current = 0
    for individual, fit in zip(population, fitnesses):
        current += fit
        if current > pick:
            return individual
```

Evalúa y va seleccionononando las cadenas/individuos de la población dependiendo de la similitud que tenga con el target.

Reproducción:

```
def crossover(parent1: str, parent2: str):
    crossover_point = random.randint(0, len(parent1) - 1)
    child = parent1[:crossover_point] + parent2[crossover_point:]
    return child
```

Es la union de ambas cadenas, el primer padre se corta en un porcentaje aleatorio y el segundo padre la otra parte restante. El hijo esta compuesto por un porcentaje aleatorio de el valor de un padre y la otra parte el valor del otro.

Mutación

```
def mutate(individual: str):
    mutated = ''.join(
        (char if random.random() > MUTATION_RATE else random.choice(ALPHABET))
        for char in individual
    )
    return mutated
```

Introduce un valor aleatorio dentro de la cadena dependiendo del valor de mutacion como threshold. Esto ayuda a la "evolución" y generacion de variantes nuevas.

¿Qué sucede si el tamaño máximo de la población es demasiado bajo?

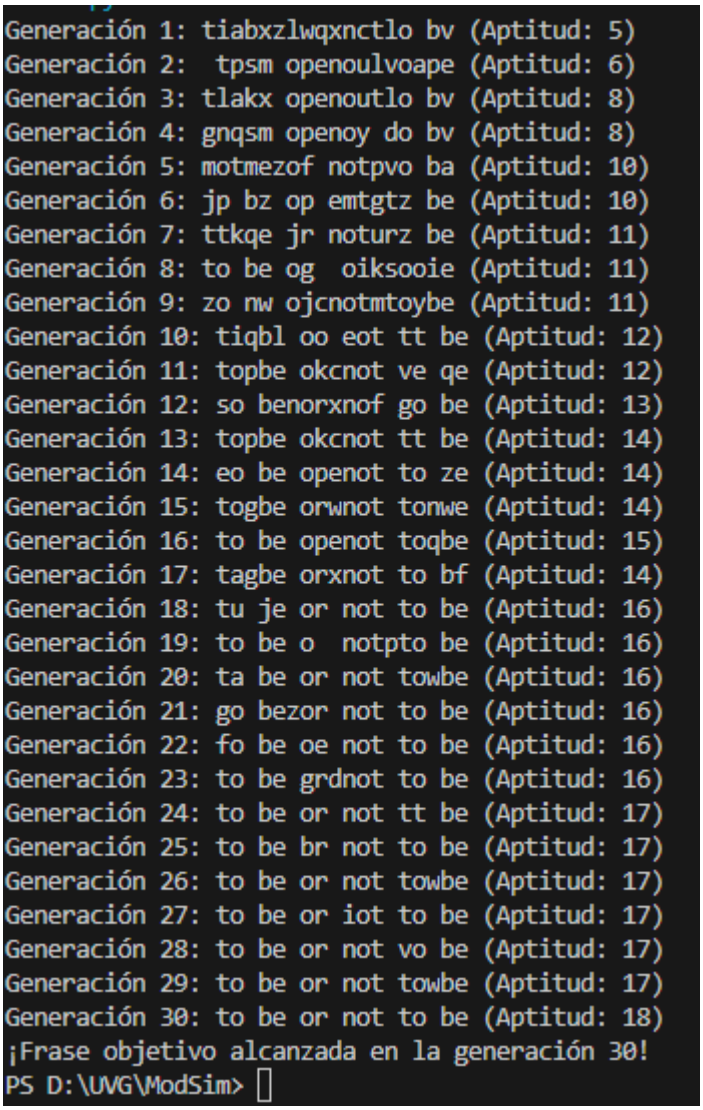
La ejecución es muy rápida pero dado que las opciones/población de la cual puede seleccionar son muy limitadas es probable que se tome demasiado tiempo o se quede generando palabras aleatorias y nunca "evolucione".

```
POPULATION_SIZE = 5
```

¿Qué ocurre si el tamaño máximo de la población es demasiado alto?

La ejecucuión se volverá más lenta, y por el tamaño del set de datos (una oracion corta), no habrá mucho beneficio en cuanto a mejroas en la cantidad de generaciones necesarias.

```
POPULATION_SIZE = 10000
```



¿Qué sucede si la tasa de mutación es del 0%?

Inicialmente no hay mucha diferencia pero las generaciones cercanas al resultado no van a mutar y será muy dificil que convergan en la respuesta correcta. (Se queda en aptitud 14-15)

```
MUTATION_RATE = 0
```

```
Generación 1: endbzlwzehvt movpe (Aptitud: 5)
Generación 2: kndbzlwzehvt movpe (Aptitud: 5)
Generación 3: toptelwzehvt movpd (Aptitud: 6)
Generación 4: toptelwzehvt movbg (Aptitud: 7)
Generación 5: wo bt beinbtcuodus (Aptitud: 7)
Generación 6: toptereo rctgbovbf (Aptitud: 7)
Generación 7:  x pelhvinbn sovbe (Aptitud: 7)
Generación 8: wo xg r  lttbga pe (Aptitud: 7)
Generación 9: ts pe oljngtyv cbg (Aptitud: 8)
Generación 10: sw xg r  nltgbovbs (Aptitud: 7)
Generación 11: tsrbbmoglnln jolbe (Aptitud: 8)
Generación 12: toioelty nlg vo pe (Aptitud: 9)
Generación 13: to pe yp vzt movbf (Aptitud: 10)
Generación 14: topte rglnoj bo be (Aptitud: 11)
Generación 15: ul pe rglnoj bo be (Aptitud: 10)
Generación 16: to belty nlg vosbf (Aptitud: 10)
Generación 17: wo beleo xzt go be (Aptitud: 11)
Generación 18: wo beleo xzt go be (Aptitud: 11)
Generación 19: to belrk vzt go be (Aptitud: 12)
Generación 20: to bereo nltylo be (Aptitud: 12)
Generación 21: to pelty nct bo be (Aptitud: 12)
Generación 22: to be aljrzt bo be (Aptitud: 12)
Generación 23: wo be to ngt vo be (Aptitud: 13)
Generación 24: wo be to ngt vo be (Aptitud: 13)
Generación 25: to be to ngt vo be (Aptitud: 14)
Generación 26: to be hv nlt bo be (Aptitud: 14)
Generación 27: to be hv nlt bo be (Aptitud: 14)
Generación 28: to be hv nlt bo be (Aptitud: 14)
Generación 29: to belov nlt bo be (Aptitud: 14)
Generación 30: to belov nlt bo be (Aptitud: 14)
Generación 31: to be ty ngt vo be (Aptitud: 14)
Generación 32: to be to nzt bo be (Aptitud: 14)
Generación 33: to be eo nzt bo be (Aptitud: 14)
Generación 34: to be oy nlt bo be (Aptitud: 15)
Generación 35: to be oy nlt bo be (Aptitud: 15)
Generación 36: to be oy nlt bo be (Aptitud: 15)
```

¿Qué pasa si la tasa de mutación es del 50%?

(Se queda en aptitud 5-6) Los valores internos se mutan casi siempre por lo que hay demasiada aleatoriedad y no puede mantenerse en curso a llegar a la frase objetivo, es posible con una poblacion mas alta probablemente pero es una tasa de reemplazo aleatorio demasiado alta para que la frase cambie nuevamente a algo aleatorio con cada generacion.

```
MUTATION_RATE = 0.2
```

```
Generación 452: adebigorlfotwtjqjf (Aptitud: 6)
Generación 453: tydbebdennsthiwvbv (Aptitud: 6)
Generación 454: epobkggr nidwqogba (Aptitud: 6)
Generación 455: ro ukuvrxmntuzzzbo (Aptitud: 5)
Generación 456: vofbyjorjxat uvjr  (Aptitud: 6)
Generación 457: tu aq qehhidkloabh (Aptitud: 5)
Generación 458: igsvgoguonotqfp bm (Aptitud: 5)
Generación 459: lisvgqguonotqfp bi (Aptitud: 5)
Generación 460: tsmzf guonotdtp uv (Aptitud: 7)
Generación 461: tsmjm gucoxtdtw uv (Aptitud: 5)
Generación 462: ehlbeamwzxojjtc uv (Aptitud: 5)
Generación 463: phlbeiqwzxob dsxbg (Aptitud: 5)
Generación 464:  qmteokoetog tpsbh (Aptitud: 5)
Generación 465: jmwbeacbptog xwwbt (Aptitud: 5)
Generación 466: vbywnwoo eutwte ee (Aptitud: 6)
Generación 467: tozbsbvxqnwjsfcnbx (Aptitud: 5)
Generación 468: gglom hrxrox ttqlr (Aptitud: 5)
Generación 469: oollenagy vt to km (Aptitud: 7)
Generación 470: rm oerbfcmotmxa ap (Aptitud: 5)
Generación 471: dtakewqrkcbbktowbf (Aptitud: 5)
Generación 472: ttablwqlbcbbktowbf (Aptitud: 5)
Generación 473: co bmwbrqtntmyfbfv (Aptitud: 5)
Generación 474: tkoskhzcnnntayosbe (Aptitud: 6)
Generación 475: teqbfnp hndsztiirz (Aptitud: 4)
Generación 476: dodi  t cxebitj ej (Aptitud: 4)
Generación 477: ck drntrcpooitr jj (Aptitud: 5)
Generación 478: xilse otlkbw toxnm (Aptitud: 6)
Generación 479:  ilse okdkskqtoxnm (Aptitud: 5)
Generación 480: yoibldfr oqtffxxwo (Aptitud: 5)
Generación 481: yoibrdzr fqtffdsx  (Aptitud: 5)
Generación 482: gogby sr fqxyfovbe (Aptitud: 8)
Generación 483: twdyz orowmegtsaje (Aptitud: 6)
Generación 484: mj aziow nwk nomff (Aptitud: 6)
Generación 485: ngqwi sr blnyga br (Aptitud: 5)
Generación 486: sypgnmornldb ywpbe (Aptitud: 5)
Generación 487: thjyq arkmdb ywcbe (Aptitud: 6)
Generación 488: tldbu arkmvb gwifi (Aptitud: 5)
Generación 489: ikmm  qg nltbjolkq (Aptitud: 5)
Generación 490: io nxsznznltbgolkf (Aptitud: 5)
Generación 491: obexw dlkqhtctwfpe (Aptitud: 4)
Generación 492: to fxobs tiedtyfez (Aptitud: 5)
Generación 493: se ferom ziirhwjbv (Aptitud: 5)
```

¿Qué ocurre si la tasa de mutación es del 100%?

(Se queda en aptitud 3-4) Los valores internos se mutan siempre por lo que hay demasiada aleatoriedad y no puede mantenerse en curso a llegar a la frase objetivo.

```
MUTATION_RATE = 1.0
```

```
Generación 289: kuzjcwhrxmqk joyum (Aptitud: 3)
Generación 290: tgfgtuofpwotasksyw (Aptitud: 4)
Generación 291: eoelgqcejepjsbo ya (Aptitud: 3)
Generación 292: mzixehwy nbunbobfw (Aptitud: 4)
Generación 293: vzqbipgokijfstq zc (Aptitud: 3)
Generación 294: qokqe cevtctydpgnp (Aptitud: 4)
Generación 295: zssfezcdzsotqdaqdp (Aptitud: 3)
Generación 296: thljhyjryn ouckuai (Aptitud: 3)
Generación 297: pesbmfzogocjltklbc (Aptitud: 3)
Generación 298: ogbb wxbgjxw zozuy (Aptitud: 3)
Generación 299: aoawlxoyvurcqroqbb (Aptitud: 4)
Generación 300: padamquu viuamotbh (Aptitud: 3)
Generación 301: kuihe potneqsphnds (Aptitud: 3)
Generación 302: vinbdohduykv rs qa (Aptitud: 3)
Generación 303: cwfhekycfpotytob n (Aptitud: 5)
Generación 304: tkffunhezvotlctcop (Aptitud: 3)
Generación 305: zcjbsdbrqncdftqgyi (Aptitud: 4)
Generación 306: hboknzgx dtfntuome (Aptitud: 3)
Generación 307: peo uww  loxktoghu (Aptitud: 4)
Generación 308: xs agnooicmpucom e (Aptitud: 4)
Generación 309: povyzkfgdnjnonv zo (Aptitud: 3)
Generación 310: cw meohqueogutvafd (Aptitud: 4)
Generación 311: musynbaaodu jqdebe (Aptitud: 2)
Generación 312: njv l pmstxtcpnwbq (Aptitud: 3)
Generación 313: yztyxwsrtzor nsrbd (Aptitud: 4)
Generación 314: lnbbegmqwsmsjvd fb (Aptitud: 3)
Generación 315: tjxmd krcpogpwuwtx (Aptitud: 4)
Generación 316: zjylebgrnputvt enx (Aptitud: 4)
Generación 317: hozzm kutsg  tdqmw (Aptitud: 4)
Generación 318: omjbvhaiymmd soxrg (Aptitud: 3)
Generación 319:  jaho of oh alm gc (Aptitud: 4)
Generación 320: yn dh kyyqoqhikzgo (Aptitud: 3)
Generación 321: be hgzhrjoomavdvzd (Aptitud: 3)
Generación 322:  wzbwhzrwnrk rnewq (Aptitud: 4)
Generación 323: kt sioofcefxveoqay (Aptitud: 3)
Generación 324: nqkmshfpifmt tpali (Aptitud: 3)
Generación 325: bnkbhhyt nvlnxraap (Aptitud: 3)
Generación 326: psdom nkyqokltysye (Aptitud: 4)
Generación 327: idcruhow wltltcrhs (Aptitud: 4)
Generación 328: ttham ck tzvsxcsha (Aptitud: 3)
Generación 329: uoqellkh wopffrxeu (Aptitud: 3)
Generación 330: ftqdkdykptis tzfbj (Aptitud: 3)
Generación 331: frgbo hvcnkhyyogyz (Aptitud: 4)
Generación 332: syrkvjyrfeutkiyvae (Aptitud: 3)
Generación 333: ld bkhopqanpokd gy (Aptitud: 4)
```