# Proyecto 2

Estrategias de diseño de algoritmos

**Objetivos**

* Implementar algoritmos de redes de flujo de forma eficiente en el lenguaje c# o python, utilizado las ventajas de OO
* Identificar la medición de la eficiencia de los algoritmos y la programación de algoritmos utilizando buenas prácticas y estructuración del código en capas.
* Determinar diversas estrategias para el diseño e implementación de algoritmos.

**Definición**

La empresa TEC-Games requiere utilizar un mecanismo de inteligencia artificial para la creación de escenarios para sus futuros lanzamientos de videos juegos. Esto a raíz de que sus gastos se están elevando y no se realizan una optimización de los recursos de la empresa.

El objetivo final de este proyecto es conseguir una población de individuos que solventen las soluciones de al menos dos juegos.

Se deben aplicar 2 estrategias de diseño de algoritmos:

* Algoritmos genéticos
* Algoritmos probabilísticos

Debe utilizar la librería retro de Python para la aplicación de los juegos. Puede usar los juegos “SuperMarioBros-Nes” y “Airstriker” para la ejecución de los algoritmos genéticos y algoritmos probabilísticos.

Recursos

* <https://github.com/ejimenezdelgado/proyecto_analisis_2022>
* <https://analyticsindiamag.com/hands-on-guide-reinforcement-learning-openai-gymretro/>
* <https://youtu.be/XDMokVOdjb4>

**Mediciones sobre los algoritmos**

Para evaluar la efectividad y eficiencia de los algoritmos, se implementarán pruebas y mediciones para cada uno de las dos estrategias utilizadas, considerando diferentes tamaños del dataset hasta determinar si los algoritmos son tratables o intratables en grandes cantidades de datos (para efectos del proyecto se considera un tamaño máximo de 1000 registros).

A continuación, una descripción de los procesos de medición y registro de datos que se implementarán para cada uno de los algoritmos:

1. Medición empírica, estadísticas básicas

* Tiempo de ejecución en segundos/milisegundos. Recuerden utilizar una precisión de 3 decimales.
* Comparaciones (menor “<”, mayor “>”, igual que “==”, distinto de “!=”) realizadas en tiempo de ejecución.
* Asignaciones realizadas en tiempo de ejecución.
* Cantidad de líneas del código que componen los algoritmos
* Cantidad total de líneas ejecutadas en cada prueba

1. Cálculo del factor de crecimiento y clasificación de los algoritmos en notación: Theta, O Grande, y Omega, ***según sus datos de entrada***.
2. Clasificación en notación O Grande ***según sus comparaciones y asignaciones***.
3. Medición analítica de cada uno de los códigos de los algoritmos y determinar su clasificación en notación O Grande.

Para registrar los datos anteriores, se adjuntan las siguientes tablas.

Medición empírica

* 1. Nombre del algoritmo **Genético.**

| Operaciones | Tamaños del arreglo | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 50 | 100 | 200 | 500 | 1000 |
| Asignaciones | 43 | 190 | 400 | 802 | 3751 | 11727 |
| Comparaciones | 64 | 313 | 633 | 1267 | 4999 | 14353 |
| Cantidad de líneas ejecutadas | 132 | 617 | 1267 | 2537 | 13684 | 44932 |
| Tiempo de ejecución (seg) | 0.649 | 1.483 | 2.586 | 4.686 | 39.682 | 147.322 |
| Cantidad de líneas del código | 30 | | | | | | |

**Crear una tabla igual para el algoritmo Probabilístico**

Determinar el factor de crecimiento

3.1 Algoritmo **Genético**

| Talla | | Factor talla | Factor Asig | Factor Comp | Factor Cantidad de líneas ejecutadas | Factor Tiempo de ejecución |
| --- | --- | --- | --- | --- | --- | --- |
| De 10 a 50 | Datos aleatorios | 5 | 190/43=4,418604651 | 313/64=4,890625 | 617/132=4,67424242 | 1,483/0,649=2,28505393 |
| De 50 a 100 | Datos aleatorios | 2 | 400/190=2,105263158 | 633/313=2,022364217 | 1267/617=2,0534846 | 2,586/1,483=1,74376264 |
| De 100 a 200 | Datos aleatorios | 2 | 802/400=2,005 | 1267/633=2,001579779 | 2537/1267=2,0023678 | 4,686/2,586=1,81206497 |
| De 500 a 1000 | Datos aleatorios | 2 | 11727/3751=3,126366302 | 14353/4999=2,871174235 | 13684/2537=3,28354282 | 147,322/39,682=3,71256489 |
| De 50 a 500 | Datos aleatorios | 10 | 3751/190=19,74210526 | 4999/313=15,97124601 | 13684/617=22,178282 | 39,682/1,483=26,7579231 |
| De 100 a 1000 | Datos aleatorios | 10 | 11727/43=272,7209302 | 14353/64=224,265625 | 44932/132=340,393939 | 147,322/0,649=226,998459 |

|  |  |  |
| --- | --- | --- |
| Clasificación del comportamiento de las **asignaciones** | Cuadrático |  |
| Clasificación del comportamiento de las **comparaciones** | Cuadrático |

|  |  |
| --- | --- |
| Clasificación según su entrada de los datos use la notación  O Grande según corresponda | |
| **Entrada de los datos** | **Aleatorios** |
| Clasificación | Cuadrático |

Crear una tabla igual para el **algoritmo Probabilístico**

Medición analítica

* 1. Algoritmo **Genético**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Código fuente  Solo se analiza el código del método **Probabilístico**. | Medición de líneas ejecutadas en el peor de los casos  (línea por línea) |
| while nivel1:  env.render()  probDisparo = random()  action = [0, 0, 0, 0, 0, 0, randint(0,1), randint(0,1), 0, 1, 0, 0]  ob, rew, done, info = env.step(action)  dormir()  if inicio and info['gameover'] ==9:  inicio = False  for i in range(len(movCorrectos)):  env.render()  ob, rew, done, info = env.step(movCorrectos[i])  dormir()  if info['gameover'] ==2:  break  if info['gameover'] ==9:  movCorrectos.append(action)  if info['gameover'] ==2:  inicio = True  obs = env.reset()  if (len(movCorrectos)>100):  for e in range(20):  movCorrectos.pop(len(movCorrectos)-e-1)  if probDisparo <= 0.30:  action = [randint(0, 1), 0, 0, 0, 0, 0, randint(0, 1), randint(0, 1), 0, 1, 0, 0]  ob, rew, done, info = env.step(action)  dormir()  if info['gameover'] == 9:  movCorrectos.append(action)  if done:  obs = env.reset() | n+1  n+1  n+1  n+1  n+1  n+1+1  n+1  n+1  n\*n  n(n+1)  n(n+1)  n(n+1+1)  n(n+1)  n(n+1)  n+1  n+1  n+1  n+1  n+1  n+1  n\*n  n(n+1)  n+2  n+1  n+1  n+1+1  n+1  n+1  n+1  n+1 |
| Total (la suma de todos los pasos) | 8n^2+29n+25 |
| Clasificación en notación O Grande | Cuadrático |

***Crear una tabla igual para el*** “**Probabilistico”**

Medición gráfica:

### Grafique el comportamiento de los 2 algoritmos en un gráfico cada uno, tomando en cuenta solo la variable de comparaciones y en un segundo gráfico las asignaciones. Para cada algoritmo obtenga la medición de 10 pares ordenados, a una escala pequeña. Recuerde indicar el nombre al gráfico, nombre a los ejes y serie clara para cada una de las gráficas.

### Realice un análisis del gráfico, compárelo con el comportamiento de la clasificación asignada a los algoritmos en la medición empírica.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

**Operaciones**

Crear los 2 algoritmos correctamente de forma eficiente y presentar en consola las respuestas para los tamaños que sean posibles. TODOS en un solo proyecto, no se revisará más de un proyecto.

**A la hora de ejecutarlo el usuario final no introducirá ningún dato.** Los datos de entrada para los grafos los genera el mismo programa en tiempo de ejecución.

Mostrar en pantalla los resultados de medición (calculados correctamente) cuando el tamaño de la prueba lo permita.

***Nota:*** Tomar en cuenta todas las aclaraciones que se dieron en día de entrega y discusión de este proyecto.

**La documentación Externa debe generar un paper**

**Título**: debe llamar la atención y a la vez comunicar el tema del trabajo.

**Resumen**: debe contener la motivación para el trabajo, el problema planteado, el objetivo de la investigación, la propuesta para resolver el problema, la evaluación para

probar/validar la propuesta y por último los resultados obtenidos.

**Introducción**: plantea el problema en relación con los desafíos del mundo real, el objetivo de la investigación debe aparecer literal al planteado en el resumen, la propuesta para resolver el problema debe ser incluida junto a las preguntas de investigación que se desea abordar, se debe agregar una vista general de la evaluación para probar/validar la propuesta y por último un párrafo que describa la estructura del documento.

**Trabajos relacionados**: es una revisión de literatura que permita aportar conocimiento para desarrollar el proyecto de software. Se debe hacer una búsqueda de artículos que aborden temáticas que aporten información para plantear una propuesta de solución al problema que se aborda.

**Solución propuesta**: es la contribución científica del proyecto. Se debe generalizar el trabajo en un nivel de abstracción posicionando éste como un marco de trabajo.

* La última solución, indique cuales son las estructuras utilizadas.
* Lógica de cómo se realizó cada algoritmo, indique todas las mejoras realizadas a los algoritmos para mejorar su eficiencia.

**Metodología de la investigación**: describe los métodos que se utilizaron, el entorno del estudio de investigación y los temas de investigación. Considere esto como un libro de cocina para alguien que está replicando lo que hizo.

**Títulos a adjuntar:**

* Diagrama de flujo de las estrategias utilizadas
* Estrategia de ramificación escogida
* Tipos de cruces genéticos realizados
* Tipo de mutación que se aplicó
* Función objetivo de la maximización de recursos
* Adjunte las tablas de todas las mediciones realizadas a sus algoritmos (empírica, analítica y la medición gráfica).
* Adjunte el cálculo del factor de crecimiento.

**Evaluación**: debe incluir la evaluación para probar/validar la propuesta planteada.

* Analice los datos obtenidos, a que se debe que un algoritmo sea mejor que otro. Indique las características o ventajas de cada algoritmo sobre el otro.

**Conclusiones y trabajos futuros**: indica los impactos de su propuesta y marcar el territorio de su trabajo futuro.

* Resultados finales, indique que partes están completas, cuales defectuosos, y cuáles no se realizaron y el porqué, que aspectos se pueden mejorar.

### Documentación Interna

Fecha de inicio y Fecha última modificación.

Usar la Notación CamelCase:

* + - LowerCamelCase: Cada variable deberá definirse con su primera palabra en minúscula y las siguientes con su primera letra en mayúsculas. Ejemplos:
      * primerNombre, numeroPuestos, correoPersonal.

Descripción para cada estructura (clase o struct) y su uso en el programa.

Describir cada función e instrucciones dentro de estas.

### Aspectos Administrativos

* La tarea debe programarse en lenguaje c# o python.
* El desarrollo de este trabajo se puede realizar en pareja como máximo.
* La fecha de entrega de la tarea: 14 de noviembre del 2022
* Si se encuentra copia en alguna parte del proyecto la calificación será de cero para todos los implicados.
* Debe entregarse la tarea a través del tec-Digital, si tiene virus o si se encuentra mal identificando se rebajarán puntos por descuido del estudiante. Si no abre el proyecto o la documentación, se calificará con una nota de 0.
* Se recomienda que se comience a trabajar desde hoy. Recuerde existen horas de consulta.
* Las mediciones en tiempo de ejecución deben coincidir con las presentadas en la documentación, en caso contrario se rebajarán los puntos correspondientes.

### Calificación General.

60% Documentación que incluye las diferentes mediciones.

40% Programación.

* Se rebajarán puntos por utilizar código o programación poco eficiente, redundante o innecesaria. Por ejemplo: declaración de variables sin usar, o de un tipo de datos incorrecto, en vez de un boolean un char o string.