Tipo de artículo: Artículo original

Una demostración del Principio de Invariancia para el estudio de la Complejidad Temporal de los Algoritmos

A demonstration of the Principle of Invariance for the study of the Time Complexity of Algorithms

Kirenia Maldonado Zuñiga [1*], https://orcid.org/0000-0002-3764-5633
Omar Antonio Quimis Sánchez [2], https://orcid.org/0000-0001-8341-7722
Franklin Jhimmy Toala Arias [3], https://orcid.org/0000-0003-2639-8208
Elio Armando Cables Fernández [4], https://orcid.org/0000-0002-3193-3968

[1] Doctorando en Tecnología de la Información y Comunicación, Universidad Nacional de Piura, Perú. Magister en Ciencias de la Educación, Licenciada en Educación Informática. Docente de la carrera de Tecnologías de la Información de la Facultad Ciencias Técnicas de la Universidad Estatal del Sur de Manabí. Jipijapa, Manabí, Ecuador. kirenia.maldonado@unesum.edu.ec
[2] Ingeniero en Contabilidad y Auditoría - Analista de Sistemas - Master en Contabilidad y Auditoría, Docente de la Carrera de Tecnologías de la Información de la Universidad Estatal del Sur de Manabí. Jipijapa – Manabí – Ecuador. E-mail: omar.quimis@unesum.edu.ec
[3] Magíster en Educación y Desarrollo Social. Docente de la carrera de Tecnologías de la Información de la Facultad Ciencias Técnicas de la Universidad Estatal del Sur de Manabí. Jipijapa, Manabí, Ecuador. franklin.toala@unesum.edu.ec
[4] Doctorando en Tecnología de la Información y Comunicación, Universidad Nacional de Piura, Perú. Máster en Matemática Aplicada e Informática para la Administración, Universidad "Oscar Lucero Moya", Holguín, Cuba. Ingeniero Informático, Universidad "Oscar Lucero Moya", Holguín, Cuba. Docente de Matemática y TIC´s en la UEP "Glenn Doman", Manta, Ecuador. E-mail: ecablesf@posgrado.unp.edu.pe

*** Autor para correspondencia:** kirenia.maldonado@unesum.edu.ec

Resumen

La complejidad de un algoritmo es determinada por el costo en recursos como, tiempo, memoria, entre otros, que emplea el agente de cómputo para su ejecución. El Principio de Invariancia proporciona una métrica a priori de la complejidad temporal de un algoritmo, independiente de las características del agente de cómputo. Una prueba formal, rigurosa del mismo, no es encontrada con regularidad en la literatura sobre la problemática. En el trabajo se presenta una demostración donde se verifica la veracidad de la conjetura presentada en el Principio de Invariancia. Así como una definición no intuitiva de algoritmo. Dando a conocer su importancia en las ciencias de la computación, para dar soluciones a diferentes problemas obteniendo un resultado eficiente. Este estudio fue realizado por docentes de la carrera de Tecnologías de la Información de la facultad Ciencias Técnicas de la Universidad Estatal del Sur de Manabí. Se utilizaron métodos científicos del nivel teóricos y empíricos los que facilitaron el análisis, síntesis y búsqueda de información relacionada al tema tratado. Como resultado se conoció que los algoritmos permiten dar solución a problemas complejos, en poco tiempo con el uso de las tecnologías. Se concluye que la demostración presentada deja abierta la posibilidad de generalizar el Principio de Invariancia de la complejidad temporal de los algoritmos a conjuntos de información continua.

Palabras clave: algoritmo; información; lenguaje; complejidad; principio de invariancia.

Abstract

The complexity of an algorithm is determined by the cost in resources such as time, memory, among others, that the computational agent uses for its execution. The Invariance Principle provides an a priori metric of the time complexity of an

algorithm, independent of the characteristics of the computing agent. A formal, rigorous proof of it is not regularly found in the literature on the problem. In the work a demonstration is presented where the veracity of the conjecture presented in the Principle of Invariance is verified. As well as a non-intuitive definition of algorithm. Making known its importance in computer science, to provide solutions to different problems obtaining an efficient result. This study was carried out by professors of the Information Technologies career of the Faculty of Technical Sciences of the State University of the South of Manabí. Scientific methods of the theoretical and empirical level were used, which facilitated the analysis, synthesis, and search for information related to the subject matter. As a result, it was known that algorithms allow solving complex problems, in a short time with the use of technologies. It is concluded that the demonstration presented leaves open the possibility of generalizing the Invariance Principle of the temporal complexity of the algorithms to sets of continuous information.

Keywords: *algorithm; information; language; complexity; principle of invariance.*

Recibido: 15/12/2022
Aceptado: 28/02/2023
En línea: 12/03/2023

Introducción

En ciencias de la computación, el análisis de algoritmos es una parte importante, en la búsqueda de algoritmo para resolver un problema. Es posible tener muchos algoritmos para resolver un problema, el reto es elegir el más eficiente.

La complejidad temporal es el número de operaciones que realiza un algoritmo para completar su tarea (considerando que cada operación dura el mismo tiempo). El algoritmo que realiza la tarea en el menor número de operaciones se considera el más eficiente en términos de complejidad temporal. Sin embargo, la complejidad espacial y temporal se ve afectada por factores como el sistema operativo y el hardware.

El aprendizaje, es visto como un proceso activo, en el proceso de alojamiento y asimilación de la información, resultan vitales, la experiencia directa, las equivocaciones y la búsqueda de soluciones. La manera en la que se presenta la información es de suma importancia. Cuando la información es introducida como una forma de respuesta para solucionar un problema, funciona como una herramienta, no como un hecho arbitrario y solitario.En este sentido el aprendizaje es completo, auténtico y positivo. (Velazquez, R. V., Maldonado Zúñiga , K., (2021).

En el estudio del Cómputo (la Informática en general), las siguientes preguntas se presentan desde el primer momento:

¿Qué problemas pueden ser resueltos por un algoritmo (programa) de cómputo?

¿Qué soluciones pueden ser obtenidas eficientemente en la práctica?

Estas preguntas parecen muy vagas. Lograr respuestas precisas a las preguntas formuladas obliga a:

- Definir rigurosamente que se entiende por algoritmo.

- Caracterizar los datos y el agente de cómputo que los procesará.
- Medir la efectividad práctica del algoritmo.

El concepto de algoritmo procede del nombre del matemático del Asia Central Al-Jwarizmi en el siglo IX, se empleaba en las matemáticas de la época para designar las reglas de las cuatro operaciones aritméticas, convirtiéndose en una de las nociones básicas de la Matemática. La disciplina que estudia los algoritmos en todos sus aspectos se ha nombrado " Algorithmique" por Donald Knuth, en un texto francés de 1976. Pero la comunidad internacional ha aceptado el término " Algorítmica".

En la actualidad los algoritmos están presentes en todas las esferas de la actividad humana. Gracias en gran medida al desarrollo y utilización de las Tecnologías de la Informática y las Comunicaciones (TIC). Siendo los algoritmos junto a la información el software y el hardware sus componentes.

Materiales y métodos

En la presente investigación se realizó una demostración del Principio de Invariancia para el estudio de la Complejidad Temporal de los Algoritmos y su importancia en las ciencias de la computación, para dar soluciones a diferentes problemas con un algoritmo eficiente. Estudio realizado en la carrera de Tecnologías de la Información de la facultad Ciencias Técnicas de la Universidad Estatal del Sur de Manabí. Para el desarrollo de esta investigación se utilizaron métodos científicos teóricos y empíricos, como:

Análisis – síntesis: se aplicó en el análisis de Invariancia para el estudio de la Complejidad Temporal de los Algoritmos.

Histórico – lógico: se empleó en la búsqueda de investigaciones relacionadas a la Complejidad Temporal de los Algoritmos.

Revisión bibliográfica: Se utilizó en la fase de recopilación de la información, mediante libros, revistas de carácter científico, entre otros documentos de carácter académico.

Resultados y discusión

EL agente de cómputo (procesador de la información) es un medio, cuya tarea es: dada una data inicial (información), ejecutar las reglas (algoritmo) para obtener nueva información (resultado), este puede ser humano, mecánico, electrónico, etc.

El mismo puede ser representado por un diagrama INPUT-OUTPUT (ENTRADA-SALIDA), como lo muestra la figura 1.

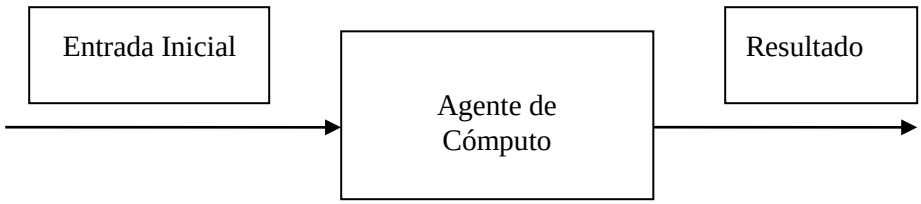

Figura 1. Diagrama conceptual de un agente de cómputo.

Definición intuitiva de algoritmo.

La definición de algoritmo es más rigurosa, pero de manera informal, popular (intuitiva) puede ser enunciado como sigue:

Un **algoritmo** es un procedimiento finito y sistemático, para procesar información simbólica discreta por un agente de cómputo, paso a paso y sin ambigüedad.

La noción intuitiva de algoritmo es inconsistente, por lo que no debe aceptarse como una definición formal y rigurosa, siendo objeto de estudio continuo de la comunidad de investigadores.

A lo largo de la historia muchos problemas han sido resueltos mediante técnicas de cómputo. Sin embargo el estudio sistemático de una definición formal y rigurosa de algoritmo comenzó a partir de los años treinta del pasado siglo.

Dos hecho contribuyeron substancialmente al crecimiento de estos estudios:

- Los resultados en los estudios de la lógica matemática de finales del siglo XIX y principios del XX.
- El desarrollo alcanzado por las computadoras electrónicas. Cuyo uso ha invadido toda la actividad humana lo cual ha requerido de estudios profundos para lograr un uso eficaz de las mismas.

Estos hechos han dado lugar al desarrollo de dos direcciones en el estudio de los algoritmos: **el diseño** (determinar los procedimientos adecuados para resolver el problema con el agente de cómputo de que disponemos) y **la teoría de la complejidad algorítmica** (estudio de la eficiencia, costo en tiempo y recursos con que el agente de cómputo ejecuta un algoritmo que resuelve el problema).

En principio, dado un problema puede determinarse un algoritmo que lo resuelva, si este puede ser formulado de manera lógica, tal que el agente de cómputo sea capaz de comprenderlo y controlar su ejecución paso a paso. Un algoritmo responde a la ecuación ALGORITMO = LÓGICA + CONTROL. Esta es una idea importante, según la cual la tarea fundamental del programador es formular lógica y apropiadamente el problema para que la solución sea buscada mediante un algoritmo.

¿Cómo debe el programador formular el problema? Esta es realmente una tarea de gran complejidad, siendo necesario realizar investigaciones teóricas y experimentales que permitan desarrollar metodologías y formalismos para enfrentar con éxito problemas de gran importancia práctica.

En la teoría clásica de la Informática se consideran dominios de información discretos. El caso de conjuntos de información continua ha sido extendido a partir de los trabajos de Blum, Shub y Smale (1989). Esta nueva extensión teórica representa un desarrollo importante que extiende los conceptos de recursividad y complejidad a nuevos campos matemáticos como el análisis y la topología (Rodríguez et al., 2021; Rodríguez et al., 2022; Rodríguez et al., 2020).

Los alfabetos, las palabras, los lenguajes:

La información se transmite por las redes mediante **palabras:** sucesión finita de elementos de un conjunto numerable de símbolos, que llamaremos **alfabeto**. Al número de símbolos de la información lo llamaremos **longitud de la información**.

Dado un alfabeto $A = \{a_1, a_2, ..., a_n, ...\}$, llamemos al conjunto de las palabras definidas sobre *A* **lenguaje** $W(A)$.

Sea M un conjunto discreto no vacío de símbolos. Una representación de *M* sobre el alfabeto *A*, es una aplicación $f: M \rightarrow W(A)$, que asocia a cada símbolo $s \in M$ alguna palabra $w \in W(A)$, entonces *f* asocia al conjunto *M* el lenguaje L_M tal que $L_M = Im(f) \subseteq W(A)$,

$$L_M = \{w: w \in W(A) \text{ y } w = f(s) \text{ para algún } s \in M\}$$

Lo anterior permite la siguiente definición:

Definición 1: Un **algoritmo** A no es más que una terna $(M; f; L_M)$.

En otras palabras, el algoritmo proporciona al agente de cómputo un método sistemático que transforma la información dada por medio de una aplicación, en una nueva información.

Complejidad de un algoritmo.

Dos direcciones se adoptan para evaluar la complejidad de un algoritmo:

En la primera, la complejidad **estática** del algoritmo, relacionada con la complejidad del texto del algoritmo (longitud, complejidad estructural, la variedad de tipos instrucción usadas). Esta medida de complejidad caracteriza al algoritmo independientemente de los datos de entrada a los que se aplica el algoritmo. La segunda, la complejidad **dinámica**, relacionada con el uso de los recursos requeridos por el agente de cómputo para llevar a fin la ejecución del algoritmo para una entrada determinada de datos (tiempo para su ejecución, memoria necesaria para la ejecución y almacenamiento).

Las direcciones, estática o dinámica, pueden ser usadas para escoger el algoritmo menos complejo entre dos o más implementaciones que resuelven el mismo problema. Lo anterior impone evaluar la complejidad intrínseca del problema a solucionar.

La complejidad (estática y dinámica) de un algoritmo depende de varios elementos: el formalismo con que es expresado el algoritmo, el agente de cómputo que lo ejecuta, y el criterio para evaluar su eficacia.

Evaluar la complejidad intrínseca de un problema es en general una tarea difícil, porque puede existir un gran número de algoritmos equivalentes, diferentes que resuelven el problema dado.

Un rasgo muy importante para la teoría de la complejidad es lograr estudios generales, independientes del agente de cómputo que ejecutará el algoritmo, lo anterior no minimiza la importancia de obtener medidas de complejidad específicas al agente de cómputo que ejecuta el algoritmo.

Axiomas de Blum

El estudio de la complejidad abstracta proporciona resultados que se sostienen para cualquier agente de cómputo, y cualquier criterio para evaluar el costo de la ejecución del algoritmo.

Considere la lista de los agentes de cómputo $(M_1,M_2,...)$, y correspondientemente, una lista $(T_1(n),T_2(n),...)$ de las funciones para evaluar el costo de ejecución de cada agente M_k, $T_k(n)$ es una función definida en el conjunto de los números naturales $T_k: \aleph \rightarrow \aleph$, la cual indica el costo de ejecución del agente M_k para la entrada n. $T_k(n)$ define el número de unidades de recurso consumidos en la ejecución del algoritmo por M_k para n.

Las función $T_k(n)$ proporciona una medida de complejidad si se satisfacen los **Axiomas de Blum**:

1. $T_k(n)$ esta definida si y sólo si $M_k(n)$ esta definida para todo k y n que pertenecen a los naturales.
2. Existe un algoritmo, que para valores arbitrarios k, n y m, decide si $T_k(n)=m$.

El primer axioma expresa: el costo de ejecución el cual es calculable para cualquier algoritmo que concluye la ejecución, pero debe ser considerado como indefinido el costo de ejecución de los algoritmos cuya ejecución no concluye. El segundo axioma exige la existencia de un algoritmo para decidir la cantidad de unidades de recurso ***m*** consumidos por M_k, para la entrada ***n***. Esto significa que a medida que la ejecución avanza se consumen más unidades del recurso, por consiguiente, para verificar si $T_k(n)=m$, podemos simplemente observa M_k para la entrada ***n*** durante un tiempo finito, esto define el período en que son consumidas las ***m*** unidades del recurso.

Como hemos expresado la complejidad de un algoritmo define el uso eficiente de los recursos empleados por el agente de cómputo para su ejecución. Generalmente la eficiencia se mide en función de: $T_k(n)$ tiempo que empleó el agente de cómputo M_k para ejecutar el algoritmo para la entrada ***n*** y $C_k(n)$ número de células (espacios de memoria) necesarias que usó el agente de cómputo M_k para la entrada ***n*** al ejecutar el algoritmo y almacenar los datos de entrada y salida, es decir, la complejidad temporal y capacitiva.

Complejidad temporal.

El tiempo de ejecución de un algoritmo va a depender de diversos factores: los datos de entrada, las características del agente de cómputo, la complejidad intrínseca del algoritmo. Con el florecer de las computadoras, la comunidad científica convino que la medida más significativa de la eficiencia temporal de un algoritmo es la expresión matemática que define su tiempo de ejecución por el agente de cómputo, la cual se expresa como función de los datos de entrada $T: \aleph \rightarrow \Re^+$, siendo $T(n)$ una función creciente de ***n***.

Es posible realizar el estudio del tiempo de ejecución de las maneras siguientes:

1. Obtener una función que acote el tiempo de ejecución para unos valores de entrada, esto proporciona una medida teórica a ***priori***, expresada por el número de operaciones elementales que realiza cada regla del algoritmo, para valores de entrada, independiente del agente de cómputo.
2. Medir el tiempo de ejecución en un agente de cómputo específico para unos valores de entrada dados, esto proporciona una medida real a ***posteriori***.

Teorema (Principio de Invarianza):

Sea ***A*** un algoritmo y I_1 e I_2 dos implementaciones de ***A*** que concluyen la ejecución. Sean $T_1(n)$ y $T_2(n)$ el tiempo de ejecución de I_1 y I_2, entonces $\exists c \in \Re,\ c > 0$ y $n_o \in \aleph$, tal que

$\forall n \geq n_0$, se verifica que $T_1(n) \leq c\ T_2(n)$.

Demostración:

Sean dos sucesiones finitas que representan el tiempo de ejecución de cada una de las instrucciones correspondientes a las implementaciones I_1 e I_2, para la entrada $n \geq n_o$, n_o es un número fijo:

$$\left\{T_1^1(n), T_1^2(n), \ldots, T_1^p(n)\right\}; \qquad \left\{T_2^1(n), T_2^2(n), \ldots, T_2^q(n)\right\}$$

Consideremos,

$$T_1^i(n) = \alpha_{ij} T_2^j(n), \quad \alpha_{ij} \in \Re^+, \qquad (1)$$

$i = 1,2,\ldots,p, \quad j = 1,2,\ldots,q, \quad \forall n \geq n_0,$

Se denota por $T_1(n)$ y $T_2(n)$ el tiempo total de ejecución de las dos implementaciones, de forma que,

$$T_1(n) = \sum_{i=1}^{p} T_1^i(n), \qquad (2)$$

$$T_2(n) = \sum_{j=1}^{q} T_2^j(n). \qquad (3)$$

Entonces, sumando en ambos miembros de la igualdad (1) respecto a j, se tiene,

$$\sum_{j=1}^{q} T_1^i(n) = \sum_{j=1}^{q} \alpha_{ij} T_2^j(n)$$

$$T_1^i(n) \leq \frac{1}{q} \max_{1 \leq j \leq q} \alpha_{ij} \sum_{j=1}^{q} T_2^j(n) = \frac{1}{q} \max_{1 \leq j \leq q} \alpha_{ij} \; T_2(n)$$

Sumando en ambos miembros de la desigualdad respecto a i se

$$\sum_{i=1}^{p} T_1^i(n) \leq \left[\frac{1}{q} \sum_{i=1}^{p} \max_{1 \leq j \leq q} \alpha_{ij}\right] T_2(n) = cT_2(n),$$

obtiene,

$$T_1(n) \leq cT_2(n), \quad \text{donde} \quad c = \frac{1}{q} \sum_{i=1}^{p} \max_{1 \leq j \leq q} \alpha_{ij}.$$

El Principio de Invariancia dice que el tiempo de ejecución de dos implementaciones de un algoritmo difieren en una constante multiplicativa.

En otras palabras un algoritmo tarda en su ejecución un tiempo del orden ***T(n),*** si existe una constante real ***c > 0*** tal que otra implementación tarda menos que ***cT(n)***, para toda entrada de tamaño ***n***.

Problema: Dadas dos implementaciones I_1 y I_2 que resuelven el problema:

Se desea conocer cuántos triángulos equiláteros, isósceles y escalenos, contiene un conjunto de n triángulos.

a) Calcular el tiempo de ejecución a priori de cada implementación.
b) Haciendo uso del Principio de Invariancia, dar una valoración de la eficiencia de las implementaciones.

// IMPLEMENTACIÓN I_1

```
// Determinar total de triángulos equiláteros, isósceles y escalenos
    Entrada n  // total de triángulos a clasificar
    CEQ = 0,  CES = 0, CIS = 0     // valores iniciales de  los contadores
    I = 0       // valor inicial para el contador del ciclo
        Mientras I ≠ n
          Entrada a, b, c   // lados del triángulo
          Si a = b y b = c
                Entonces
                  Clas = triángulo equilátero
                    CEQ = CEQ +1
               Sino Si a = b o b = c o a = c
                                Entonces
                             Clas = triángulo isósceles
                                    CIS= CIS+1
                          Sino
                          Clas = triángulo escaleno
                                CES = CES +1
               Fin Si
          Fin Si
```

```
        I = I + 1
      Fin Mientras
      Salida  Clas      // clasificación del triángulo.
   Salida CEQ, CIS, CES // total de triángulos equiláteros, isósceles y   escalenos
  //Fin del algoritmo
```

// IMPLEMENTACIÓN I_2

```
// Determinar  total de triángulos equiláteros, isósceles y escalenos
    Entrada n  // total de triángulos a clasificar
    CEQ = 0,  CES = 0, CIS = 0     // valores iniciales de  los contadores
        Para I =1 hasta n
          Entrada a, b, c   // lados del triángulo
          Si a = b y b = c
                Entonces
                  Clas = triángulo equilátero
                    CEQ = CEQ +1
               Sino Si a = b o b = c o a = c
                                Entonces
                             Clas = triángulo isósceles
                                    CIS= CIS+1
                           Sino
                           Clas = triángulo escaleno
                                 CES = CES +1
                Fin Si
           Fin Si
      Salida  Clas      // clasificación del triángulo.
    Próximo I
    Salida CEQ, CIS, CES // total de triángulos equiláteros, isósceles y   escalenos
 //Fin del algoritmo
```

Solución:

Para I_1 obtenemos $T_1(n) = 20(n-1) + 9$, con I_2 se obtiene $T_2(n) = 20n + 5$,

$$T_1(n) = 20n - 11 \leq 20n + 5 = T_2(n)$$

$$T_1(n) = 20n - 11 \leq 5(4n + 1) = T_2(n), \text{ para todo } n,$$

Conclusiónes: $\forall n, c = 5$, I_1 *es más eficiente que* I_2.

$\forall n, c = 1, \exists I_3$ *con* $T_3(n) = (4n + 1)$ *más eficiente que* I_1.

En esta demostración se conoció que en la actualidad los algoritmos están presentes en todas las áreas de la actividad humana, los cuales haciendo uso de las Tecnologías permite dar solución a diferentes problemas, llegando a un resultado eficaz en poco tiempo.

El diseño de algoritmos permitió determinar los procedimientos adecuados para resolver el problema, las tecnologías cumplen un rol importante en este proceso, así como la teoría de la complejidad algorítmica, facilita el estudio de la eficiencia, el costo en tiempo y recursos con que el agente de cómputo ejecuta un algoritmo para dar solución al problema.

En un problema puede determinarse un algoritmo que lo resuelva, si este puede ser formulado de manera lógica, tal que el agente de cómputo sea capaz de comprenderlo y controlar su ejecución paso a paso.

El tiempo de ejecución de un algoritmo va a depender de diversos factores tales como: los datos de entrada, las características del agente de cómputo, la complejidad intrínseca del algoritmo y las tecnologías aplicadas.

Conclusiones

El estudio teórico (abstracto) de la complejidad temporal de los algoritmos basado en el Principio de Invariancia proporciona resultados que se sostienen independiente del agente de computó, permitiendo comparar la eficiencia de las implementaciones de los algoritmos. Dando una valoración teórica a priorí, invariante ante los datos de entrada. Abriendo interrogantes para la búsqueda de algoritmos más refinados.

La demostración presentada deja abierta la posibilidad de generalizar el Principio de Invariancia de la complejidad temporal de los algoritmos a conjuntos de información continua.

Conflictos de intereses

Los autores no poseen conflictos de intereses

Contribución de los autores

1. Conceptualización: irenia Maldonado Zuñiga, Omar Antonio Quimis Sánchez, Franklin Jhimmy Toala Arias, Elio Armando Cables Fernández.
2. Curación de datos: Franklin Jhimmy Toala Arias, Elio Armando Cables Fernández.
3. Análisis formal: Kirenia Maldonado Zuñiga, Omar Antonio Quimis Sánchez.
4. Investigación: Kirenia Maldonado Zuñiga, Omar Antonio Quimis Sánchez.
5. Metodología: Kirenia Maldonado Zuñiga, Omar Antonio Quimis Sánchez.
6. Software: Franklin Jhimmy Toala Arias, Elio Armando Cables Fernández.
7. Validación: Franklin Jhimmy Toala Arias, Elio Armando Cables Fernández.
8. Visualización: Kirenia Maldonado Zuñiga, Omar Antonio Quimis Sánchez.
9. Redacción – borrador original: Kirenia Maldonado Zuñiga, Omar Antonio Quimis Sánchez, Franklin Jhimmy Toala Arias, Elio Armando Cables Fernández.
10. Redacción – revisión y edición: Kirenia Maldonado Zuñiga, Omar Antonio Quimis Sánchez, Franklin Jhimmy Toala Arias, Elio Armando Cables Fernández.

Financiamiento

La investigación fue financiada por los autores.

Referencias

Amosov A. A. & Dubinskii Y. A.& Konpchenova N. V. (1991). Métodos de cálculo para la solución de problemas de ingeniería. Ed. MEI.

Cormen. H.Thomas & Leiserson. E. Charles & Rivest. L Ronald & Stein Clifford. (2010). Introduction to Algorithms. Cambridge, MA: MIT Press and McGraw-Hill.

Complejidad de los algoritmos(1999). Universidad de Malaga. España.

Fortnow & Lance & Horner & Steven. (2002). A short History To Computational Complexity. Bulletin of the EATCS 80: 95-133.

Laureano-Cruces Ana Lilia (2013). Curso de Programación Estructurada Parte

Departamento de Sistemas UAM, Unidad Azcapotzalco. Recuperado: hptt://ce.azc.uam.mx/profesores/clc/03_docencia/.../i.../ProgEst_1.ppt

Fabricio L. & Otros. (2000). Matematics of Computing. CISM, UNESCO Project.

Frolov G. Y & Kuztnetsov E. (1991). Elementos de Informática. Ed. Nauka, Moscú.

Johnsonbaugh R. (2004). Matemáticas Discretas. Vol. 1, Cap. 3 y 5, Ed. Felix Varela.

Garbatov. V.A(1988): Fundamentos de la Matemática Discreta. Ed. Mir Moscú.

Rodríguez, A., Castro, V. F. R., Gonzalez, A. D. C. R., Baque, N. A. C., & Tarragó, J. C. P. (2021). Aplicaciones de la Inteligencia Artificial en técnicas de minería de procesos. Serie Científica de la Universidad de las Ciencias Informáticas, 14(7), 136-155. https://dialnet.unirioja.es/servlet/articulo?codigo=8590663

Rodríguez, A., Lucas, H. B. D., Mero, C. J. Á., Pisco, R. J. L., & Castro, F. I. G. (2022). Método computacional de recomendación sobre la evaluación del aprendizaje bajo el paradigma constructivista. Serie Científica de la Universidad de las Ciencias Informáticas, 15(1), 178-187. https://dialnet.unirioja.es/servlet/articulo?codigo=8590599

Rodríguez, A., Tarragó, J. C. P., Gálvez, D. L. D., & Pisco, R. L. (2020). Modelo de formación constructivista en el proceso Enseñanza-Aprendizaje virtual. Serie Científica de la Universidad de las Ciencias Informáticas, 13(11), 175-184. https://dialnet.unirioja.es/servlet/articulo?codigo=8590367

Otero D.A.M & Castellano P.L.O (2008) . Introducción al diseño y análisis de los algoritmos. Recuperado de: http://www.monografias.com/trabajos67/teoria-de-algoritmos/teoria-de-algoritmos.shtml

Velazquez, R. V., Maldonado Zúñiga , K., Castro Piguave, C., & Batista Garcet , Y. (2021). Metodología del aprendizaje basado en problemas como una herramienta para el logro del proceso de enseñanza- aprendizaje: Metodología del aprendizaje basado en problemas. Revista Científica Sinapsis, 1(19). https://doi.org/10.37117/s.v19i1.465

Sipser & Michael. (2006). Introduction to the theory of computation. Thomson Course Tecnology.