25-11-2025

REPORTE
Equipo E-A-B-M-O-D-E-L
PROYECTO FINAL

Análisis de Algoritmos

Equipo: **Códigos:**

Eduardo Ramos Ochoa	304489918
Andres Santiago Aguirre Macias	220293594
Cesar Emmanuel Gómez Martínez	214843698

Profesor:

JORGE ERNESTO LOPEZ ARCE DELGADO

Modulo:

D01

Centro Universitario de Ciencias Exactas e Ingenierías

INTRODUCCIÓN.

El presente proyecto final, se traduce como, la suma de las diferentes técnicas o enfoques algorítmicos estudiados durante el semestre, partiendo desde; fuerza bruta, divide y vencerás, programación dinámica y, algoritmos voraces, todos los enfoques anteriormente señalados, siendo aplicados al algoritmo base con el que se ha venido trabajando desde inicios del curso, siendo nuestro caso, el cracking de contraseñas.

Es importante señalar que, no todos los algoritmos tratados durante el curso, han sido aplicables, atendiendo a cuestiones de funcionalidad y/o lógica, por poner un ejemplo, se puede hablar del algoritmo de "Hoffman", mismo que no resultó pertinente ya que, su enfoque es la compresión de caracteres, lo que lo vuelve incompatible con nuestro proyecto, no así, al tratar de otros de enfoque voraz como lo es, el algoritmo de PRIM, así entonces, presentamos nuestro proyecto final.

OBJETIVOS.

I. **GENERAL. -** Desarrollo de un algoritmo que compile y muestre la evolución de nuestro algoritmo base, que en el caso que ahora nos ocupa es, "el cracking de contraseñas" (con un ambiente controlado), a través de la aplicación de las distintas técnicas algorítmicas estudiadas durante el semestre, a saber; fuerza bruta, divide y vencerás, programación dinámica y, algoritmos voraces.

II. **ESPECIFICO. -** Breve explicación de sus funcionamientos y diferencias, así como la comparación de sus correspondientes desempeños en la práctica.

DESARROLLO.

De forma concreta, el proceso de desarrollo de nuestro proyecto final, ha consistido en:

I. Como lenguaje de programación, se ha empleado **Python**, en su **versión 3.10+** conjuntamente con **Tkinter para creación de las distintas GUI** que se despliegan al momento de ejecutar/compilar nuestro programa.

II. La aplicación o suma de los distintos enfoques o técnicas algorítmicas estudiadas a lo largo del semestre, en concreto, al algoritmo base originalmente elegido, esto es, **"el cracking de contraseñas"**, partiendo desde, **fuerza bruta y, continuando con divide y vencerás, programación dinámica, concluyendo con enfoque voraz**, todo, en el mismo orden mencionado.

III. Para la debida ejecución y/o funcionamiento de las distintas técnicas algorítmicas, se implementaron diversas librerías adicionales de Python, como lo son; threading que nos permite trabajas con hilos, Time para trabajo con pausas y tiempo, math tal vez una de las más importantes gracias al trabajo con funciones matemáticas, heapq para trabajar con colas, itertools para el trabajo con iteraciones, hashlib para funciones de hashing

(particularmente importante en nuestro algoritmo base) random para la generación de valores aleatorios.

IV. Nuestro programa final, al ejecutarse, muestra un menú que permite elegir; el enfoque o técnica algorítmica a emplear, el wordlist a cargar (toda vez que se trata de un ambiente controlado), iniciar y/o detener funcionamiento, concluyendo con la muestra de tiempos de ejecución y resultados gráficos del algoritmo empleado.

- **FUERZA BRUTA –Problema / Algoritmo Base**

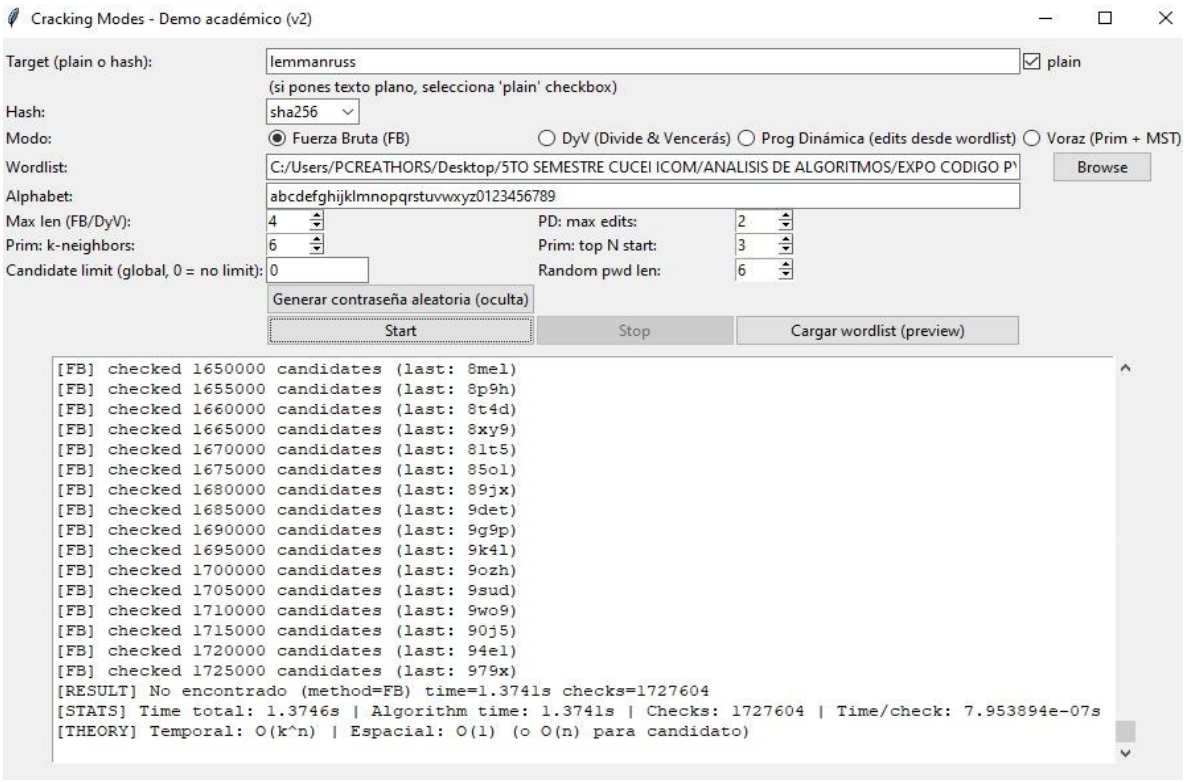


Figura 1- Enfoque de Fuerza Bruta

El primer enfoque aplicado es, Fuerza Bruta, el cual, a grandes rasgos consiste en probar todas las posibles combinaciones (de contraseñas) con el objeto de encontrar la solución al problema, en este caso, la búsqueda de la contraseña esperada.

Como puede notarse en la figura 1, al ejecutar el programa, tuvo un **tiempo de ejecución de 1.3741 segundos**, teniendo un crecimiento exponencial, esto, en razón de que, conforme la base de opciones a iterar (en nuestro caso, las contraseñas del wordlist) aumente, igualmente lo hará el tiempo de ejecución o complejidad temporal.

La principal ventaja que este enfoque ofrece es, garantizar encontrar la contraseña si esta, se encuentra en el espacio de búsqueda, aunque, por el contra, su desventaja es su escalamiento exponencial.

❖ **Divide y Venceras**

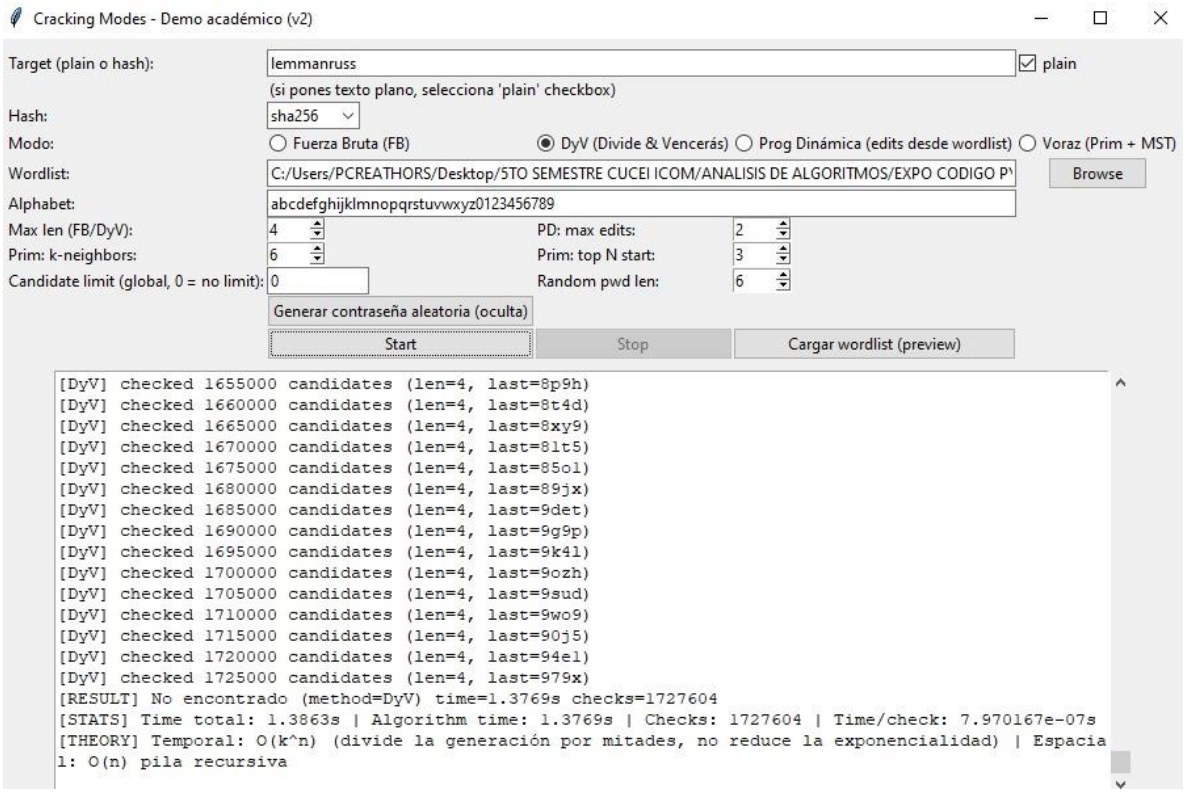


Figura 2- Enfoque de Divide y Vencerás

El segundo enfoque o técnica aplicada ha sido, Divide y Vencerás, el cual, consiste en dividir el problema general (en nuestro la wordlist) en subproblemas a fin de poder manejarlos de forma individual y más fácil, esto es, mejorar la manipulación del wordlist de contraseñas con el objetivo de encontrar la combinación indicada siempre que la misma se encuentre en el rango proporcionado.

Como puede notarse en la figura 2, al ejecutar el programa, tuvo un tiempo de **ejecución de 1.3769 segundos**, teniendo un crecimiento exponencial, aunque con levemente, muy levemente superior al de fuerza bruta, al igual que ocurre con el enfoque de fuerza bruta, su tiempo de ejecución o complejidad temporal, aumentará en medida que el rango de búsqueda lo haga.

La principal ventaja que ofrece este enfoque es, el de reestructurar el problema base, en subproblemas más fáciles de manejar a efecto de resolverlos de forma ordenada y recursiva, aunque, su desventaja, para el problema que nos ocupa es, precisamente, no reducir la complejidad exponencial, dicho de otro modo, no presenta una mejora significativa a la fuerza bruta.

- **Programación Dinámica**

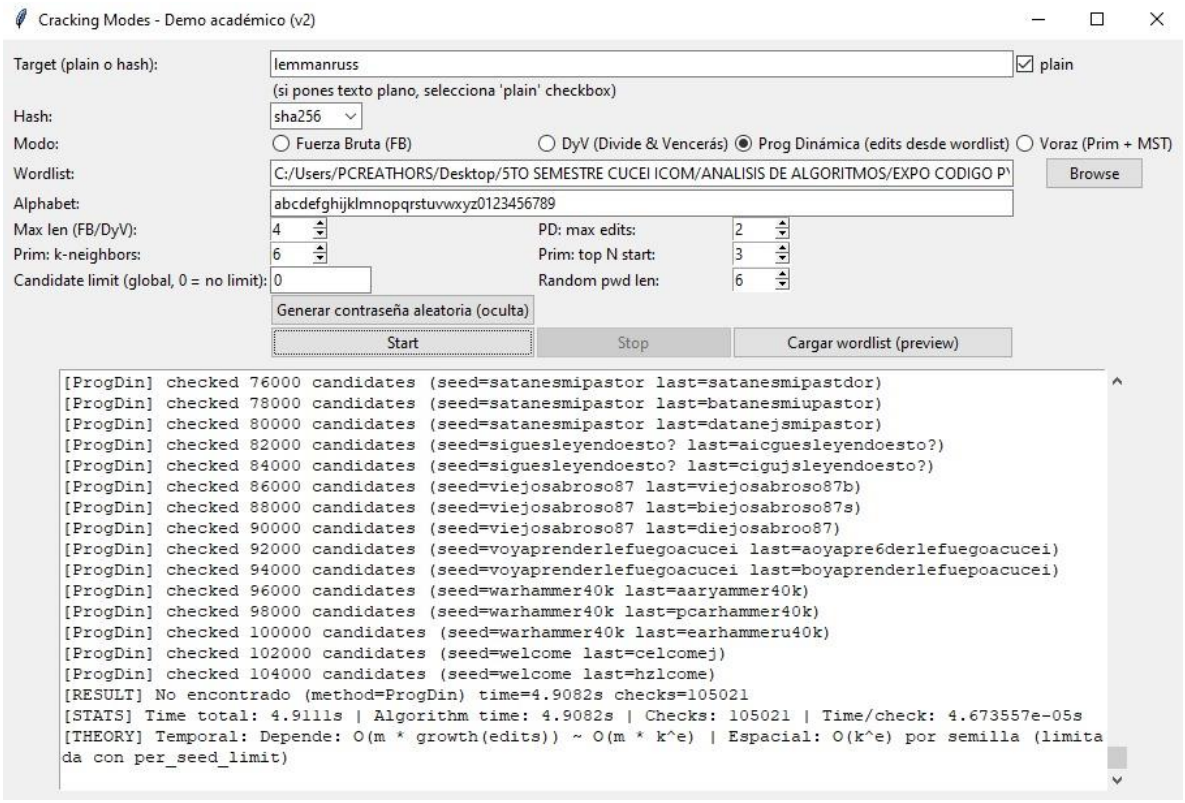


Figura 3- Enfoque de Programación Dinámica

El tercer enfoque aplicado ha sido el de Programación Dinámica, el cual se sustenta en la idea de partir de un conjunto base de posibles contraseñas y, a partir de estas, ir haciendo variaciones, a través de operaciones de edición tales como inserciones, eliminaciones o sustituciones de caracteres, de aquí lo de “dinámico”.

Como se aprecia en la figura 3, la ejecución del programa bajo este enfoque presentó un **tiempo de 4.9082 segundos**, de todos los enfoques que forman parte del mismo. Lo anterior, en virtud a la complejidad temporal de este enfoque, la cual, depende tanto del tamaño de la wordlist como medida de crecimiento exponencial asociado a la cantidad de modificaciones permitidas por palabra.

La principal ventaja de este enfoque radica en su mayor realismo y eficiencia práctica en escenarios donde las contraseñas se asemejan a palabras o estructuras comunes dentro de una wordlist, pues reduce el espacio de búsqueda frente a Fuerza Bruta y DyV. No obstante, su eficacia está totalmente condicionada por la calidad y cobertura del diccionario: si la contraseña objetivo no se encuentra directamente en la wordlist o dentro de un rango cercano de variaciones, este enfoque no logrará encontrarla aun cuando incremente el tiempo de ejecución. En conclusión, se trata de un método muy útil siempre que se conozca o estime el comportamiento del usuario al momento de elegir su contraseña.

❖ **Algoritmo Voraz**

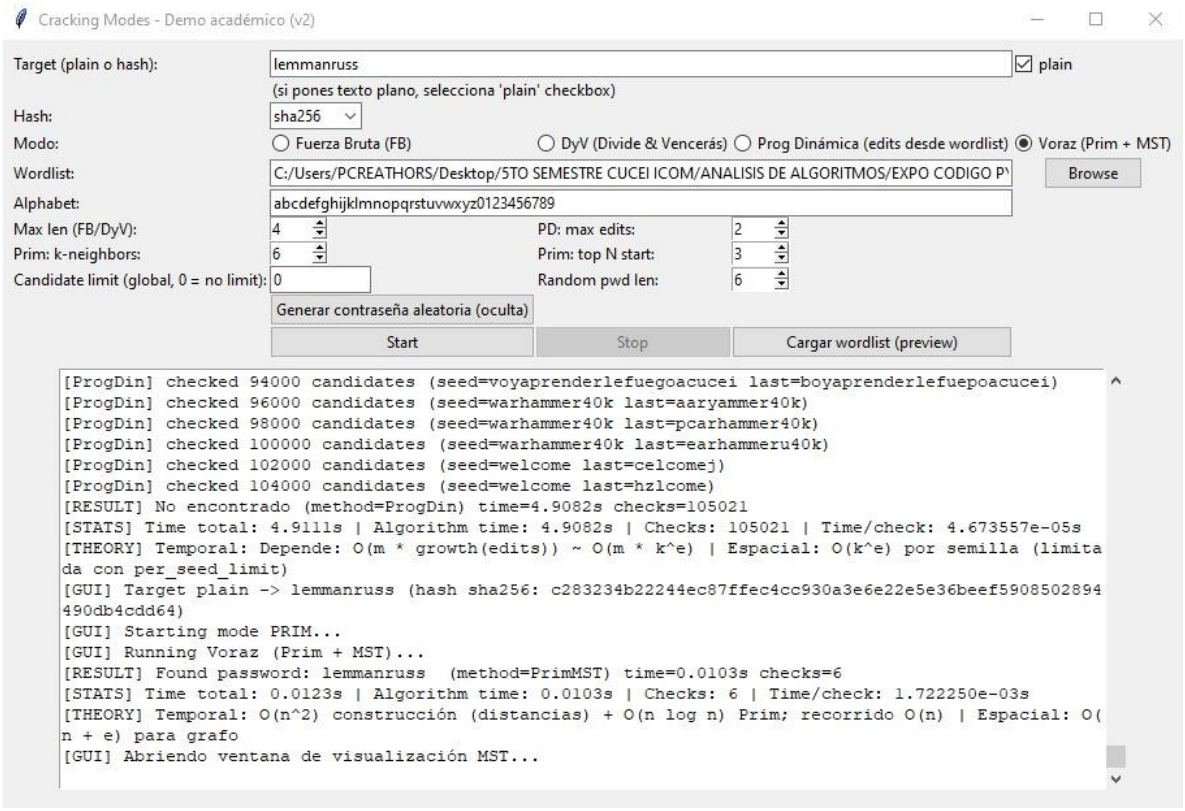


Figura 4- Enfoque Voraz, empleando algoritmo de PRIM

El cuarto y último de los enfoques empleados, se trata del enfoque de algoritmos voraces, concretamente, empleando el algoritmo de PRIM, si bien es verdad, existen otros algoritmos dentro de este grupo (voraces), como fue el caso de “Hoffman”, esté último no resultó pertinente para los fines de nuestro algoritmo, toda vez que, su objetivo es el de comprimir caracteres, lo que lo vuelve inviable cuando se trata de cracking de contraseñas.

Ahora bien, el algoritmo de PRIM, funciona trazando el MST o árbol de expansión menor dentro de un grafo construido tomando como base las posibles contraseñas del wordlist, recorriendo los candidatos que son representados por nodos, como puede apreciarse en la figura 4, este método obtuvo un tiempo aproximado de **0.010 segundos**, convirtiéndose en **el más rápido de los cuatro enfoques** implementados, debido a su complejidad **$O(n^2 + n \log n)$**, mucho más eficiente que la exponencial de los métodos anteriores.

Así pues, la principal ventaja que tiene este algoritmo es el hecho de reducir drásticamente el espacio de búsqueda al enfocarse en candidatos prometedores; sin embargo, al igual que en Programación Dinámica, su rendimiento depende de que la contraseña objetivo guarde relación con el diccionario base o wordlist.

No pasamos por alto presentar el MST obtenido de la ejecución de PRIM

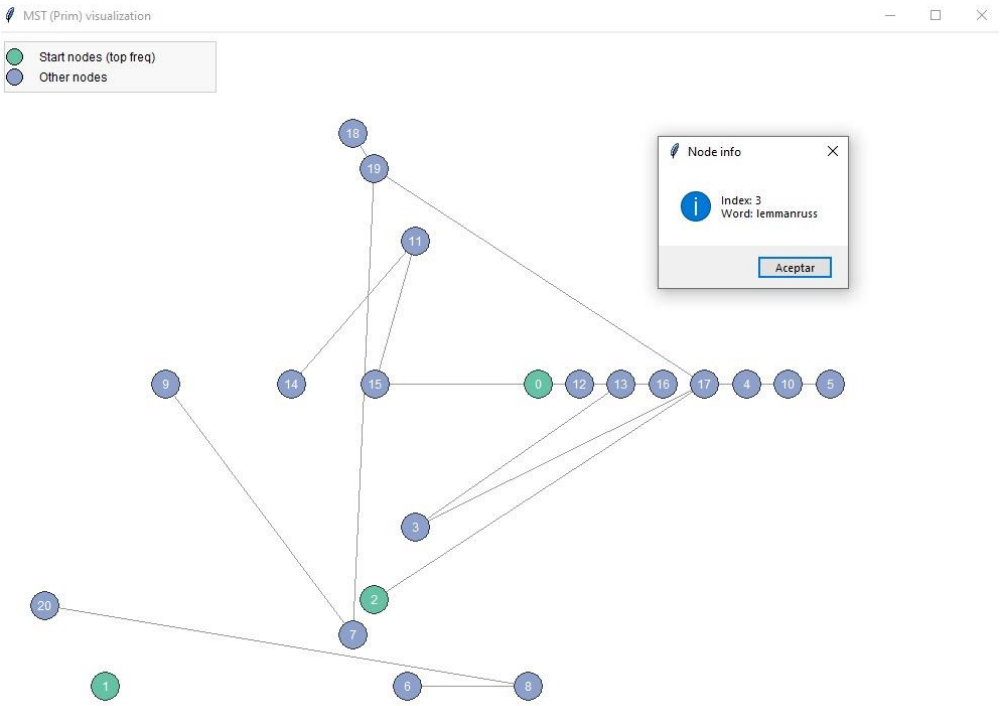
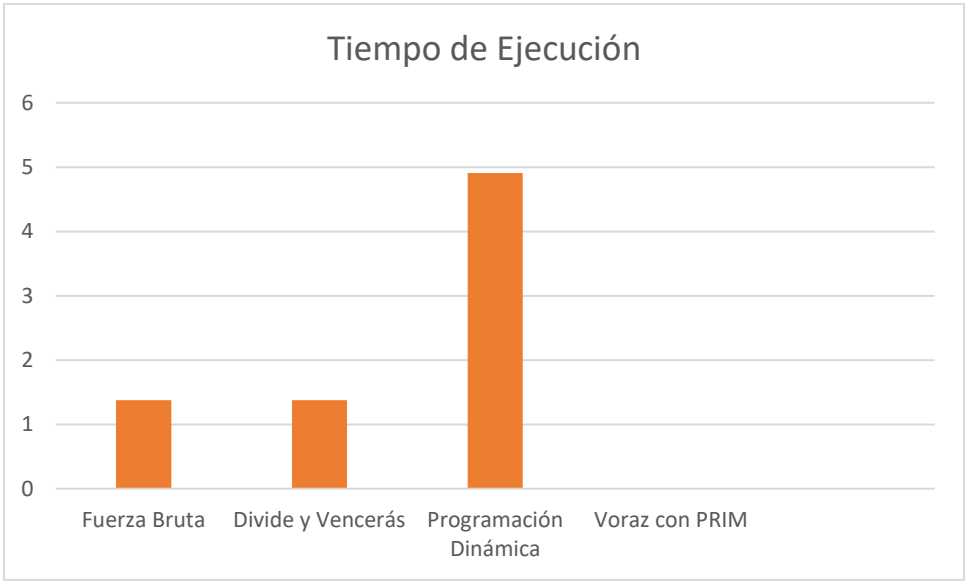


Figura 5.- Árbol de Expansión Menor o MST de PRIM

Así entonces, al realizar una comparación entre el comportamiento de los enfoques algorítmicos empleados para nuestro proyecto final, la misma queda así:



CONCLUSIONES

Como puede notarse, si bien es verdad que, a nuestro algoritmo base, se le puede aplicar diversos enfoques o técnicas algorítmicas, cierto resulta también que, cada una de estas tiene sus ventajas y desventajas, traducidas en complejidad temporal y/o espacial.

Por lo anterior, consideramos que, el punto principal reside en, elegir el algoritmo que consuma menores recursos o bien, realice la función deseada con mayor eficiencia.

Consideramos que nuestro proyecto final puede ser catalogado como exitoso.

REFERENCIAS

- **Algoritmos a profundidad, de cero a Senior. (s. f.). Hotmart.**

https://hotmart.com/es/marketplace/productos/algoritmos-a-profundidad-de-cero-a-senior-por-adevsays/X92565754V

- **Wirth, N. (1976). Algorithms + Data Structures = Programs. Prentice-Hall.**
- **Dromey, R. G. (1982). How to Solve it by Computer. Prentice-Hall.**