Algoritmo Divide y Vencerás (Elemento Mayoritario)

Danny Toala, Erick Sanchez y Vicente Saavedra

Universidad Central del Ecuador

Facultad de Ingeniería, Ciencias Físicas y Matemáticas, Universidad Central del Ecuador, Av. Universitaria, Quito 170129.

Ingeniería en Sistemas de Información

Algoritmos

Grupo 5

Email: [danniruxorellana@hotmail.com](mailto:danniruxorellana@hotmail.com)

[erick-velarde1@hotmail.com](mailto:erick-velarde1@hotmail.com)

vicentesaavedra45@gmail.com

10 de diciembre de 2019

Algoritmo Divide y Vencerás

Danny Toala, Erick Sanchez y Vicente Saavedra

Universidad Central del Ecuador

Docente o Tutor

Mauro Leonardo Rosas Lara

Facultad de Ingeniería Ciencias Físicas y Matemáticas

Algoritmo Divide y Vencerás (Elemento Mayoritario)

Quito

2019

Contenido

[Resumen 4](#_Toc26209014)

[Objetivo General 4](#_Toc26209015)

[Objetivos Específicos 4](#_Toc26209016)

[Marco Teórico 4](#_Toc26209017)

[Materiales 6](#_Toc26209018)

[Resultados 6](#_Toc26209019)

[Algoritmo Divide y Vencerás 7](#_Toc26209020)

[Algoritmo Iterativo 9](#_Toc26209021)

[Tablas y Gráficos 10](#_Toc26209022)

[Procedimiento: 13](#_Toc26209023)

[Observaciones: 13](#_Toc26209024)

[Conclusiones: 14](#_Toc26209025)

[Recomendaciones: 14](#_Toc26209026)

[Referencias 15](#_Toc26209027)

# Resumen

En la práctica se realizarón dos algoritmos para la resolución de un mismo problema. El problema solicitaba que se encuentra el número mayoritario si existe, dentro de una colección de números enteros considerando como mayoritario al número que se repite más de n/2 veces siendo n el número de elementos de la colección. El primer algoritmo aplica la técnica de Divide y Vencerás, mientras que el segundo resuelve el problema de forma iterativa, al final se realizará una comparación de los dos métodos.

## Objetivo General

* Observar por medio de un ejemplo cómo funciona el paradigma Divide y Vencerás

## Objetivos Específicos

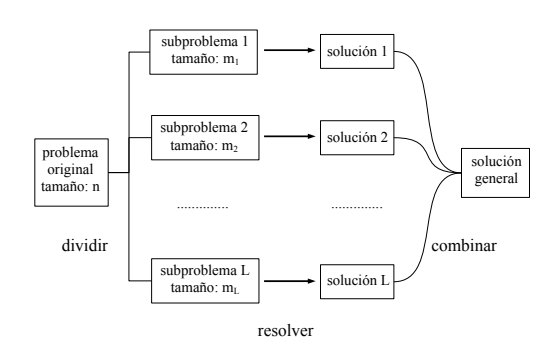
* Comprobar que tan eficiente es resolver un problema aplicando el paradigma Divide y Vencerás, respecto a resolverlo de forma iterativa.
* Comparar tiempos de ejecución.

# Marco Teórico

El *paradigma Divide y Vencerás separa un probelma en subproblemas, que se parecen al problema original, estos subproblemas se resuelven de manera recursiva, y por último se combinan las soluciones de cada subproblema resolviendo así el problema original* (khan Academy, 2015)

Hay que tener en cuenta que cada subproblema debe ser más pequeño que el orignal, ya que estos se resuelven de manera recursiva, y sobre todo debe exister un caso base para evitar bucles infinitos este paradigma se compone de 3 partes que son las siguientes:

* Divide: El problema principal, divide en subproblemas más pequeños
* Vence: Los subproblemas se van resolviendo de forma recursiva
* Combina: Las soluciones de los subprobelmas se combinan, para así obtener la solución al problema original

*El término Divide y Vencerás, tiene una aceptación muy amplia y se lo considera como algo más que una técnica de diseño de algoritmos, se lo considera como una filosofía general para resolver problemas que es muy utilizada en varios ámbitos* (Rodríguez L, 2012)

Rodríguez, L. (2019). [Figura 1]: Diagrama Divide y Vencerás. Recuperado de *Técnicas de diseño de algoritmos Divide y Vencerás.*

Se puede denotar que en la Figura 1 expone de manera abreviada el pseudofuncionamiento de un algoritmo de divide y venceras.

Esta técnica cuenta varios ventajas y desventajas las cuales son:

* Ventajas
* Simple, robusto y elegante
* Buena legibilidad
* Fácil depuraión y mantenimiento del código
* Desventajas
* Mayor coste espacial por el uso intesivo de la pila del sistema
* En ocasiones no se reduce, sino que incluso aumenta la complejidad temporal respecto a soluciones iterativas

Exiten varios algoritmos que aplican esta técnia, sin embargo uno de los que mejor lo ejemplifica es el algoritmo de búsqueda binaria

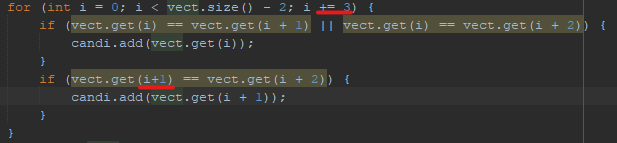
# Materiales

* Procesador: Core i7 de 2.70 GHz
* Memoria: 12 de RAM
* Sistema Operativo Windows 10 Home de 64 bit
* Disco Sólido de 250 GB
* NetBeans IDE 8.2
* Word 10 Home

# Resultados

Se probaron los dos algoritmos, con colecciones de diferentes tamaños desde 100 elementos a 1.000.000 de elementos, por lo que se optó en llenar las colecciones únicamente con dos valores, uno se repetiría mas que el otro para que se pueda imprimir una solución. Ya que si, se llenaba de manera aleatoria las colecciones era complicado que se encuentre una solución, porque es poco probable que de 1000 números que se crearon aleatoriamente se repitan 501 elementos, entonces casi nunca se encontraba una solución si, se hacía de esa manera.

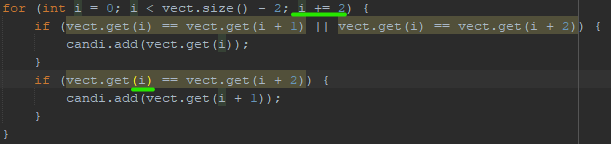
## Algoritmo Divide y Vencerás

El algoritmo que se usó para esta práctica se encuentra en internet, no se encontró al autor del código, pero se adjunta el enlace del blog dónde se encontró el código. A este código se le hizo dos modificaciones por que no concordaba con el diagrama de flujo que nos facilitó nuestro docente.

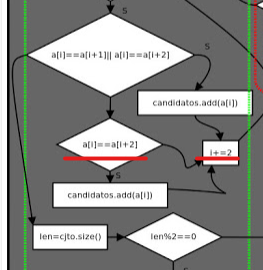
Saavedra, V. (2019). [Figura 2]: Código Divide y Vencerás original. Elaboración propia

En la Figura 2 se observa el código original descargado del blog antes mencionado, cuando se probó este código con 12 datos, dio resultados que no concordaban con lo que se solicitaba en el problema.

* El número 6 (Aparece 6 veces), el resto números para rellenar. El resultado esperado es que no haya un número mayoritario, ya que no se cumple la condición de que se debe repetir **más de n/2 veces,** y efectivamente cuando se probó el algoritmo arrojo este resultado.
* El número 6(Aparece 7 veces), el resto números para rellenar. El resultado esperado es que haya un número mayoritario, ya que se repite más de n/2 veces, pero el programa dio como resultado que **no hay** un número mayoritario
* El número 6(Aparece 8 veces), el resto números para rellenar. El resultado esperado es que haya un número mayoritario, ya que se repite más de n/2 veces, pero el programa dio como resultado que **no hay** un número mayoritario
* Sólo cuando el número 6(Aparece 9 veces), el programa dio como resultado que efectivamente hay un número mayoritario, y es el 6.

Como los resultados no eran los esperados, se realizaron los siguientes cambios:

Saavedra, V. (2019). [Figura 3]: Código Divide y Vencerás modificado. Elaboración propia

En la Figura 3 se observa los cambios que se realizaron, estos se los realizaron en base al diagrama de flujo, ya que en el mismo consta que se deben poner esos valores. (Véase Figura 4)

Anónimo. (2012). [Figura 4]: Diagrama de flujo Divide y Vencerás, número mayoritario. recuperado de: <http://misiniciosenjava.blogspot.com/p/algoritmos-divide-y-venceras.html>

Se probó el nuevo código con los anteriores casos, dando resultados satisfactorios cumpliendo en todos los casos con lo que nos pide el problema, que el número es mayoritario cuando se repite n/2 veces. Ahora se procedió a probar este nuevo código con colecciones mucho más grandes, arrojando los siguientes resultados

El algoritmo Divide y Vencerás modificado registró mejores tiempos únicamente cuando se usaron 100 y 500 datos, cuando se usaron 1000 y 5000 datos el tiempo de ejecución subió considerablemente, además cuando se intento probar el algoritmo con 10 000 datos, nos dio como el siguiente excepción *“Exception in thread "main" java.lang.StackOverflowError”* que nos indica que nos hemos quedado sin memoria, claramente no se espera estos resultados, de un algoritmo recursivo, así que se opto por volver al código sin modificaciones, y dio resultados mucho más satisfactorios mejorando los tiempos de ejecución del algoritmo iterativo y permitiendo probar el algoritmo con colecciones de 1 000 000 de elementos, sin que ocurra ningún error.

Por último, se encontró una particularidad cuando se realizaban las pruebas en el **algoritmo modificado**, teniendo en cuenta la siguiente entrada: 7-7-7-6-6-6

* Lo esperado es que, no se encuentre un número mayoritario pero el programa nos da como resultado, que si existe un número mayoritario y es 7. Lo curioso es que cuando se coloca los siguientes datos 7-7-7-5-4-3, el programa ahora sí nos dice que no se ha encontrado un número mayoritario, se intentó hacer otras modificaciones para solucionar esta falla pero no se pudo encontrar.

## Algoritmo Iterativo

El algoritmo iterativo, se lo realizó en grupo y se usó los casos bases del algoritmo Divide y Vencerás, en el resto del código se aplicó la lógica de cada uno de los integrantes del grupo. El algoritmo no presentó inconvenientes con colecciones pequeñas ni con colecciones grandes. Lo que se puede destacar es que los tiempos de ejecución fueron mayores a los del algoritmo Divide y Vencerás, y que el tiempo de ejecución variaba dependiendo de los datos ingresados, por ejemplo:

* Con un 1 000 000 de datos, se ingresaron 5(El que más se repetía) y 10(El que menos se repetía), con estos datos el tiempo de ejecución fue de 83,7 milisegundos
* De igual manera con la mima cantidad de datos, pero con los números 1 y 2 el tiempo de ejecución fue 35,32 milisegundos.

Se llegó a la conclusión de que esto se debe a la lógica programacional que aplicamos en el algoritmo.

# Tablas y Gráficos

|  |  |  |
| --- | --- | --- |
| Número de datos | Algoritmo Iterativo | Algoritmo Divide y Vencerás (Modificado) |
| 100 | 20,96 | 10,15 |
| 500 | 24,47 | 22,03 |
| 1000 | 22,46 | 54,21 |
| 5000 | 30,23 | 187,55 |
| 10 000 | 35,5 | Exception in thread "main" java.lang.StackOverflowError |
| 50 000 | 46,91 | Exception in thread "main" java.lang.StackOverflowError |
| 100 000 | 65,49 | Exception in thread "main" java.lang.StackOverflowError |
| 500 000 | 106,03 | Exception in thread "main" java.lang.StackOverflowError |
| 1 000 000 | 110.67 | Exception in thread "main" java.lang.StackOverflowError |

Saavedra, V. (2019). [Tabla 1]: C: Datos obtenidos del algoritmo DYV (modificado) y el algoritmo iterativo. Elaboración propia

|  |  |  |
| --- | --- | --- |
| Número de datos | Algoritmo Iterativo | Algoritmo Divide y Vencerás |
| 100 | 20,96 | 9,13 |
| 500 | 24,47 | 11,23 |
| 1000 | 22,46 | 15,22 |
| 5000 | 30,23 | 10,65 |
| 10 000 | 35,5 | 14,34 |
| 50 000 | 46,91 | 17,6 |
| 100 000 | 65,49 | 29,9 |
| 500 000 | 106,03 | 50,5 |
| 1 000 000 000 | 110.67 | 77,5 |

Saavedra, V. (2019). [Tabla 2]: Datos obtenidos del algoritmo DYV (sin modificar) y el algoritmo iterativo. Elaboración propia

Véase en las Tablas 1 y 2, el tiempo de ejecución en milisegundos, del algoritmo Divide y Vencerás tanto el código modificado como el sin modificar, y el algoritmo iterativo a continuación se presentan unos gráficos, obtenidos de las tablas antes mostradas

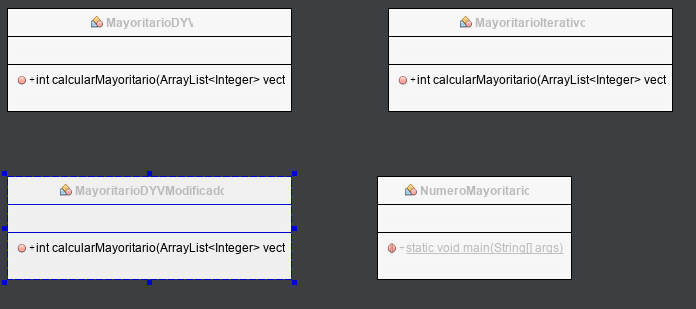
Saavedra, V. (2019). [Grafica 1]: Datos obtenidos del algoritmo DYV(sin modificar) y el algoritmo iterativo. Elaboración propia

La gráfica 1, es una comparación entre el algoritmo iterativo, y el algoritmo Divide y Vencerás sin modificar, se observa claramente como aplicando el paradigma Divide y Vencerás la ejecución es mucho más rápida a pesar de que con una cantidad de datos menor los resultados no son los esperados, trabaja muy rápido respecto al algoritmo iterativo, y es de esperarse ya que la recursividad es mucho más rápida que las técnicas iterativas

Saavedra, V. (2019). [Grafica 2]: Datos obtenidos del algoritmo DYV(Modificado) y el algoritmo iterativo. Elaboración propia

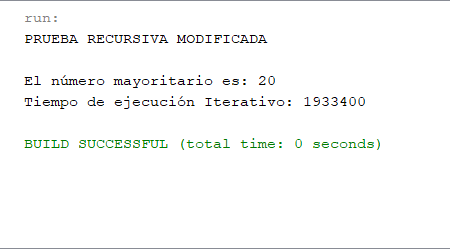
En la gráfica 2, de igual manera de muestra una comparación entre el algoritmo Divide y Vencerás modificado y no modificado, al cambiar esas dos líneas en el código descargado el tiempo ejecución aumenta considerablemente con cantidades elevadas de datos, y cuando se pasa de 5000 datos el algoritmo simplemente deja de funcionar, sin embargo, cuando se manejan pocos datos, los resultados son los esperados a diferencia del código que se descargó que muestra resultados diferentes.

# Procedimiento:

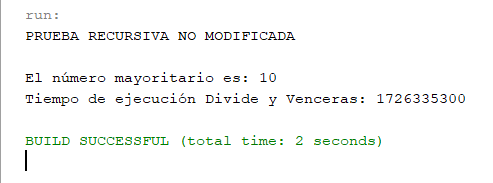
1. Diagrama de Clases. (Véase Figura 5)

Saavedra, V. (2019). [Figura 5]. Diagrama de Clases del Programa Divide y Vencerás. Elaboración propia

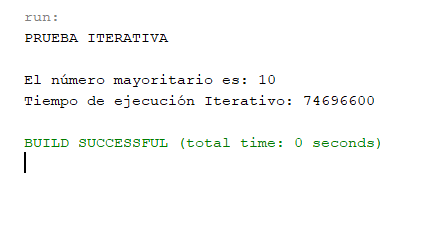
1. Se probó cada uno de los algoritmos con diferentes cantidades de datos, para comprobar que funcionen correctamente.
2. Se encontraron algunos errores en los algoritmos, por que se procedió a realizar algunos cambios que solucionen el problema
3. Se procedió a realizar las pruebas correspondientes con cada algoritmo, y con una cantidad datos diferentes en cada prueba, se registraron los resultados en una tabla de datos. (Véase Figura 6, 7, 8)



Toala, D. (2019). [Figura 6]. Prueba Recursiva Modificada con 100 elementos. Elaboración propia



Toala, D. (2019). [Figura 7]. Prueba Recursiva No Modificada con 10000000 elementos. Elaboración propia



Toala, D. (2019). [Figura 8]. Prueba Iterativa con 1000000 elementos. Elaboración propia

1. Se tabularon los resultados para obtener gráficas, que muestren de mejor manera los resultados, y nos permitan obtener conclusiones.

# Observaciones:

* Realizar pequeñas modificaciones en un código, puede cambiar completamente su funcionalidad
* Sin el uso de excepciones, hubiera sido complicado mostrar los resultados que presentaban los algoritmos

# Conclusiones:

* Los tiempos de ejecución aplicando el paradigma DYV, fueron mucho más rápidos que aplicando el método iterativo. Sin embargo, el que mejor resultados presentó fue el algoritmo iterativo tanto con colecciones pequeñas como grandes. En algunos casos el algoritmo DYV, no presentaba las soluciones esperadas a pesar de haber realizado algunos cambios.
* Al dividir un problema general en subproblemas más pequeños mediante la técnica Divide y vencerás, se puede llegar a una solución más rápida y efectiva.
* Al implementar el manejo de excepciones se logró dar una alternativa de salida a la solución en caso de no encontrar el elemento mayoritario.
* Se logró realizar un análisis de efectividad entre el método iterativo y la técnica divide y vencerás.

# Recomendaciones:

* Cuando se obtiene código de internet, es importante revisarlo, comprenderlo y comprobar que funcione correctamente, y mencionar al autor original
* Utilizar el método nanoTime(); para obtener un tiempo de ejecución más exacto
* Realizar las pruebas que hagan falta, hasta estar seguros que el código no contiene errores

# Referencias

* Luis Rodríguez. (2012). *Técnicas de diseño de algoritmos Divide y Vencerás.* Universidad Politécnica de Valencia (UPV). España
* Anónimo (2012). Mis inicios en java. Pinceladas de código en java y javascript. Algoritmo Divide y Vencerás recuperado de: <http://misiniciosenjava.blogspot.com/p/algoritmos-divide-y-venceras.html>
* Khan Academy (2015). Ordenamiento por mezcla-Algoritmos de divide y vencerás*.* Recuperado:<https://es.khanacademy.org/computing/computerscience/algorithms/merge-sort/a/divide-and-conquer-algorithms>

Vicente Saavedra

CI: 1756175228

Danny Toala

CI: 1723463053

Erick Sánchez

CI: 0604272393