

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**INGENIERÍA DE SOFTWARE**

**ESTRUCTURA DE DATOS**

**NRC:** 3253

**DOCENTE:** ING. DORYS QUIROZ

**GRUPO N°6:**

YOSELIN ANDRANGO

WUIDEZON BURGA

MICHAEL COBACANGO

FREDDY PAEZ

BRYAN YAGUARSHUNGO

PULLAGUARI AXEL

**SANGOLQUÍ, FEBRERO 2021**

[INTRODUCCIÓN: 3](#_Toc67141167)

[METODOLOGÍA DE SOLUCIÓN ALGORÍTMICA 3](#_Toc67141168)

[**BÚSQUEDA EXHAUSTIVA:** 3](#_Toc67141169)

[**ALGORITMO VORACES:** 4](#_Toc67141170)

[**DIVIDE Y VENCERÁS:** 5](#_Toc67141171)

[**PROGRAMACIÓN DINÁMICA:** 6](#_Toc67141172)

[**BACKTRACKING:** 7](#_Toc67141173)

[CUADRO COMPARATIVO  : 7](#_Toc67141174)

[ALGORITMOS: 8](#_Toc67141175)

[BIBLIOGRAFÍA 8](#_Toc67141176)

# INTRODUCCIÓN:

# METODOLOGÍA DE SOLUCIÓN ALGORÍTMICA

## **BÚSQUEDA EXHAUSTIVA:**

**Definición:** Es una técnica general de resolución de problemas, consiste en mirar todas las posibles soluciones, típicamente de una forma recursiva, parte de una solución parcial que se va completando poco a poco, y descarta las soluciones parciales que no llevan a ningún lado. La búsqueda se suele realizar recorriendo un árbol con el que se representan las posibles soluciones, hay que agregar que hay algunos métodos de recorrer un árbol como son: recorrido en anchura, backtracking, branch and bound(ramificación y acotación)(Nacho Cabanes,  2013).

**Implementación:**

Se puede aplicar para la resolución mayormente en los problemas de optimización ya que solo pueden resolverse de manera exacta, ¿Cómo conseguimos esto?, pues explorando todas las combinaciones posibles de los elementos y tratando de encontrar la solución, para posteriormente hacer una elección entre ellas una cualquiera o aquella que sea más optimice una determinada función objetivo, además la búsqueda exhaustiva se usa habitualmente cuando el número de soluciones candidatas no es elevado, o dicho de otra forma cuando este puede reducirse previamente usando algún otro método heuristico. También puede ser utilizado cuando es más importante una implementación sencilla que una mayor rapidez, además que es útil como método “base” cuando se desea comparar el desempeño de otros algoritmos metaheurísticos (es un método heurístico para resolver un tipo de problema computacional general).

**Ventajas:**

* Se le puede asignar varias tareas automatizadas en tiempo real.
* Su eficacia está en hacer programas de forma rápida y presentable.
* Evalúa las soluciones potenciales una a una descalificando las no factibles y manteniendo un registro de la mejor encontrada.

**Desventajas:**

* Se realiza una búsqueda exhaustiva y sistemática en el espacio de soluciones, en consecuencia, puede resultar ineficiente.
* Es muy complejo, puede ocurrir que, si no se ejecuta en un ordenador muy potente, este puede bloquearse.
* Puede resultar ambiguo o confuso en los diferentes lenguajes de programación.

**Conclusión:**

Esta técnica puede resultar muy favorable además de ser muy conocida ya que hace una búsqueda exhaustiva y sistemática en el espacio de soluciones, pero trae consigo varios inconvenientes, ya que puede ser muy difícil a la hora de codificar, y para el buen funcionamiento de este método debe cumplir algunos requisitos, uno de ellos es el hardware debe ser el adecuado ya que este método puede hacer colapsar el equipo llegando a bloquearse.

## **ALGORITMO VORACES:**

**Definición:** (USalesiana Virtual, n.d.) Nos menciona que el algoritmo voraz es una estrategia de búsqueda por la cual se sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima. El término voraz se deriva de la forma en que los datos de entrada se van tratando, realizando la elección de desechar o seleccionar un determinado elemento una sola vez.

El objetivo de estos es que cada elemento a considerar se evalúa una única vez, siendo descartado o seleccionado, de tal forma que si es seleccionado forma parte de la solución, y si es descartado, no forma parte de la solución ni volverá a ser considerado para la misma.

**Implementación:** (Course Hero, 2012). Nos menciona que la implementación de los algoritmos voraces, se caracteriza por la existencia de un bucle. Los algoritmos voraces se utilizan en problemas tales como:

* Árbol de recubrimiento mínimo de un grafo.
* El algoritmo de Kruskal.
* El algoritmo de Prim.
* Caminos mínimos en grafos:  Encontrar la ruta más corta posible entre caminos.
* Algoritmo de Dijkstra
* El problema de la mochila. Consiste en llenar una mochila de manera que el contenido sea lo máximo posible.
* El problema de planificación de tareas. Estos problemas tratan de planificar varias tareas de manera que se maximice la ganancia obtenida o el número de tareas realizadas.
* El problema del cambio de la devuelta. Consiste en retornar una devuelta, la cual está conformada con el menor número de tareas posibles.

**Ventajas:**

* El algoritmo voraz arroja soluciones que están muy cerca de las soluciones exactas
* Rapidez en hallar una solución
* De implementación sencilla

**Desventajas:**

* El enfoque que aplican es muy corto y toma decisiones basándose en la información que tienen disponibles de modo inmediato, sin tener en cuenta los efectos que estas decisiones puedan tener en el futuro.
* No resuelven problemas complejos y en algunas ocasiones no se encuentra solución.

**Conclusión:**

* A Pesar de que no siempre es capaz de encontrar una respuesta óptima, se lo utiliza con frecuencia dado que es muy rápido.
* Sigue una estrategia sencilla pero eficaz, Simplemente, se trata de elegir la opción óptima en cada paso local, con la esperanza de llegar a una solución general óptima

## **DIVIDE Y VENCERÁS:**

**Definición:** En las ciencias de la computación, el término divide y vencerás (DYV)

hace referencia a uno de los más importantes paradigmas de diseño algorítmico. El método está basado en la resolución recursiva de un problema dividiéndolo en dos o más subproblemas de igual tipo o similar. El proceso continúa hasta que éstos llegan a ser lo suficientemente sencillos como para que se resuelvan directamente. Al final, las soluciones a cada uno de los subproblemas se combinan para dar una solución al problema original.

**Implementación:**

Los algoritmos de “divide y vencerás” pueden ser implementados como procesos que son recursivos. Poniendo este ejemplo al existir un problema grande este puede ser resuelto de una forma sencillo dividiendo en problemas más pequeños, aquí puede aplicarse y almacenarse el resultado en una pila que contenga todas las llamadas de los procesamientos.

Los algoritmos de divide y vencerás también pueden ser implementados procesos no recursivos que almacena los subproblemas parciales en alguna estructura de datos explícita, tales como una pila, una cola, o una cola de prioridad. Este enfoque permite más libertad a la hora de elegir los subproblemas a resolver después, una característica que es importante en algunas aplicaciones, por ejemplo, en la búsqueda en anchura y en el método de ramificación y acotación o de optimización de subproblemas.

**Ventajas:**

* Resolución de problemas complejos: Este modelo algorítmico es una herramienta potente para solucionar problemas complejos, tales como el clásico juego de las torres de Hanói.
* Paralelismo: Este tipo de algoritmos se adapta de forma natural a la ejecución en entornos multiprocesador, especialmente en sistemas de memoria compartida donde la comunicación de datos entre los procesadores no necesita ser planeada por adelantado, por lo que subproblemas distintos se pueden ejecutar en procesadores distintos.
* Acceso a memoria: Los algoritmos Divide y vencerás, tienden naturalmente a hacer un uso eficiente de las memorias cachés.
* Control del redondeo: Un algoritmo de divide y vencerás podría dar resultados más exactos que un problema iterativo equivalente superficialmente.

**Desventajas:**

* La principal desventaja de este método es su lentitud en la repetición del proceso recursivo: los gastos indirectos de las llamadas recursivas a la resolución de los subproblemas, junto con el hecho de tener que almacenar la pila de llamadas (el estado en cada punto en la repetición), pueden empeorar cualquier mejora
* La dificultad o incluso inconveniencia de aplicar el método a situaciones en las que la solución al problema general no se deriva de la suma directa y simple de los subproblemas (partes). Es decir, al tratar de solventar subproblemas pueden generar más subproblemas haciendo que la complejidad a solventar se incremente.

**Conclusión:** Algunos problemas pueden ser difíciles de solucionar cuando tratamos

de abordarlos por completo. Sin embargo, sí podemos dividirlos en partes más simples que sí sepamos resolver, estaremos en el buen camino. La estrategia divide y vencerás puede aplicarse a un buen número de ellos.

## **PROGRAMACIÓN DINÁMICA:**

**Definición:** En informática, la programación dinámica es un procedimiento para minimizar la era de ejecución de un algoritmo por medio de la implementación de subproblemas superpuestos y subestructuras (EcuRed contributors, 2010).

**Implementación:**

Donde tiene más grande aplicación la Programación Dinámica es en la resolución de inconvenientes de mejora. En esta clase de inconvenientes tienen la posibilidad de exponer diversas resoluciones, cada una con un costo, y lo cual se quiere es hallar la solución de costo óptimo (máximo o mínimo).

La solución de inconvenientes por medio de esta técnica se fundamenta en el denominado inicio de óptimo enunciado por Bellman en 1957 y que dice: “En una serie de elecciones óptima toda subsecuencia ha de ser además óptima”.

Para que un problema logre ser resuelto con la técnica de programación dinámica, debería cumplir con ciertas propiedades:

* Naturaleza secuencial de las decisiones: el problema puede ser dividido en etapas.
* Cada etapa tiene un número de estados asociados a ella.
* La decisión óptima de cada etapa depende solo del estado actual y no de las decisiones anteriores.
* La decisión tomada en una etapa determina cuál será el estado de la etapa siguiente.

Este principio parece evidente pero no siempre es aplicable y por tanto es necesario verificar (UDB, s.f.).

Jaramillo, (2016) nos describe las siguientes ventajas y desventajas:

**Ventajas**

* Eficaz para resolver problemas de gran complejidad al dividirlo y ordenarlo.
* Resuelve cada sub problema una sola vez.
* Los cálculos de cada etapa se organizan y se guardan de manera eficiente, facilitando su consulta para posteriores análisis.

**Desventajas**

* Si la red es muy grande se vuelve laborioso.
* No aplicable a todo tipo de problemas.
* Si hay un error en alguna tabla afecta a todo el problema.

**Conclusión**

* La programación dinámica es para cierto tipo de problemas, aquellos que logran subdividirse en problemas más pequeños.
* Principalmente es eficiente en problemas pequeños y simples, donde se guardan en una tabla, un arreglo, etc.
* Esto se aplica cuando los métodos más especializados fallan, pero pueden llegar a ser menos eficientes.

## **BACKTRACKING:**

**Definición:** Es un método de búsqueda de soluciones exhaustiva sobre grafos dirigidos acíclicos (Grafo dirigido acíclico: Es un grafo dirigido que no tiene ciclos, en otras palabras, esto significa que para cada vértice v, no hay camino directo que empiece y termine en v).

También se lo puede definir como un método de retroceso o vuelta atrás y es una técnica general de resolución de problemas.

Donde cada decisión nos lleva a un nuevo conjunto de decisiones, alguna secuencia de decisiones (y puede que más de una) puede solucionar nuestro problema.

Es añadir y quitar todos los elementos. Probar todas las combinaciones, por ello suele ser muy ineficiente.

**Implementación:** El objetivo de su implementación es encontrar soluciones para algún problema.

Donde se dice que:

-Su recorrido tiene éxito si: se puede definir por completo una solución, y en este caso el algoritmo puede detenerse si y sólo si necesitamos encontrar una solución o puede continuar si necesitamos alternativas de soluciones.

Por lo general se utiliza en la optimización de juegos.

**Ventajas:**

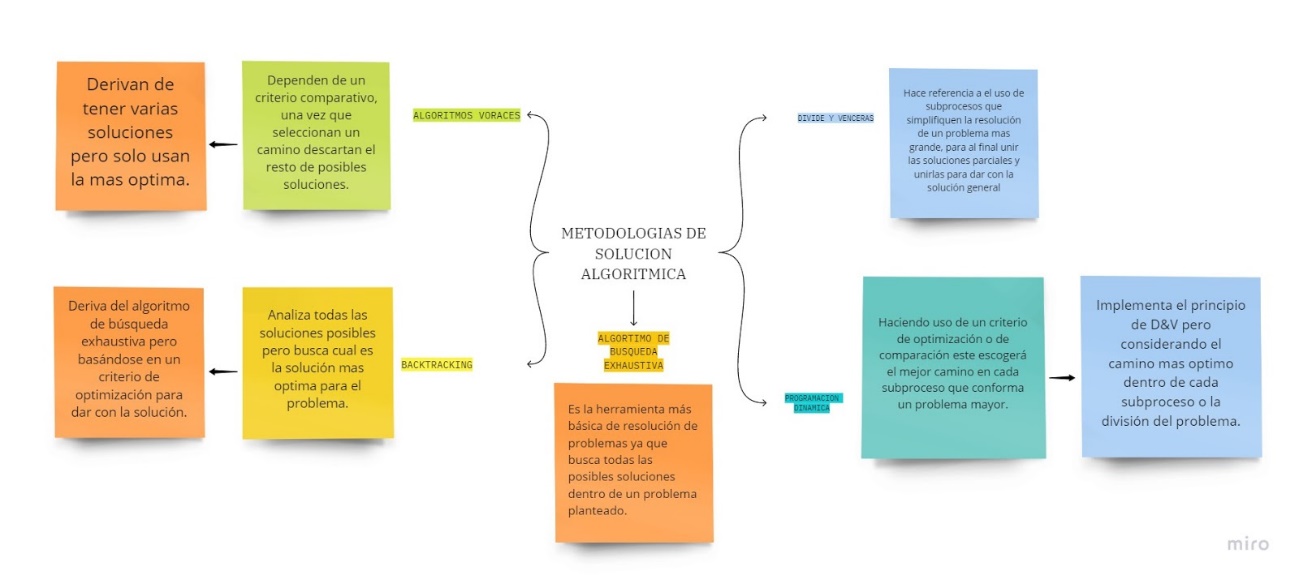
* Analiza todas las posibles soluciones en caso de ser necesario.
* Es fácil y sencillo de implementar en los problemas a resolver.
* Puede ser adaptado a las características en específico de un problema.

**Desventajas:**

* Si la solución es infinita no se encontrará nunca.
* Consume mucha memoria puesto que tiene que almacenar un árbol de soluciones.

# 

# CUADRO COMPARATIVO:

****

# ALGORITMOS:

# BIBLIOGRAFÍA:

EcuRed contributors. (12 de 11 de 2010). *Programación dinámica*. Obtenido de EcuRed: https://www.ecured.cu/index.php?title=Programaci%C3%B3n\_din%C3%A1mica&oldid=254760

Jaramillo, D. G. (12 de Noviembre de 2016). slideshare. Obtenido de Programación dinámica: <https://es.slideshare.net/DanielGmez3/programacin-dinmica-68791263>

Maquiavelo, Nicolás. DIVIDE ET IMPERA, Recuperado de: https://www.divideyvenceras.com/algoritmo/diseno-e-implementacion/

Dartmouth Computer Science Thomas Cormen y Devin Balkcom, Recuperado de: https://es.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms

UDB. (s.f.). Programación IV. Guía No. 12. Obtenido de Programación Dinámica.: <http://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/programacion-iv/2019/ii/guia-12.pdf>

Nacho Cabanes,(Abril del 2013).Jugando a crear. Optimizacion de codigo. Recuperado de: <https://nachocabanes.blogspot.com/2013/04/busqueda-exhaustiva-backtracking.html?fbclid=IwAR2MGwCZikx1By8H4_dR2sKlENf9A5yIQONqZnej45xMTHrU-EaDjoo5xCA>

