FICHA DE LECTURA

APELLIDOS Y NOMBRES:	CHANALUISA RONQUILLO DENNIS ISMAEL
CICLO:	SEGUNDO SEMESTRE
MATERIA:	LOGICA DE PROGRAMACION
FECHA DE LECTURA:	19-02-2022
TÍTULO (LIBRO)	ALLGORITMOS DE BRACKTRAKING
AUTOR: Marin	**PÁGINAS: 10**

VALORACION DEL LIBRO:	BUENO
DIFICULTAD DEL VOCABULARIO:	FACTIBLE
OBRAS SIMILARES QUE HAYA LEÍDO:	SISTEMAS
ESCRIBA 3 SUSTANTIVOS O ADJETIVOS QUE DEFINAN LAS CARACTERÍSTICAS DEL LIBRO:	1.INTERESANTE 2.ARGUMENTATIVO 3.LIBRE
TEMA DE LECTURA: BACKTRAKING	

VOCABULARIO

Grafo: Representación simbólica de los elementos constituidos de un sistema o conjunto, mediante esquemas gráficos.

Exhaustiva: Que agota la materia de que se trata o es muy completo.

TEMA QUE LE HA LLAMADO LA ATENCIÓN Y EL MOTIVO:

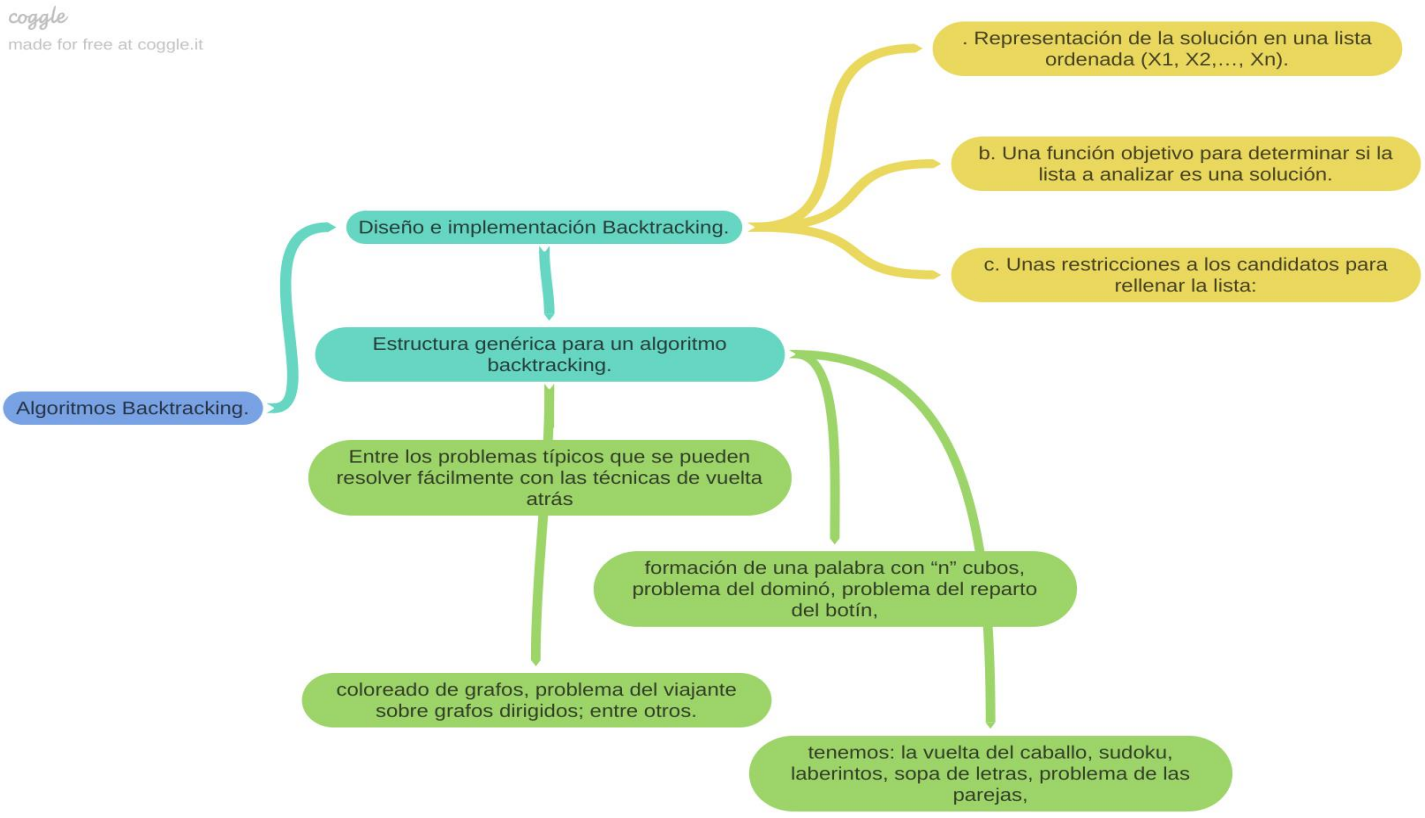
Diseño e implementación Backtracking.

Esencialmente, la idea es encontrar la mejor combinación posible en un momento determinado, por eso, se dice que este tipo de algoritmo es una búsqueda en profundidad. Normalmente, se suele implementar este tipo de algoritmos como un procedimiento recursivo. La diferencia con la búsqueda en profundidad es que se suelen diseñar funciones de cota, de forma que no se generen algunos estados si no van a conducir a ninguna solución, o a una.

solución peor de la que ya se tiene. De esta forma se ahorra espacio en memoria y tiempo de ejecución. Al diseñar un algoritmo backtraking debemos considerar los siguientes elementos: a. Representación de la solución en una lista ordenada (X1, X2,…, Xn). b. Una función objetivo para determinar si la lista a analizar es una solución. c. Unas restricciones a los candidatos para rellenar la lista: ¬ Implícitas del problema. Valores que puede tomar cada valor Xi. ¬ Explícitas o externas al problema. Por ejemplo, problema de la mochila, el peso no debe superar la capacidad de la mochila. d. Una

función de poda para eliminar partes del árbol de búsqueda e. Organización del problema en un árbol de búsqueda. f. Construir la solución al problema en distintas etapas. g. En cada paso se elige un candidato y se añade a la solución, y se avanza en la solución parcial. h. Si no es posible continuar en la construcción hacia una solución completa, se abandona ésta y la última componente se cambia por otro valor. i. Si no quedan más valores por probar, se retrocede al candidato anterior, se desecha, y se selecciona otro candidato. Para diseñar un algoritmo con la técnica backtracking, debemos seguir los siguientes pasos: 1. Buscar una representación del tipo (X1, X2,…, Xn) para las soluciones del problema. 2. Identificar las restricciones implícitas y explícitas del problema. 3. Establecer la organización del árbol que define los diferentes estados en los que se encuentra una (sub)solución. 4. Definir una función solución para determinar si una lista ordenada es solución. 5. Definir una función de poda Bk (X1, X2,…, Xk) para eliminar ramas del árbol que puedan derivar en soluciones poco deseables o inadecuadas. 6. Aplicar la estructura genérica de un algoritmo backtracking.

ESQUEMA DE LAS PARTES EN QUE PUEDE DIVIDIR LA LECTURA:



RESÚMEN:

En su forma básica, la idea de backtracking se asemeja a un recorrido en profundidad dentro de un grafo dirigido. El grafo en cuestión suele ser un árbol, o por lo menos no contiene ciclos. Sea cual sea su estructura, existe sólo implícitamente. El objetivo del recorrido es encontrar soluciones para algún problema. El recorrido tiene éxito si, procediendo de esta forma, se puede definir por completo una solución. En este caso el algoritmo puede bien detenerse (si lo único que se necesita es una solución del problema) o bien seguir buscando soluciones alternativas (si deseamos examinarlas todas). Por otra parte, el recorrido no tiene éxito si en alguna etapa la solución parcial construida hasta el momento no se puede completar. En tal caso, el recorrido vuelve atrás exactamente igual que en un recorrido en profundidad, eliminando sobre la marcha los elementos que se hubieran añadido en cada fase. Cuando vuelve a un nodo que tiene uno o más vecinos sin explorar, prosigue el recorrido de una solución. Los problemas que deben satisfacer un determinado tipo de restricciones son problemas completos, donde el orden de los elementos de la solución no importa. Estos problemas consisten en un conjunto (o lista) de variables a la que a cada una se le debe asignar un valor sujeto a las restricciones del problema. La técnica va creando todas las posibles combinaciones de elementos para obtener una solución. Su principal virtud es que en la mayoría de las implementaciones se puede evitar combinaciones, estableciendo funciones de acotación (o poda) reduciendo el tiempo de ejecución. La técnica backtracking (vuelta atrás) está muy relacionada con la búsqueda binaria**.**

Vuelta atrás (*Backtracking*) es una estrategia para encontrar soluciones a problemas que satisfacen restricciones. El término "backtrack" fue acuñado por primera vez por el matemático estadounidense D. H. Lehmer en la década de 1950.

Los problemas que deben satisfacer un determinado tipo de restricciones son problemas completos, donde el orden de los elementos de la solución no importa. Estos problemas consisten en un conjunto (o lista) de variables a la que a cada una se le debe asignar un valor sujeto a las restricciones del problema. La técnica va creando todas las posibles combinaciones de elementos para obtener una solución. Su principal virtud es que en la mayoría de las implementaciones se puede evitar combinaciones, estableciendo funciones de acotación (o poda) reduciendo el tiempo de ejecución.

Vuelta atrás está muy relacionado con la búsqueda combinatoria.

Diseño e implementación

Esencialmente, la idea es encontrar la mejor combinación posible en un momento determinado, por eso, se dice que este tipo de algoritmo es una búsqueda en profundidad. Durante la búsqueda, si se encuentra una alternativa incorrecta, la búsqueda retrocede hasta el paso anterior y toma la siguiente alternativa. Cuando se han terminado las posibilidades, se vuelve a la elección anterior y se toma la siguiente opción (hijo [si nos referimos a un árbol]). Si no hay más alternativas la búsqueda falla. De esta manera, se crea un árbol implícito, en el que cada nodo es un estado de la solución (solución parcial en el caso de nodos interiores o solución total en el caso de los nodos hoja).

Normalmente, se suele implementar este tipo de algoritmos como un procedimiento recursivo. Así, en cada llamada al procedimiento se toma una variable y se le asignan todos los valores posibles, llamando a su vez al procedimiento para cada uno de los nuevos estados. La diferencia con la búsqueda en profundidad es que se suelen diseñar funciones de cota, de forma que no se generen algunos estados si no van a conducir a ninguna solución, o a una solución peor de la que ya se tiene. De esta forma se ahorra espacio en memoria y tiempo de ejecución.

Heurísticas

Algunas heurísticas son comúnmente usadas para acelerar el proceso. Como las variables se pueden procesar en cualquier orden, generalmente es más eficiente intentar ser lo más restrictivo posible con las primeras (esto es, las primeras con menores valores posibles). Este proceso poda el árbol de búsqueda antes de que se tome la decisión y se llame a la subrutina recursiva.

Cuando se elige qué valor se va a asignar, muchas implementaciones hacen un examen hacia delante (FC, Forward Checking), para ver qué valor restringirá el menor número posible de valores, de forma que se anticipa en a) preservar una posible solución y b) hace que la solución encontrada no tenga restricciones destacadas.

Algunas implementaciones muy sofisticadas usan una función de cotas, que examina si es posible encontrar una solución a partir de una solución parcial. Además, se comprueba si la solución parcial que falla puede incrementar significativamente la eficiencia del algoritmo. Por el uso de estas funciones de cota, se debe ser muy minucioso en su implementación de forma que sean poco costosas computacionalmente hablando, ya que lo más normal es que se ejecuten en para cada nodo o paso del algoritmo. Cabe destacar, que las cotas eficaces se crean de forma parecida a las funciones heurísticas, esto es, relajando las restricciones para conseguir mayor eficiencia.

ranch & Bound (Ramificación y poda)

Este método busca una solución como en el método de backtracking, pero cada solución tiene asociado un costo y la solución que se busca es una de mínimo costo llamada óptima. Además de ramificar una solución padre (branch) en hijos se trata de eliminar de consideración aquellos hijos cuyos descendientes tienen un costo que supera al óptimo buscado acotando el costo de los descendientes del hijo (bound). La forma de acotar es un arte que depende de cada problema. La acotación reduce el tiempo de búsqueda de la solución óptima al "podar" (pruning) los subarboles de descendientes costosos.

HIPÓTESIS, TESIS, IDEAS CENTRALES DEL TEXTO:

Con el objetivo de mantener la solución actual con coste mínimo, los algoritmos vuelta atrás mantienen el coste de la mejor solución en una variable que va variando con cada nueva mejor solución encontrada. Así, si una solución es peor que la que se acaba de encontrar, el algoritmo no actualizará la solución. De esta forma, devolverá siempre la mejor solución que haya encontrado. Y recuerden, la vida es un backtracking.

PALABRAS Y EXPRESIONES CLAVES:

Heurística: De la heurística o relacionado con ella.

En su forma básica, la idea de backtracking se asemeja a un recorrido en profundidad dentro de un grafo dirigido. El grafo en cuestión suele ser un árbol, o por lo menos no contiene ciclos. Sea cual sea su estructura, existe sólo implícitamente. El objetivo del recorrido es encontrar soluciones para algún problema. Esto se consigue construyendo soluciones parciales a medida que progresa el recorrido; estas soluciones parciales limitan las regiones en las que se puede encontrar una solución completa. El recorrido tiene éxito si, procediendo de esta forma, se puede definir por completo una solución.

OBSERVACIONES PERSONALES E INTERPRETACIÓN:

En este caso el algoritmo puede bien detenerse (si lo único que se necesita es una solución del problema) o bien seguir buscando soluciones alternativas (si deseamos examinarlas todas). Por otra parte, el recorrido no tiene éxito si en alguna etapa la solución parcial construida hasta el momento no se puede completar. En tal caso, el recorrido vuelve atrás exactamente igual que en un recorrido en profundidad, eliminando sobre la marcha los elementos que se hubieran añadido en cada fase. Cuando vuelve a un nodo que tiene uno o más vecinos sin explorar, prosigue el recorrido de una solución.