Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

PROYECTO 3 – DESARROLLO DE API

201800951 – Pablo Gerardo Schaart Calderon

Resumen

El programa trabaja transdormando matrices de frecuencia de acceso en matrices reducidas convirtiéndola primero en una matriz binaria lo cual tendrá un gran impacto en la solución de los problemas combinatorios NP-Hard los cuales tiene un fuerte requerimiento de tiempo y una fuerte demanda de recursos.

El método propuesto para resolver el problema NP-Hard consiste en aplicar una metodología de agrupamiento.

El programa presentado a continuación tiene como función principal eficientar el almacenamiento de datos de las bases de datos detectando patrones y reduciendo las matrices de frecuencia de acceso.

La solución además presenta los datos tanto gráficamente como un archivo con extensión XML para mejor análisis e interpretación si se desea trabajar con algún intérprete.

La forma gráfica se presenta haciendo utilización de herramientas de software utilizando el lenguaje descriptivo DOT.

Palabras clave:

Tupla: es una secuencia o lista ordenada finita de n objetos

Abstract

The program works by transforming access frequency matrices into reduced matrices, converting it first into a binary matrix, which will have a great impact on solving the NP-Hard combinatorial problems which have a strong time requirement and a strong demand for resources.

The proposed method to solve the NP-Hard problem consists of applying a grouping methodology.

The program presented below has the main function of making the data storage of the databases more efficient by detecting patterns and reducing the access frequency matrices.

The solution also presents the data both graphically as a file with an XML extension for better analysis and interpretation if you want to work with an interpreter.

The graphic form is presented using software tools using the DOT descriptive language.

Keywords

Tuple: is a sequence or finite ordered list of n objects

Introducción

En el entorno del mercado actual, la competitividad y la rapidez de maniobra de una empresa son imprescindibles para su éxito. Para conseguirlo existe cada vez una mayor demanda de datos y, por tanto, más necesidad de gestionarlos. Esta demanda siempre ha estado patente en empresas y sociedades, pero en estos años se ha disparado debido al acceso multitudinario a las redes integradas en Internet y a la aparición de los dispositivos móviles que también requieren esa información.

En informática se conoce como dato a cualquier elemento informativo que tenga relevancia para un usuario. Desde su nacimiento, la informática se ha encargado de proporcionar herramientas que faciliten la manipulación de los datos. Antes de la aparición de las aplicaciones informáticas, las empresas tenían como únicas herramientas de gestión de datos los ficheros con cajones, carpetas y fichas de cartón. En este proceso manual, el tiempo requerido para manipular estos datos era enorme. Pero la propia informática ha adaptado sus herramientas para que los elementos que el usuario utiliza en cuanto a manejo de datos se parezcan a los manuales. Por eso se sigue hablado de ficheros, formularios, carpetas, directorios,….

Desarrollo del tema

La optimización combinatoria es una rama de la optimización en matemáticas aplicadas y en ciencias de la computación, relacionada con la investigación de operaciones, Teoría algorítmica de la información y teoría de la complejidad computacional. También está relacionada con otros campos, como la inteligencia artificial e ingeniería de software. Los algoritmos de optimización combinatoria resuelven instancias de problemas que se creen ser difíciles en general, explorando el espacio de soluciones (usualmente grande) para estas instancias. Los algoritmos de optimización combinatoria logran esto reduciendo el tamaño efectivo del espacio, y explorando el espacio de búsqueda eficientemente.

Los algoritmos de optimización combinatoria a menudo son implementados en lenguajes imperativos como C y C++ entre otros softwares inteligentes en lenguajes de programación lógicos tales como Prolog, o incluso en lenguajes multi-paradigma tales como Oz.

Mediante el estudio de la teoría de la complejidad computacional es posible comprender la importancia de la optimización combinatoria. Los algoritmos de optimización combinatoria se relacionan comúnmente con problemas NP-hard. Dichos problemas en general no son resueltos eficientemente, sin embargo, varias aproximaciones de la teoría de la complejidad sugieren que ciertas instancias (ej. "pequeñas" instancias) de estos problemas pueden ser resueltas eficientemente. Dichas instancias a menudo tienen ramificaciones prácticas muy importantes.

Complejidad algorítmica

Es difícil realizar un análisis simple de un algoritmo que determine la cantidad exacta de tiempo que este requiere para ser ejecutado, porque depende en gran parte del algoritmo y de la computadora en que se ejecute. Además, conocer el tiempo exacto que tardará un programa de cómputo en dar resultados es una tarea difícil de determinar.

En su lugar, es mejor calcular la cantidad de operaciones que se realizan de acuerdo con los datos de entrada del problema a tratar, lo que se conoce como cálculo de la función temporal.

Así, una vez que se cuenta con un algoritmo que funciona de manera idónea, es necesario definir los criterios que permitan medir su rendimiento o comportamiento. Estos deben considerar el uso eficiente de los recursos y la simplicidad del algoritmo. El que sea sencillo no le demerita calidad, ya que su simplicidad facilita su mantenimiento, su verificación y su eficiencia.

Memoria ocupada

La eficiencia de la memoria o complejidad espacial de un algoritmo muestra la cantidad de memoria que ocupan todas las variables utilizadas por este, es decir, la suma del almacenamiento necesario para ejecutarlo, que está constituida por la memoria estática y la memoria dinámica.

Para el cálculo de la memoria estática solo hay que sumar la memoria ocupada por cada una de las variables declaradas en el algoritmo. El cálculo de la memoria dinámica depende de la ejecución del algoritmo y puede ser liberada, se modifica de forma permanente; concretamente, es un espacio de almacenamiento que se solicita en el tiempo de ejecución.

La complejidad en función de la memoria utilizada radica en el consumo del espacio de esta en la computadora. Un problema con costo espacial elevado gastará una cantidad de memoria con incremento exponencial conforme el problema aumente en tamaño, pues el número de variables a utilizar también aumentará. Esto en algún momento causaría que el algoritmo no se realice correctamente por falta de memoria en la computadora

Tiempo de ejecución

La complejidad temporal de un algoritmo computacional se evalúa generando una función, llamada función temporal, la cual define el número de instrucciones ejecutadas por el programa cuando se resuelve un problema. De este número de instrucciones se puede obtener el tiempo aproximado si se conoce el tiempo que el equipo de cómputo en uso tarda en ejecutar una instrucción. De esa manera se puede representar el número de unidades de tiempo requeridas para que un programa o algoritmo de cualquier entrada de datos de tamaño n produzca un resultado. El tiempo de ejecución de un algoritmo depende del número de datos de entrada n del problema y de la velocidad que el equipo de cómputo posea.

Conclusiones

La complejidad algorítmica de la solución a los problemas Np-Hard tiene muchos puntos a tomar en cuenta desde la ocupación de memoria hasta el consumo de recursos durante la ejecución por lo cual es obligatorio el uso de teoría de grafos para la solución de la misma

El buen uso y manejo de los recursos computacionales puede resultar en un gran ahorro de tiempo y eficacia por lo que permite resolver problemas mas complejos en menor tiempo.

Referencias bibliográficas

Jcc.dcc.fceia.unr.edu.ar
URL:https://jcc.dcc.fceia.unr.edu.ar/2016/slides/2016
Paredes-restrepo.pdf

Redalyc.org. 2021. [online] Available at:
<https://www.redalyc.org/jatsRepo/5075/507555007
001/507555007001.pdf> [Accessed 9 March 2021].