Enunciado 3, Problema de Goloso

Análisis e Implementación

Dany Efraín Rubiano Jiménez
Departamento de Ingeniería Informática
Universidad de Santiago de Chile
Santiago de Chile, Chile
dany.rubiano@usach.cl

***Resumen*— En el presente documento se analiza la aplicación del método de goloso, con el fin de resolver una problemática dada, aplicando su definición y explicando cómo se contextualiza en el desarrollo de esta.**

Index Terms—Goloso, Algoritmo, C.

I. Introducción

A pesar de que existen incontables algoritmos, se han logrado identificar un número de técnicas de diseño suficiente mente generales como para poder aplicarse a un gran número de problemas. Estas técnicas dan como resultado algoritmos eficientes (a veces...) y que son de muy diversos dominios de a plicación.

Para esta ocasión se aborda un problema dado, para el cual se aplicará la técnica de resolución de goloso, buscando encontrar una solución que cumpla con las condiciones y restricciones dadas.

II. Descripción del Problema

El problema dado, al cual se aplicará la técnica de resolución de goloso se contextualiza en una sala de clases, donde un profesor se encuentra aburrido del desorden que mantienen sus alumnos en clases. Para solucionar el problema el profesor ha clasificado a los alumnos en 4 grupos (A, B, C y D) y solicita que se haga un programa que redistribuya a los estudiantes dentro de la sala de clases. Esta sala es representada por una matriz de n x m dimensiones, y debe reorganizarse de tal modo que no queden juntos dos niños del mismo grupo (ni vertical ni horizontalmente dentro de la matriz).

Para ello se entrega una matriz, la cual representa el estado actual en que están organizados los alumnos, estos últimos están identificados con el grupo al que pertenecen, un ejemplo de ello, se encuentra en la *Tabla 1.* Esta Entrada está contenida en un archivo de nombre "entrada.in" que contiene dicha matriz.

TABLE I. Tabla Complejidad Algorítmica

Filas	**Columnas**				
	Col₁	***Col₂***	***Col₃***	***Col₄***	***Col₅***
Fila₁	A	A	B	D	C
Fila₂	A	B	D	B	C
Fila₃	A	C	C	B	D
Fila₄	C	A	A	B	D
Fila₅	C	D	A	D	A

Para el resultado, se debe generar un archivo de nombre "salida.out" donde se escriba la matriz que contenga la nueva representación de la sala de clases y conteniendo la redistribución que cumpla con las condiciones impuestas por el profesor. Un ejemplo para la salida se representa en la Tabla II.

TABLE II. Tabla Complejidad Algorítmica

Filas	**Columnas**				
	Col₁	***Col₂***	***Col₃***	***Col₄***	***Col₅***
Fila₁	A	C	A	D	A
Fila₂	B	A	C	A	D
Fila₃	C	B	D	B	A
Fila₄	D	C	A	C	B
Fila₅	A	D	B	D	C

III. Explicación de la Solución

En el problema dado, se requiere encontrar una redistribución de los alumnos en la sala que cumpla los criterios indicados por el profesor. Para lo anterior, se debe en primer lugar obtener la cantidad de alumnos que hay en la sala, además de obtener la cantidad de grupos y alumnos que componen a cada uno. Ya encontrados estos valores, se debe aplicar las restricciones que indicó el profesor y redistribuir

todos los alumnos encontrando así una solución para el problema.

La técnica requerida en la resolución del problema es la de goloso. Siguiendo este método, se debe tener una restricción, o mejor dicho un criterio Goloso, para generar la reasignación. Resulta lógico entonces, que el criterio goloso utilizado sean las indicaciones del profesor, estas son, que según la clasificación dada de los alumnos por grupos, no quede junto ningún alumno de un mismo grupo ni a derecha, izquierda, arriba o abajo. De esta manera, se puede encontrar una solución al problema dado.

IV. Método Utilizado

Para la resolución del problema, tal como se mencionó anteriormente se utiliza la técnica de goloso, esta es una de las técnicas más versátiles y simples de entender, también se le denomina como "algoritmo voraz" (Greedy algorithm). A pesar de que no siempre es capaz de encontrar una respuesta óptima, se lo utiliza con frecuencia dado que es muy rápido.

La técnica de goloso 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. Este tipo de algoritmo, es el que menos dificultades presenta a los investigadores que deben diseñar y comprobar el funcionamiento de diferentes estrategias. El nombre "voraz" se debe a que, en cada paso, el algoritmo escoge el mejor "pedazo" que es capaz de "comer" sin preocuparse de los pasos que restan hasta encontrar la solución. Un algoritmo de este tipo nunca deshace una decisión ya tomada: una vez incorporado, un "candidato a la solución" formará parte de la solución.

Una vez aclarado el hecho de que la técnica de goloso procede por pasos, se puede ver en detalle cómo es su estructura. Se parte de un conjunto de candidatos que se encuentra vacío, es decir, no hay ninguna solución. Luego, en cada paso, se intenta añadir al conjunto el mejor candidato entre las soluciones que aún no han sido escogidas, a través de una función de selección. Luego de cada incorporación se comprueba si el conjunto de candidatos resultante es una solución del problema. Más formalmente, Dado un conjunto finito de entradas C, goloso devuelve un conjunto S (seleccionados) tal que S $\subseteq$ C y que además cumple con las restricciones del problema inicial. A cada conjunto S que satisfaga las restricciones se le suele denominar prometedor, y si este además logra que la función objetivo se minimice o maximice (según corresponda) diremos que S es una solución óptima.

La técnica incluye los siguientes elementos:

- El conjunto C de candidatos, entradas del problema.
- Función solución. Comprueba, en cada paso, si el subconjunto actual de candidatos elegidos forma una solución (no importa si es óptima o no lo es).
- Función de selección. Informa de cuál es el elemento más prometedor para completar la solución. Éste no puede haber sido escogido con anterioridad. Cada elemento es considerado una sola vez. Luego, puede ser rechazado o aceptado y pertenecerá a C \ S.
- Función de factibilidad. Informa si a partir de un conjunto se puede llegar a una solución. Lo aplicaremos al conjunto de seleccionados unido con el elemento más prometedor.
- Función objetivo. Es aquella que queremos maximizar o minimizar, el núcleo del problema.

El algoritmo escoge en cada paso al mejor elemento x $\in$ C posible, conocido como el elemento más prometedor. Se elimina ese elemento del conjunto de candidatos (C <¬ C /{x}) y, acto seguido, comprueba si la inclusión de este elemento en el conjunto de elementos seleccionados (S $\cup${x}) produce una solución factible.

Todo esto se resume en un esquema genérico, el cual es:

función voraz(C:conjunto):conjunto
{ C es el conjunto de todos los candidatos }
S <= vacio { S es el conjunto en el que se construye la solución}
mientras ¬solución(S) y C <> vacío hacer
x <= el elemento de C que maximiza seleccionar(x)
C <= C {x}
si completable(S U {x}) entonces S <= S U {x}
si solución(S)
entonces devolver S
si no, devolver no hay solución

V. Descripción de la Solución

Para dar paso a establecer la solución del problema, primero se debe pasar la matriz de entrada a una estructura que almacene todos los datos del archivo dado, y así poder operarlos. Puesto que los alumnos pertenecen a un grupo, y los grupos son cuatro en total, entonces se puede tener una estructura fija que contenga estos, de esta manera, se puede guardar la cantidad de alumnos por grupo que hay en la sala. Se escoge como la estructura a utilizar a un arreglo, debido a que si se guarda la representación como una matriz, para recorrer esta, se hace necesario dos ciclos anidados, en cambio con un arreglo, se hace necesario un ciclo. En el arreglo, cada posición representa a un grupo, es decir, la posición 0 corresponde al grupo 'A', la posición 1 al grupo 'B', y así sucesivamente para todos los grupos.

Luego, se debe leer el archivo y guardar los datos en el arreglo.

Se necesita así mismo tener las dimensiones de la sala para generar la reasignación. Para esto, se cuentan el número de líneas que presenta el archivo y así obtener la cantidad de filas, en el caso de las columnas, se cuenta la cantidad de alumnos y se divide en el número de filas obtenidas. Ya con estos valores,

se genera una matriz que muestra la representación de la sala, y permite genera la salida.

Para encontrar una solución, como método elegido, se recorre el arreglo obtenido buscando la posición, es decir, el grupo, donde se encuentra la mayor cantidad de alumnos para insertarla en la nueva representación de la sala, siempre y cuando se cumpla con el criterio goloso, que cumpla que no se repita un alumno del mismo grupo en la posición siguiente, tanto arriba, como abajo, derecha e izquierda. Todo esto, para todos los alumnos. Si se inserta un elemento, se debe restar al valor de la posición insertada en el arreglo de los grupos.

Ya obtenida la solución, basta solo con imprimir el resultado en una archivo de texto.

Las funciones implementadas para llevar a cabo esta solución son:

- *contarLineasDeArchivo( )*
  Esta función cuenta el número de líneas que tiene el archivo, lo que corresponde a la cantidad de filas que tiene la sala.

- *leerArchivo( )*
  Lee el archivo y guarda en el arreglo de tamaño cuatro, la cantidad de alumnos por cada grupo, en donde cada posición representa al grupo, tal como se dijo anteriormente.

- *buscarMayor( )*
  Recorre el arreglo que contiene los grupos y busca en cual grupo está el valor de mayor cantidad de alumnos, luego se retorna la posición.

- *reasignar( )*
  Para la totalidad de los alumnos, busca paso por paso el grupo con mayor cantidad de alumnos, mediante el llamado a la función 'buscarMayor', y se van insertando si se cumple con las condiciones indicadas.

- *mostrarResultado( )*
  Debido a que en la función 'reasignar', se insertan los alumnos en una representación de los grupos como números, entonces se debe pasar estos elementos a la representación inicial.
  Posteriormente esta función imprime la matriz solución en un archivo de texto.

- *main( )*
  Función principal en la que se van llamando las funciones anteriores.

VI. Complejidad Algorítmica

La complejidad algorítmica representa la cantidad de recursos (temporales) que necesita un algoritmo para resolver un problema y por tanto permite determinar la eficiencia de dicho algoritmo. En esta ocasión, se representa el orden de cada una de las funciones utilizadas, calculado por el número de ciclos. El cálculo del orden para cada una de las funciones establecidas en la resolución del problema se encuentran en la Tabla III.

TABLE III. Tabla Complejidad Algorítmica

Función	Orden
contarLineasDeArchivo	O (n)
leerArchivo	O (n)
buscarMayor	O (n)
Reasignar	O (n^4)
mostrarResultado()	O (n^2)
Main	O (n^3)

VII. Traza

En la Tabla IV, se muestra una traza simple recortada en base a la entrada propuesta en la Tabla I.

TABLE IV. Tabla Traza Simple

Función	Resultado - Traza
contarLineasDeCodigo( )	5
leerArchivo( )	[8, 5, 6, 6]
Int buscarMayor(int grupos[])	// Se usa en la siguiente función
void reasignar (int n, int m, int z, int grupos[], int salida[n][m])	// Donde el primer elemento corresponde al valor de buscarMayor(), y los siguientes dos elementos indican donde se inserta el elemento en la matriz de salida. 0 , [0][0] 0 , [0][2] 0 , [0][4] 2 , [1][0] 3 , [1][1] 0 , [1][3] 1 , [1][4] 2 , [2][1] 3 , [2][2] 0 , [2][4] 1 , [3][0] 2 , [3][2] 3 , [3][3] 0 , [4][0] 1 , [4][1] 2 , [4][3] 3 , [4][4] 0 , [2][0] 1 , [2][3] 2 , [3][4] 3 , [4][2] 0 , [3][1] 1 , [0][1] 2 , [0][3]

Función	Resultado - Traza
	// De esta forma, la matriz resultante es: 0 1 0 2 0 2 3 -1 0 1 0 2 3 1 0 1 0 2 3 2 0 1 3 2 3
void mostrarResultado(int n, int m, int salida[n][m], char out[n][m])	A B A C A C D - A B A C D B A B A C D C A B D C D
main( )	15234

VIII. Eficiencia

La eficiencia de cada una de las funciones empleadas en la resolución del problema, es de alguna manera aceptable, ya que en su solución entregan el resultado esperado, y en un tiempo mínimo, que no alcanza a ser ni siquiera un segundo.

Al utilizar un arreglo para la representación inicial, en vez de guardar la misma matriz ilustrada en el archivo, se redujo el orden del algoritmo en varias de las funciones implementadas, debido a que se evitó un for anidado más para el recorrido de la estructura y posterior uso de los elementos en las otras funciones. Estas funciones son, 'leerArchivo( )' y 'reasignar( )'.

De por si, por el hecho de utilizar la técnica de goloso, da una eficacia en la resolución del problema.

IX. Conclusiones

Como conclusiones se puede afirmar que se logró de alguna manera llegar a los resultados correctos, si lo probamos con los ejemplos que están en el enunciado del laboratorio.

Las estructuras de datos utilizadas se usaron en base a los algoritmos planteados y cumplen la función para lo que fueron declaradas.

Las técnicas de desarrollo de algoritmos nos permiten encontrar solución a los problemas que se nos presentan y deben ser solucionados por el computador, estas técnicas están orientadas para utilizarse en cada uno de los niveles de complejidad y variedad o alternativas para las cuales se aplican los algoritmos, en el caso de la fuerza bruta, para la mayoría de problemas reales, el número de soluciones candidatas es prohibitivamente elevado, por lo cual es ineficiente ante casos muy grandes. El tiempo de ejecución y la eficiencia de la solución con este método, depende del tamaño del problema.

Si se compara la técnica de goloso con la fuerza bruta, obviamente, este tiene una mayor eficiencia en la utilización de recursos y en el tiempo de ejecución, es por esto, que esta técnica es ampliamente utilizada.

Referencias

[1] http://webdiis.unizar.es/asignaturas/EDA/ea/slides/2-Algoritmos%20Voraces.pdf

[2] http://es.wikibooks.org/wiki/Algoritmia/Algoritmos_voraces