Algoritmos Avanzados Laboratorio 3

NICOLÁS ROMERO FLORES

Profesora: Mónica Villanueva

Ayudante: Gerardo Zúñiga

Fecha: August 3, 2018

Compiled August 3, 2018

En el presente informe se muestra el problema y la solución correspondiente al tercer laboratorio del curso de Algoritmos Avanzados. La solución se elabora utilizando la técnica de resolución de problemas conocida como Goloso. Se realiza un análisis de los resultados obtenidos, viendo la eficacia y eficiencia de la solución.

1. INTRODUCCIÓN

El laboratorio numero tres del curso de Algoritmos Avanzados consiste en la aplicación de la técnica de resolución de problemas conocida como Goloso, y de esta forma desarrollar un algoritmo que resuelva el problema planteado en el enunciado. el cual consiste en dado una distribución de colonias de bacterias inicial, encontrar la secuencia de agrupamientos que de como resultado el menor numero de colonias separadas. El problema sera explicado con mas detalles en secciones posteriores. Finalmente, se hará un análisis del algoritmo desarrollado, es decir, se ve la eficacia y eficiencia del algoritmo. Para el caso de la eficiencia, se analiza el tiempo de ejecución y orden de complejidad del algoritmo, de esta manera se puede determinar efectivamente lo eficiente del algoritmo.

A. Objetivo Principal

1. Desarrollar un algoritmo que mediante el uso de Goloso resuelva el problema planteado en el enunciado, el cual consiste en encontrar la secuencia de agrupamientos que de el menor numero de colonias separadas.

B. Objetivos Secundarios

1. Obtener la secuencia de agrupamientos que de el menor numero de colonias separadas.
2. Responder las preguntas principales de un algoritmo: ¿Se detiene?, Cuando se detiene ¿entrega la solución correcta?, ¿Es eficiente?, ¿Se puede mejorar?, ¿Existen otros métodos?
3. Para el caso de la eficiencia, estimar el orden de complejidad del algoritmo desarrollado usando la cota superior asintótica (notación O grande).

C. Estructura del informe

En el informe se presentan la descripción del problema , luego un marco teórico que define conceptos clave que se utilizaran a lo largo del informe, se procede a describir y analizar la solución obtenida para el problema para finalmente, hacer un análisis de esta respondiendo las preguntas anteriormente planteadas, haciendo énfasis en la eficiencia del algoritmo.

2. DESCRIPCIÓN DEL PROBLEMA

Un grupo de investigadores ha descubierto una nueva especia de bacteria, las cuales tienen a agruparse en colonias de 3 tipos, catalogadas como tipo 1, 2 y 3. En condiciones hostiles, dos colonias de distinto tipo pueden unirse, convirtiéndose así en el tipo que ninguna de las dos era.

Los investigadores piden que se desarrolle un programa escrito en Lenguaje de programación C que utilizando la técnica de resolución de problemas Goloso, obtenga la secuencia de agrupamientos que finalice con el menor numero de colonias separadas.

La distribución inicial de las colonias sera entregada mediante un archivo de texto con nombre "entrada.in", y la salida debe ser entregada en un archivo de texto plano llamado "salida.out", el cual debe listar linea a linea los pasos que se requieren para obtener la distribución mas apta para sobrevivir en el ambiente.

3. MARCO TEÓRICO

Se describirán algunos de los conceptos usados a lo largo del informe:

1. **Algoritmo:** Secuencia de instrucciones finita que permite encontrar la solución a un determinado problema.

2. **Goloso:** Corresponde a la técnica de resolución de problemas que sigue la heurística de encontrar la solución óptima local en cada etapa con la esperanza de llegar al óptimo global, situación que no siempre se logra.

3. **Backtracking:** Consiste en una forma de resolución de problemas en la cual se van encontrando lo candidatos a solución usando búsqueda en profundidad, y que si se encuentra una alternativa incorrecta o que no sea mejor que la que ya se ha encontrado, la búsqueda retrocede un paso atrás y toma la siguiente opción, de esta manera se "poda" el árbol de búsqueda, evitando visitar estados que de antemano se saben no entregaran una solución mejor que la ya encontrada.

4. **Fuerza bruta:** Consiste en una forma de resolución de problemas en la cual se deben encontrar todos los candidatos posibles a solución, y luego dentro de ese conjunto encontrar el subconjunto que cumpla las condiciones dadas.

5. **Eficiencia de un algoritmo:** Corresponde a las propiedades del algoritmo que nos permite identificar cuanto tiempo tarda un algoritmo en completarse y cuantos recursos usara. Por lo tanto una mayor eficiencia significara que el algoritmo usara menos recursos y tardara menos tiempo en resolver un problema.

6. **Tiempo de ejecución:** Expresión algebraica que indica el tiempo en que demora ejecutar un algoritmo, en función de la entrada de este. Las instrucciones mas básicas poseen un tiempo de ejecución constante 1.

7. **Orden de complejidad:** Corresponde a la estimación del tiempo de ejecución usando la cota superior asintótica de esta.

8. **Lenguaje de programación C:** Lenguaje de programación imperativo-procedural lanzado en 1972. Es un lenguaje de nivel medio que destaca por el manejo manual de memoria, poder crear tipos de datos compuestos y estructuras, además de las características propias de su paradigma.

4. DESCRIPCIÓN DE LA SOLUCIÓN

Para encontrar la secuencia de agrupamientos que dé el menor numero de colonias separadas, se toma lo distribución inicial y se generan todos los estados siguientes, de esos estados, se elige el menor "numero" formado por la distribución y se repite el procedimiento. De esta forma se aplica la técnica de resolución de problemas goloso, ya que se elige siempre el óptimo local con el deseo de poder llegar al óptimo global.

A. Funciones y procedimientos

Las principales funciones, procedimientos y algoritmos utilizados para resolver el problema serán descritos en esta sección:

- **unir:** Función que une dos tipos de colonias y entrega la nueva distribución, además si la distribución encontrada menor a la menor encontrada previamente la reemplaza. **Entrada:** Distribución actual, valor a insertar, tamaño de la distribución actual, posición en la que hacer el cambio **Salida:** Distribución nueva

- **Algoritmo unirse:** Es el algoritmo principal que se encarga de encontrar el óptimo local en cada etapa, es decir de encontrar la distribución que entregue el menor numero en cada etapa. **Entrada:** Archivo de salida, distribución inicial. **Salida:** Secuencia de agrupamientos que da el menor numero de colonias separadas.

```
size = len(distribucion)+1
min = distribucion
flag = true
cambio = false
Mientras flag:
    i = 0
    Mientras i + 1 < size:
        Si distribucion[i] == 1:
            Si distribucion[i+1] == 2:
                aux = unir(distribucion, 3, size, i)
            Si no:
                aux = unir(distribucion, 2, size, i)
            Si aux < min:
                min = aux
            cambio = true
        Si distribucion[i] == 2:
            Si distribucion[i+1] == 3:
                aux = unir(distribucion, 1, size, i)
            Si no:
                aux = unir(distribucion, 3, size, i)
            Si aux < min:
                min = aux
            cambio = true
        Si distribucion[i] == 3:
            Si distribucion[i+1] == 1:
                aux = unir(distribucion, 2, size, i)
            Si no:
                aux = unir(distribucion, 1, size, i)
            Si aux < min:
                min = aux
            cambio = true
        i++
    Si cambio:
        Si size > 0:
            cambio = false
        Si no:
            flag = false
        escribirLineaEnArchivo(salida, min)
    Si no:
        flag = false
    size--
    distribucion = min
```

En el algoritmo mostrado, se evito mostrar las conversiones de tipo entre string e integer, sin embargo al implementar este algoritmo en el lenguaje de programación C fue necesario hacerlo.

5. ANÁLISIS DE LA SOLUCIÓN Y RESULTADOS

1. **¿El algoritmo se detiene?** Si se detiene, pues cuando solo quede una sola colonia o varias colonias del mismo tipo el algoritmo se detendra.

2. **¿Resuelve el problema?** Al usar la técnica de resolución de problemas goloso no se puede asegurar que encuentra la solución global optima.

3. **¿Es eficiente?** Es rápido, ya que va descartando soluciones en cada etapa, sin embargo, no se puede asegurar que encuentre la solución global óptima.

4. **¿Se puede mejorar?** Si, en la solución se escribió cada combinación de colonias, por lo que se si descubriera otra colonia este código quedaría obsoleto. Para mejorar esto se pueden generar las combinaciones de colonias iterativamente.

5. **¿Existen otros métodos?** Si, por ejemplo backtracking. La solución tardaría mucho mas en encontrarse, pero si seria la mejor solución.

A. Complejidad de la solución

La complejidad del algoritmo dependerá del tamaño de la entrada de este, en este caso el tamaño de la entrada corresponde a la cantidad de colonias iniciales que vengan en el archivo de entrada.

1. **unir:** Recorre la cadena de caracteres (la distribución de colonias) y junta las colonias indicadas. Por lo tanto su orden de complejidad es:

$$O(n) \quad \textbf{(1)}$$

2. **unirse:** Para cada distrubucion (n), une (n) todas las colonias posibles buscando la que de el menor numero para este estado (n), por lo tanto el orden es de:

$$O(n^3) \quad \textbf{(2)}$$

Por lo tanto, acotando superiormente, el orden de complejidad de la solución propuesta es de:

$$O(n^3) \quad \textbf{(3)}$$

Cabe destacar que a pesar de dar un orden de complejidad alto, el tiempo de ejecución seria menor de todas formas a otros métodos como fuerza bruta, ya que no revisa cada estado posible.

6. TRAZA DE LA SOLUCIÓN

En la figura 1 se muestra un extracto de la traza de la solución dado un archivo de entrada.

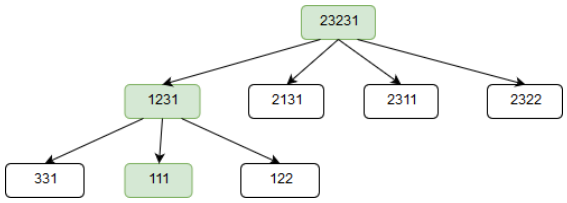

Fig. 1. Traza de la solución.

7. CONCLUSIONES

La solución implementada cumple efectivamente con lo que se pide en el enunciado, que corresponde a desarrollar un programa que use la técnica de resolución de problemas Goloso, pues en el código desarrollado se busca el óptimo local en cada etapa (la combinación de colonias que de el menor numero). El algoritmo propuesto responde correctamente a las preguntas de análisis del algoritmo. En cuanto a la eficiencia de este, el algoritmo tiene un orden de complejidad indicado en la ecuación 4, el cual en primera instancia puede parecer alto, sin embargo, el tiempo de ejecución si se ve reducido en comparación a otros métodos, ya que evita tener que revisar todos los estados posibles. Este método (Goloso), sin embargo, no necesariamente entrega la solución óptima.

$$O(n^3) \quad \textbf{(4)}$$

Sera necesario implementar una solución con otra técnica de resolución de problemas si se desea encontrar la solución óptima.