TALLER N°1

El problema de los bidones de
agua con el algoritmo A* y heurística

usach

Taller de programación 2-2024

Fecha: 2/11/2024

Autor: Vicente Aníbal Aninat Norambuena

TALLER N°1

El problema de los bidones de agua con el algoritmo A* y heurística

Explicación breve del algoritmo

El programa que resuelve los bidones de agua primero despliega un menú por terminal por el que se escribe el nombre del archivo .txt que contiene la información de los bidones a operar, luego dicho archivo se lee y se extraen los datos para después entrar a la fase de cálculo de la solución. Primero se calcula la distancia entre 0 y los números de las cantidades de agua a la que se busca llegar como si de un plano n dimensional se tratase. Luego se calculan todas las operaciones posibles de llenar, vaciar y trasvasar bidones en una iteración de un conjunto de bidones, para luego hacer lo mismo con el resto de los estados derivados hasta llegar a la solución, es decir, una distancia 0 entre el punto del estado y el punto solución.

Heurísticas o técnicas utilizadas

A continuación se detallan cada una de las heurísticas o técnicas utilizadas en favor de la eficiencia del programa:

- Comprobación temprana del estado: Cada estado primero se verifica que sea la solución antes de realizar las operaciones de llenado, vaciado y trasvasado sobre este.
- Comprobación de realización de estados: Para evitar operaciones redundantes en los estados, se realizan verificaciones preliminares. Por ejemplo, no se puede llenar un bidón lleno, no se puede vaciar un bidón vacío y no se pueden trasvasar bidones cuando: El primero está vació, el segundo está lleno, ambos están vacíos o ambos están llenos.
- Prioridad de distancia: Luego de realizar las operaciones, los nuevos estados se ordenan en base a su distancia a la solución. Los estados más cercanos se encolan primero para ser operados y el resto queda en una cola de rezagados para ver trabajados si es que la cola principal se agota antes de llegar a la solución, esto en cada iteración.
- Salto de fé: Se elige un estado al azar de entre los disponibles y se realizan sus operaciones, si las distancias de estas operaciones son menores a las distancias de los demás estados, se sigue operando solamente por los nuevos estados y todos los demás quedan rezagados, si las distancias no son menores el salto de fé falla y se realizan las operaciones normalmente.
- Tabla Hash: Se utiliza una tabla Hash para reducir los tiempos de búsqueda e inserción de los estados ya visitados.

Si las cantidades de agua de cada bidón construyen un espacio de n dimensiones, estamos buscando el camino más corto entre el origen y un punto en el espacio. Si tenemos un vector que se desplaza por el espacio representando el proceso de cálculo computacional, la filtración por menor distancia genera un cilindro alrededor del vector que lo dirige en su desplazamiento por el espacio, el cálculo de esta distancia está basado en fórmula de

distancia euclidiana descrita por Arce, Castillo y González. (*Arce S. et al. (2003)*); y la frecuencia descendiente del "salto de fé" causa que mientras más cerca se esté de la solución menos conveniente es arriesgarse y elegir un estado al azar. De esta forma el algoritmo de búsqueda se puede interpretar como un vector que se desplaza por el espacio en forma de taladro entre un punto y otro. **[Anexo 1]**

Esta heurística posee similitudes con otras como el Enfriamiento simulado por la forma en la que al acercarse más a la solución, menor es la frecuencia de decisiones aleatorias.

Funcionamiento del programa

Para mayor entendimiento se separará el código en dos secciones: Menú y lectura; y Solución.

Menú y lectura

- Inicia la ejecución del programa.
- Se solicita por consola el nombre del archivo con la información de los bidones a operar.
- Si el archivo no es encontrado por su nombre:
    - Se muestra un mensaje de error y termina la ejecución
- Se inicia la lectura del archivo.
- Si el archivo no tiene el formato solicitado:
    - Se muestra un mensaje de error y termina la ejecución.
- Se lee la primera línea del archivo para contar la cantidad de bidones del problema a resolver.
- Se crea un arreglo de bidones de largo igual a la cantidad de bidones.
- Se leen las dos primeras líneas del archivo, líneas referentes a la capacidad máxima de los bidones y la cantidad de agua a la que se quiere llegar.
- Se crean bidones por cada par de datos leídos y se guardan en un arreglo.
- Se calcula que las capacidades máximas de los bidones en el arreglo sean coprimas con el mínimo común múltiplo.
- Se obtiene el estado inicial del problema.

Para mayor entendimiento, consultar el diagrama de flujo ***[Anexo 2]***.

Solución

- Se inicializa el largo máximo de las listas de todos los estados, los no visitados y una lista auxiliar de no visitados.
- Se añade el estado inicial en la primera posición de cada lista.
- Se calcula la distancia entre el origen y el punto final.
- Se define la frecuencia con la que se tomarán "saltos de fe".
- Se inicializa un contador de ciclos.
- Mientras no se haya encontrado una solución:

- Si el contador coincide con la frecuencia del “salto de fé”:
    - Se elige un estado al azar de entre los no visitados
    - Se calculan sus operaciones
    - Si las distancias entre los estados nuevos y la solución son menores a la distancia entre el estado original y la solución:
        - Se actualiza el listado de no visitados con los estados nuevos.
- Sino:
    - Para cada Bidón dentro del estado:
        - Si todas las cantidades de agua actuales coinciden con las buscadas:
            - Se imprime la solución y se termina la ejecución.
    - Para cada Bidón dentro del estado:
        - Si no está lleno:
            - Se llena.
            - Si el estado no existe ya:
                - Se añade a las listas de no visitados auxiliar y de todos los estados.
                - Se calcula la distancia entre el estado y la solución
        - Si no está vació:
            - Se vacía.
            - Si el estado no existe ya
                - Se añade a las listas de no visitados auxiliar y de todos los estados.
                - Se calcula la distancia entre el estado y la solución
        - Para cada bidón:
            - Si dos bidones son diferentes, no están llenos, no están vacíos, el primero no está vació y el segundo no está lleno:
                - El primer bidón se trasvasa al segundo.
                - Si el estado no existe ya:
                    - Se añade a las listas de no visitados auxiliar y de todos los estados.
                    - Se calcula la distancia entre el estado y la solución
        - Si las capacidades de las listas de no visitados auxiliar o de todos los estados están a un ¾ de su máximo:
            - Se crean nuevas listas de largo 10 veces superior y se copian los datos respectivos.

- Se actualiza la lista de no visitados con los valores de la mitad más cercana a la solución en la lista auxiliar.

Para mayor entendimiento consultar el diagrama de flujo ***[Anexo 3]***.

Aspectos de implementación y eficiencia

A continuación se detallan las implementaciones de las clases dentro del programa y su relación con la eficiencia de este:

- Clase Bidon: Clase que permite instanciar objetos bidones, cada uno con sus atributos de agua actual, agua máxima y agua objetivo. La razón del uso de esta clase es poder representar el problema y el funcionamiento del programa de manera sencilla de entender.
- Clase GrupoBidones: También puede conocerse como estado, puesto que esta clase contiene todos los bidones que componen un estado dentro de la resolución del problema. Sus atributos son un arreglo de punteros a Bidones, la operación de llenado, vaciado o trasvasado de la que fue creado, la cantidad de bidones que contiene su arreglo y la distancia entre el estado y el resultado esperado. Su eficiencia radica en la contención punteros a Bidones para evitar casos de sobreescritura que perjudiquen el proceso de búsqueda de la solución y el atributo distancia, este se calcula en el momento de la creación de la instancia de la clase, lo que permite consultas mucho más sencillas y rápidas que calcular la distancia cada vez que se vaya a utilizar.
- Clase Operacion: Clase interfaz de la que heredan las siguientes clases, esto para una mejor relación entre clases, puesto que todas las siguientes comparten el hecho de ser operaciones:
    - Clase OperacionLlenar: Crea un nuevo estado con el Bidon seleccionado lleno y un puntero a su padre, el estado ingresado para el llenado, esta metodología al igual que las siguientes, evita casos de sobreescritura de estados o instancias de GrupoBidones.
    - Clase OperacionVaciar: Crea un nuevo estado con el Bidon seleccionado vacío y un puntero a su padre.
    - Clase OperacionTrasvasijar: Crea un nuevo estado con los Bidones seleccionados pasados por el trasvase de agua y un puntero a su padre.
- Clase Leer: Clase encargada de la lectura de archivos, recibe el nombre del archivo y retorna el que será el estado inicial del problema. El proceso de lectura verifica estrictamente que la información del archivo esté en la forma y formato correctos para reducir al mínimo los casos de errores fatales dentro del programa.
- Clase Menu: Clase que instancia un menú por la consola de comandos, a través de esta se da la bienvenida al usuario, se llama a la lectura del archivo y se llama a cálculo de la solución, esto con el fin de separar las etapas del código para un mayor entendimiento.

- Clase Queue: Clase que define instancias de tipos de datos Cola para su posterior uso en el cálculo de la solución, tiene una implementación genérica pero su utilidad radica en la simplificación del proceso de operación de los diferentes estados sin tener que recurrir a un arreglo estático.
- Clase JugProb: Clase que calcula la solución del problema de los bidones de agua pertinente. Su funcionamiento se detalla en el apartado *Funcionamiento del programa* y sus aspectos de eficiencia en *Heurísticas o técnicas utilizadas*.
- Clase Main: Clase principal del programa, a partir de esta se llama a los demás procesos y permite una ejecución sencilla por línea de comandos. Detallada en el apartado *Ejecución del código*.
- Makefile: Archivo vital para el compilado rápido, sencillo y eficiente de todo el programa, es un añadido imprescindible tanto para el desarrollador como para usuarios externos sin experiencia en el código y solamente se interesan en su uso.

Ejecución del código

El código debe ejecutarse en Sistema operativo Linux. Por medio de la terminal del Sistema operativo abierta en la carpeta de código fuente se debe escribir el comando ‘make’ para compilar el programa, luego, se pueden probar ejemplos de cada una de las clases del programa con los comandos ‘./test_Clase’. Para usar el programa escribir el comando ‘./Main’, en este se solicitará por consola escribir solamente el nombre (sin extensión) del archivo que contiene la información de los bidones a resolver, debe ser un archivo txt. Los archivos ‘ejemplo_n’ contienen ya configuraciones de bidones de ejemplo. Finalmente para borrar los archivos de objeto creados durante la compilación, escribir ‘make clean’.

Bibliografía

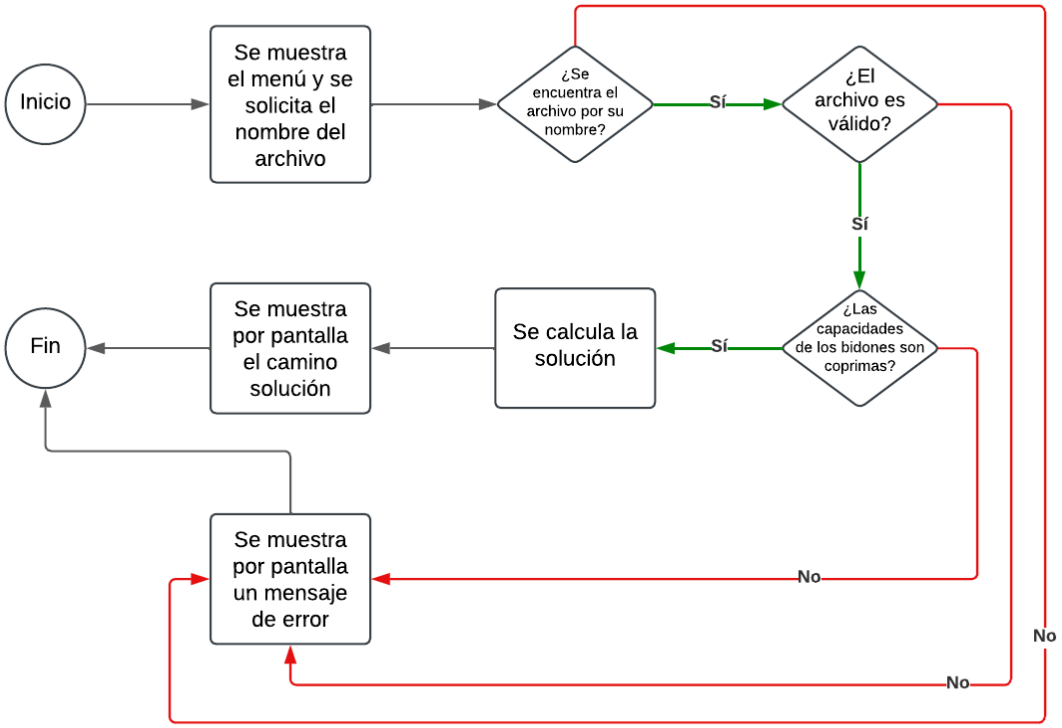
Arce S., C., Castillo E., W., & González V., J. (2003). *Álgebra lineal* (3.ª ed.). Pag(175).

https://nickpgill.github.io/files/2014/07/libro-algebra-lineal.pdf

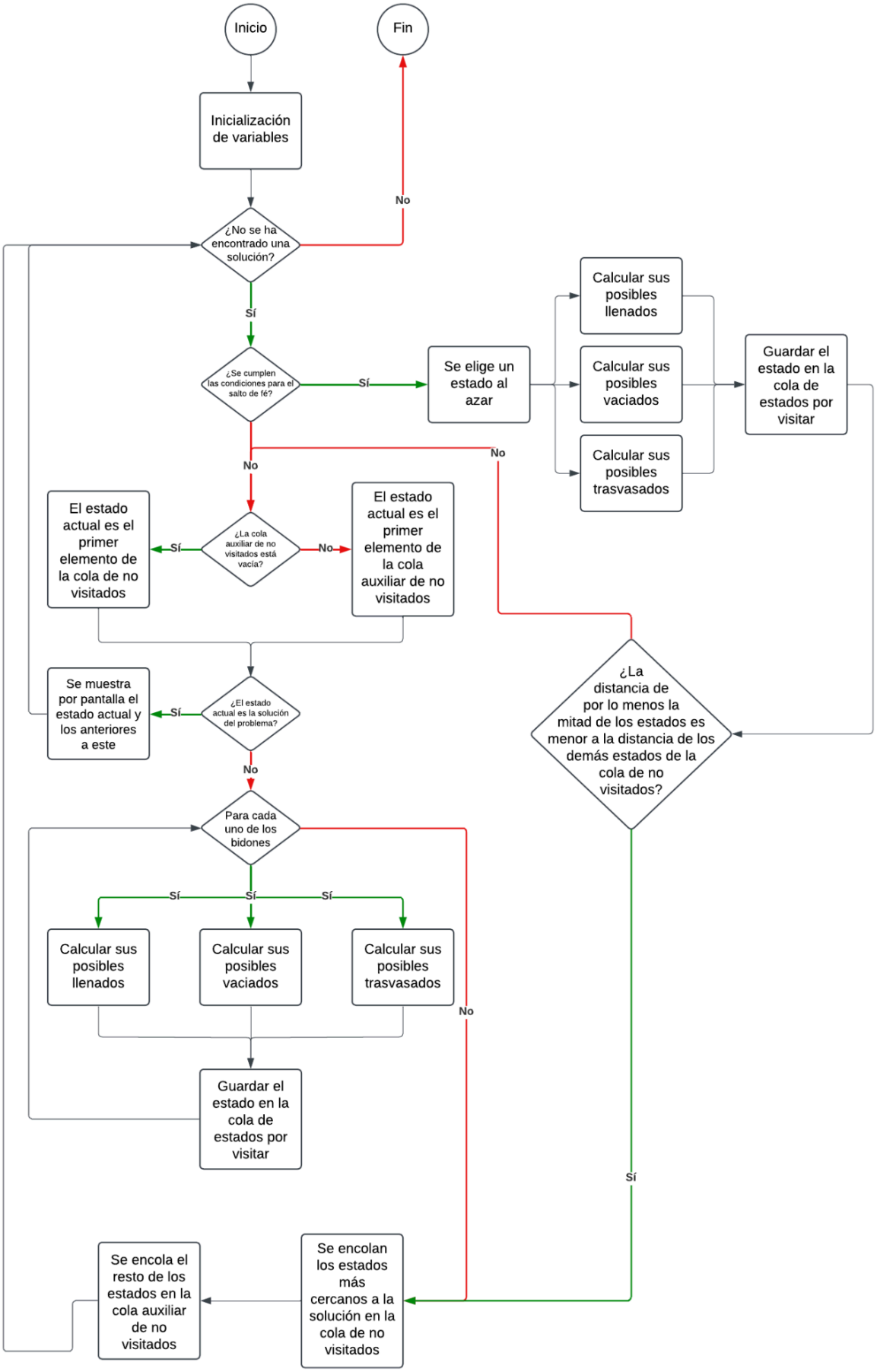
Anexos

Anexo 1: Interpretación gráfica del algoritmo.

Anexo 2: Diagrama del menú y la lectura del programa



Anexo 3: Diagrama de flujo del cálculo de la solución


Inicio
Fin
Inicialización de variables
¿No se ha encontrado una solución?
No
Sí
¿Se cumplen las condiciones para el salto de fé?
Sí
Se elige un estado al azar
Calcular sus posibles llenados
Calcular sus posibles vaciados
Calcular sus posibles trasvasados
Guardar el estado en la cola de estados por visitar
No
¿La cola auxiliar de no visitados está vacía?
Sí
No
El estado actual es el primer elemento de la cola de no visitados
El estado actual es el primer elemento de la cola auxiliar de no visitados
¿La distancia de por lo menos la mitad de los estados es menor a la distancia de los demás estados de la cola de no visitados?
¿El estado actual es la solución del problema?
Sí
Se muestra por pantalla el estado actual y los anteriores a este
No
Para cada uno de los bidones
Sí
Sí
Sí
Calcular sus posibles llenados
Calcular sus posibles vaciados
Calcular sus posibles trasvasados
No
Guardar el estado en la cola de estados por visitar
Sí
Se encolan los estados más cercanos a la solución en la cola de no visitados
Se encola el resto de los estados en la cola auxiliar de no visitados