Programación Aeronave

Informe Final

Santiago Neusa Ruiz

Área de Ciencias Fundamentales

Escuela de Ciencias Aplicadas e Ingeniería

28/05/2023

1. Algoritmo

1.1. Macroalgoritmo

La función main recibe las primeras líneas de información sobre el caso a evaluar y repite esta acción mientras la primera línea no sea igual a 0.

Después de recibir la información inicial, se llama a la función crear aeropuerto, que procesa el contenido de una matriz que representa el aeropuerto. Esta función almacena las líneas de entrada en una matriz y luego construye un diccionario que contiene los datos de cada parqueadero, su disponibilidad, avión que lo ocupa y conjunto de estacionamientos aledaños. El diccionario resultante se devuelve al final de la función.

De vuelta en main, el diccionario obtenido se guarda en una variable llamada aeropuerto y se procede a tomar los eventos (aterrizajes y despegues), que se almacenan en una cola.

Tanto el aeropuerto como los eventos se pasan como parámetros a una nueva función llamada verificar. Esta función decide si, con los datos recopilados hasta el momento, es posible resolver el problema. Verifica si el aeropuerto está completo (es decir, tiene lugares para aterrizar) o si le faltan parqueaderos para satisfacer la demanda de aviones. La función devuelve el número máximo de aviones simultáneos posibles, a menos que se produzca un caso de fallo, en cuyo caso retorna 0.

Después de ejecutar verificar, el número máximo de aviones simultáneos posibles se almacena en una variable. Si en algún punto del proceso se determina que no habrá suficiente espacio en el aeropuerto, se decide que el camino actual no es correcto y se detiene el back tracking. Si el resultado de verificar es 0, se imprime un mensaje indicando que el caso no es posible.

Si el resultado de verificar es diferente de 0, se crea una variable llamada historial que almacena en una lista de listas los eventos, intentos y parqueaderos relacionados. Esta lista de listas, junto con un puntero, funciona como una pila.

Finalmente, se llama a la función back tracking, a la cual se le pasan como parámetros el aeropuerto, el historial y el número máximo de aviones simultáneos posibles. La función back tracking se encarga de resolver el problema mediante un algoritmo de back tracking.

En back tracking cobra mucha importancia la variable historial. Como fue descrito, historial contiene por cada evento una lista. La primera posición de cada lista es el evento, la segunda el intento (cuántas veces se ha intentado realizar este evento, por ejemplo, la cantidad de veces que se le ha tratado de asignar un parqueadero) y el parqueadero que ocupa dicho evento.

Con un puntero se avanza o se devuelve posiciones en historial (esto sería el back tracking). Si es un aterrizaje se llama a la función bloquear, que bloquea el parqueadero solicitado y se buscan los 'alcanzables', parqueaderos a los que se puede llegar sin pasar por el parqueadero que se va a tapar. Si todos los aledaños son alcanzables, se para la ejecución de la función. Si no hay alcanzables, se bloquean todos los parqueaderos del aeropuerto y se para la ejecución. Al no cumplirse ninguna de las dos, se buscan aquellos que se pueden bloquear y se bloquean. Lo primero que se debe hacer es incluir los pseudocódigos, explicando cómo se afrontan cada uno de estos pasos.

Si el evento es negativo se llama a desbloquear, en donde se crea una lista de desbloqueables. Se sacan los que están ya disponibles (en caso de estar todos sus aledaños disponibles, se libera y retorna True). Si todos a su alrededor están ocupados no se puede liberar nada y retorna False. Como hay algunos ocupados y otros no, se puede liberar el espacio, pero se buscan los aledaños que se pueden liberar también y se retorna True.

Con ayuda de una variable de control llamada 'encontrado' se verifica si se pudo realizar el evento (se convierte en True) o no se pudo realizar (se queda en False). En caso de que no se pueda realizar el evento, se llama a back tracking deshaciendo el evento (si se bloqueó se desbloquea, y si se desbloqueó se bloquea) y se mueve el puntero a una celda anterior de la pila historial.

1.2. Pseudo-código

Main	- Recibe la primera línea de input - Crea el aeropuerto - Crea la cola de eventos con la última línea de input - Crea máximos aviones llamando a verificar - Si verificar encontró un error devuelve 0, y se imprime 'Case: No', de lo contrario se crea una variable output y se llama a back tracking. - Si el tamaño de output es 0, es porque no se encontró solución, así que se imprime 'Case: No'. De lo contrario se imprimen los parqueaderos.
Verificar	- Verifica si hay '==' y haya espacios suficientes para la cantidad de aviones. - Mientras va revisando cuenta la cantidad máximo de aviones que puede haber a la vez y retorna ese resultado si no hubo alguna excepción
Crear Aeropuerto	- Recibe las líneas de input que constituyen la matriz y las guarda en una variable - Se para en los '==' y empieza a hacer un BFS, explorando cada parqueadero aledaño, para crear una lista de adyacencia que va a ser el aeropuerto. Se desprecian los espacios en blanco y bloqueos.
Crear Historial	- Convierte la cola de eventos en una pila que en cada celda tiene listas con: [evento, intento:0, parqueadero:0]
Bloquear	- Se bloquea el parqueadero solicitado y se buscan los 'alcanzables', parqueaderos a los que se puede llegar sin pasar por el parqueadero que se va a tapar. - Si todos los aledaños son alcanzables, se para la ejecución de la función. Si no hay alcanzables, se bloquean todos los parqueaderos del aeropuerto y se para la ejecución - Al no cumplirse ninguna de las dos, se buscan aquellos que se pueden bloquear y se bloquean.

Desbloquear	- Se crea una lista de desbloqueables. Se sacan los que están ya disponibles (en caso de estar todos sus aledaños disponibles, se libera y retorna True). - Si todos a su alrededor están ocupados no se puede liberar nada y retorna False. - Como hay algunos ocupados y otros no, se puede liberar el espacio, pero se buscan los aledaños que se pueden liberar también y se retorna True.
Back Tracking	- Con un puntero va a ir hacia adelante o atrás en el historial (así va a manejar los eventos) mientras el puntero esté en los límites y el intento del primer evento no sea el último intento. - Cuando el evento sea positivo es un aterrizaje, así que le busca un parqueadero y llama a la función bloqueo. Luego se hace la verificación de que haya espacios suficientes para estacionar los aviones que vienen. Si no hay espacios suficientes de marca como 'encontrado = False', de lo contrario 'encontrado = True' - Cuando es negativo busca el parqueadero que está ocupando el avión y llama a la función desbloquear. Si el avión no se puede sacar se marca 'encontrado = False', de lo contrario 'encontrado = True' - Para ver si un evento se realizó se utiliza la variable booleana 'encontrado'. Se llama a back tracking si encontrado es Falso, deshaciendo el evento (si se bloqueó se desbloquea, y si se desbloqueó se bloquea) y se mueve el puntero a una celda anterior de la pila historial. Si encontrado es Verdadero, se aumenta el puntero para avanzar al siguiente evento.

2. Representación de la información

Aviones	Se representa como una variable entera con el mismo nombre y se utiliza para hacer la comprobación en la función 'Verificar' de que la cantidad de aviones dada en el input concuerde con lo procesado en eventos.

Filas	Dos valores enteros que se utilizan como referencia al hacer los ciclos que van a guardar la información de la matriz del parqueadero.
Columnas	
Matriz	Inicialmente se guarda la información del input en una matriz, pero luego se transforma en 'Aeropuerto'. La única información que queda es la de los puntos de aterrizaje y los números de parqueaderos. Los bloqueos y espacios en blanco se desprecian.
Eventos	Se guarda inicialmente como una pila, y se envía a verificar para comprobar que no haya más despegues que aterrizajes y otras comprobaciones. Luego dicha información se transforma en la 'pila_historial'

3. Proceso de asignación de los parqueaderos

- **¿Cuál es el criterio para seleccionar cuál es el siguiente avión al que se le va a asignar parqueadero?**

Los aviones se procesan tal y como aparecen en la última línea de input para dicho caso.

- **¿Cuál es el criterio para seleccionar cuál es el parqueadero que se va a asignar?**

El ordenamiento de los parqueaderos es pseudoaleatorio, ya que dependerá de cómo estén organizados los parqueaderos en el diccionario 'Aeropuerto'. Los parqueaderos asignan por orden de aparición en el diccionario.

Al seleccionar un parqueadero se itera sobre los elementos del diccionario y para estacionar allí se tiene en cuenta que: no sea '==', el lugar esté disponible y que no se haya seleccionado anteriormente (esto con ayuda de los intentos, valores que aparecen en la pila de historial).

- **¿Cómo se verifica que el parqueadero sí se pueda utilizar?**

La estructura del aeropuerto es un diccionario que tiene en las Keys los parqueaderos (1, 2, 3, 4…) y en los values una lista con los siguientes valores:

Parqueadero → [ocupación, avión que le ocupa, conjunto de parqueaderos adyacentes]

Como se mencionó, se mira si el lugar está disponible (o sea, en Parqueadero [0]).

- **¿Qué se hace en caso de que no se encuentre un parqueadero que se pueda asignar al avión?**

El programa tiene una variable que se llama 'max_airplanes', explicada anteriormente. Dicha variable sirve como control para conocer una estimación de los aviones que faltan todavía por entrar. De esta manera, cuando se ingresa un avión, se ocupa una plaza y se bloquean algunos lugares que tapa ese avión en ese parqueadero, se verifica si la cantidad de parqueaderos restantes es suficiente para satisfacer la cantidad máxima de aviones que faltan.

Si la cantidad de parqueaderos disponibles es mayor o igual a la cantidad de aviones máximos (eso evita que los aviones al aterrizar no tengan dificultad), se sigue ejecutando el programa, si no, se hace back tracking moviendo el puntero de la lista de eventos al evento anterior.

4. Validación de los despegues

En el informe se debe incluir una descripción de cómo se valida que los despegues sí se puedan realizar. Esta descripción debe incluir por lo menos:

- **¿Cómo determinan si un avión puede despegar o no, cómo construyen la ruta?**

EL programa no construye una ruta directamente, sino que hace una serie de comprobaciones para determinar si se puede liberar el espacio o no. De esta manera, la función retorna un valor booleano.

1. Si todas las plazas aledañas al parqueadero que se quiere liberar están desocupadas, se libera sin problema (se retorna True).
2. Si todas las plazas aledañas al parqueadero que se quiere liberar están desocupadas y el parqueadero no es aledaño a '==', se retorna False.
3. Como ya se comprobó que por lo menos 1 plaza está disponible (o sea que puede salir) y hay otras ocupadas, significa que el parqueadero que se quiere liberar está obstaculizando el paso hacia otros parqueaderos, así que con un BFS se buscan los parqueaderos que están ocupados y no tienen un avión dentro, y se liberan.

- **¿Qué se hace en caso de que un avión no pueda despegar?**

Se mueve el puntero al evento anterior, se hace back tracking.

5. Componentes técnicos

El último componente del informe se refiere a os detalles técnicos de la implementación:

- El lenguaje de programación utilizado y por qué fue seleccionado

Python por su simple sintaxis y porque era el lenguaje que se venía trabajando desde hacía algunos cursos.

- La definición de cada función. Siguiendo la propuesta en CORBA deben incluirse los siguientes elementos:

Nota: ninguna función produce excepciones.

Main	No tiene parámetros y no retorna	Lleva control del programa, recibiendo los datos de input y enviándolos a una función u otra para buscar la solución de un parqueadero con una serie de eventos.
Verificar	Recibe aviones:int, aeropuerto:dict, eventos:cola Retorna máximos aviones:int	Hace verificaciones debido a que hay algunos casos que están incompletos y, por ende, deben evaluarse como falsos. Además, provee la variable de máximos aviones que es de mucha utilidad.
Crear Aeropuerto	Recibe filas:int, columnas:int Retorna aeropuerto:dict	Recibe los datos del parqueadero y los convierte en el diccionario 'Aeropuerto' con el que se va a trabajar durante el programa. Esta función emplea un BFS para crear el conjunto de aledaños de cada parqueadero y hacer la exploración de la matriz.

Crear Historial	Recibe eventos:cola Retorna historial:pila	Es la estructura con la que se van a procesar los eventos. Su estructura es que en cada celda de la pila se guarda una lista con la siguiente información: [evento, intento: 0, parqueadero que ocupa: 0]
Bloquear	Recibe aeropuerto:dict, parqueadero:int, avión:int No retorna nada	Se bloquea el parqueadero solicitado y se buscan los 'alcanzables', parqueaderos a los que se puede llegar sin pasar por el parqueadero que se va a tapar. Si todos los aledaños son alcanzables, se para la ejecución de la función. Si no hay alcanzables, se bloquean todos los parqueaderos del aeropuerto y se para la ejecución Al no cumplirse ninguna de las dos, se buscan aquellos que se pueden bloquear y se bloquean.
Desbloquear	Recibe aeropuerto:dict, parqueadero:int Retorna valor booleano	Se crea una lista de desbloqueables. Se sacan los que están ya disponibles (en caso de estar todos sus aledaños disponibles, se libera y retorna True). Si todos a su alrededor están ocupados no se puede liberar nada y retorna False. Como hay algunos ocupados y otros no, se puede liberar el espacio, pero se buscan los aledaños que se pueden liberar también y se retorna True.
Back Tracking	Recibe aeropuerto:dict, historial:pila, máximos aviones:int Retorna output:dict	Con un puntero va a ir hacia adelante o atrás en el historial (así va a manejar los eventos). Cuando el evento sea positivo es un aterrizaje, así que le busca un parqueadero y llama a la función bloqueo

		Cuando es negativo busca el parqueadero que está ocupando el avión y llama a la función desbloquear. En caso de que no se pueda realizar el evento, se llama a back tracking deshaciendo el evento (si se bloqueó se desbloquea, y si se desbloqueó se bloquea) y se mueve el puntero a una celda anterior de la pila historial.

- Descripción de las variables mediante una tabla que contenga la siguiente información:

Estas son las variables principales y en las que el programa basa su funcionalidad:

Aeropuerto	Diccionario que en las keys tiene parqueaderos y en los values listas con: [ocupación, avión, aledaños] Variable local	Se utiliza para guardar los parqueaderos donde se va a estacionar y despegar los aviones. Es la estructura que viene como remplazo a la matriz que dan en el input. Se crea en main, se manipula en back tracking y se destruye en main.
Historial	Pila que en cada celda tiene listas con: [evento, intento, parqueadero] Variable local	Es la estructura de control que ayuda a saber cuándo hacer cada evento y lleva le cantidad de intentos por evento. Además de ser quien colabora con el back tracking. Al emplear un puntero puede ir adelante o atrás en la pila (hacer back tracking) Se crea en main, se manipula en back tracking y se destruye en main.
Máximos aviones	Entero positivo que ayuda a llevar un conteo de los aviones que entran y salen Variable local	Es una variable que disminuye con cada aterrizaje y aumenta con cada despegue.

		Ayuda a saber cuántos aviones faltan por estacionar, y por ende, una idea de cuántos parqueaderos hay que tener disponibles. Se crea en main, se manipula en back tracking y se destruye en main.