UNIVERSIDAD EAFIT

ESTRUCTURA DE DATOS Y ALGORITMOS 2

INFORME PROGRAMACIÓN DE AERONAVE

KRISTIAN RESTREPO OSORIO

ID: 1000257630

25/05/2023

El problema 11208 del juez en línea de la Universidad de Valladolid (Valladolid, s.f.) llamado Airlplane Scheduling fue en sí mismo un reto importante para cada uno de los estudiantes de la asignatura, ya que, debido a su alta dificultad, se reforzaron conocimientos necesarios para abordar cualquier problema en el ámbito de la programación, y a su vez, permitió dar desarrollo a los temas del curso planteados por el docente Juan G. Lalinde-Pulido (EAFIT, s.f.).

Se puede representar la solución al problema en el siguiente pseudocódigo, resumido en 3 funciones:

1. Función que almacena los distintos tipos de datos necesarios para abordar el problema e imprime la solución:

Mientras que el usuario desee conocer una solución factible a un aeropuerto dado:

Se asigna el número del caso actual.

Se le pide al usuario que ingrese el numero de aviones, las filas y columnas del aeropuerto.

Se le pide al usuario que ingrese el aeropuerto (los símbolos correspondientes a cada coordenada de la matriz).

Se le pide al usurario ingresar los eventos relacionado con las salidas/entradas de los aviones.

Se obtiene la matriz de pesos y los parqueaderos alcanzables desde los puntos de inicio/salida a partir de los datos ingresados. (2)

Se verifica si existe una solución factible al aeropuerto y la lista de eventos ingresada, y en caso de existir una solución, la imprime. (3)

2. Función que le asigna un peso a cada casilla del aeropuerto, y guarda los parqueaderos accesibles desde los puntos de entrada/salida:

Se crea la matriz de pesos correspondiente a cada una de las casillas de la matriz original.

Se guardan las coordenadas de los puntos entrada/salidas desde los cuales se va a empezar a recorrer y así ponderar la matriz.

Mientras existan casillas a las cuales hay que mirar sus vecinos para ponderar:

Se observa los adyacentes de la coordenada actual que se está evaluando

Si el adyacente se encuentra dentro de los limites y aún no se ha ponderado:

Si ese adyacente es un bloqueo, se marca la casilla como -11 y ponderada.

Si ese adyacente es una casilla blanca (camino), este tendrá el mismo peso de la casilla actual, se agregan las coordenadas de la casilla adyacente a las casillas por evaluar a sus vecinos (se agrega al principio de la estructura en donde vamos a almacenar estos datos), y finalmente, se marca como ponderada.

Si ese adyacente es un parqueadero, ese tendrá el peso de la casilla actual + 1, se agregan las coordenadas de la casilla adyacente a las casillas por evaluar a sus vecinos (se agrega al final de la estructura en donde se guardarán los datos), y la casilla se marca como ponderada. Además, se guarda la coordenada del parqueadero en la estructura en donde se almacenarán todos los parqueaderos accesibles.

Finalmente, se devuelve la matriz ponderada y los parqueaderos accesibles desde los puntos entradas/salidas.

3. Función que utiliza un algoritmo de backtracking para determinar si existe una solución factible al aeropuerto y la lista de eventos ingresada:

Si la lista de eventos ingresada se recorrió completamente:

Se devuelve que existe una solución.

Si hay más entradas de aviones consecutivas que parqueaderos:

Se devuelve que no existe una solución.

Si el evento es un aterrizaje:

Se recorren los parqueaderos disponibles, y para cada uno de ellos:

Se añade el parqueadero actual a la solución.

Se guarda la coordenada y el numero del parqueadero que tiene ese lugar donde se estableció el avión.

La casilla se marca como ocupada y se vuelve a ponderar la matriz, ya que algunos caminos se ven afectados.

Se hace el llamado recursivo para evaluar el siguiente evento, si el llamado recursivo devuelve verdadero, significa que cada avión pudo despegar y aterrizar satisfactoriamente, por lo tanto, se devuelve verdadero al anterior llamado con lo que se concluye que esa posición fue la adecuada para establecer el avión. De lo contrario, se deshace todo lo hecho anteriormente para el aterrizaje, y se prueba la posibilidad de una solución factible parqueando el evento actual en el siguiente parqueadero disponible

Si se recorrió toda la lista de parqueaderos y el avión no se ubicó satisfactoriamente, se debe volver a evaluar los anteriores posicionamientos de los llamados recursivos para que me generen una solución factible para ubicar el avión actual. Por lo tanto, se devuelve falso, y se reevalúan los eventos.

Si es un despegue:

Se ubica la casilla en la que el avión esta parqueado y se mira cada uno de sus vecinos:

Si el vecino está ponderado, significa que existe un camino desde un punto de salida hacia el mismo, por lo tanto, la casilla actual también tiene salida por este mismo camino, de esto:

Se libera nuevamente el parqueadero del evento actual.

Se vuelve a ponderar la matriz para los caminos que se veían afectados por el avión que despegó y se guardan los parqueaderos disponibles gracias a que este evento se llevase a cabo.

Se hace el llamado recursivo para evaluar el siguiente evento con los nuevos parqueaderos disponibles.

Si el llamado recursivo es verdadero, significa que todo evento posterior al actual se llevó a cabo satisfactoriamente, por lo tanto, se devuelve verdadero a los anteriores llamados recursivos. De lo contrario, se para de evaluar si alguno de los vecinos puede salir, ya que se comprobó que la manera en que se establecieron los aviones no es una solución factible.

Si al recorrerse todos los vecinos de la casilla actual, ninguno de ellos tiene una conexión con la salida, o se encontraron eventos posteriores que no generan una solución factible, se deshace todos los pasos mencionados anteriormente en el despegue, y se devuelve falso para cambiar la solución planteada en anteriores llamados recursivos, reevaluando los eventos.

Es importante entender a la hora de la implementación del algoritmo la noción del tipo de dato que se está utilizando. Para comprender claramente como se ven estos reflejados, se construye las siguientes tablas donde se le da la especificación correspondiente a cada dato de entrada y salida:

Nombre del dato	Número de aviones
Descripción del dato	Dato que busca almacenar el número de aviones que ingresarán al aeropuerto, y serán evaluados en el algoritmo.
Representación del dato en la entrada o salida	El dato en la entrada se representa como un numero entero encargado de simbolizar la cantidad de aviones que entrarán al aeropuerto.
Representación del dato en el algoritmo	En el algoritmo, el dato es una variable de tipo INT donde se almacena dicha cantidad de aviones.
Conversión	El dato en la entrada es texto escrito por el usuario (por defecto como lo recibe el lenguaje), y en el algoritmo al recibirlo se

	convierte dicho valor en un entero con la función INT para almacenarlo finalmente.

Nombre del dato	Número de filas
Descripción del dato	Dato que busca almacenar el número de filas que tiene el aeropuerto, ya que el mismo se representa como una matriz, lo cual permite definir la cantidad de filas que tendrá esta cuando se cree en el algoritmo.
Representación del dato en la entrada o salida	El dato en la entrada se representa como un numero entero encargado de simbolizar la cantidad de filas que tendrá la matriz de símbolos/aeropuerto.
Representación del dato en el algoritmo	En el algoritmo, el dato es una variable de tipo INT donde se almacena dicha cantidad de filas.
Conversión	El dato en la entrada es texto escrito por el usuario (por defecto como lo recibe el lenguaje), y el algoritmo al recibirlo lo convierte en un valor entero finalmente por medio de la función INT, para almacenarlo.

Nombre del dato	Número de columnas
Descripción del dato	Dato que busca almacenar el número de columnas que tiene el aeropuerto, ya que el mismo se representa como una matriz, lo cual permite definir la cantidad de columnas que tendrá esta cuando se cree en el algoritmo.

Representación del dato en la entrada o salida	El dato en la entrada se representa como un numero entero encargado de simbolizar la cantidad de columnas que tendrá la matriz de símbolos/aeropuerto.
Representación del dato en el algoritmo	En el algoritmo, el dato es una variable de tipo INT donde se almacena dicha cantidad de columnas.
Conversión	El dato en la entrada es texto escrito por el usuario (por defecto como lo recibe el lenguaje), y el algoritmo al recibirlo lo convierte en un valor entero finalmente por medio de la función INT, para almacenarlo.

Nombre del dato	Matriz de símbolos o aeropuerto
Descripción del dato	Dato que busca almacenar el aeropuerto que va a contener los parqueaderos, los caminos, los bloqueos y los puntos de entrada/salida. Este será fundamental a la hora de asignar parqueaderos, ya que es el "tablero" en el que se mueve todo el algoritmo buscando soluciones factibles.
Representación del dato en la entrada o salida	El dato se ingresa como texto separado por espacios, donde por cada espacio hay un conjunto de símbolos, ya sea bloqueo ("##"), camino (".."") o parqueadero (el número del parqueadero). Por línea de texto habrá tantos conjuntos de dos símbolos separados por espacios como el numero de columnas que se ingresó, y se repetirá este proceso por la cantidad de

	filas que se ingreso en un principio, estando los símbolos en el orden en que el usuario desee “armar” el aeropuerto, generando así la representación de una matriz.
Representación del dato en el algoritmo	El dato se representa en el algoritmo como una lista de listas, o, en otras palabras, una matriz, donde cada fila es la lista que se genera al partir la línea de texto por los espacios, y cada columna es el conjunto de dos símbolos que había entre los espacios.
Conversión	Los datos como tal se siguen manejando de la misma manera, no se genera ninguna conversión en cuanto a los símbolos que se proponían inicialmente, simplemente el algoritmo se encarga de partir cada línea de texto ingresada por espacios, y, la lista que genera el proceso anterior se guarda en una posición de la lista externa, representando así una matriz con filas y columnas.

Nombre del dato	Eventos
Descripción del dato	Dato que busca almacenar la actividad con la que los aviones entran y salen del aeropuerto. Este es determinante a la hora de aplicar el algoritmo ya que permite saber que se hace con el evento. Un aterrizaje se representa con un numero positivo, y un despegue con un numero negativo.

Representación del dato en la entrada o salida	El dato en la entrada se representa como una línea de texto con eventos de aterrizaje, simbolizados de la forma +1, +2, +3…etc. Y eventos de despegue, simbolizados de la forma -1,-2,-3…etc.
Representación del dato en el algoritmo	En el algoritmo, el dato se ve representado por una lista de números enteros, tanto positivos como negativos, que simbolizan de igual forma tanto los aterrizajes como los despegues.
Conversión	Se parte por espacios la línea de texto, y a cada valor de la lista resultante por medio de la función INT se convierte a un entero, transformando así una lista de textos, a una lista de enteros.

Nombre del dato	Número del caso
Descripción del dato	Dato que almacena cual es el caso actual al que se le esta evaluando si existe una solución con el algoritmo planteado.
Representación del dato en la entrada o salida	La representación del dato en la salida se acompaña del resultado en si se llegó a una solución factible o no. A la hora de devolver el resultado, se devuelve como un texto enfatizando el número del caso y la respectiva respuesta del algoritmo.
Representación del dato en el algoritmo	En el algoritmo, el dato se representa como un contador que almacena un número entero, el cual se va sumando a medida que

	se va ingresando casos para hallar una solución.
Conversión	Como tal el dato no cambia, pero a la hora de devolver el mismo, se convierte el entero a texto para que este se pueda concatenar con el resto de información.

Nombre del dato	Solución parqueaderos
Descripción del dato	Dato que busca almacenar la ubicación en la que se debe establecer cada uno de los aviones para que se lleve a cabo satisfactoriamente la lista de eventos en el aeropuerto dado.
Representación del dato en la entrada o salida	La representación en la salida es una línea de texto que contiene el parqueadero en el que se debe establecer cada avión para llegar a dicha solución factible, separado por espacios. Cabe recalcar que los mismos se almacenan en orden, es decir, al avión uno, le corresponde el primer parqueadero, etc.
Representación del dato en el algoritmo	En el algoritmo, el dato se ve representado en una lista en la cual, en cada posición se almacena el parqueadero correspondiente al avión evaluado (el símbolo del parqueadero).
Conversión	El dato no cambia como tal su representación entre ambas partes, únicamente se imprime cada posición de la lista sin salto de línea, y con un espacio

	entre cada parqueadero que se imprime en la solución.

Una vez mencionado lo anterior, se podría considerar que una de las partes mas importantes del algoritmo se centra en la asignación de parqueaderos y en la validación de los despegues, por lo tanto, se entrará en detalle sobre como el algoritmo se encarga de hacer estos dos procesos.

En el caso de la asignación de los parqueaderos, el criterio para seleccionar cual es el siguiente avión al cual se le va a realizar este proceso, radica en que el evento a evaluar sea positivo (de aterrizaje), y para asignarle un parqueadero a el mismo, se debe almacenar los lugares de estacionamiento alcanzables desde los puntos de entrada/salida para así ser considerados lugares adecuados para establecer el avión, pero ¿Cómo se sabe que dicho parqueadero es alcanzable desde una entrada/salida?, aquí entra en juego el algoritmo que se encarga de ponderar cada una de las casillas, puesto que, al recorrer cada uno de los puntos alcanzables y asignarle un peso, se está definiendo que si la casilla se ponderó, existe un camino al punto de salida y es el más corto, por lo tanto, todo parqueadero que esté en la estructura donde se almacenan los mismos a la hora de hacer el proceso de ponderación, son parqueaderos alcanzables y disponibles para la ubicación de un evento. De esto, se justifica el porque se debe estar reevaluando la ponderación cada que el avión aterriza o despega, ya que la presencia de este puede generar un impacto en el camino a los demás aviones, y, por lo tanto, se debe verificar que parqueaderos están disponibles después de que alguno despega o aterriza. El algoritmo se encarga de posicionar el avión en el primer parqueadero que está en la estructura y hace el llamado recursivo para los siguientes eventos, y si no se llegó a una solución factible, el algoritmo deshace el lugar donde parqueó el avión (lo marca nuevamente como disponible), y prueba en el siguiente parqueadero alcanzable, y en el caso en que no hayan mas parqueaderos disponibles para establecer el avión, devuelve falso para una reubicación de los eventos anteriores a él, o en el caso de que no existan mas llamados recursivos, se concluye que no existe una solución para el aeropuerto y la lista de eventos dada.

Para la validación de los despegues, una vez identificado que el evento a realizar es negativo, lo que se hace es encontrar en que coordenada está el parqueadero en el que se ubicó el avión

actual. Cabe recalcar, que una vez se parquea el avión, siempre se guarda el lugar en donde se estableció el mismo, lo cual facilita el proceso a la hora de verificar su posición en el despegue. Una vez encontrada la posición, se observa cada uno de los adyacentes a la misma (arriba, derecha, abajo, izquierda), y por cada una de ellas se pregunta si está ponderada, esto, en otras palabras, quiere decir si alguno de sus vecinos tiene un camino hacía la salida, y en caso de que esto sea correcto, donde está ubicado el avión actualmente también tiene un camino hacía dicha salida, por lo tanto, se marcar como disponible el parqueadero del evento, se vuelve a ponderar la matriz y se almacena los nuevos parqueaderos disponibles, para hacer el llamado recursivo de la función y calcular los siguientes eventos. Si no se llegó a una solución factible en el llamado recursivo, se debe devolver falso a los anteriores llamados para que se reubiquen los aviones establecidos en el aeropuerto y se pueda reevaluar los eventos.

Finalmente, una vez explicado el funcionamiento del algoritmo para la solución del problema, se entra en detalle de algunos componentes técnicos sobre la implementación del mismo, empezando por el lenguaje de programación utilizado, el cual fue para este caso Python. La razón del porque se escogió este radica principalmente en la facilidad de uso que brinda el lenguaje al programador, destacando su sintaxis clara y legible, como también sus estructuras de control y expresiones concisas las cuales facilitan la implementación y comprensión de algoritmos que tienen una gran dificultad. También, una razón importante para escoger el mismo se centra en la amplia documentación existente para aquellos obstáculos que se presenten en el camino.

Por otro lado, es importante especificar la definición de cada función, que se puede ver a continuación con las siguientes tablas:

Nombre de la función	“casilla_es_valido”.
Parámetros que recibe	La función recibe cuatro parámetros: la cantidad de filas y columnas de la matriz, como también la coordenada en x y en y a la que se quiere acceder.
Tipo de dato que retorna	BOOL.
Excepciones que produce	No tiene.

Descripción corta de qué hace	Devuelve si la posición a la que se quiere acceder se encuentra dentro de los límites de la matriz.

Nombre de la función	“casilla_es_negra”.
Parámetros que recibe	Recibe tres parámetros: la matriz de símbolos o aeropuerto, y la coordenada en x y en y que se pretende evaluar.
Tipo de dato que retorna	BOOL.
Excepciones que produce	No tiene.
Descripción corta de qué hace	Verifica si la matriz de símbolos en la coordenada ingresada a la función es un bloqueo (“##”).

Nombre de la función	“casilla_es_blanca”.
Parámetros que recibe	Recibe tres parámetros: la matriz de símbolos o aeropuerto, y la coordenada en x y en y que se pretende evaluar.
Tipo de dato que retorna	BOOL.
Excepciones que produce	No tiene.
Descripción corta de qué hace	Verifica si la matriz de símbolos en la coordenada ingresada a la función es un camino (“..”).

Nombre de la función	“casilla_es_parqueadero”.
Parámetros que recibe	Recibe tres parámetros: la matriz de símbolos o aeropuerto, y la coordenada en x y en y que se pretende evaluar.
Tipo de dato que retorna	BOOL.

Excepciones que produce	No tiene.
Descripción corta de qué hace	Verifica si la matriz de símbolos en la coordenada ingresada a la función es un parqueadero (un digito).

Nombre de la función	“ponderar”.
Parámetros que recibe	Recibe 6 parámetros: la matriz de símbolos o aeropuerto, la cual contiene todos los símbolos correspondientes ingresados por el usuario; la matriz a ponderar, la cual contiene en cada casilla el peso que le corresponde a las posiciones de la matriz de símbolos; la cola doblemente terminada, en este caso llamada 'pesos', la cual contiene las coordenadas a evaluar; la lista de movimientos, la cual al recorrer la misma se permite evaluar a cada uno de los vecinos; el numero de filas de la matriz y el numero de columnas de la matriz.
Tipo de dato que retorna	TUPLE. (matriz ponderada, parqueaderos alcanzables)
Excepciones que produce	No tiene.
Descripción corta de qué hace	La función se encarga de darle un peso a cada una de las casillas partiendo desde los puntos de entrada/salida, según si es un parqueadero, un camino, o un bloqueo, lo cual permite deducir si existe ruta a una casilla, y también permite identificar que ese es el camino más corto. Además, a medida que recorre la matriz, almacena los

	parqueaderos alcanzables para recorrerlos en la función de backtracking.

Nombre de la función	"matriz_pesos".
Parámetros que recibe	Recibe 2 parámetros: el numero de filas y el número de columnas.
Tipo de dato que retorna	LIST.
Excepciones que produce	No tiene.
Descripción corta de qué hace	Crea una matriz idéntica a la matriz de símbolos original, pero cada elemento de la matriz es el peso asociado a cada casilla de la matriz original. Inicialmente como no se ha ponderado, todos los valores están en "None" por defecto.

Nombre de la función	"pre_ponderacion".
Parámetros que recibe	Recibe 3 parámetros: la matriz de símbolos o aeropuerto, la cantidad de filas y la cantidad de columnas de la matriz
Tipo de dato que retorna	TUPLE. (matriz ponderada, parqueaderos alcanzables).
Excepciones que produce	No tiene.
Descripción corta de qué hace	Se encarga de encontrar los puntos de entrada/salida de la matriz, almacenar la coordenada de estos en la cola doblemente terminada, y darle un valor en esa coordenada a la matriz de ponderación de 0. También se define la lista de movimientos y se llama a la función ponderar. Básicamente se encarga de dejar

	las cosas listas para que el algoritmo de ponderación se pueda llevar a cabo.

Nombre de la función	"resolver_problema".
Parámetros que recibe	Recibe 8 parámetros: la matriz de símbolos, la cual se va a modificar a la hora del aterrizaje o despegue de un avión; la matriz ponderada, la cual define si existe un camino desde el punto entrada/salida a cualquier casilla accesible; la lista de eventos, la cual se recorre para determinar si realizar un aterrizaje o un despegue; el numero de filas de la matriz; el numero de columnas de la matriz; la lista de solución, que es donde se almacenará el parqueadero en donde se ubico cada uno de los aviones; el diccionario de posicionamientos, el cual almacena en que coordenada de la matriz se establecieron los aviones para poder ubicarlos en el despegue de los mismos y la lista de parqueaderos, la cual almacena los estacionamientos disponibles en los que algún evento de aterrizaje se puede establecer en ese momento.
Tipo de dato que retorna	BOOL.
Excepciones que produce	No tiene.
Descripción corta de qué hace	Función de backtracking, que se encarga de probar todas las posibilidades existentes para realizar la lista de eventos en un aeropuerto dado.

Nombre de la función	“principal”.
Parámetros que recibe	Ninguno
Tipo de dato que retorna	NONE.
Excepciones que produce	No tiene.
Descripción corta de qué hace	Función que se encarga de pedir los datos al usuario de cada uno de los casos que el mismo desee evaluar, en base a un aeropuerto y su lista de eventos. Esta función almacena todos los datos respectivos, realizará la primera ponderación, creará la lista donde se almacenará la solución de los parqueaderos en caso de existir, llamará a la función que determina si existe una solución o no, pasándole todos los datos necesarios, para finalmente imprimir el número del caso, la respuesta del algoritmo, y en caso de que exista un posicionamiento de los aviones correcto, se imprima el parqueadero para cada uno de los aviones.

Nombre de la función	“casilla_es_aeropuerto”.
Parámetros que recibe	Recibe tres parámetros: la matriz de símbolos o aeropuerto, y la coordenada en x y en y que se pretende evaluar.
Tipo de dato que retorna	BOOL.
Excepciones que produce	No tiene.
Descripción corta de qué hace	Verifica si la matriz de símbolos en la coordenada ingresada a la función es un punto de entrada/salida (“==”).

Una vez dada la especificación de cada función, se entra en detalle de la definición de cada una de las variables:

Nombre de la variable	"numero_caso".
Tipo de dato	INT.
Para que se utiliza	Determina el caso actual al que se le busca una solución factible.
Visibilidad	La variable se define en la función "principal"
Ciclo de vida	Desde que se inicia la ejecución del programa, hasta que ya no se desea conocer la solución de mas aeropuertos, dicho en otras palabras, hasta que finaliza el programa.

Nombre de la variable	"numero_aviones".
Tipo de dato	INT.
Para que se utiliza	Almacena el numero de aviones ingresado por el usuario, y permite verificar si se debe seguir ejecutando el programa o no.
Visibilidad	La variable se define en la función "principal".
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este.

Nombre de la variable	"numero_filas".

Tipo de dato	INT.
Para que se utiliza	Determina el número de filas que tendrá el aeropuerto dado por el usuario, también sirve para generar la matriz de pesos y para verificar si dada una coordenada, esta se encuentra dentro de los límites de la matriz.
Visibilidad	La variable se define en la función “principal”, y las funciones “casilla_es_valido”, ”ponderar”, ”matriz_pesos”, “pre_ponderacion” y “resolver_problema” tienen acceso a ella.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este.

Nombre de la variable	“numero_columnas”.
Tipo de dato	INT.
Para que se utiliza	Determina el número de columnas que tendrá el aeropuerto dado por el usuario, también sirve para generar la matriz de pesos y para verificar si dada una coordenada, esta se encuentra dentro de los límites de la matriz.
Visibilidad	La variable se define en la función “principal”, y las funciones “casilla_es_valido”, ”ponderar”,

	”matriz_pesos”, “pre_ponderacion” y “resolver_problema” tienen acceso a ella.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este.

Nombre de la variable	“lista_solucion”.
Tipo de dato	LIST.
Para que se utiliza	Almacena los parqueaderos en los que cada uno de los aviones se debe ubicar para que exista una solución factible, junto a el evento correspondiente a ese parqueadero.
Visibilidad	La variable se define en la función “principal” y la función “resolver_problema” tiene acceso a ella.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este.

Nombre de la variable	“diccionario”.
Tipo de dato	DICT.
Para que se utiliza	Se utiliza para almacenar en que posición se estableció cada avión y poder desmarcar o marcar la casilla como ocupada. Se guarda como llave el evento (positivo), y como valor la coordenada en donde esta

	ubicado el avión y el símbolo que tenía originalmente.
Visibilidad	La variable se define en la función “principal” y la función “resolver_problema” tiene acceso a ella, ya que esta realiza modificaciones en la misma cada que se establece un evento, o se necesita deshacer una acción realizada anteriormente.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este.

Nombre de la variable	“matriz_simbolos”.
Tipo de dato	LIST.
Para que se utiliza	En esta variable se almacena la matriz relacionada con cada uno de los símbolos del parqueadero, la cual se modificará a la hora de que un avión aterrice o despegue, y la misma permitirá realizar la ponderación respectiva cada que sucede un evento así logrando encontrar los parqueaderos disponibles en cada caso.
Visibilidad	La variable se define en la función “principal”, y las funciones “resolver_problema”, ”ponderar”, “pre_ponderacion”, “casilla_es_aeropuerto”,

	“casilla_es_parqueadero”, “casilla_es_blanca”, “casilla_es_negra” y “casilla_es_valido” tienen acceso a ella.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este. Cabe recalcar que está variable se va modificando en el proceso de la asignación de los parqueaderos.

Nombre de la variable	“eventos”.
Tipo de dato	DEQUE (cola doblemente terminada)
Para que se utiliza	Almacena cada uno de los eventos a realizarse en el aeropuerto.
Visibilidad	La variable se define en la función “principal” y la función “resolver_problema” tiene acceso a ella.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este. Cabe recalcar que esta lista se va modificando a medida que se va procesando cada evento (se va expulsando los mismos al ser procesados, o en caso de que se deshace una acción, lo vuelve a agregar).

Nombre de la variable	“resultado”.

Tipo de dato	BOOL.
Para que se utiliza	Variable que almacena si se llegó a una solución factible o no, para imprimir la respuesta correspondiente más adelante.
Visibilidad	La variable se define en la función “principal”.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este.

Nombre de la variable	“lista_solucion_organizada”.
Tipo de dato	LIST.
Para que se utiliza	Almacena los estacionamientos organizados de forma que el parqueadero numero uno, le corresponda al avión número uno, y así sucesivamente con cada uno de los aviones del caso.
Visibilidad	La variable se define en la función “principal”.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este.

Nombre de la variable	“matriz_ponderar”.
Tipo de dato	LIST.

Para que se utiliza	Se almacena una matriz idéntica a la de símbolos, pero en cada casilla se guarda el peso que existe desde un punto inicio/salida. Esto permite identificar si existe un camino hacia la casilla que se pretenda evaluar.
Visibilidad	La variable se define en la función "principal", y las funciones "pre_ponderacion", "ponderar" y "resolver_problema", tienen acceso a ella.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende buscar una solución a un caso, hasta que se determina si existe o no una solución de este. Cabe recalcar que está variable se somete a modificaciones cada que se procesa un evento en la función recursiva, debido a que se vuelve a ponderar y se sobre escribe el valor de la variable.

Nombre de la variable	"parqueaderos".
Tipo de dato	SET.
Para que se utiliza	Sirve para almacenar los parqueaderos alcanzables desde los puntos inicio/salida.
Visibilidad	Se define en la función "principal" al hacer el llamado a la función "pre_ponderacion", y la función "resolver_problema" tiene acceso a ella.
Ciclo de vida	Su ciclo de vida es la iteración actual del bucle, es decir, desde que se pretende

	buscar una solución a un caso, hasta que se determina si existe o no una solución de este. Cabe recalcar que está variable se somete a modificaciones cada que se procesa un evento en la función recursiva. Ya que, según el evento, se desocupan o se ocupan nuevos parqueaderos.

Nombre de la variable	“pesos”.
Tipo de dato	DEQUE (cola doblemente terminada).
Para que se utiliza	Permite almacenar las coordenadas de las casillas a las cuales hay que evaluarle sus vecinos, para así generar toda la ponderación del aeropuerto.
Visibilidad	Se define en la función “pre_ponderacion” y la función “ponderar” tiene acceso a ella.
Ciclo de vida	Se crea cuando se busca obtener la ponderación de la matriz, y se destruye cuando se realiza la ponderación a toda esa matriz.

Nombre de la variable	“movimientos”.
Tipo de dato	LIST.
Para que se utiliza	Esta lista define los movimientos con el funcionamiento de observar los vecinos de la casilla actual, observado arriba, derecha, abajo, e izquierda.
Visibilidad	Se define en las funciones “resolver_problema”, cuando un avión despega, y también en “pre_ponderacion”,

	para el movimiento a través de la matriz y la ponderación de cada casilla.
Ciclo de vida	Se crea cuando se hace el llamado de la función para un evento de despegue, y se destruye cuando finaliza el llamado, es decir, cuando se devuelve un resultado. O también, se crea cuando se busca obtener la ponderación de la matriz, y se destruye cuando se realiza la ponderación a toda esa matriz.

Nombre de la variable	“evento_actual”.
Tipo de dato	INT.
Para que se utiliza	Almacena el evento a procesar, el cual sirve para comparar si este es un despegue o un aterrizaje, para hacer lo correspondiente a cada caso.
Visibilidad	Se define en la función “resolver_problema”.
Ciclo de vida	El ciclo de vida de esta es desde que empieza el llamado de la función, hasta que el llamado a la función devuelve un resultado (si existen llamados recursivos, es hasta que el llamado recursivo devuelva un resultado).

Nombre de la variable	“contador”
Tipo de dato	INT.
Para que se utiliza	Se encarga de almacenar el valor de cuantos eventos de aterrizajes hay

	consecutivos, lo cual permite minimizar los casos a la hora de determinar que hay más eventos consecutivos positivos que parqueaderos disponibles.
Visibilidad	Se define en la función "resolver_problema".
Ciclo de vida	El ciclo de vida de esta es desde que empieza el llamado de la función, hasta que el llamado a la función devuelve un resultado (si existen llamados recursivos, es hasta que el llamado recursivo devuelva un resultado).

El algoritmo presentado cumple los casos pertenecientes a la plataforma de uDebug (uDebug, s.f.), pero, este no es validado por el jurado de la Universidad de Valladolid por límite de tiempo.

Referencias

EAFIT. (s.f.). *Docentes e investigadores EAFIT*. Obtenido de https://www.eafit.edu.co/docentes-investigadores/Paginas/juan-lalinde.aspx

uDebug. (s.f.). *uDebug*. Obtenido de https://www.udebug.com

Valladolid, U. d. (s.f.). *THEONLINEJUDGE*. Obtenido de 11208 - airplane scheduling: https://onlinejudge.org/index.php?option=com_frontpage&Itemid=1