ESTRUCTURA DE DATOS Y ALGORITMOS 2

INFORME AVANCE SEMANA

PROYECTO #1 – 11208 AIRPLANE SCHEDULING

PROFESOR: JUAN GUILLERMO LALINDE PULIDO

KRISTIAN RESTREPO OSORIO

EAFIT

2023

Para el avance de esta semana, primero, nos dimos cuenta que, como se mencionó erróneamente en el primer documento, los casos de ejemplo de las matrices grandes de uDebug si son correctos, más bien por mi parte no había entendido correctamente como retornar el arreglo de solución, por ejemplo, antes, si teníamos una lista de eventos de la siguiente manera: +3+2+1-3-2-1, en el arreglo que contiene la solución de donde se parquearon los aviones, en primera posición aparecía el parqueadero donde se posiciono el avión 3, en la segunda donde se posiciono el avión 2, y en la tercera donde se posiciono el avión 1, cuando no debería ser así, con lo que esta ”desorganizado”, por así decirlo, puesto que si tenemos una lista de eventos de ese tipo y que se cumple satisfactoriamente el despegue y aterrizaje de los aviones, la lista de solución en primera posición debe tener donde se parqueó el avión con id “1”, en la segunda posición el avión con id “2”, y en la tercera posición el avión con id “3”. Para solucionar esto, ahora, a la lista de solución, una vez establece un parqueadero, no se le añade únicamente el símbolo del parqueadero, sino también, el id del avión que está aterrizando en dicho parqueadero, por lo tanto, en dicho arreglo, se guardan tuplas con el numero de parqueadero donde se estableció el avión y el id del avión, para posteriormente, una vez finalizada la función recursiva y encontrada una solución valida para ubicar los aviones, utilizamos la función “sorted” sobre dicha lista de eventos desorganizada, y establecemos como parámetro de organización, lo que esta en la tupla en la segunda posición, es decir, organizamos la lista de solución de menor a mayor, teniendo como parámetro para organizar los mismos el id del avión. Esta solución se guarda en un nuevo arreglo llamado “lista_solucion_organizada”, la cual, una vez los valores estén ubicados correctamente allí, se iterará sobre la misma imprimiendo la solución de donde se parquearon los aviones (si se llegó a una solución valida, claramente).

Además, se realizó ciertos cambios importantes al código:

- Se volvió a recorrer toda la matriz para el aterrizaje del avión en un parqueadero, sin la propiedad de ubicarlo en la de mayor ponderación. Por lo tanto, ahora la función ponderar no retorna la lista de parqueaderos, sino únicamente la matriz ponderada.
- Se elimino la variable “temporal”, en el aterrizaje de los aviones, ahora para almacenar el elemento de la casilla actual accedemos a la matriz en la posición i, j.

Esto permitió que se solucionara el problema que teníamos con que un caso se quedaba procesando, dicha variable “temporal”, además de ser innecesaria, estaba provocando problemas en el llamado recursivo de la función con los cambios en el valor de la misma. Además, se quitó la propiedad de ubicar los aviones en el mayor peso, puesto que, un algoritmo debe satisfacer una respuesta para cualquier caso posible en cierto contexto, esta propiedad no es una propiedad absoluta que mejore el rendimiento del algoritmo para todo caso ingresado, solo para casos en específico, lo cual si, puede conllevar a soluciones mas optimas en algunos casos, como más pesadas en algunos otros, así que lo dejamos como se mencionó anteriormente.

Finalmente, podemos concluir que este es el código mas cercano que se ha desarrollado en todas las entregas por mi persona a la solución del problema, pues ahora sí, definitivamente cumple todos los casos de uDebug, tanto para las matrices grandes, como para las pequeñas, pero, no se pudo comprobar con el jurado de Online Judge puesto que nos excedemos en tiempo. Específicamente el caso 2 de uDebug, que se nos demora 14 min, y el caso 21 de uDebug, que se nos demora 5 min,

aunque retornen una solución valida, son los que demuestran el “talón de Aquiles” de nuestro algoritmo, su rendimiento (el resto de los casos de uDebug los ejecuta en menos de un segundo).