Justificación de la Corrección del Método Auxiliar para encontrar la Ruta Óptima

Edgar Aguadé Nadal

May 2024

```
void River::findOptimalRoute(BinTree<string> structure, Travel
        current_travel, Travel &best_travel) const
```

Precondición $travel_{current}$ contiene el estado del viaje realizado para llegar a la ciudad actual.

Postcondición $travel_{best}$ guarda el mejor viaje realizado hasta el momento y $travel_{current}$ contiene el estado del viaje realizado una vez pasada a la ciudad actual.

1 Justificación

1.1 Caso Base

1.1.1 *structure.empty*()

$structure.empty() \iff \nexists structure.left(), structure.left() \iff$ No hay más nodos que explorar y por lo tanto se ha acabado el camino.

En este caso no se actualiza $travel_{best}$ ni $travel_{current}$ por lo que sigue cumpliendo la *Postcondición*.

1.1.2 $\neg$*structure.empty*()

$\neg structure.empty() \implies structure.value()$ es una ciudad de $travel_{current} \implies$ la longitud del viaje aumenta $\wedge$ se debe intentar comerciar con dicha ciudad.

En haber cambiado el estado de $travel_{current} \implies$ hay que asegurar que $travel_{best}$ sigue siendo el mejor viaje por lo que comparamos ambos viajes y en caso que $travel_{current}$ sea mejor lo remplazamos. Así pues, se cumple la *Postcondición*.

1.2 Caso Inductivo

Hipotesi El árbol no es vacio. Es decir, $\neg structure.empty()$

Aislando el caso directo en que $travel_{current}.objectiveAchived()$ *es cierto (el barco ya ha vendido y comprado todo lo posible y por lo tanto no debe continuar la ruta), pues en este caso* $\neg structure.empty() \implies Postcondición$. *Veamos entonces el caso inductivo que es el interesante:*

$\neg structure.empty() \implies \exists structure.left() \wedge structure.right()$ Por lo que podemos ver dos casos

- $structure.left().empty() \wedge structure.right().empty()$

  En este caso no podemos continuar el camino, puesto que ya hemos llegado al final del árbol. Por otro lado, en ser $\neg structure.empty() \implies$ *Postcondición.*

- $\neg structure.left().empty() \wedge \neg structure.right().empty()$

  $\implies \exists structure.left().value(), structure.right().value()$ Por lo tanto, como se cumple la *Precondición* se cumplirá la *Postcondición.*

Decrecimiento

$structure.left(), structure.right() \subseteq structure$ por lo que las llamadas se hacen cada vez con un árbol más pequeño, asegurando así que eventualmente se llega a una hoja/final.