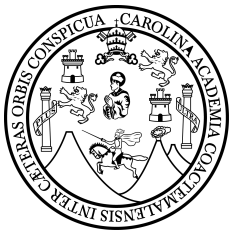
División de Ciencias de la Ingeniería
Centro Universitario de Occidente
Universidad de San Carlos de Guatemala



Manual Tecnico
Práctica 1 Laboratorio Estructura de Datos 1.

03/01/2022

MARCO JOSE MUNGUIA ALVA - 201931804

Ingreso de apuestas

```
private void saveBets(String[] data, NodeList<Bet> bets) {
        try {
            Gambler gambler = new Gambler(data[0].trim());       // 1
            Double amount = Double.valueOf(data[1].trim());      // 1
            Horse[] horses = new Horse[data.length - 2];         //1
            readHorses(horses, data, 0);                         //1
            bets.addAtHead(new Bet(gambler, horses, amount));    //1
            tmpSteps++;
        } catch (Exception e) {
            System.out.println("Error while saving data " + e.getMessage());
        }
    }
```

Por tanto la complejidad de mi Algoritmo es de O(1)

Verificación de Apuestas

```
public void verifyBets(Node<Bet> bet, NodeList<Bet> bets, int actualHorse, int
tmpHorse) {
        if (bet != null) {
            verifyBet(bet, bets, actualHorse, tmpHorse);              //n
            bet = bet.getNext();                                      //1
            tmp = true;                                               //1
            verifyBets(bet, bets, actualHorse, tmpHorse);             //T(n)
        } else {

        }

    }
```

Resolviendo la recursion T(n)

$$T(n) = T(n) + 2$$
$$= T(n+1) + 4$$
$$= T(n+2) + 6$$
$$= T(n-k) + 2k$$

$n-k = 0 \Rightarrow k=n$

$$\Rightarrow T(n) = T(0) + 3n$$
$$= 2n + 1$$
$$T(n) = n$$

->Funcion de complejidad final: $O(n) = n$

Por tanto la complejidad de mi Algoritmo es de O(n)

Cálculo de Resultados

```
public void giveScores(Node<Bet> bet, NodeList<Bet> bets, Horse[] horsePodium) {
       if ((bet != null)) {
           if ((bet != null) && (bet.getData().isValid())) {          //1
               giveScore(bet, bets, horsePodium);                     //n
               bet = bet.getNext();                                   //1
               giveScores(bet, bets, horsePodium);                    //T(n)
           } else {
               bet = bet.getNext();
               giveScores(bet, bets, horsePodium);
           }
       } else {

       }

   }
```

Resolviendo la recursion T(n)

$T(n) = T(n) + 2$
$= T(n+1) + 4$
$= T(n+2) + 6$
$= T(n-k) + 2k$
$n-k = 0 \Rightarrow k=n$
$\Rightarrow T(n) = T(0) + 3n$
$= 2n + 1$
$T(n) = n$

->Funcion de complejidad final: $O(n) = n$

Por tanto la complejidad de mi Algoritmo es de O(n)

Ordenamiento de Resultados

```
public void orderingResults(Node<Bet> bet, NodeList<Bet> bets) {
       if ((bet != null)) {
           if ((bet != null) && (bet.getData().isValid())) {
               if (bet.getData().getGambler().getGamblerScore() >
bets.getHead().getData().getGambler().getGamblerScore()) {
                   Bet tmp = bet.getData();
                   bets.addAtHead(tmp);
                   bet = bet.getNext();
                   orderingResults(bet, bets);                    //T(n)
               } else {
               }
```

```
            }
        } else {


        }
    }
```

Resolviendo la recursion T(n)

$$T(n) = T(n) + 2$$
$$= T(n+1) + 4$$
$$= T(n+2) + 6$$
$$= T(n-k) + 2k$$

$n-k = 0 \Rightarrow k=n$

$$\Rightarrow T(n) = T(0) + 3n$$
$$= 2n + 1$$
$$T(n) = n$$

->Funcion de complejidad final: $O(n) = n$

Por tanto la complejidad de mi Algoritmo es de O(n)