Prueba  
Nombre: Diego Torreblanca

Rut: 19.669.326-2

Profesor: Travis Gagie



func X(X []int) (int,int,int,bool) {

for k:=len(X)-1;k >=0;k--{

for i, j := 0, len(X) - 1; i < len(X) && j >= 0; {

if X[i] + X[j] < X[k] {

i++

}else if X[i] + X[j] > X[k] {

j--

}else{

return i, j, k , true

}

}

}

return -1, -1, -1, false

}

Se modifico el algoritmo para que se pudiese cumplir la condición de encontrar xi+xj=xk , para eso se aplico un for dentro de otro donde se sabe que el tiempo en este caso seria de O(n2), aparte se cambiaron algunas condiciones para poder funcionar en con el arreglo en desorden.



func X(X []int) (int,int,int,bool) {

for k:=len(X)-1;k >=0;k--{

for i, j := 0, len(X) - 1; i < len(X) && j >= 0; {

if X[i] + X[j] + X[k] < 0 {

i++

}else if X[i] + X[j] + X[k] > 0 {

j--

}else{

return i, j, k , true

}

}

}

return -1, -1, -1, false

}

Se modifico el algoritmo desde las condiciones solamente para no perder la complejidad que tiene, prácticamente se cambiaron las condiciones de los if para poder encontrar xi+xj+xk=0.



func BusquedaBinaria(arr []int, valor int,pos int) int {

inf := 0

sup := len(arr)-1

for inf <= sup {

medio := (inf + sup) / 2

if arr[medio] < valor {

inf = medio + 1

} else {

sup = medio - 1

}

}

if inf == len(arr) || arr[inf] != valor {

return -1

} else {

return inf

}

}

func BusquedaExponencial(arr []int, tamano int, valor int) int {

if (tamano == 0) {

fmt.Println("no existe")

return -1

}

pos := 1

for pos < tamano && arr[pos] < valor {

pos \*= 2

}

fmt.Println(arr, valor, Min(pos/2, tamano))

i:=BusquedaBinaria(arr, valor, Min(pos/2,tamano))

return i

}

Se aplico búsqueda exponencial para encontrar la posición mas cercana y después búsqueda binaria para encontrar la posición exacta.