1. Als het array gesorteerd is, kan het lineair zoeken uit de vorige vragen dan nog verder versneld worden? Zo ja, geef ook dit algoritme (zitErinC). Toon met een test het verschil met het vorige algoritme aan.

Ja, je kan er voor zorgen dat als je een getal tegen komt dat groter is dan dat je zoekt, je er voor kan zorgen dat je stopt met zoeken. Deze methode verminderd de worstcasetijd niet (merkbaar), maar wel de worstcasescenario. Ook wordt de averagecasetijd lager, aangezien je stopt als het volgende getal groter is dan het gezochte getal. Het verschil van de worst case zit hem in het feit dat de worstcase niet meer is dat het getal niet in de array staat maar dat hij op de laatste plaats staat.

1. Uitgaande van dezelfde gegevens wat kan je dan zeggen over de zoektijd van het binaire zoekalgoritme? Stel dat het array verdubbeld wordt, dus uit 2 miljoen elementen bestaat, wat kan je dan zeggen over tijdsduur bij het lineair zoeken en bij het binair zoeken?  
   Implementeer en test het binair zoeken in onze getalrij (zitErinD).

Bij een binair zoekalgoritme is de gebruikte tijd een stuk minder dan bij een lineair zoekalgoritme. Als arrays worden verdubbeld, zullen de lineaire algoritmes er 2 keer zolang over doen, terwijl de binaire algoritme er een paar stappen bij doet, en er dus niet veel verschil is.

1. Wat is de orde van het lineaire zoekalgoritme? Hier verwachten we dus een antwoord in de vorm van *O(…)*, met op … een bepaalde functie van n.

public boolean zitErinA( int zoekWaarde )  
{  
 int counter = 0; // 1 stap  
 boolean found = false; // 1 stap  
 while (counter < getallen.length) // n stappen  
 {  
 if (getallen[counter] == zoekWaarde) // 1 stap  
 found = true; // 1 stap  
 counter++; // 1 stap  
 }  
 return found; // 1 stap  
}

De formule voor deze functie is: f(x) = 3n + 3

Hierdoor is de orde: O(n)

1. Wat is de orde van het binaire zoekalgoritme? Geef een uitleg waarom dat zo is.

Omdat er elke keer een getal door de helft gedaan wordt is er een veel kleiner aantal stappen. Stel er zijn 32 (25) voorwerpen, 0 tot en met 31

De eerste check is op positie 15 (er wordt geteld vanaf 0, dus 15 is het midden)

Dan 7 of 23

Dan 3, 11, 19 of 27

Dan 1, 5, 9, 13, 17, 21, 25 of 29

Dan 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28 en 30.

Binnen 5 stappen zijn alle mogelijke gegevens doorgelopen. Zou je 64 (26) gebruiken, zou je maar 1 stap meer nodig hebben. Dit komt er op neer dat het binaire zoekalgoritme een orde heeft van log2(n).

1. Verzameling van training set is 4 (alle auto’s)

Verzameling van features is 3 (ac, abs, catogorie)

ac, abs, catogorie

Tac = Yes No

Tabs = Yes No

Tcategorie = Hoog midden laag

|Tac| = 2

|Tabs| = 2

|Tcategorie| = 3

Frequentie van hoog is 2

Frequentie van midden is 1

Frequentie van laag is 1

Frequentie van Yes (abs) = 2

Frequentie van no (abs) = 2

Frequentie van Yes (ac) = 2

Frequentie van no (ac) = 2

M = 2 (ja en nee)