Verslag opdracht 1

Victor van de Riet & Rob logtenberg

2015

# De analyse

## Legale permutaties

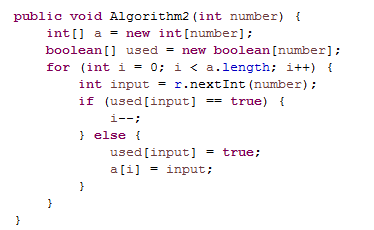
## Big-Oh schatting

Algoritme 1:



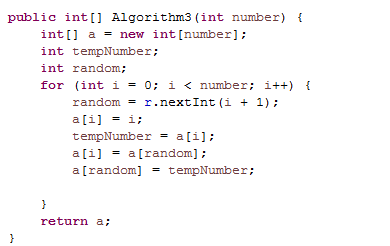
De Big-OH is: O(n2). Dit komt omdat we in het algoritme 2 for-loops gebruiken. De eerste for-loop gaat tot “n” en de tweede for-loop gaat in het slechtste geval tot “n”. De constante waarden mag je verwaarlozen.

Algoritme 2:



Bij algoritme 2 is de Big-Oh: O(n). Hier is maar 1 for-loop en dat levert maar één “n” op. Contante getallen mag je verwaarlozen.

Algoritme 3:



Ook bij algoritme 3 is de Big-Oh: O(n). Ook hier is maar 1 for-loop en levert maar één “n” op. Constante getallen mag je verwaarlozen.

# Ontwerp

Algoritme 1:

Bij dit algoritme wordt eerst een willekeurig getal gegenereerd hierna wordt gekeken of dit getal al in de array staat. Als dit niet zo is wordt het getal in de array gezet als dit wel zo is wordt er een nieuw getal gegenereerd en weer gekeken of hij er al instaat. Dit gaat zo door tot de array volledig gevuld is.

Algoritme 2:

Dit algoritme werkt bijna op dezelfde manier. Echter wordt er hier een array used bijgehouden. Als een getal in de hoofd array wordt geplaatst, wordt er in de array used op de plaats wat het getal was true gezet. Op deze manier kun je makkelijker controleren of het getal al een keer gebruikt is.

Algoritme 3:

Hierbij wordt het algoritme stap voor stap gevuld, zodra een plek is gevuld wordt deze direct verwisseld met een willekeurige reeds gevulde positie.

# Implementatie:

Zie code

# Resultaten

Meetresultaten

|  |  |
| --- | --- |
| Elementen (N) | Tijd (ms) |
| 0 | 0 |
| 5000 | 92 |
| 10000 | 409 |
| 20000 | 1838 |
| 50000 | 13604 |
| 100000 | 59948 |

De grafiek neemt kwadratisch toe, dus de Big-Oh schatting klopt wel

Als we naar N = 10000 kijken t.o.v. N = 5000 dan is N verdubbelt maar omdat de schatting kwadratisch is moet T 4x zo groot zijn. Als we naar de bijbehorende tijden kijken klopt dit. Als we verder gaan vergelijken dan klopt dit ook voor de rest van de Elementen met de bijbehorende Tijd. Alleen hoe groter de N hoe groter de afwijking t.o.v. de N in de rij daarboven.

Algoritme 2:

|  |  |
| --- | --- |
| Elementen (N) | Tijd (ms) |
| 0 | 0 |
| 100000 | 22,5 |
| 500000 | 140 |
| 1000000 | 283 |
| 5000000 | 2542,5 |
| 10000000 | 7416,5 |

De grafiek is kwadratisch, dus de Big-Oh schatting klopt niet. Waarschijnlijk komt dit omdat het algoritme ook moet checken of het getal al een keer gebruikt is. Dit neemt veel tijd in beslag.

Als we hier naar deze tabel kijken zien we dat 500000, t.o.v. 10000, 5x zo groot is als 100000 dus moet de tijd ook 5x zo groot zijn. Want we hebben immers geschat dat de grafiek lineair zou zijn. Dit gaat ook goed als we naar 1000000 kijken t.o.v. 500000 maar vanaf 5000000 is er geen touw meer aan vast te knopen. De tijd bij 5000000 t.o.v. 1000000 is (3542,5 / 283) ≈ 12 en dat is niet (ongeveer) 566 zoals we verwacht hadden. Dus hier kunnen we vrij weinig mee.

Algoritme 3:

|  |  |
| --- | --- |
| Elementen (N) | Tijd (ms) |
| 0 | 0 |
| 5000000 | 231,5 |
| 10000000 | 502 |
| 20000000 | 1076,5 |
| 40000000 | 2328 |
| 80000000 | 4737 |

De grafiek is lineair, dus onze geschatte Big-Oh komt wel overeen met de praktijk.

Als we kijken naar 10000000 t.o.v. 5000000 is dat een verdubbeling en als we daarna kijken naar de bijbehorende tijd zien we ook een verdubbeling (ongeveer), dit is ook het geval bij 20000000, 40000000 en 80000000 dus de grafiek klopt bij de geschatte Big-Oh.