1.6

Quick Sort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1000 | 10000 | 30000 | 30000 (sorted) | 30000 (reversed sorted) |
| 0.0029914379.. | 0.038895606.. | 0.1196835.. | Stack overflow | Stack overflow |

Quick sort heeft een complexiteit van O(n log n). Je moet minstens een keer door de gehele lijst heen loopen. Quick sort zet 1 element in de juiste positie door de gehele array op te delen in ‘groter’ en ‘kleiner’. In deze sub arrays wordt dan weer 1 element op de juiste positie gezet, en de arrays zelf weer verder opgedeeld. Dit blijft zich herhalen totdat alle elementen zijn gesorteerd. Een lijst van 8 kan maximaal 3 keer worden opgedeeld. Een lijst van 16 maar 4 keer, een lijst van 32 5 keer etc. Hier komt de log n vandaan. (Let op: deze complexiteit geldt alleen voor de best en average case).

De worst case voor een quick sort is een gesorteerde/reverse gesorteerde lijst. Hierbij eindig je namelijk steeds met een lege lijst en een lijst die bijna alle elementen bevat. Hierdoor neemt de complexiteit drastisch toe (O(n2)). Dit is ook de reden dat de laatste twee tests gefaald zijn; het algoritme moest teveel stappen zetten en raakte te diep in de stack.

De best case is wanneer alle opdelingen voor ongeveer even grote sub arrays zorgen. Iedere functie call hoeft dan alleen een lijst die de helft van het originele formaat heeft te processen. We hoeven dan ook maar log n functie calls te maken voordat we een lijst van 1 hebben.

De average case is de willekeurig gesorteerde lijst. Dit is namelijk niet altijd de best case, want doordat het willekeurig gesorteerd is, is er een kans dat er zich toch wat stukken in de array bevinden die zorgen voor grote verschillen in de sub arrays.