Algoritmiek



Groep: 21

Klas: 2A

Groepsleden: Diego Bencherif & Marle Mulder

Datum: 31 januari 2020

# Inhoud

[**Inhoud**](#_lglwr95wf7pg) **2**

[**Inleiding**](#_ttw0346lhnsj) **2**

[**Welke algoritmes hebben we toegepast?**](#_fmvp3n1acn10) **2**

[**Wat is onze speedup?**](#_37zwu1lnoz6x) **2**

# Inleiding

Het thema van deze opdracht is het iteratief optimaliseren van een simulatie van een gevecht tussen een aantal eenheden. De simulatie wordt gebruikt om bepaalde gevecht scenario's te analyseren. Het is van belang dat deze simulaties snel kunnen draaien zodat op tijd een nieuwe strategie ontwikkeld kan worden.

# Welke algoritmes hebben we toegepast?

Wij hebben de volgende algoritmes toegepast:

* Quick sort
* Spatial partitioning

Wij hebben van de insertion sort een quicksort gemaakt. De quick sort is een ‘divide and conquer algorithm’. Bij de quicksort wordt er als eerst uit een lijst een ‘pivot’ gekozen. Dit kunt u zien als een punt ergens in de lijst. Van dit punt wordt het middelpunt gemaakt en worden er twee aparte lijsten gemaakt. Een lijst die onder het punt zit en een lijst die boven het punt zit. Daarna worden er in de lijsten weer geshuffeld, totdat we een oplopende lijst hebben. We hebben de quicksort gekozen, omdat hij tot wel drie keer zo efficiënt kan zijn dan de insertion sort of de merge sort.

Verder hebben we gebruikt gemaakt van spatial partitioning. Bij spatial partitioning worden er objecten efficiënt gelokaliseerd door ze op te slaan in een datastructuur die is georganiseerd door hun posities. Helaas hebben wij deze niet compleet werkend gekregen voor de deadline en hebben wij besloten deze gecommend in de code te laten. Hierdoor werkt de rest van de code wel, maar kunt u nog steeds zien wat we hebben gemaakt voor spatial partitioning.

# Wat is onze speedup?

Onze speedup bij de quicksort was 1,3. Hierdoor werd de simulatie 5 seconden sneller. Ook veranderde de Big0 van 0(n\*2) naar 0(log).