Rapportage

Probleembeschrijving

Stel je voor je bent een groot fan van Sudoku en andere puzzels en je gaat dus door boeken heen alsof het niks is. Dit is het geval voor mijn oma ze is super snel met het oplossen en heeft dus erg veel afgeronden puzzelboeken thuis liggen die niet meer gebruikt worden. Dit kan natuurlijk best een kostenpost worden en over het algemeen zijn de sudoku's in de boeken niet de allermoeilijkste.

Hierom heb ik een applicatie gemaakt die op 4 verschillende niveau's sudoku's genereert die bijna nooit hetzelfde zijn van elkaar. Het minimale aantal cijfers nodig voor een unieke oplossing van een sudoku board is 17 cijfers. Dit is dus ook de grootste uitdaging en deze uitdaging is ook speelbaar in de applicatie.

Eisen

- Recursive Backtracking algoritme voor oplossen sudoku
- Duidelijk visuele weergave Sudoku
- Meerdere niveau's aanwezig
- Goede documentatie van de code
- Alle code is OOP
- Solve algorithm is efficient en snel

Algorithm

Voor dit project heb ik een Recursive Backtracking algorithm gemaakt op basis van het onderstaand vermeld en meegeleverd artikel. Het originele algoritme en het artikel is geschreven in 2016. Sinds toen zijn de computers veel sneller geworden, dus de originele 20s die beschreven staat in het artikel zijn door de nieuwe programmeertaal en mijn laptop, Apple Macbook Pro M1, is ook veel sneller. Ik heb mijn code getest en hier zijn de resultaten. Zoals u kunt zien is de tijd dat het algoritme erover doet heel erg wisselvallig.

```
The total number of times a new puzzle has been created and solved.
Total           : 20
Total duration  : 172.145493375
Single duration : 8.60727466875
```

```
The total number of times a new puzzle has been created and solved.
Total           : 50
Total duration  : 182.454493625
Single duration : 3.6490898725
```

Beschrijving

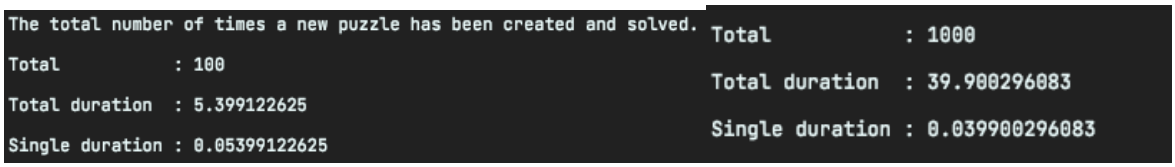
Het algoritme wat ik geschreven heb gaat op zoek naar een lege cell. Vervolgens vult die met een loop cijfers van 1 tot 9 in. Per cijfer worden er checks uitgevoerd of het valid is in die cell. Als dat het geval is word het algoritme zelf aangeroepen, dit is de recursie. Het moment dat er geen valid cijfers zijn. Word de huidige cell waar net cijfers in geprobeerd zijn terug veranderd naar 0 en word de voorgaande cel een cijfer groter gekozen, dit is het backtracking gedeelte. Vervolgens word dit gedaan zolang als er geen lege plekken meer zijn in het Sudoku board.

Resultaat

Ik heb een Sudoku game gemaakt in deze game kan je zelf een puzzel oplossen, nieuwe puzzels opstarten en oplossen en ook je gegeven antwoorden verwijderen om nog een keer dezelfde puzzel te doen. De puzzels worden door een algoritme compleet random gegenereerd zodat je (bijna) nooit dezelfde puzzel zal krijgen.

Het spel heeft 4 verschillende niveau's, het programma is flexibel opgezet daardoor is het heel erg makkelijk om meer niveau's toe te voegen naar wens. Het moeilijkste niveau is bijna onmogelijk deze bevat 17 getallen en dat is het absolute minimum wat nodig is om het nog een unieke oplossing te geven.

Ook heb ik een algoritme gecreëerd die binnen gemiddeld 3-8 sec bij het moeilijkste niveau de puzzel helemaal voor je oplost. Bij de makkelijkere puzzels word dit al heel wat sneller te zien bij de screenshot hieronder.



Het artikel

http://www.journal.bonfring.org/abstract.php?id=2&archiveid=484