Kenan de Vries

Studentnummer:1811470

Klas V2A

***Verslag Bucket Sort, HPP, 13/02/2023***

**Inleiding**

Voor het vak High-Performance Programming ben ik aan de slag gegaan met het implementeren van een Bucket Sort algoritme in de codeer taal C++. Dit is gedaan aan de hand van door de docent geleverde pseudocode. Deze opdracht geld ter introductie van het vak, maar was desalniettemin een redelijke uitdaging. Dit kwam voornamelijk door een lange tijd niet met C++ gewerkt te hebben. Het doel van dit (korte) verslag is de geschreven code toe te lichten.

**Algoritme in het kort**

Bucket Sort begint met een een-dimensionale array van n nummers die gesorteerd worden en een twee-dimensionale array van integers, met rijen genummerd van 0 tot 9 en kolommen geïndexeerd van 0 tot n-1. Elke rij in de twee-dimensionale array heeft een bucket. Het algoritme werkt dan alsvolgt:

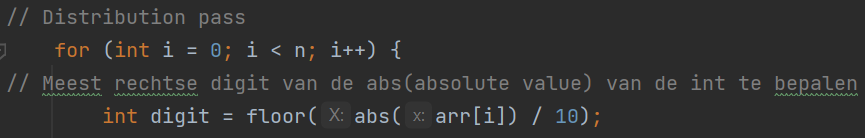
1.Plaats elke waarde van de een-dimensionale array in een rij van de bucket array, gebaseerd op het meest rechtse cijfer in het getal (de "een"-waarde). Bijvoorbeeld, 97 wordt geplaatst in rij 7, 3 wordt geplaatst in rij 3 en 100 wordt geplaatst in rij 0. Deze stap heet de distribution pass.

2.Loop door de bucket array rij voor rij, en kopieer de waardes terug in de originele array. Deze stap heet de gathering pass. De volgorde van de hierboven genoemde getallen is dus nu 100, 3, 97.

3.Herhaal dit proces voor elke volgende digit-positie (dus voor de tientallen, honderdtallen, etc.). Na de laatste gathering pass is de array gesorteerd.

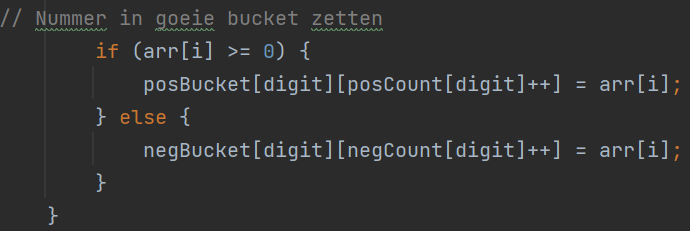
**Distribution pass**

Het doel van de distribution pass is het meest rechtse getal bepalen en deze vervolgens in de goeie bucket te zetten. Voor de eerste stap hiervan is de volgende code gebruikt:



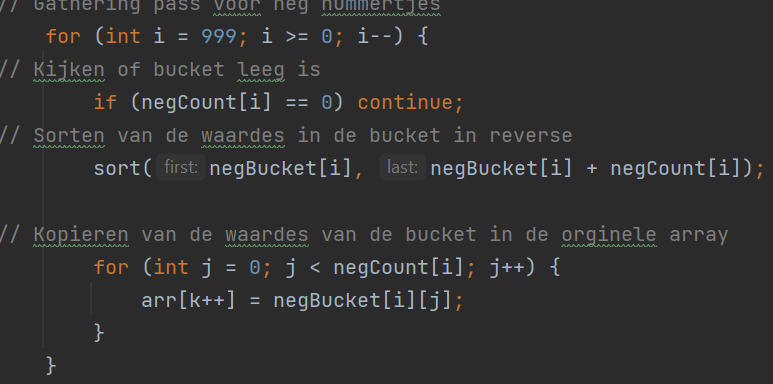
Hierbij is het n de index van de twee-dimensionale array. Wat opvalt is dat hier de c++ functie floor wordt gebruikt. Dit is omdat het data type double vaak in de code wordt gebruikt, zodat het mogelijk is om mijn implementatie van het algoritme te laten werken met decimalen getallen. Dit leverde echter wat problemen op waarover verder in het verslag meer te lezen is.

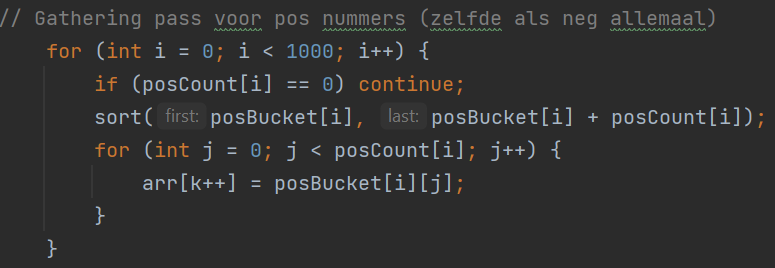
Voor de tweede stap van de Distribution pass moet de digit in de goeie bucket geplaatst worden. Hierbij wordt gekeken of deze groter of gelijk is aan 0. Zo niet, dan wordt digit in een bucket met negatieve getallen geplaatst.



**Gathering pass**

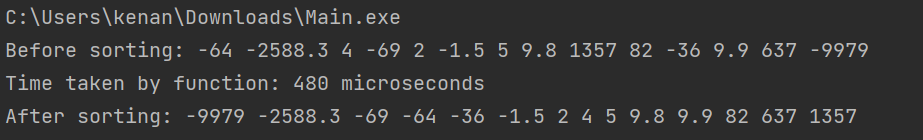
Voor de gathering pass zijn 2 verschillende variaties gemaakt.



Doordat mijn array size 1000 is, en mijn input getallen dus nooit groter kunnen zijn dan 9999 of –9999 (wederom hier later meer over) beginnen we de loop bij 999 en eindigen we bij 0. In dit proces verzamelt hij alle waardes die negatief zijn. Doordat we aan het begin de algorithm library hebben geïmporteerd kunnen we de sort functie aanroepen om (gek genoeg) de bucket correct te sorten. In de laatste stap wordt een for loop gebruikt om alles goed te zetten. Aan het begin is k 0 en deze loopt door tot de index van het langste getal. Bij een lijst van [[12,1423],[343,454]] wordt k dus max 3. De arr is de originele meegegeven lijst. Voor positieve getallen werkt dit hetzelfde maar dan loopen we de andere kant op. 

**Discussie**

Mijn implementatie van het algoritme heeft meerdere problemen. Mijn eerste fout was het gebruiken van de floor functie. Dit was mijn manier om decimalen te kunnen gebruiken in de bucket sort. Echter moest ik hierdoor een limit aan de lengte van de getallen zetten omdat er anders een crash ontstaat. Na onderzoek te hebben gedaan had ik (zoals op canvas in de tips staat) een vector moeten gebruiken. De bucket sort is meerdere keren gerund en de output klopt verder altijd.



Ik ben redelijk tevreden met de code, alleen het crashen bij lange getallen laat een zure nasmaak in mijn mond.