2016

Verslag opdracht 2

VICTOR VAN DE RIET & ROB LOGTENBERG

De analyse

De random getallen die in de file staan moeten gesorteerd worden in een heap. De is een zo lang mogelijke reeks opeenvolgende getallen. De verwachting is dat voor aantal getallen N en heap grootte H het aantal runs is uit te rekenen met (N/H)/2. We laten ons programma het verwachte aantal runs berekenen en bijhouden hoeveel runs er echt zijn nodig geweest. Hiermee kunnen we kijken of het programma voldoet aan de verwachtingen.

Het ontwerp

HeapBuilder.createRuns()

Met de createRuns wordt het programma gestart, hierin wordt de heap gebouwd, worden de getallen weggeschreven en als deze methode is doorlopen gebeurd er ook niet veel meer met de heap.

FileBeheer.getArrayFromFile(String fileName, String seperator)

Met deze methode kan een int array worden gehaald uit een bestand.

FileBeheer. writeArraytoFile(ArrayList<Integer> intArray)

Hiermee worden de items in de meegegeven ArrayList weggeschreven naar het bestand

HeapBuilder.BuildHeap()

Deze methode maakt de eerste heap van elke run aan, dit moet namelijk eerst gebeuren voordat we kunnen beginnen met het wegschrijven van nummers naar de output file.

HeapBuilder.getNext()

Deze methode leest een nieuw getal uit de input array zodat deze in de heap kan worden gezet.

Heap.swap(int pos1, int pos2)

Met de swap methode kunnen 2 getallen in de heap van positie wisselen, zonder deze methode kan de heap nooit goed worden opgebouwd.

Heap.insert(int elem)

Met de insert methode wordt een nieuw getal toegevoegd aan de heap, ook controleerd deze methode of er misschien met een reeds bestaand element in de heap van positie moet worden gewisseld met de Heap.swap methode.

Heap.setElement(int index, int element)

Het deze methode wordt een element op een positie in de heap array gezet

Heap.setMin(in newMin)

Dit is een zeer simpele methode, maar het is belangrijk want zonder deze methode kan er geen nieuwe root worden gezet (al zal setElement met index 0 ook werken maar dit is netter).

Heap.naarBeneden(int position)

Met deze methode wordt de waarde van de meegegeven positie naar beneden verplaatst door hem te verwisselen met zijn kleinste kind

Heap.printHeap()

Deze methode is geschikt voor testen, het print de heap uit zodat je kan controleren of de opbouw is zoals je zou verwachten op het moment dat je deze methode aanroept.

De implementatie

Zie code.

Resultaten

We hebben het programma meerdere keren uitgevoerd met verschillende waarden voor N en H (h=heap en a= aantal getallen). Dit was het resultaat van 2 keer testen:

```
h50, a1000: runs: 10 . Verwachte runs: 10
h50, a10000: runs: 100 . Verwachte runs: 100
h10, a1000: runs: 50 . Verwachte runs: 50
h50, a5000: runs: 50 . Verwachte runs: 50
h10, a5000: runs: 245 . Verwachte runs: 250
h5, a100: runs: 11 . Verwachte runs: 10
h5, a1000: runs: 96 . Verwachte runs: 100
h10, a10000: runs: 491 . Verwachte runs: 500
```

```
h50, a1000: runs: 10 . Verwachte runs: 10
h50, a10000: runs: 100 . Verwachte runs: 100
h10, a1000: runs: 50 . Verwachte runs: 50
h50, a5000: runs: 51 . Verwachte runs: 50
h10, a5000: runs: 249 . Verwachte runs: 250
h5, a100: runs: 10 . Verwachte runs: 10
h5, a1000: runs: 98 . Verwachte runs: 100
h10, a10000: runs: 489 . Verwachte runs: 500
```

Hier valt te zien dat we het verwachtte aantal runs benaderen en dat er een kleine variatie zit in de resultaten, vooral als de runs wat meer worden.