UNIVERSITEIT GENT

Taak 3: Het partitieprobleem

Algoritmen en Datastructuren I

Academiejaar 2010-2011

1 Probleem

Veronderstel dat drie personen de boeken op een boekenplank moeten doorkijken om een bepaald stuk informatie op te zoeken. De boeken worden zo "eerlijk" (zie definitie) mogelijk verdeeld over de personen, maar wel zodanig dat elke persoon een stuk van de boekenplank toegewezen krijgt (zodat geen herschikken van de boeken nodig is). Gevraagd is om een goede verdeling van de boekenplank te bepalen. Wanneer alle boeken evenveel pagina 's bevatten, dan kan de boekenplank in gelijke delen worden opgesplitst. Bijvoorbeeld,

```
100 100 100 | 100 100 100 | 100 100 100
```

en iedereen heeft 300 pagina 's te verwerken. Deze benadering werkt uiteraard niet als de boeken verschillende dikte hebben. Bijvoorbeeld, in de volgende verdeling

```
100 200 300 | 400 500 600 | 700 800 900
```

heeft de eerste persoon slechts 600 pagina 's te bekijken, terwijl de derde persoon er 2400 moet verwerken. Voor deze boekenplank is de volgende verdeling

```
100 200 300 400 500 | 600 700 | 800 900
```

uiteindelijk de eerlijkste.

Algemeen wordt het partitieprobleem als volgt geformuleerd. Gegeven een rij S van niet-negatieve getallen $s_1, s_2, ..., s_n$ en een positief geheel getal k. Gevraagd is om S te partitioneren in k deelrijen, zodanig dat de maximale som over alle deelrijen geminimaliseerd wordt.

2 Opgave

Ontwerp en implementeer twee algoritmen voor het partitieprobleem.

1. Het eerste algoritme moet een **recursief backtracking-algoritme** zijn.
2. Het tweede algoritme moet een **gretig algoritme** zijn. Merk op dat een gretig algoritme streeft naar zo goed mogelijke, maar niet noodzakelijk optimale resultaten.

3 Implementatie

Je recursief backtracking-algoritme komt in een klasse `BacktrackingPartitionSolver`. Je gretig algoritme komt in een klasse `GreedyPartitionSolver`. Beide klassen implementeren de interface `PartitionSolver`. Deze interface heeft slechts één methode:

```
int[] solve(int[] row, int k);
```

De parameters `row` en `k` geven respectievelijk de rij zelf en het aantal gewenste partities weer. Merk op dat om k partities te bekomen, slechts $k-1$ scheidingen dienen geplaatst te worden. Deze methode retourneert dan ook een tabel van grootte $k-1$ die de **index** (geteld vanaf 0) van het eerste element van elke deelrij bevat. Uiteraard begint altijd op positie 0 een deelrij. De positie 0 neem je dan ook **niet** op in deze tabel.

Bijvoorbeeld, voor de verdeling

```
100 200 300 400 500 | 600 700 | 800 900
```

retourneer je

```
5 7
```

en dus **niet** `4 6`, niet `0 5 7` en niet `0 4 6`.

4 Verslag

In een **bondig** verslag van **1 à 3 bladzijden** beschrijf je duidelijk de werking van je beide algoritmen. Vermeld ook alles wat je eventueel gedaan hebt om je algoritmen te optimaliseren/versnellen. Geef een **voorbeeld** waarbij je gretig algoritme een **suboptimale** oplossing geeft. Beschrijf duidelijk **op welke manier** je gretig algoritme tot deze suboptimale oplossing komt en geef ook de correcte oplossing. Er worden voor deze taak **geen complexiteiten of tijdsmetingen** gevraagd.

5 Nog enkele algemene opmerkingen

- Plaats je code **niet in een package-structuur** en voorzie je code van voldoende duidelijke **commentaar**.
- De beide gevraagde klassen moeten een **default-constructor** (zonder parameters) hebben. Dit wordt gecontroleerd m.b.v. de klasse `ConstructorTest`.
- Zeer belangrijk: **schrijf niets naar het scherm**, ook geen debuginformatie of foutmeldingen, niet naar stdout en niet naar stderr.
- Voor je verslag gebruik je de structuur van de gegeven **template** (`TemplateVerslag.odt`). Het is ook toegelaten dezelfde structuur over te nemen in LaTeX of in een andere tekstverwerker.

6 Indienen

Om in te dienen bundel je je verslag en code in een zip-bestand. Dit bestand bevat **enkel** een bestand `verslag.pdf` en je zelfgeschreven javabestanden. We verwachten dat je dit bestand indient voor **maandag 2 mei om 17u** via `http://indianio.ugent.be` bij het project: "Algoritmen en Datastructuren I: Taak 3".

Hiernaast dient er ook een **papieren versie** van het verslag ingediend te worden. Dit kan tot de deadline in bureau 40.09.110.016 (Bart) of 40.09.110.032 (Stéphanie).

7 Belangrijke opmerking

Het project is **individueel** en dient dus door jou persoonlijk gemaakt te worden. Het is uiteraard toegestaan ideeën uit te wisselen maar het is ten strengste verboden code uit te wisselen, op welke manier dan ook. Het overnemen van code beschouwen we als fraude van beide betrokken partijen en zal in overeenstemming met het examenreglement behandeld worden.