DM536/DM574: INTRODUKTION TIL PROGRAMMERING
Gruppeprojekt — Efterår 2024 — Fase II
Afleveringsfrist: d. 29. november 2024, kl.18

Oversigt

I denne fase af projektet udvikles de topmoduler i programmet, som bygger et sorteringsnetværk på et bestemt antal kanaler. Af hensyn til strukturering deles programmet i tre moduler, `generate.py`, `prune.py` og `network_finder.py`. Modul `prune.py` er udleveret.

Til at hjælpe med dette formål skal der også defineres en ny datatype, som udbydes i et modul `filter.py`.

Modul `filter.py`

Dette modul definerer en datatype `Filter`, som består af et comparatornetværk samt med dets binære outputs. Formålene med datatypen er, at undgå at man gentagne gange skal beregne de outputs for et bestemt netværk.

Et af formålet med filtre er, at kunne bestemme hvilke comparators er "interessante" at tilføje til et netværk, i den forstand, at de bidrager til at sortere mere. En comparator c siges at være *redundant* ift. et netværk hvis antallet af outputs ikke ændres, når c tilføjes til netværket.

Udover definitionen af datatypen skal modulet indeholde:

- en funktion `make_empty_filter(n: int) -> Filter`, som returnerer et filter bestående af det tomme netværk og alle binære input af den givne længde;
- to funktioner `net(f: Filter) -> Network` og `out(f: Filter) -> list[list[int]]` som returnerer hhv. netværket og outputsene i et filter;
- en funktion `is_redundant(c: Comparator, f: Filter) -> bool`, som tjekker om en comparator er redundant iftėt filter;
- en funktion `add(c: Comparator, f: Filter) -> Filter`, som tilføjer `c` i enden af det netværk i filteret;
- en funktion `is_sorting(f: Filter) -> bool`, som tjekker om netværket i filteret er et sorteringsnetværk.

Modul `generate.py`

Dette modul indeholder kun en funktion `extend(w: list[Filter], n: int) -> list[Filter]`, som udvider alle filtre i en liste med en ikke-redundant comparator på alle mulige måder. Det betyder at, for hver filter og hver comparator, skal der testes om comparatoren er redundant ift. filteret, og hvis dette ikke er tilfældet, skal det udvidede filter være med i resultatet. Argumentet `n` fortæller, hvor mange kanaler der er i alt.

Modul `network_finder.py`

Dette modul inkluderer slutprogrammet. Den skal spørge brugeren om antallet af kanaler, og bruge generate-and-prune algoritme til at finde et sorteringsnetværk for input af den længde. Algoritmen starter med at danne en liste med et tomt filter. Et opdaterings trin består af:

- et opkald til `generate.extend()` for at udvide alle filtre i listen med alle ikke-redundante comparators;
- et opkald til `prune.prune()` for at fjerne ikke-relevante filtre fra listen.

Processen gentages så længe listen ikke indeholder et sorteringsnetværk. Når et sorteringsnetværk er fundet, skal programmet fortælle om dens længde og udskrive en Python implementering af den, som sorterer en liste gemt i en variabel `v` ved brug af hjælpevariablen `aux`.

Nedenunder vises et eksempel på, hvordan interaktionen med programmet kunne være. Bemærk, at resultatet er meget afhængig af hvordan de to funktioner `to_program` er implementerede i de moduler fra fase I.

```
What should the size of the network be? 2
Iteration 0
Size 1
Found a sorting network on 2 channels with size 1.
The Python implementation for sorting a list v of size 2 is:
if v[0] > v[1]:
    aux = v[0]
    v[0] = v[1]
    v[1] = aux
```

Kommentarer, forklaringer og tips til implementering

- Den algoritme, der skal implementeres, har en køretid der vokser supereksponentielt med antallet af kanaler. Det betyder, at dit program sandsynligvis ikke vil kun håndtere mere end 4 eller 5 kanaler :-)
- Det er mening, at der bruges kontraktbaseret programmering til udvikling af de her moduler. Det betyder nemlig, at alle antagelser om inputtet skal dokumenteres tydeligt, sådan at klienter til disse moduler kan være sikker på, at de opfylder dem.
- Der udleveres en implementering af moduler `comparator.py` og `network.py`, som kan bruges i stedet for dem, der udvikledes under fase I. Det er mening, at de nye moduler fungerer uanset hvordan `comparator.py` og `network.py` er implementeret.

Aflevering

Hver gruppe skal aflevere en zipfil med navn `groupXX.zip` (hvor `XX` er gruppens nummer), som indeholder:

- filer `generate.py`, `filter.py` og `network_finder`, som implementerer de kontrakter beskrevet tidligere;

- en fil `reportXX.pdf`, hvor `XX` er gruppens nummer.

Rapporten skal bl.a. beskrive, hvordan modulerne er designet – de valg, der er blevet taget ifm. definitionen af datatyper, samt alle relevante overvejlelser ifm. implementering af funktioner. Det er vigtigt at inkludere eksempler og forklare, hvordan modulerne er blevet testede. Rapporten skal også inkludere kildekode for de tre moduler som bilag. Grupper skal vælge en fasekoordinator, som skal identificeres tydeligt på rapportens forside.

Vigtige pointer.

- Hvis instruktioner om formatet til aflevering ikke følges, kan projektet blive afvist.
- Rapporten er basis for evalueringen.
- I takt med kursets formål er det mening at alle funktionalitet implementeres af gruppen. Det betyder at det ikke er tilladt at importere nogen af Pythons biblioteker, undtaget `functools` og `dataclasses`.