Programmering B — Rekursion og dynamisk programmering — Miniprojekt

Uffe Thorsen, utho@zbc.dk

1 Guldsmed & matematiker

En guldsmed modtager lænker til armbånd og halskæder i lange lænker lavet i et land med billigere arbejdskraft. For selv at kunne sælge halskæder og armbånd klipper han den lange lænke til i passende længder.

Vores guldsmed lever i en fiktiv verden med uendelig efterspørgsel på smykker, så uanset hvor mange armbånd han laver, kan han altid sælge dem for 100 kroner, og uanset hvor mange halskæder han laver kan han sælge dem til 300 kroner.

Guldsmeden overvejer hvordan han kan optimere den værdi han får for en lænke. En del af hans overvejelser er selvfølgelig at der skal bruges 13 led til at armbånd og 30 led til en halskæde.[1] Samtidig er der et led der går tabt hver gang der klippes. Et enkelt led (enten tabt fordi der er klippet i det eller til overs fordi der ikke længere er led nok til et helt armbånd eller en halskæde) kan genanvendes og har en værdi på 1 krone.

Tilfældigvis er guldsmeden også matematiker og ved at hvis han ser bort fra at han mister et led hver gang han klipper kan problemet ret let beskrives som et eksempel på Rod Cutting problemet, der kan beskrives med en forholdsvis simpel rekursionsligning, hvor n angiver længden af den oprindelige lænke og p_n angiver hvor meget der maksimalt kan tjenes på denne:

$$p_0 = 0$$

$$p_n = \max_{l \in \{1,13,30\}} p_{n-l} + v_l$$

Hvor $v_1 = 1$, $v_{13} = 100$ og $v_{30} = 300$.

Der betyder noget i stil med at hvis lænken har længde 0 kan der tjenes 0 kroner ($p_0 = 0$). For at finde ud af hvor meget han kan tjene med en lænke med n led, skal han tageste det højeste tal af $p_{n-1} + 1$ (han klipper et enkelt led af), $p_{n-13} + 100$ (han klipper nok led af til et armbånd) og $p_{n-30} + 300$ (han klipper nok led af til en halskæde).

[1] Nej, jeres underviser ved absolut intet om længden af smykker.

2 Opgave — Hjælp den stakkels guldsmed

Guldsmedens evner stopper desværre her, så han har bedt en programmør om hjælp.

Der er flere udfordringer i det ovenstående, der hver især er behandlet nedenfor. Rækkefølgen de er beskrevet i er ikke nødvendigvis den letteste rækkefølge at arbejde med dem.

2.1 Den fulde beskrivelse af problemet

Guldsmedens beskrivelse er en god start, men ser bort fra de led der mistes når der klippes. Lav en udvidet beskrivelse der tager højde for dette.

Skriv et kort specifikationsafsnit der beskriver det konkrete problem der skal løses så kort og præcist som muligt.

Beskriv den fulde udgave af rekursionen, enten som en rekursionsligning eller med ord.

2.2 En rekursiv løsning

Med en rekursiv beskrivelse af et problem er det typisk lettest at lave en rekursiv løsning. Design og implementér et program der løser guldsmedens problem.

Skriv kode samt et velformuleret designafsnit over dit arbejde. Tilføj et kort implementeringsafsnit.

I designafsnittet skal din løsning beskrives ved hjælp af pseudokode.

2.3 Dynamisk programmering

Ofte er en dynamisk programmeringsløsning langt hurtigere end en ren rekursiv. Undersøg hvor grænserne for hurtig løsning af den rekursive løsning er.

Design og implementér en dynamisk løsning af problemet.

Skriv kode samt et velformuleret designafsnit over dit arbejde. Tilføj et kort implementeringsafsnit.

I designafsnittet skal din løsning beskrives ved hjælp af pseudokode.

3 Bonusopgaver — Udfordr dig selv

Der er mange andre ting man kan arbejde med i forbindelse med ovenstående problem. Overvej at inkludere noget af følgende:

- Hvor lang tid tager det faktisk for de to løsninger?
- Kan man hjælpe guldsmeden yderligere, så han faktisk ved hvordan han skal klippe og ikke bare hvor meget han kan tjene?
- Kan vi generalisere problemet ovenfor så andre værdier kan håndteres, hvad skal der gælde for at problemet kan løses.
- Kan vi udføre test for forskellige værdier af n og forskellige priser? Er det muligt at skrive test til filer og så udføre testen på disse?
- Kan vi finde lignende problemer, der kan løses på tilsvarende vis?

4 Formalia

Der må arbejdes sammen i grupper på op til to personer. Der forventes i bruger omkring 8 timer til opgaven (heraf ligger de 4 i undervisningstiden).

De konkrete produkter der skal afleveres er:

- Kildekode.
- Rapport-dele:
  - Specifikation.
  - Desing.
  - Implementering (helt kort).

Der afleveres en enkelt `.zip`-fil, med sædvanlig struktur.