Institut for Matematik og Datalogi — 5. oktober 2004
Syddansk Universitet, Odense — LMF

DM02 Eksamen – Obligatorisk Opgave
Avanceret Strengsøgning

1 Problemet

I denne opgave skal I lave et program, som kan lave avanceret strengsøgning. Strengene kan f.eks. være tekster på nettet, DNA-sekvenser, proteinstrenge, eller noget helt fjerde (se afsnit 2).

Programmet skal søge i en streng T efter et mønster sammensat af flere delmønstre $p_1, p_2, \ldots, p_k$. Delmønstrene skal optræde i den angivne rækkefølge, men skal ikke nødvendigvis følge umiddelbart efter hinanden. Generelt er der en nedre grænse ℓ_i og en øvre grænse u_i for afstanden mellem p_i og p_{i+1}, $1 \leq i \leq k-1$. Hvis f.eks. $\ell_1 = 2$ og $u_1 = 5$, skal der optræde 2, 3, 4 eller 5 tegn imellem første og andet delmønster.

Input er en streng T, en sekvens $P = \langle p_1, p_2, \ldots, p_k \rangle$ af delmønstre, to sekvenser $U = \langle u_1, u_2, \ldots, u_{k-1} \rangle$ og $L = \langle \ell_1, \ell_2, \ldots, \ell_{k-1} \rangle$ af øvre og nedre grænser samt et heltal q mellem 1 og k. Opgaven er nu at afgøre, om T indeholder mindst q af delmønstrene, i den angivne rækkefølge og med indbyrdes afstande i overensstemmelse med L og U.

Hvis vi lader $\overline{q} = k - q$, kan man altså udelade op til $\overline{q}$ af delmønstrene. Hvis f.eks. $k = 4$ og $q = 3$, vil $\langle p_1, p_3, p_4 \rangle$ være en gyldig forekomst af det sammensatte mønster, hvis afstandene mellem p_1 og p_3 og mellem p_3 og p_4 er "rigtige". For at være rigtig skal afstanden mellem p_1 og p_3 ligge mellem $\ell_1 + \ell_2$ og $u_1 + u_2$. Afstanden mellem p_3 og p_4 skal selvfølgelig ligge mellem ℓ_3 og u_3. Mere generelt: hvis delmønstrene $p_i, p_{i+1}, \ldots, p_j$ er udeladt i det fundne mønster, skal afstanden mellem p_{i-1} og p_{j+1} ligge mellem $\ell_{i-1} + \ell_i + \ldots + \ell_j$ og $u_{i-1} + u_i + \ldots + u_j$.

For nemheds skyld antager vi, at mønstrene $p_1, \ldots, p_k$ hver især består af kun ét tegn. Vi antager også, at alle delmønstrene er forskellige, dvs. $p_i \neq p_j$, hvis $i \neq j$.

Man skal ikke finde alle forekomster af det sammensatte mønster — blot afgøre, om det forekommer i strengen T.

Eks. 1:

$T_1 =$ "datalogi er sjovt"
$T_2 =$ "algoritmer og datastrukturer er sjovest"
$P = \langle \mathrm{a}, \mathrm{i}, \mathrm{e}, \mathrm{o} \rangle$
$L = \langle 4, 5, 6 \rangle$
$U = \langle 11, 15, 7 \rangle$
$q = 3$

T_2 indeholder mindst en forekomst af det sammensatte mønster, da $4 + 5 \leq 25 \leq 11 + 15$ og $6 \leq 7 \leq 7$:

$$\underbrace{\underline{\text{a}}\text{lgoritmer og datastruktur}}_{25}\underbrace{\underline{\text{e}}\text{r er sj}}_{7}\underline{\text{o}}\text{vest}$$

Det er let at overbevise sig om, at mønsteret derimod ikke optræder i T_1: den eneste kombination af delmønstre, som opfylder afstandskravene, er $\langle \mathrm{a}, \mathrm{i} \rangle$, og den indeholder kun to delmønstre.

2 Mulige anvendelser

En algoritme, som kan løse ovenstående problem, har mange anvendelsesmuligheder. Forestil dig f.eks., at du gerne vil finde en bestemt bog på nettet. Du mener, den hedder “Introduction to Computational Molecular Biology”, men du er ikke helt sikker på, om ordet “Molecular” indgår, og måske hedder den i virkeligheden heller ikke noget med “Introduction to”, men rækkefølgen af ordene ligger fast. Da giver det mening at søge efter et mønster med mindst to af de fire delmønstre “Introduction to”, “Computational”, “Molecular” og “Biology”. I dette eksempel vil samtlige øvre og nedre grænser være 0, men man kan sagtens forestille sig eksempler, hvor det ikke vil være tilfældet.

Strengen T kunne også være en biologisk sekvens, f.eks. DNA eller protein. Mønstre i sådanne sekvenser består ofte af delmønstre, som kan optræde med varierende afstand, og af og til mangler nogle af delmønstrene, enten fordi de ikke findes i den virkelige sekvens, eller fordi der er fejl i de data, man har til rådighed.

I begge tilfælde vil strengen typisk være mange størrelsesordener længere end i Eks. 1. Da bliver det vigtigt at bruge en effektiv metode til at afgøre, om mønsteret optræder i strengen eller ej.

3 Overordnet fremgangsmåde

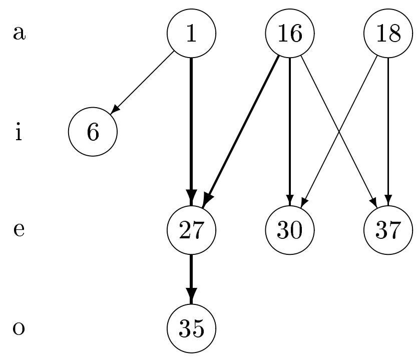
Problemet kan løses effektivt vha. en orienteret graf G, hvor hver knude svarer til en forekomst af et af de k delmønstre i T. Strengen søges igennem fra en ende af, og samtidig opbygges grafen. Ideen er følgende:

- Når man finder en forekomst af et delmønster p_j, tilføjer man en knude til grafen svarende til denne forekomst, medmindre man allerede har fundet bevis for, at forekomsten ikke kan komme til at indgå i det sammensatte mønster. Dvs.:
  - Hver gang et delmønster $p_j \in \{p_1, p_2, \ldots, p_{\overline{q}+1}\}$ findes i T, oprettes en knude v (husk, at $\overline{q} = k - q$).
  - Hver gang et delmønster $p_j \in \{p_{\overline{q}+2}, p_{\overline{q}+3}, \ldots, p_k\}$ findes, oprettes en knude, hvis der tidligere er fundet et delmønster, som kunne komme før p_j i det sammensatte mønster. Mere præcist oprettes der en ny knude v svarende til den netop fundne forekomst af p_j, hvis der allerede er oprettet en knude u svarende til en forekomst af et andet delmønster p_i, hvor
  $$\max\{1, j - \overline{q} - 1\} \le i < j \tag{1}$$
  og afstanden d mellem de to forekomster opfylder
  $$\ell_i + \ldots + \ell_{j-1} \le d \le u_i + \ldots + u_{j-1}. \tag{2}$$
  Bemærk, at afstanden d er antallet af tegn imellem de to forekomster. Dvs. hvis p_i optræder på plads x i T, og p_j optræder på plads y i T, da er $d = y - x - 1$.
- Man indsætter en kant mellem to knuder, hvis de tilsvarende forekomster af delmønstre kan efterfølge hinanden i det sammensatte mønster. Dvs. for en knude v opretter man en kant fra u til v for hver knude u, som opfylder betingelserne (1) og (2) ovenfor.

Det sammensatte mønster optræder i T, netop hvis der findes en vej af længde mindst $q - 1$, dvs. med mindst q knuder og $q - 1$ kanter, i G.

Eks. 2:

For Eks. 1 med T_2 ser grafen ud som vist nedenfor. Tallet i hver knude angiver den plads, delmønstret optræder på i T_2. For overskuelighedens skyld er knuderne tegnet i fire lag svarende til de fire delmønstre.



De to veje af længde to svarer til de to forekomster af mønsteret:

algoritmer og datastrukturer er sjovest
algoritmer og datastrukturer er sjovest

4 Programmet

Dit program skal have køretid $O(n^2)$, hvor n er længden af strengen T (det antages, at $k \leq n$).

Input er en fil med fem linier. Den første linie skal indeholde strengen T. Anden linie skal indeholde delmønstrene, som hver især består af blot ét tegn. Tredje linie skal indeholde de nedre grænser, adskilt af komma. Fjerde linie skal indeholde de øvre grænser, adskilt af komma. Femte linie skal indeholde tallet q. Input-filen svarende til Eks. 1 med T_1 ser altså sådan ud:

```
datalogi er sjovt
aieo
4,5,6
11,15,7
3
```

Når du skriver programmet, må du gerne antage, at input har det rigtige format. Dvs. det er i orden, at dit program fejler, hvis input har et andet format end det beskrevet ovenfor.

Output skal være 0 eller 1 og intet andet. For Eks. 1 med T_1 vil output være tallet 0, og for Eks. 1 med T_2 vil output være tallet 1.

Hovedklassen skal hedde StringSearch (to store S'er). Man skal kunne afvikle programmet på følgende måde:

```
java StringSearch <inputfil>
```

hvor filen <inputfil> indeholder input som beskrevet ovenfor.

4.1 Indlæsning af filer

Indlæsning af filer kan klares med

```
FileReader fr = new FileReader(fileName);
BufferedReader inFile = new BufferedReader(fr);
```

hvorefter man kan læse en linie ad gangen med

```
s = inFile.readLine();
```

hvor `s` er en `String`.

5 Test

Testen skal designes, inden du skriver programmet. Du skal give en overordnet beskrivelse af din teststrategi; dvs. du skal, uden at referere til programmet, forklare, hvilke specialtilfælde man kan komme ud for, og hvordan du tester, at programmet virker korrekt i alle tilfælde.

Hvis du gerne vil have en ide om, hvad vi senere vil sige til det output, dit program producerer, kan du skrive `DM02check`. Du skal, før du afgiver kommandoen, placere dig i det katalog, som alle dine oversatte `Java`-programmer ligger i. Så vil dit program blive kørt på testfilerne i `/home/IMADA/courses/dm02/Tests`. Testen indeholder kun ganske få testeksempler, så hvis der ikke findes fejl, betyder det ikke nødvendigvis, at dit program fungerer korrekt.

Det er en rigtig god ide at køre DM02check, inden du afleverer. På den måde sikrer du bl.a., at dit program bruger det korrekte input- og output-format. Programmer, som ikke kan testes med DM02check godkendes ikke; vi har ikke tid til at teste hvert enkelt program manuelt.

Programmet `DM02check` kan afbrydes med `Ctrl-c`.

6 Krav til rapport og program

Først og fremmest skal alle krav beskrevet i denne opgaveformulering naturligvis tilfredsstilles.

Programmet skal være velstruktureret og kommenteret i passende omfang. Rapporten skal indeholde en udskrift af hele programmet. Denne udskrift skal være identisk med det elektronisk afleverede program. Der skal være en beskrivelse af de væsentligste valg, der er truffet i forbindelse med implementationen, samt begrundelser herfor.

Man skal argumentere for, at den samlede køretid af programmet er $O(n^2)$, og det skal forklares, hvorfor fremgangsmåden beskrevet i afsnit 3 giver det ønskede resultat.

Desuden skal det forklares, hvordan programmet er afprøvet (se afsnit 5). Det er *ikke* tilstrækkeligt blot at have kørt `DM02check`, da denne test ikke er udtømmende.

Eventuelle mangler i program eller rapport skal beskrives.

Endelig skal rapporten underskrives.

7 Aflevering

Programmet skal afleveres elektronisk senest torsdag d. 28/10 kl. 12:00, og rapporten skal afleveres på IMADAs sekretariat senest tirsdag d. 2/11 kl. 12:00.

Den elektroniske aflevering foregår på følgende måde: Opret et katalog (directory), som indeholder alle dine JAVA-filer til opgaven og *intet andet*. Stil dig i dette katalog. Brug først `ls -la` for at sikre, at du står det rigtige sted, og at de rigtige filer er der. Afgiv derefter kommandoen `DM02aflever`.

D.v.s. gør følgende.

```
cd <opgave-katalog>
ls -la
DM02aflever
```

hvor `<opgave-katalog>` indeholder alle jeres JAVA-filer til opgaven og intet andet.

Bemærk, at du (inden afleveringsfristen) kan aflevere flere gange. Kun den sidste aflevering tæller (de andre slettes).

Rapporten skal afleveres på sekretariatet på Institut for Matematik og Datalogi. Denne opgaveformulering er vedhæftet en forside, som skal anvendes ved afleveringen. Husk for jeres egen skyld at få en kvittering (også vedhæftet) på, at I har afleveret.

Husk, at det er et krav, at det elektronisk afleverede program er præcis det samme som det, der afleveres en udskrift af i rapporten.

8 Evaluering

For at kunne gå til eksamen i DM02 til januar skal man have bestået den obligatoriske opgave. Har man tidligere lavet en obligatorisk DM02-opgave og fået den godkendt, er det tilstrækkeligt.

Der er i princippet kun to mulige bedømmelser: "godkendt" eller "ikke godkendt". Dog kan der i enkelte grænsetilfælde være mulighed for genaflevering.

9 Yderligere formalia

Den obligatoriske opgave er et individuelt eksamensprojekt. En overtrædelse af dette er eksamenssnyd og vil blive behandlet som sådan. Man har pligt til selv at beskytte sine noter og filer mod læsning af andre. Dette kan f.eks. gøres med `chmod 700 <opgavedirectory>`. Begge parter involveret i en eventuel plagiering kan blive holdt ansvarlige.

I må gerne snakke sammen om den overordnede løsning og lære af hinanden, men når I går i gang med at skrive ting ned, såvel program som rapport, skal I arbejde selvstændigt. Der er overraskende mange måder at skrive selv så lille et program på, og vi kan godt se, om man har arbejdet sammen — så lad være!

Programmet skal kunne køre på IMADAs maskiner. Man må gerne lave det hjemme, men det er helt på eget ansvar. Tekniske problemer derhjemme er ingen undskyldning. Man er også selv ansvarlig for, at ens filer kan flyttes uden problemer.

Der SKAL afleveres til tiden. Undtagelser herfra kræver særlige omstændigheder, som vil kunne holde i studienævnet. Vi har pligt til at behandle jer ens.

God arbejdslyst!

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

5. oktober 2004
LMF

DM02 eksamen, Efteråret 2004
Obligatorisk Opgave

Skriv tydeligt (maskinskrift eller blokbogstaver)

Navn: ....

Fødselsdato: ....

Brugernavn (login): ....

Instruktor	Martin	Rune	Steffen
Sæt kryds			

Besvarelsen omfatter .... nummererede sider.

Kvittering for aflevering af
obligatorisk opgave i DM02, efteråret 2004

Udfyldes inden afleveringen

Navn: ....

Fødselsdato: ....

Brugernavn (login): ....

Udfyldes af sekretariatet

Modtaget den .... kl. .... af ....
(dato) (klokken) (initialer)