Anvendelser av diskret matematikk

Noen eksempler

Håvard Utne Terland

April 15, 2024

RegEx - Regulære uttrykk i praksis

Regulære uttrykk kan brukes til å blant annet søke i tekst eller sjekke at tekst er på riktig form.

- Si at du designer en nettside hvor brukeren må skrive inn en dato.
- En dato på formen dd-mm-yyyy vil for eksempel i Python kunne matches med uttrykket

```
[0-9]{2}-[0-9]{2}-[0-9]{4}
```

RegEx - Regulære uttrykk i praksis

Regulære uttrykk kan brukes til å blant annet søke i tekst eller sjekke at tekst er på riktig form.

- Si at du designer en nettside hvor brukeren må skrive inn en dato.
- En dato på formen dd-mm-yyyy vil for eksempel i Python kunne matches med uttrykket

```
[0-9]{2}-[0-9]{2}-[0-9]{4}
```

- Vi kan da sjekke om en bruker har skrevet et gyldig input ved å sjekke om input ligger i det regulære språket beskrevet av uttrykket over. Dette kaller vi å *matche* utrykket.

RegEx - søking i tekst Vi kan bruke regulære uttrykk til å søke i tekst, som en "avansert ctrl+f".

- Vi leter da etter delstrenger i en tekst som matcher et gitt regulært uttrykk.
- I strengen `aabbbaa8a8` vil for eksempel uttrykket `a*8` matche fem ulike substrenger. Merk: hele strengen matcher *ikke* uttrykket.

RegEx - søking i tekst

Si at vi i en enorm tekstfil leter etter en viss feilmelding på formen `Error: ???? failed to compile`. Vi leter etter meldingen for å kunne finne ut hvilke program det er problemer med. Si at vi vet at programmet har et navn på fire tegn, hvorav de første to er sifre. Hvordan kan vi effektivt søke etter denne teskten?

RegEx - søking i tekst

Si at vi i en enorm tekstfil leter etter en viss feilmelding på formen `Error: ???? failed to compile`. Vi leter etter meldingen for å kunne finne ut hvilke program det er problemer med. Si at vi vet at programmet har et navn på fire tegn, hvorav de første to er sifre. Hvordan kan vi effektivt søke etter denne teskten?

```
Error: [0-9][0-9].{2} failed to compile
```

RegEx - søking i tekst

Si at vi i en enorm tekstfil leter etter en viss feilmelding på formen `Error: ???? failed to compile`. Vi leter etter meldingen for å kunne finne ut hvilke program det er problemer med. Si at vi vet at programmet har et navn på fire tegn, hvorav de første to er sifre. Hvordan kan vi effektivt søke etter denne teskten?

```
Error: [0-9][0-9].{2} failed to compile
```

Merk: Ganske avhengig av valg av syntax i den valgte regex-motoren. I python måtte vi erstattet mellomrom med '\s':

```
Error:\s[0-9][0-9].{2}\sfailed\sto\scompile
```

Grafer: korteste vandring

Utfordring: Si at du har fått i oppdrag å lage en ny app for en bussoperatør. De vil ha en funksjon som lar deg finne raskeste måte å komme fra et sted til et annet, ved bruk av utelukkende buss.

- En strategi: Vi representerer bussnettverket som en graf.

Grafer: korteste vandring

Utfordring: Si at du har fått i oppdrag å lage en ny app for en bussoperatør. De vil ha en funksjon som lar deg finne raskeste måte å komme fra et sted til et annet, ved bruk av utelukkende buss.

- En strategi: Vi representerer bussnettverket som en graf.
- Hvert buss-stopp tilsvarer en node.

Grafer: korteste vandring

Utfordring: Si at du har fått i oppdrag å lage en ny app for en bussoperatør. De vil ha en funksjon som lar deg finne raskeste måte å komme fra et sted til et annet, ved bruk av utelukkende buss.

- En strategi: Vi representerer bussnettverket som en graf.
- Hvert buss-stopp tilsvarer en node.
- For hver busslinje, tegn kanter mellom etterfølgende stopp til den linjen.
- På en gitt kant lagrer vi hvor lang tid i minutter det tar å reise

Grafer: korteste vandring

Utfordring: Si at du har fått i oppdrag å lage en ny app for en bussoperatør. De vil ha en funksjon som lar deg finne raskeste måte å komme fra et sted til et annet, ved bruk av utelukkende buss.

- En strategi: Vi representerer bussnettverket som en graf.
- Hvert buss-stopp tilsvarer en node.
- For hver busslinje, tegn kanter mellom etterfølgende stopp til den linjen.
- På en gitt kant lagrer vi hvor lang tid i minutter det tar å reise
- Gitt at man nå ønsker å bevege seg fra et stopp til et annet tilsvarer dette nå en vandring i denne grafen.
- Merk: Korteste vandring er alltid en *sti*; det er ikke noe poeng å gå fram og tilbake mellom to stopp.

Minimale spenntrær

- Si at du ønsker å koble sammen noen avsideliggende hytter med et veinett. Du vil bruke færrest mulig penger, men nok til at det mellom ethvert par av hytter finnes en vandring mellom dem - altså vil vi koble sammen hyttene.

Minimale spenntrær

- Si at du ønsker å koble sammen noen avsideliggende hytter med et veinett. Du vil bruke færrest mulig penger, men nok til at det mellom ethvert par av hytter finnes en vandring mellom dem - altså vil vi koble sammen hyttene.
- La H være mengden hytter. Da kan vi lage en graf $G = (H, E)$ hvor en kant indikerer at det er mulig å bygge vei. Til hver kant har vi også en pris $p\colon E \to \mathbb{N}$, hvor mange kroner det koster å bygge vei mellom de to hyttene koblet sammen av den kanten.

Minimale spenntrær

- Si at du ønsker å koble sammen noen avsideliggende hytter med et veinett. Du vil bruke færrest mulig penger, men nok til at det mellom ethvert par av hytter finnes en vandring mellom dem - altså vil vi koble sammen hyttene.
- La H være mengden hytter. Da kan vi lage en graf $G = (H, E)$ hvor en kant indikerer at det er mulig å bygge vei. Til hver kant har vi også en pris $p\colon E \to \mathbb{N}$, hvor mange kroner det koster å bygge vei mellom de to hyttene koblet sammen av den kanten.
- En sammenkobling av hyttene tilsvarer da en undergraf av G, si $T = (H, F)$, slik at T er sammenhengende. Prisen på å bygge T er da lik

$$\sum_{e \in F} v(e)$$

Minimale spenntrær

- Si at du ønsker å koble sammen noen avsideliggende hytter med et veinett. Du vil bruke færrest mulig penger, men nok til at det mellom ethvert par av hytter finnes en vandring mellom dem - altså vil vi koble sammen hyttene.
- La H være mengden hytter. Da kan vi lage en graf $G = (H, E)$ hvor en kant indikerer at det er mulig å bygge vei. Til hver kant har vi også en pris $p\colon E \to \mathbb{N}$, hvor mange kroner det koster å bygge vei mellom de to hyttene koblet sammen av den kanten.
- En sammenkobling av hyttene tilsvarer da en undergraf av G, si $T = (H, F)$, slik at T er sammenhengende. Prisen på å bygge T er da lik

$$\sum_{e \in F} v(e)$$

- Hvordan finne billigste slik T? Merk: Billigste T er nødvendigvis et tre (vi antar $T(e) > 0$ for alle $e \in E$).

Kruskals algoritme

Det viser seg at vi kan lage minimale spenntrær med en *grådig* algoritme: Ser vi for oss hytte-eksempelet vil følgende algoritme fungere:

- Bygg iterativt den billigste tilgjgenlige veien, så lenge den *ikke* skaper en sykel om vi tar den med.
- Når alle mulige kanter vil skape en sykel, har vi funnet et spenntre, og dette er et minimalt spenntre.