INF 4130 Oppgavesett 2, 13/09-2011

Vi skal starte litt mykt med noen korte oppgaver omkring algoritmers kjøretid og analyse av dette. Som dere vet bruker vi oftest *O*-notasjon (eller riktigere *asymptotisk notasjon*) til å angi kjøretiden. I notatet dere fikk på forelesningen (som også ligger på kurssidene) er det beskrevet fire ulike varianter av slik notasjon: *O*, Θ, Ω og *o.*

Oppgave 1

a) Vis at $n+1$ er $O(n)$.
b) Vis at $n \log n$ er $O(n^2)$.
c) Er $2^{n+1} = O(2^n)$?
d) Er $\frac{10n+16n^3}{2} = O(n^2)$?

Oppgave 2

a) Hva vet vi om en algoritmes kjøretid om denne er $O(n!)$?
b) Hva vet vi om en algoritmes kjøretid om denne er $\Omega(n)$?
c) Hva vet vi om en algoritmes kjøretid om denne er $\Theta(2^n)$?
d) Hva vet vi om en algoritmes kjøretid om denne er $O(n^2)$?
e) Utsagnet «Denne algoritmen har kjøretid minst $O(n^2)$.» kan virke litt pussig. Har det mening?

Så fortsetter vi litt med oppgaver om strengsøk, delvis hentet fra læreboka. Bruk gjerne litt tid på gruppa til å repetere/diskutere hvorfor/hvordan de ulike skift-strategiene til Knuth-Morris-Pratt og forenklet Boyer-Moore (Horspool) virker.

Oppgave 3 (Oppgave 20.3 i Berman & Paul)

Simuler CreateNext side 637-8, bruk patternet “abracadabra”.

Oppgave 4

Beregn arrayet Shift[a:z] for patternet P = "announce" - simuler CreateShift side 639.

Oppgave 5

Tegn ukomprimerte suffikstrær for strengene "BABBAGE" og "BAGLADY". Sjekk så om "BAG" er en felles substreng. Kan du klare deg med bare ett tre?

Ekstraoppgave

NB: Denne oppgaven krever litt kunnskap om regulære språk og NFA-er/DFA-er.

Som en generell problemstilling kunne vi tenke oss å søke etter en streng som passer med et gitt regulært uttrykk R, i en lengere streng S. Vi kan da i stedet si at vi forsøker å finne en delstreng fra starten av S som stemmer med det regulære uttrykket ".* R"

Her betyr "." et hviket som helst tegn i alfabetet, og stjernen betyr at det som står foran kan gjentas null eller flere ganger. Dermed betyr ".*" alle mulige strenger, inklusive den tomme.

Vi kan da løse søkeproblemet som følger: Vi lager først en NFA (ikkedeterministisk endelig automat) som tilsvarer ".*R" (dette er lett, enten intuitivt eller med en såkalt Thompson-konstruksjon), og lager denne om til en DFA (deterministisk endelig automat) med standard-algoritmen for NFA->DFA.

Denne DFA-en er svært lett å lage om til et program som leser S i lineær tid, og hver gang vi kommer til en sluttilstand i DFA-en vet vi at vi akkurat har lest noe som passer med R.

Spørsmål: Hvorfor er dette ikke en så rask algoritme som det i første omgang kan synes? Hva er det som begrenser hastigheten på denne algoritmen? Når vil den være rask?

[slutt]