INF 4130 Oppgavesett 3, 20/09-2011

Oppgave 1

1.1 Løs oppgave 20.19 (B&P), (a) er vist på forelesningen og kan vel bare repeteres, men løs (b).

1.2 Kjør algoritmen med noen enkle eksempler, f.eks. "algori" og "logari", og med to helt like ord.

1.3 Vis at man kan programmere algoritmen slik at man bare tar plass til en kolonne (eller rad), samt en enkel-variabel.

Oppgave 2

2.1 Løs oppgave 20.20 (B&P) Hint: Forsøk å gjøre lure forandringer i initialiseringen.

2.2 Kjør algoritmen med noen eksempler, f.eks.: P="hvis" og T="haiehus" med K=2, og "lag" og "varelager" med K=1. (Haiehuset er selvfølgelig der hvor hushaiene bor……)

Oppgave 3

Vi skal løse pengevekslerens problem med dynamisk programmering: Problemet består i å gi igjen K øre med så få mynter som mulig. La $\{v_1, v_2, \ldots, v_n\}$, være myentenes verdier. Vi kan f.eks ha: $v_1 = 25$, $v_2 = 10$, $v_3 = 5$, $v_4 = 1$. Vi antar verdiene er heltallige og at $v_{i-1} < v_i$. Videre antar vi at $v_n = 1$, slik at det alltid finnes en løsning. Og selvfølgelig antar vi at vi har uendelig med vekslepenger av alle aktuelle verdier.

For pengesystemet beskrevet over vil en grådig strategi virke. (Og for interesserte kan det jo være en passelig utfordring å bevise nettopp det.) MEN hvis vi fjerner femøringen, havner vi i trøbbel: da vil en grådig algoritme som skal gi igjen 30 øre gi en 25-øring og fem 1-øringer, seks mynter totalt, mens det optimale er å gi tre 10-øringer. Dynamisk programmering finner en optimal løsning for alle pengesystemer, også de hvor en grådig algoritme feiler.

La $C[j]$ være det antall mynter i en løsning som gir igjen j øre. Hvis en løsning bruker en mynt med verdi v_i, vil $C[j] = C[j - v_i] + 1$.

a) Sett opp en rekursiv formel for verdien av en optimal løsning.
b) Skriv (pseudo)kode for en algoritme basert på formelen i a).
c) Kjør algoritmen med $v_1 = 30$, $v_2 = 24$, $v_3 = 12$, $v_4 = 6$, $v_5 = 3$, $v_6 = 1$, og K = 48. (Så gæernt var det faktisk i England for ikke lenge siden!) [Grådig-løsningen er for øvrig 30+12+6.]
d) Hva blir kjøretiden til algoritmen?

Oppgave 4 (om man har tid)

Se på det å bruke memoisering, altså at man har en tabell som forvanlig dyn. progr., men at man skriver algoritmen rekursivt ut fra utfyllings-formlen. Trikset er da at hvert kall da først slår opp i tabellen, og om svaret er beregnet tidligere ligger det her og kan bare leveres. Om det ikke er beregnet gjøres det nå, og man både setter ned og leverer svaret.

a) Skriv en slik algoritme for spøsmålet i oppgave 20.19.
b) Kjør den pr hånd på noen enkle eksempler (f.eks. de angitt i oppgave 20.19 over), og se hvor mange verdier du slipper å beregne.

[slutt]