Institut for Matematik og Datalogi
Syddansk Universitet, Odense

13. november 2003
finie02@imada.sdu.dk

DM02 – Opgaver fra ugeseddel 9

Øvelsesopgaver 4/11, 5/11 og 6/11

1. Cormen 22.2-1

   Vis alle d og π værdier der resulterer fra kørslen af breadth-first søgningen på grafen vist i figur 22.2-a (s.528), med knude 3 som startknude.

   Først initialiseres knuderne

   $$d \leftarrow \infty, \pi \leftarrow NIL$$

   Herefter initialiseres startknuden

   $$d \leftarrow 0, \pi \leftarrow NIL$$

   Efter kørslen ser tabellen således ud:

	1	2	3	4	5	6
d	∞	3	0	2	1	1
π	NIL	4	NIL	5	3	3

2. Cormen 22.2-3

   Hvad er køretiden for BFS hvis inputgrafen er repræsenteret ved en adjacensmatrice og algoritmen er modificeret til at håndtere dette input?

   Modifikationen sker i hvad der svarer til pseudokodens linie 12 (s.532):

   for each vertex $v \in Adj[u]$

   Når algoritmen kører på en adjacens**liste** skal denne løkke blot gå et enkelt skridt videre i den hægtede liste for knuden u, hvorimod, når vi bruger en adjacens**matrice**, skal vi lede efter den næste nabo i rækken for u.
   Hver række har længden $|V|$, så når vi har fundet alle naboer for u har vi kigget på $|V|$ felter. Altså for hver $u \leftarrow$ DEQUEUE(Q), undersøger vi $|V|$ elementer.
   Ialt, efter at have kørt DEQUEUE højst en gang for hver knude, har vi i worstcase undersøgt alle felter i adjacensmatricen, dvs. $|V|^2$.

   Den samlede køretid bliver (initialisering + nabosøgning) = $O(V+V^2) \in O(V^2)$ (se evt. analysen af køretid s. 534)

3. Cormen 22.2-5

   Giv et eksempel på en orienteret graf $G = (V, E)$, en startknude $s \in V$, og et sæt af kanter $E_\pi \in E$, så for hver knude $v \in V$ er den unikke sti i grafen $(V, E_\pi$ fra s til v en korteste sti i G, selvom kanterne E_π ikke kan opnås ved BFS på G uanset knudernes rækkefølge i adjacenslisten.

   (udeladt med vilje)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

13. november 2003
finie02@imada.sdu.dk

4. Cormen 22.2-5

   Good guys vs. Bad guys
   Givet n professionelle brydere og en liste r af rival-par, lav en $O(n+r)$ algoritme som undersøger om bryderne kan opdeles i to lejre således at der ikke findes rivalpar inden for den samme lejr, og i givet fald returnere opdelingen.

   Problemet kan omformuleres til et grafproblem $G = (V, E)$:

   - Hver bryder v repræsenteres af en knude i grafen $v \in V$
   - Hver rivalpar af brydere (u, v) repræsenteres ved en kant (u, v)
   - En bryders tilhørsforhold angives ved farvning af knuden i en af to farver f.eks. sort og hvid.
   - Hvis det er muligt at farvelægge grafen således at ingen knude er adjacent til en knude af samme farve svarer det til at ingen bryder er rival med en bryder fra sin egen lejr.

   Algoritme

   (a) Lav en adjacensliste repræsentation af grafen ud fra rivallisten r.

   (b) Lav en bredde-først søgning fra en tilfældig knude $s \in V$ hvor knuderne farvelægges efter afstand d til roden: if $d[u]$ lige then $color[u]$=sort else $color[u]$=hvid.
   Hvis der under gennemløbet findes en knude v med en naboknude u som allerede er farvet så $color[u] == color[v]$, returneres False, ellers returneres True.

   Køretid
   Adjacenslisterepræsentationen kan oprettes i tid $\Theta(r)$ og bredde-først søgningen kører i $O(n + r)$. Samlet har vi køretid $O(n + r)$.

5. Eksamen Januar 1996, opg. 1

   Bæltegrafer
   Bæltegrafer beskrives ved n: antal knuder, m: antal kanter og k: antal ringe.

   (a) Hvad er m udtrykt ved n og k i en vilkårlig bæltegraf?
   Betragter kanter der udgår fra højresiden af en knude

   - $n - 1$ knuder har mindst én kant mod højre
   - k knuder har to kanter mod højre
   - ingen knuder har mere end to kanter mod højre

   $$m = (n - 1) + k$$

   (b) Forklar med ord hvordan man altid kan finde den korteste vej mellem endepunkterne u og v i en bæltegraf i tid $O(n)$.
   Vi kan beregne længden af den korteste vej til en samleknude (en knude med fire kanter) ved at følge de to stier hver vej rundt i ringen fra u. For hvert skridt tilføjer vi en kant til den sti som indtil videre har den korteste vej (laveste samlet vægt). Hermed finder vi korteste vej til samleknuden ved at kigge på hver kant i ringen højst én gang.
   Samleknudens resultat kan vi bruge til at beregne længden af korteste vej til næste samleknude og så fremdeles indtil vi når slutknuden v.

Køretiden er lineær i antallet af kanter $O(m) \in O(n)$, for $k < n$ jf. opgave a.

6. Eksamen Januar 1996, opg. 3

Invester p kroner i d virksomheder
Meget lig opgaven med skakbrættet fra sidste uge, får vi givet en funktion $udbytte(p, c)$ som returnerer forventet udbytte ved at investere p kroner i virksomhed c. $udbytte(p, c)$ kører i konstant tid.
Opgaven giver pseudokode for en funktion $invester(p, c)$.

(a) Forklar kort i ord hvad $invester(p, c)$ beregner.

Virksomhederne er nummereret $1 \ldots d$
$Invester(p, c)$ beregner det maksimale udbytte ved a tinvestere p kroner i virksomhederne $c \ldots d$.
For hvert muligt investeringsbeløb, dvs. $i = 0$ til p, beregnes udbyttet for at investere i kroner i virksomhed c og investere resten af pengene i virksomhederne $c + 1 \ldots d$.
Det maksimale udbytte returneres.

(b) Betragt $Invester(p, 1)$. Gør rede for at de samme resultater beregnes flere gange.

(tegn et rekursionstræ for de første par rekursive kald fra $Invester(p, 1)$ indtil det er tydeligt at de samme kald bliver lavet flere gange)

(c) Konstruer en bedre løsning ved at anvende dynamisk programmering.
Hvad bliver kompleksiteten udtrykt ved p og d?

Vi opretter en tabel i størrelsen $(p + 1) \cdot d$ til at holde alle delresultater af $Invester(p, c)$.

Bemærk: Beregningen af feltet $[p, c]$ (dvs. $Invester(p, c)$) er afhængig af felterne $[i, c + 1]$, hvor $0 \leq i \leq p$.

Iterativ algoritme
Vi kan udregne en søjle af gangen uden at genregne delresultater hvis vi starter med udregning af søjlen $[i, d]$, hvor $0 \leq i \leq p$, og fortsætter søjlevis til vi når $[p, 1]$.

Memoized algoritme
Vi kan også vælge at bruge samme rekursive algoritme som angivet i opgaven og blot modificere den til at gemme sit resultat i tabellen og bare returnere tabelværdien hvis den allerede er beregnet.

Køretid
Vi laver en beregning for hvert felt $[p, c]$ i tabellen. Beregningen finder max ved at lave opslag i p felter i forrige søjle sammen med p kald til $udbytte$ som kører i konstant tid ($\Theta(1)$).

Ialt operationer $O((p + 1) \cdot d \cdot p \cdot \Theta(1)) \in O(p^2 d)$