Linköpings Universitet
Institutionen för datavetenskap
Peter Jonsson

TDDD20 Konstruktion och analys av algoritmer

Hemtentamen. **Exempel.**

Hjälpmedel: Kursboken "Introduction to Algorithms" (Cormen, Leiserson, Rivest & Stein). Ordlista.

Poäng: Totalt kan 40 poäng erhållas. För godkänt krävs ca 16 poäng.

Jourhavande lärare: Peter Jonsson, `peter.jonsson@liu.se`, tel 013 28 24 15.

Allmänt: Skriv läsligt. Onödigt komplicerade lösningar kan leda till poängavdrag. Bristfälligt motiverade lösningar leder ofelbart till poängavdrag. Om inte annat anges skall presenterade algoritmer vara körbara på en processor av standardtyp (vilket utesluter exempelvis kvantdatorer och massiv parallellism). Slumpbitar får användas av algoritmer vid behov och de förutsätts kunna genereras i konstant tid.

Uppgifterna i tentamen är inte ordnade efter svårighetsgrad.

Lycka till!

Peter

Uppgift 1. (8p)
Ett linjärt ekvationssystem är *homogent* om varje ekvation har en nolla i högerledet. Ett exempel är

$$\begin{array}{rcrcrcl} 2x & + & 3y & - & 3z & = & 0 \\ & & y & + & z & = & 0 \\ x & - & 4y & - & z & = & 0 \end{array}$$

Man kan notera att varje sådant system är lösbart eftersom noll-lösningen är en giltig lösning. Konstruera en polynomisk algoritm som testar om ett linjärt homogent ekvationssystem har en heltalslösning skild från noll-lösningen. Antag för enkelhets skull att alla koefficienter är heltal.

Uppgift 2. (8p)
Antag att vi har en mängd S av n intervall $[a_i, b_i]$ på tallinjen. Två intervall är *disjunkta* om de inte har någon gemensam punkt. Konstruera en algoritm som i $O(n \log n)$ tid beräknar det maximala antalet disjunkta intervall i S. Antag för enkelhets skull att intervallens start- och slutpunkter alltid är positiva heltal. Aritmetiska operationer mellan heltal får antas gå att beräkna i $O(1)$ tid.

Uppgift 3. (8p)
En oriktad graf $G = (V, E)$ är 3-färgningsbar om och endast om det existerar en funktion $f : V \to \{0, 1, 2\}$ med följande egenskap

om $\{v, w\} \in E$ så gäller $f(v) \neq f(w)$.

Konstruera en randomiserad algoritm som tar en oriktad graf $G = (V, E)$ och i polynomisk tid konstruerar en graf $G' = (V, E')$ med följande tre egenskaper.

1. $E' \subseteq E$,
2. det förväntade antalet bågar i E' är lika med eller större än $2|E|/3$ och
3. G' är 3-färgningsbar.

Uppgift 4. (8p)
En *hamiltonväg* i en graf är en väg som besöker varje nod exakt en gång. Konstruera en polynomisk algoritm som avgör om riktade acykliska grafer innehåller hamiltonvägar.

Uppgift 5. (8p)
Låt F vara en godtycklig satisfierbar 3SAT-formel. Låt V vara mängden variabler som F innehåller. Man säger att $f : V \to \{0, 1\}$ är en *komplementär modell* om f är en satisfierande tilldelning och den "motsatta" tilldelningen $f'(x) = 1 - f(x)$ också är satisfierande.

Antag att det finns en algoritm A som testar satisfierbarhet för 3SAT-formler i $O(c^n)$ tid där n är antalet variabler och $c > 1$ någon konstant. Konstruera en algoritm som gör följande:

- Om F saknar komplementär modell så svarar algoritmen "nej".
- Om F har en komplementär modell så returnerar algoritmen en komplementär modell f.

Algoritmen ska gå i $O(c^n \cdot p(n))$ tid där n är antalet variabler i den givna formeln och p är ett fixt polynom. Algoritmen A får naturligtvis användas som en subrutin i lösningen.

Lösningsförslag

Uppgift 1.
Låt $Ax = 0$ vara ett godtyckligt homogent och linjärt ekvationssystem där $x = (x_1, \ldots, x_n)$. Betrakta följande algoritm.

Lös följande $2n$ linjärprogram

$$\begin{array}{llll} \max x_1 & \max x_2 & \ldots & \max x_n \\ Ax = 0 & Ax = 0 & \ldots & Ax = 0 \end{array}$$

$$\begin{array}{llll} \min x_1 & \min x_2 & \ldots & \min x_n \\ Ax = 0 & Ax = 0 & \ldots & Ax = 0 \end{array}$$

Om alla optimum är lika med 0 så svara "nej" och annars svara "ja".

Denna algoritm är polynomisk eftersom linjärprogrammen kan lösas i polynomisk tid om man exempelvis använder Karmarkars algoritm.

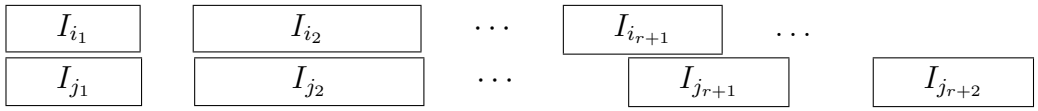
Om samtliga linjärprogram har optimalt värde 0 så kan vi omedelbart dra slutsatsen att systemet $Ax = 0$ endast har noll-lösningen. Därmed är algoritmens svar korrekt då den svarar "nej". Antag istället att något av programmen har ett optimalt värde som inte är 0. Detta innebär att programmet har lösningen $(a_1, \ldots, a_n)$ där minst ett a_i är nollskilt. Notera att om $(a_1, \ldots, a_n)$ är en lösning till $Ax = 0$ så är $(d \cdot a_1, \ldots, d \cdot a_n)$ också en lösning för godtyckligt d. Eftersom koeffecienterna i A är heltal så innehåller en optimal lösning endast rationella tal, dvs. $(a_1, \ldots, a_n) = (\frac{b_1}{c_1}, \ldots, \frac{b_n}{c_n})$ där b_i, c_i är heltal och alla c_i är nollskilda. Detta betyder att $(a_1 \cdot C, \ldots, a_n \cdot C)$ är en nollskild heltalsvektor om $C = c_1 \cdot \ldots \cdot c_n$. Det följer att algoritmens svar är korrekt då den svarar "ja".

Uppgift 2.
Antag att intervallen $I_1, \ldots, I_n$ är sorterade så att $b_1 \leq b_2 \leq \ldots \leq b_n$ och betrakta algoritmen

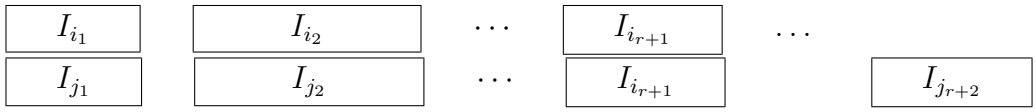
$A = \emptyset$
for $j = 1$ **to** n
    if I_j är disjunkt med alla intervall i A **then** $A = A \cup \{I_j\}$
return $|A|$

Att sortera intervallen tar $O(n \log n)$ tid och algoritmen kan implementeras så att den går i $O(n)$ tid: Kom hela tiden ihåg vilket intervall $I_* = [a_*, b_*]$ som senast lades till i A, och testet i varv j blir då helt enkelt om $a_j > b_*$.

Vi visar korrekthet genom ett motsägelsebevis. Antag att algoritmen inte är korrekt. Låt $I_1, \ldots, I_n$ vara en mängd intervall där algoritmen misslyckas med att producera en optimal lösning. Låt $I_{i_1}, \ldots, I_{i_k}$ vara den lösning som algoritmen ger. Vi antar att dessa intervall är sorterade så att $b_{i_1} \leq b_{i_2} \leq \ldots \leq b_{i_k}$. Låt $I_{j_1}, \ldots, I_{j_m}$ vara en optimal lösning med $m > k$. Bland alla optimala lösningar väljer vi $I_{j_1}, \ldots, I_{j_m}$ så att $I_{i_1} = I_{j_1}, I_{i_2} = I_{j_2}, \ldots, I_{i_r} = I_{j_r}$ och r är så stort som möjligt. Vi jämför de två lösningarna.



Notera att $b_{i_{r+1}} < b_{j_{r+1}}$ på grund av det giriga valet i algoritmen: I_{i_r} är det intervall med minst b_{i_r} som kan kombineras med intervallen $I_{i_1}, \ldots, I_{i_{r-1}}$. Detta innebär att den optimala lösningen kan byggas om på följande sätt: Intervallet $I_{j_{r+1}}$ kan tas bort och ersättas med $I_{i_{r+1}}$.



Den nya lösningen är fortfarande optimal men den motsäger valet av r. Vi har därmed visat att algoritmens lösning är optimal.

Uppgift 3.
Antag att grafen $G = (V, E)$ innehåller noderna $v_1, \ldots, v_n$. Tilldela varje nod något av talen $0, 1, 2$ med lika sannolikhet.[1] Låt E' innehålla bågen $\{v, w\} \in E$ om och endast om v och w tilldelats olika färger. Denna algoritm går i polynomisk tid.

Vi ser att (V, E') uppenbarligen är 3-färgningsbar och att $E' \subseteq E$. Vi uppskattar nu storleken på E'. Antag att bågarna i E är $e_1, \ldots, e_m$. Låt den stokastiska variabeln $\mathbf{X}_i$ vara 1 om bågen e_i finns i E', och låt variabeln vara 0 annars. Då är det förväntade antalet bågar lika med

$$E(\mathbf{X}_1 + \ldots + \mathbf{X}_m) = /\text{linearitet}/ = E(\mathbf{X}_1) + \ldots + E(\mathbf{X}_m).$$

Sannolikheten för att en båge är med i E' är $6/9 = 2/3$ eftersom en bågens noder kan färgas på totalt nio sätt och sex av dessa är en giltig färgning.

[1] Hur gör man detta om man bara har tillgång till slumpbitar?

Eftersom $\mathbf{X}_i$ är en 0/1-variabel så gäller $E(\mathbf{X}_i) = 2/3$. Därav följer att $2/3 \cdot |E|$ bågar förväntas finnas i E'.

Uppgift 4.

algoritm $A(G)$

Sortera grafen topologiskt med resultatet $v_1 \to v_2 \to \ldots \to v_n$.

Kontrollera att det finns en båge från v_i till v_{i+1} för alla $1 \leq i \leq n-1$

Om så är fallet svara "ja" och annars "nej".

Algoritmen kan enkelt implementeras med kända algoritmer så att den går i polynomisk tid.

Antag att algoritmen svarar "ja". Då är $v_1 \to v_2 \to \ldots \to v_n$ en hamiltonväg i G.

Antag att G innehåller en hamiltonväg H. Då finns bara ett sätt att topologiskt sortera G. Antag (med sikte på en motsägelse) att det finns två olika topologiska sorteringar T_1, T_2 av G. Då finns två noder u, v sådana att u är före v i T_1 men v är före u i T_2. Om u är före v i H så finns en väg från u till v i G och det motsäger existensen av T_2. Om v är före u i H så finns en väg från v till u i G och det motsäger existensen av T_1. Alltså kan G bara topologiskt sorteras på ett sätt T. Vidare så "följer" hamiltonvägen nodernas ordning i T via samma resonemang. Detta medför att algoritmen svarar "ja".

Uppgift 5.

Antag att $F = C_1 \wedge \ldots \wedge C_m$. Bilda formeln F' genom att vända polariteten på literalerna i klausulerna. Exempel: Om $F = (x \vee \neg y \vee z) \wedge (\neg x \vee z \vee w)$ så är $F' = (\neg x \vee y \vee \neg z) \wedge (x \vee \neg z \vee \neg w)$. Notera att formeln $G = F \wedge F'$ är satisfierbar om och endast om F har en komplementär modell. Vidare gäller att varje modell till G är en komplementär modell till F. Vi kan enkelt kontrollera om G är satisfierbar med algoritmen A. Om så inte är fallet så vet vi att F saknar komplementära modeller. Antag istället att G är satisfierbar—då måste vi hitta en modell till G. Detta kan man göra med följande algoritm.

algoritm $B(G)$ där G är satisfierbar och innehåller variablerna $x_1, \ldots, x_n$.

for $i = 1$ **to** n

    if $A(G \wedge (x_i))$ ="ja" **then** $G = G \wedge (x_i)$ **else** $G = G \wedge (\neg x_i)$

returnera de n sista klausulerna i G (och notera att de beskriver en modell)

Vi gör totalt $n+1$ anrop till algoritmen A och varje formel som A appliceras på innehåller n variabler. Hela proceduren tar sålunda $O(c^n \cdot p(n))$ tid för ett fixt polynom.