Algoritmer och komplexitet
hösten 2007

Mästarprov 1: Algoritmer

Mästarprovet ska lösas **individuellt** och redovisas både skriftligt och muntligt. Inget samarbete är tillåtet, se vidare hederskodexen.

Skriftliga lösningar ska lämnas senast **20 nov** på föreläsningen eller senast klockan 10.00 samma dag i kursens inlämningslåda på studentexpeditionen på Osquars backe 2 plan 2. Det är viktigt att du lämnar in i tid! Se till att spara en kopia av dina lösningar så att du kan läsa på inför den **muntliga redovisningen** som kommer att ske senare samma vecka för någon av lärarna. Boka tid för en femton minuters muntlig redovisning med kommandot `bok new algokomp07`
Redovisningstider kommer att läggas upp senast den 19 nov.

Mästarprovet är ett obligatoriskt och betygsatt moment i kursen. Det består av tre uppgifter med olika svårighetsgrad. För godkänt d.v.s. betyg E krävs helt rätt på en av uppgifterna. Helt rätt på två av uppgifterna ger betyg C och alla rätt ger betyg A. Läs mer om betygskriterier och slutbetyg i kurs-PM eller på kursens webbsida.

1. **Jobbplanering (mindre svår)**
   Vi har en uppsättning av n jobb med start- och sluttider angivna som $[s_i, f_i]$. Varje jobb tar alltså tid $t_i = f_i - s_i$ att utföra. Alla tider antas vara positiva heltal. Vi skall göra ett urval av intervallen så att inga av dem överlappar varandra. Vi tillåter dock att ett jobb påbörjas i samma ögonblick som ett annat slutar. Så exempelvis överlappar $[6, 10]$ och $[10, 13]$ inte varandra. Vi vill hitta ett urval av jobb som går att utföra tillsammans, d.v.s. inte överlappar varandra och som dessutom utnyttjar tiden så väl som möjligt. Det betyder att vi vill maximera den sammanlagda totaltiden vi arbetar.

   Vi löser problemet på följande sätt. Vi tänker oss att vi sorterat intervallen efter stigande sluttid: $f_1 \leq f_2 \leq \ldots f_n$. Vi sätter sedan $M[i] =$ den bästa totaltiden om jobb $1, \ldots, i$ får användas. Då får vi följande rekursionsformel

   $$M[i] = \max(M[i-1], M[k] + t_i)$$

   där k är det största k så att $f_k \leq s_i$.

Utgående från denna rekursion skall du implementera en dynamisk programmeringsalgoritm i pseudokod och analysera algoritmens tidskomplexitet. Beskriv också hur du kan modifiera algoritmen så att vi får reda på vilka jobb som ingår i det optimala urvalet. Ange även tidskomplexiteten för denna modifierade algoritm.

2. **Att hitta delsträngar (medelsvår)**
   Vi har en sträng $x = x_1x_2 \dots x_n$ där varje element x_i är en symbol i ett ändligt alfabet A. Låt nu $y = y_1y_2 \dots y_k$ vara en annan sträng över samma alfabet med $k \leq n$. Vi säger att y är en *delsträng* till x om det finns en talföljd $i_1, i_2, \dots, i_k$ med $i_1 < i_2 < \dots < i_k$ så att $x_{i_1} = y_1, x_{i_2} = y_2, \dots x_{i_k} = y_k$. Din uppgift är nu att konstruera en algoritm som givet en sträng x av längd n och en sträng y av längd $k \leq n$ avgör om y är en delsträng till x eller inte. Alfabetet A antas vara givet och fixt. Ange algoritmen i pseudokod. Analysera din algoritm med avseende på korrekthet och tidskomplexitet. Tidskomplexiteten skall vara polynomiell i n och k och alfabetets storlek. Använd någon av de metoder som presenterats i kursen.

3. **Positiva grafvägar (svår)**
   Vi har en riktad graf G där alla kanter har vikt 1 eller -1. Vi har också två speciella noder s givna. Vi spelar nu ett slags spel på följande sätt. Vi startar i s och går om möjligt över en framåtriktad kant med vikt 1. Vi fortsätter och går via framåtriktade kanter. Varje gång vi går över en kant med vikt 1 får vi en krona. Varje gång vi går över en kant med vikt -1 betalar vi en krona. Vi får aldrig hamna på negativt konto d.v.s. summan över de kanter vi gått över måste vara ≥ 0. Vad spelet går ut på är att försöka ta oss från s till t. Konstruera en algoritm som avgör om det är möjligt att ta sig från s till t. Ange algoritmen i pseudokod. Analysera den med avseende på korrekthet och tidskomplexitet.