Algoritmer och Komplexitet ht 09. Övning 5
Flöden. Reduktioner

Förändrat flöde

a) Beskriv en effektiv algoritm som hittar ett nytt maximalt flöde om kapaciteten längs en viss kant *ökar* med en enhet.
   Algoritmens tidskomplexitet ska vara linjär, alltså $O(|V| + |E|)$.

b) Beskriv en effektiv algoritm som hittar ett nytt maximalt flöde om kapaciteten längs en viss kant *minskar* med en enhet.
   Algoritmens tidskomplexitet ska vara linjär, alltså $O(|V| + |E|)$.

Snabb lådpackning *Lådpackningsproblemet* (*the bin packing problem* på engelska) är följande problem. Du får n stycken metallprylar som var och en väger mellan 0 och 1 kg. Du får också ett antal stora men sköra lådor. Målet är att hitta det minsta antal lådor som behövs för att förvara alla n metallprylarna utan att någon låda innehåller mer än 1 kg.

Detta är ett känt problem som är svårt att lösa exakt (det är ett så kallat *NP-fullständigt* problem). Därför ska vi nu nöja oss med att hitta en lösning som inte är optimal med hjälp av följande enkla algoritm:

Anta att både metallobjekten och lådorna är numrerade från 1 till n. Ta en metallpryl i taget (i tur och ordning) och stoppa den i den första låda som rymmer den (dvs den låda med lägst nummer som inte blir för tung om man stoppar i prylen).

Din uppgift är att beskriva hur denna algoritm kan implementeras så att den går i tid $O(n \log n)$ (i värsta fallet och med enhetskostnad). För att uppnå detta kommer du att behöva konstruera en heapliknande datastruktur i vilken du snabbt kan söka upp den första låda som rymmer den aktuella prylen.

Reduktion som ger negativt resultat I förra uppgiften beskrevs en algoritm som hittar en approximativ lösning till lådpackningsproblemet. Algoritmen fungerar så att den placerar varje pryl i den första låda som rymmer den. Uppgiften på övning 4 var att implementera algoritmen i tid $O(n \log n)$. Visa att $\Omega(n \log n)$ är en undre gräns för algoritmens tidskomplexitet.

Reduktion som ger positivt resultat Ett ofta användbart sätt att lösa problem på är att hitta en reduktion till ett problem som man redan vet hur man ska lösa. Du ska nu använda denna metod för att lösa kantsammanhängandegradsproblemet som definieras på följande sätt.

> INMATNING: En sammanhängande oriktad graf $G = (V, E)$ och ett positivt heltal K mellan 1 och $|V|$.
> PROBLEM: Är det tillräckligt att ta bort K kanter från grafen G för att göra den osammanhängande (dvs uppdelad i flera sammanhängande komponenter)?

Reduktioner mellan besluts-, optimerings- och konstruktionsproblem Anta att algoritmen GraphColouring(G,k) på tid $T(n)$ (där n är antalet hörn i G) svarar 1 omm hörnen i G kan färgas med k färger utan att någon kant har likfärgade ändpunkter.

a) Konstruera en algoritm som givet en graf G med n hörn bestämmer det minimala antalet färger som behövs för att färga G. Tiden ska vara $O(\log n \cdot T(n))$.

b) Konstruera en algoritm som givet en graf G med n hörn färgar hörnen med minimalt antal färger i tid $O(P(n)T(n))$ där $P(n)$ är ett polynom.
