Viggo Kann 2021-08-26

DD2350 Algoritmer, datastrukturer och komplexitet

Uppgifter till övning 10

NP-fullständiga problem

Konstruera kappsäckslösning Kappsäcksproblemet är ett välkänt NP-fullständigt problem (se föreläsning 21). Indata är en mängd P med prylar med vikt w_i och värde v_i, en kappsäcksstorlek S och ett mål K. Frågan i kappsäcksproblemet är ifall det går att välja ut prylar av sammanlagt värde minst K så att deras sammanlagda vikt är högst S. Alla tal i indata är ickenegativa heltal.

Anta nu att vi har en algoritm A som löser ovanstående beslutsproblem. Konstruera en algoritm som med hjälp av anrop till A löser det konstruktiva kappsäcksproblemet, det vill säga med samma indata som A dels besvarar kappsäcksproblemet och dels, ifall svaret är ja, talar om precis vilka prylar som ska packas ned i kappsäcken. Algoritmen får anropa A $O(|P|)$ gånger men får i övrigt inte ta mer än polynomisk tid.

Du ska alltså konstruera och analysera en turingreduktion av det konstruktiva kappsäcksproblemet till det vanliga kappsäcksproblemet (som är ett beslutsproblem).

Tillförlitlighet hos Internet Det händer alltför ofta att datorer eller enskilda förbindelser mellan datorer är trasiga. För att detta inte ska störa möjligheten att kommunicera inom resten av Internet vill man att det ska finnas alternativa vägar mellan varje par av datorer i nätet. Om det till exempel finns tre olika vägar genom Internet från datorn A till datorn B och dessa vägar dessutom är helt disjunkta (utom ändpunkterna) så kan två stycken datorer, vilka som helst, försvinna utan att möjligheten att kommunicera mellan A och B försvinner.

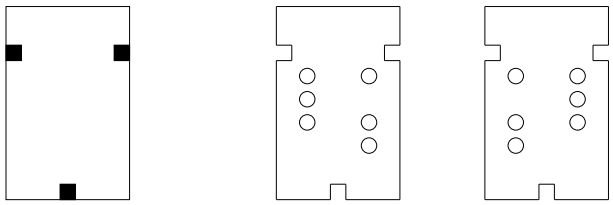
I det här problemet tänker vi oss Internet som en oriktad graf där hörnen motsvarar datorerna i Internet och varje kant (X, Y) motsvarar en *möjlig direktförbindelse* mellan datorerna X och Y. För varje par av datorer (X, Y) har vi också en önskad tillförlitlighet $T(X, Y)$ som anger hur många disjunkta vägar det ska finnas i Internet mellan X och Y. Till sist finns det en budget B som anger hur många direktförbindelser man har råd att konstruera.

Frågan är: går det att plocka ut en delgraf bestående av B kanter så att det för alla par av hörn (X, Y) finns minst $T(X, Y)$ disjunkta stigar i delgrafen mellan X och Y.

Visa att detta problem är NP-fullständigt!

Håstads leksaksaffär Håstads leksaksaffär säljer ett pussel som består av en låda och ett antal kort, se figur 3. Kort kan placeras i lådan på två sätt, med framsidan upp eller med baksidan upp, eftersom det finns urgröpningar i korten som måste passa in i lister som sitter i lådan. På varje kort finns två kolumner med hål, men vissa av hålen är igenfyllda. Målet med pusslet är att lägga korten i lådan på ett sånt sätt att hela botten täcks; för varje hålposition måste alltså åtminstone ett av korten ha ett igenfyllt hål.

Bevisa att språket $\textsc{Toys} = \{\langle c_1, c_2, \ldots, c_n\rangle$: ($c_i$ beskriver kort) $\wedge$ (det går att lösa pusslet)$\}$ är NP-fullständigt.



Figur 3: Håstads pussel. Lådan med tre lister, ett kort med hål och urgröpningar samt samma kort lagt med baksidan upp.

Processorschemaläggning Givet n jobb som ska exekveras och ett tillräckligt antal processorer. Varje jobb tar en tidsenhet att utföra. Det finns också villkor på formen *jobb i kan inte exekveras samtidigt som jobb j*. Vi kallar problemet att avgöra ifall det går att schemalägga jobben så att alla jobb kan exekveras på tre tidsenheter för SCHEDULE. Visa att detta problem är NP-fullständigt.

Är problemet fortfarande NP-fullständigt om vi kräver att jobben ska vara exekverade efter k tidsenheter där $k > 3$? Är det sant för $k = 2$?

Lapptäcke Patchwork, en produkt från programvaruföretaget Stitcher, är en programvara som hjälper dig att pussla ihop rektangulära lappar till ett lapptäcke, som inte har några hål och där inga lappar överlappar varandra.

Indata till programmet är en lista av rektangulära lappar. För varje lapp anges en bredd och en höjd i hela millimeter. Man anger också den önskade storleken (bredd och höjd) på det färdiga (rektangulära) lapptäcket.

Om det är möjligt att pussla ihop en delmängd av lapparna till ett täcke av rätt storlek så producerar programmet en beskrivning av hur detta kan gå till. För varje lapp som ska användas får man veta var i lapptäcket den ska placeras. Man får också veta om man behöver vrida på lappen för att den ska passa.

Formulera detta konstruktionsproblem med matematisk notation och visa att det tillhörande beslutsproblemet (existerar en lösning?) är NP-fullständigt.

Lösningar

Lösning till Konstruera kappsäckslösning

```
Kappsäck(P,S,K)=
  if not A(P,S,K) then return "Ingen lösning"
  foreach p in P do
    if A(P-p,S,K) then P:=P-{p}
  return P
```

Algoritmen anropar A högst $|P| + 1$ gånger. Den returnerade mängden P är en lösning för den uppfyller följande två kriterier.

- *P innehåller inga överflödiga element* eftersom forslingan löper över alla element i P, och i varje varv kollas om elementet p är överflödigt; om det är det plockas det bort ur P.

- *P innehåller en lösning* eftersom detta är en invariant för slingan. Om man kommer till slingan är detta uppfyllt, och efter varje varv i slingan är det uppfyllt.

□

Ledning till Tillförlitlighet hos Internet
Reducera problemet *Hamiltonsk krets*.

Solution to Håstads leksaksaffär
Given cards in the box you can just by a glance verify that every hole position is covered. Thus the problem is in NP.

What remains is to show that every problem in NP can be reduced to TOYS. We do that by reducing CNF-SAT, known to be NP-complete, to TOYS. More precise, we must show that every CNF-formula

$$\varphi = (l_1 \vee l_2 \vee \ldots \vee l_i) \wedge (l_j \vee \ldots \vee l_k) \wedge \ldots \wedge (\ldots)$$

with n variables and m clauses reduced to a problem γ in TOYS is satisfiable if and only if γ is solvable. Consider the following rules.

1. Variable x_i corresponds to a card c_i.
2. There are two columns with m positions on every card, so there is one row for each clause.
3. If x_i occurs in clause j then the hole in position j in the left column is covered.
4. If $\bar{x}_i$ occurs in clause j then the hole in position j in the right column is covered.
5. There are holes in all other positions.
6. There is an additional card c_{n+1} with holes only in the left column.

If φ is satisfied for an assignment of $x_1, \ldots, x_n$ then the corresponding puzzle γ is solvable, because if $x_i = 1$ then card c_i can be put down "right side up" and thus covering the holes in the left column corresponding to the clauses x_i satisfies. If $\bar{x}_i = 1$ then the card c_i should be placed "up-side-down" and in the same way covering holes. Since the formula was satisfied, every clause must be satisfied and therefore the left holes in every row are covered. The extra card c_{n+1} is used for covering the holes in the right column that might occur when all variables in a clause are set to 1, so all right holes are covered too.

Conversely, if the puzzle γ is solved, we can easily find an assignment to $x_1, \ldots, x_n$ satisfying φ from the following rule:

$$x_i = \begin{cases} 1, & \text{if } c_i \text{ is right side up} \\ 0, & \text{otherwise} \end{cases}$$

We also need to check the placing of the extra card; if it is up-side-down we simply invert the assignment of all variables. When all holes are covered we also have, by construction, that all clauses are satisfied. We have shown that CNF-SAT can be reduced to TOYS. □

Solution to Processorschemaläggning
First we need to show that the problem is in NP. This is true because we can guess a scheduling of the jobs and then easily verify that it is valid.

Next, to prove that SCHEDULE is NP-complete, we have to reduce an NP-complete problem to it. We choose to reduce the NP-complete problem 3-COLORING to SCHEDULE. This means that, given a graph $G = (V, E)$, we want to create an instance of SCHEDULE such that the graph can be colored with 3 colors if and only if the jobs can be scheduled in 3 time units.

We create a job for each vertex in the graph and if the vertex i is a neighbor to the vertex j we add the condition that job i can not be executed at the same time as job j. Let each color

correspond to a time unit. Then we claim that the graph can be colored with 3 colors if and only if the jobs can be executed in 3 time units. First, if the graph can be colored with 3 colors let C_1 be the vertices that is colored with the first color. There is no edge between the vertices in C_1 and hence the corresponding jobs can all be executed in the first time unit. The same is true for colors 2 and 3 which gives that all jobs can be executed in 3 time units. On the other hand, if the jobs can be executed in 3 time units, the vertices that correspond to the jobs in each time unit can be given one color so that the graph can be 3-colored.

In this way we have reduced 3-COLORING to SCHEDULE and, consequently, SCHEDULE is NP-complete.

With the same argument we can reduce k-COLORING to SCHEDULE with k time units so that the latter problem is NP-complete for $k \geq 3$.

If we have 2 time steps, the problem can be solved by performing the reduction in the opposite way. That is, given the jobs and the conditions, we construct a graph which we then try to color with 2 colors. This can be done in linear time with depth first search, since if a vertex v has color 1, all neighbors to v must have color 2. □

Lösning till Lapptäcke

Låt indata till konstruktionsproblemet LK vara en lista med talpar (b_i, h_i), $1 \leq i \leq n$, där b_i och h_i är bredden och höjden på en rektangulär lapp, och två tal B och H som anger bredden och höjden på lapptäcket. Samtliga tal är positiva heltal.

En lösning till problemet är en lista (l_i, x_i, y_i, v_i), där l_i, $1 \leq l_i \leq n$, anger numret på den lapp som ska placeras med sitt nedre vänstra hörn i position (x_i, y_i) och v_i är en boolesk variabel som anger om lappen ska vridas. Lapparna får inte överlappa och måste täcka samtliga punkter med koordinater (x, y) där $0 \leq x < B$ och $0 \leq y < H$. Inga andra punkter får täckas.

Vi kallar beslutsproblemet, att avgöra om det finns en lösning till LK, för LB. För att visa att LB är NP-fullständigt visar vi att LB ligger i NP och att SUBSET SUM, ett känt NP-fullständigt problem, kan reduceras till LB.

För att visa att LB ligger i NP så måste vi visa att man på polynomisk tid kan verifiera att en föreslagen lösning till LK är korrekt. Vi behöver kontrollera att lapparna ligger inom lapptäcket, att de inte överlappar varandra, att ingen lapp används mer än en gång och att deras totala yta är BH. Lösningen innehåller högst n lappar och man kan kontrollera om en lapp ligger inom lapptäcket på $O(1)$ tid. Det finns $O(n^2)$ par av lappar och man kan kontrollera om två lappar är disjunkta på $O(1)$ tid. För att kontrollera att ingen lapp används mer än en gång kan vi pricka av lapparna på en lista; detta kan göras på $O(n)$ tid. Att beräkna den totala ytan kan också göras på $O(n)$ tid. Hela kontrollen kan alltså göras på polynomisk tid.

Notera att storleken på lapparna är kvadratisk i ***talen i indata****. Det betyder att storleken på lapparna är exponentiell i* ***längden på indata****, eftersom det bara tar logaritmiskt många bitar att representera ett tal i indata. En algoritm som går igenom alla kvadratmillimetrar i lapptäcket för att kontrollera om det finns några hål tar därför exponentiell tid i längden på indata, men beskrivet med talen i indata blir komplexiteten polynomisk. Detta är därför ett exempel på en* ***pseudopolynomisk algoritm****.*

En instans av SUBSET SUM består av en mängd positiva heltal $p_1, p_2, \ldots, p_n$ och ett positivt heltal K. Man ska avgöra om det finns en delmängd av dessa tal vars summa är K.

Betrakta LB-instansen $(p_i, 1)$, $1 \leq i \leq n$, där p_i och 1 är bredden och höjden på en rektangulär lapp, och två tal K och 1 som anger bredden och höjden på lapptäcket. Om SUBSET SUM har en lösning så går det att konstruera ett sådant lapptäcke. Å andra sidan, om SUBSET SUM inte har en lösning så finns det inget lapptäcke. Reduktionen är därför korrekt. Denna Karp-reduktion tar polynomisk tid och vi har alltså visat att SUBSET SUM $\leq$ LB. □
