Analiza Algoritmilor

Tema 2 - Reduceri polinomiale
Responsabil: Cristian Pavel

Termen de predare: **11.01.2019**
Ultimul update: 09.01.2019

Obiectivele temei

Obiectivele temei sunt identificarea unei reduceri $HCP \leq_p SAT$ si implementarea acesteia prin automatizarea procesului de rezolvare.

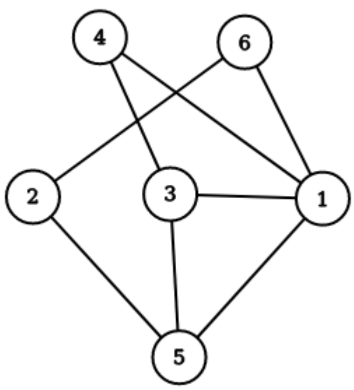
Introducere

Problema **Hamiltonian Cycle**

Un drum hamiltonian intr-un graf neorientat, este un drum care viziteaza fiecare nod o singura data. Un ciclu hamiltonian este un drum hamiltonian care reprezinta, totodata, un ciclu[1]. Problema **Hamiltonian Cycle** consta in determinarea daca un astfel de ciclu exista sau nu intr-un graf.

[1]Un ciclu intr-un graf $G = (V, E)$ reprezinta o succesiune de noduri $v_1, v_2, v_3, ..., v_n$, $v_i \in V, \forall i \in \{1, 2, ..., n\}$, cu proprietatea ca $v_1 = v_n$ si $(v_i, v_{i+1}) \in E, \forall i \in \{1, 2, ..., n-1\}$.

Pentru graful de mai jos un exemplu de ciclu este $1, 6, 2, 5, 1$ si un exemplu de **ciclu hamiltonian** $1, 6, 2, 5, 3, 4, 1$.



Problema **SAT**

Problema SAT este cea discutata la curs.

Problema

In cadrul acestei teme ne propunem sa realizam o transformare **T**, polinomiala, prin care sa reducem Hamiltonian Cycle Problem la SAT.

$$HCP \leq_P SAT$$

Astfel, transformarea **T** va primi ca input o instanta a problemei Hamiltonian Cycle (un graf) si va trebui sa intoarca o instanta a problemei SAT (o expresie booleana).

Info: Astfel de reduceri sunt foarte utile in a rezolva probleme la care nu s-a gasit inca un algoritm optim. De exemplu, daca avem o problema X din NP, stiind ca problema SAT a fost studiata in detaliu si exista foarte multe librarii care implementeaza $SAT - Solvere$, putem reduce problema noastra X la SAT, rezolvam, folosind un $SAT - Solver$, problema SAT si avem un rezultat si pentru X.

Transformarea va trebui implementata in **C** sau **Java**.

Input

Graful neorientat va fi citit dintr-un fisier denumit **graph.in**. Pe prima linie se va afla un numar natural **N**, ce reprezinta numarul de noduri din graf.

Pe urmatoarele linii se vor gasi perechi de numere (x, y) $1 \leq x, y \leq N$ (**nodurile vor incepe de la 1**), cu semnificatia ca in graf exista o muchie de la x la y.

Pe ultima linie se va afla -1, simbolizand sfarsitul input-ului.

Exemplu

```
4
1 2
3 4
2 3
-1
```

Output

Expresia booleana va fi scrisa intr-un fisier **bexpr.out** pe o singura linie. Spatiile vor fi ignorate in verificarea temei. Operatiile permise intre atomi (variabile) sunt AND, pe care il codificam cu $\&$, OR, codificat cu $|$ si NOT, codificat cu $\sim$ (tilda). Ex. $x_1 \& x_2 \& \sim x_3$ reprezinta formula $x_1 \wedge x_2 \wedge \sim x_3$. De asemenea, pentru ca intre operatorii AND si OR nu a fost definita o ordine de precedenta este necesar sa folositi paranteze pentru a seta o astfel de ordine. De exemplu, expresia $x_1 \& x_2 | x_3$ este una ambigua si vor trebui introduse paranteze: $(x_1 \& x_2)|x_3$ sau $x_1 \& (x_2 | x_3)$. In schimb, expresia $x_1 | \sim x_2 | x_3$ este una valida si nu trebuie introduse paranteze.

Restrictii

Valoarea variabilei N este restrictionata: $1 \leq N \leq 100$

Pentru a usura construirea reducerii impunem restrictia ca variabilele din expresiile booleene sa aiba urmatoarele semnificatii:

$$x_{i-j} = \begin{cases} 1, & \text{daca muchia (i, j) apartine drumului ales} \\ 0, & \text{altfel} \end{cases} \quad 1 \leq i, j \leq N$$

$$a_{i-j} = \begin{cases} 1, & \text{daca cea mai scurta cale de la 1 la j, \textbf{in drumul ales}, are lungimea i} \\ 0, & \text{altfel} \end{cases} \quad 1 \leq i \leq N/2{+}1, 1 \leq j \leq N$$

Observatie Variabila a_{0-1} nu este valida. Lungimea, specificata prin indexul i, va incepe de la 1 si va fi maxim $N/2 + 1$.

Este indicat sa se foloseasca aceste informatii pentru a se gasi o reducere corecta.

Exemplu Pentru graful de mai jos in fisierul output va trebui sa scrieti e expresie echivalenta cu urmatoarea:

Exemplu de graf primit ca input

```
3
1 2
2 3
1 3
-1
```

Expresia corespunzatoare scrisa in bexpr.out

$$
\begin{multline}
((x1{-}2\&x1{-}3))\&((x2{-}1\&x2{-}3))\&(a1{-}2|a2{-}2)\&((x3{-}1\&x3{-}2))\& \\
(a1{-}3|a2{-}3)\&((x1{-}2| \sim x2{-}1)\&(\sim x1{-}2|x2{-}1))\& \\
((x1{-}3| \sim x3{-}1)\&(\sim x1{-}3|x3{-}1))\&((x2{-}3| \sim x3{-}2)\& \\
(\sim x2{-}3|x3{-}2))\&((a1{-}2| \sim x1{-}2)\&(\sim a1{-}2|x1{-}2))\& \\
((a1{-}3|\ x1{-}3)\&(\sim a1{-}3|x1{-}3))\& \sim a1{-}1\& \\
((a2{-}2| \sim (((a1{-}1\&x1{-}2)|(a1{-}3\&x3{-}2))\& \sim (a1{-}2)))\& \\
(\sim a2{-}2|(((a1{-}1\&x1{-}2)|(a1{-}3\&x3{-}2))\& \sim (a1{-}2))))\& \\
((a2{-}3| \sim (((a1{-}1\&x1{-}3)|(a1{-}2\&x2{-}3))\& \sim (a1{-}3)))\& \\
(\sim a2{-}3|(((a1{-}1\&x1{-}3)|(a1{-}2\&x2{-}3))\& \sim (a1{-}3))))
\end{multline}
$$

Arhiva

Arhiva va trebui sa fie de tipul **zip** si sa contina fisierele sursa necesare compilarii temei, un fisier **Makefile** cu urmatoarele reguli: **build** (compileaza tema), **run** (ruleaza tema) si **clean** (sterge fisierele obiect si executabilul) si un fisier **README** care sa includa detaliile implementarii cat si o demonstratie a faptului ca transformarea aleasa este polinomiala.

Checker

Tema va fi punctata automat. Pentru a rula checker-ul veti avea nevoie de o versiune gcc >= 5.1.0.