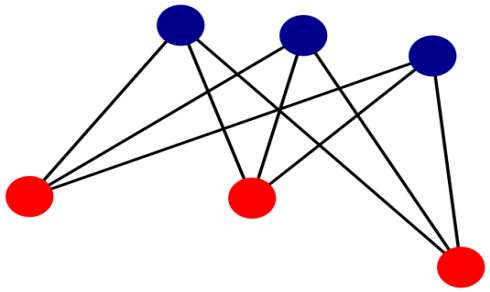
Câteva probleme și variații de probleme

2-COLORING

Uneori prescurtat 2-COL, 2-COLORING este problema în care ni se dă un graf și trebuie să răspundem la întrebarea dacă graful este sau nu 2-colorabil (nodurile pot fi colorate folosind 2 culori, fără ca doi vecini să aibă aceeași culoare).

Problema este **polinomiala**! Este, de asemenea, echivalentă cu a verifica dacă un graf este bipartit.



HORN-SAT

clauză Horn = clauză (disjuncție de literali) în care toți literalii sunt negați, mai puțin **maxim** unul

În continuare ne vom referi la literalii negați ca *literali negativi*, și la literalii simpli ca *literali pozitivi*.

Exemple:

- un literal pozitiv și restul negativi: $\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4$
- niciun literal pozitiv: $\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4$
- un singur literal, și acela este pozitiv: x_1 (numit unit clause)

formulă Horn = conjuncție de clauze Horn

HORN-SAT este o variaţie a problemei SAT în care ni se dă o formulă Horn (la SAT clasic ni se dă o formulă oarecare) şi trebuie să spunem dacă este satisfiabilă.

Cazul când nu avem niciun unit clause:

$$(P \vee \neg Q \vee \neg R) \wedge (Q \vee \neg S) \wedge (\neg W \vee S) \wedge (\neg W \vee \neg X \vee \neg P) \wedge (\neg S \vee \neg P \vee X)$$

Indiferent cum este dată formula (oricât de lungă, etc.), atâta timp cât ea nu conţine niciun unit clause, o putem satisface dând tuturor variabilelor valoarea False. Ce înseamnă faptul că nu avem niciun unit clause? Înseamnă că fiecare clauză va avea cel puţin un literal negativ! Dacă acel literal este False, clauza va fi evaluată la True. Aşadar, dând tuturor literalilor valoarea False, avem garanţia că în fiecare clauză va exista cel puţin un False negat, adică True.

Cazul când avem cel puţin un unit clause:

$$(P \vee \neg Q \vee \neg R \vee W) \wedge (Q \vee \neg S) \wedge (\neg W \vee S) \wedge W \wedge (\neg W \vee \neg X \vee \neg P) \wedge (\neg S \vee \neg P \vee X)$$

Mai sus, W este un unit clause. Îl considerăm pe W ca fiind True şi eliminăm toate clauzele care îl conţin, deoarece acestea vor fi adevărate. Dacă negaţia lui W (adică $\neg W$) este conţinută într-o clauză, vom elimina $\neg W$ din clauză deoarece $\neg W$ va fi fals deci nu mai are relevanţă pentru clauza respectivă.

$$(P \vee \neg Q \vee \neg R \vee W) \wedge (Q \vee \neg S) \wedge (\neg W \vee S) \wedge W \wedge (\neg W \vee \neg X \vee \neg P) \wedge (\neg S \vee \neg P \vee X)$$

Am marcat cu albastru clauzele care conţin W, şi cu verde clauzele care conţin $\neg W$. Pe cele cu albastru le vom elimina complet (sunt adevărate deja):

$$(Q \vee \neg S) \wedge (\neg W \vee S) \wedge W \wedge (\neg W \vee \neg X \vee \neg P) \wedge (\neg S \vee \neg P \vee X)$$

Din cele verzi vom elimina $\neg W$:

$$(Q \vee \neg S) \wedge (S) \wedge W \wedge (\neg X \vee \neg P) \wedge (\neg S \vee \neg P \vee X)$$

În final îl ştergem şi pe însuşi W:

$$(Q \vee \neg S) \wedge (S) \wedge (\neg X \vee \neg P) \wedge (\neg S \vee \neg P \vee X)$$

Acest pas se repetă pentru fiecare unit clause, până le epuizăm pe toate. După ce le-am epuizat pe toate, aplicăm regula de la cazul 1 (setăm toate variabilele

False). În exemplul de mai sus, s-a obținut un nou unit clause, și anume ***S***. Tehnica se numește *unit propagation*.

Luând în considerare algoritmul de mai sus, **HORN-SAT se rezolvă în timp polinomial!** Este considerată printre cele mai "grele" probleme din P, fiind de asemenea **P-Complete** (= problemă care este în P, și care are proprietatea că toate problemele din P se pot reduce la ea).

2-SAT

2-SAT este variația SAT în care fiecare clauză are cel mult 2 literali. Această problemă se rezolvă în timp **polinomial**!

Cunoaștem deja forma normală conjunctivă. Să ne uităm la felurile în care o clauză poate fi scrisă folosind **implicație**:

$$(x_0 \vee \neg x_3) \equiv (\neg x_0 \Rightarrow \neg x_3) \equiv (x_3 \Rightarrow x_0)$$

Dacă toate clauzele din formulă sunt sub formă de implicație, atunci spunem că formula este în *forma normală implicativă*. Vom presupune în continuare că am convertit formula noastră inițială la această formă.

2-SAT cu grafuri:

Ne amintim că, într-un graf orientat, o componentă tare conexă este o submulțime (de obicei ne dorim maximală) de noduri cu proprietatea că din orice nod se poate ajunge în orice alt nod.

Vom construi un graf orientat care corespunde formulei noastre în felul următor: dacă $L_1 \Rightarrow L_2$ (L_1, L_2 literali), fiecărui literal îi va corespunde un nod în graf și vom avea muchie orientată de la L_1 la L_2.

A fost demonstrat că ***formula este satisfiabilă dacă și numai dacă nicio variabilă nu se găsește în aceeași componentă tare conexă ca negația ei***.

2-SAT cu rezoluție (în caz că îi era cuiva dor):

Ne gândim la cazul când avem într-o clauză un literal, iar în altă clauză negația

lui. În această situaţie putem deduce o a treia clauză astfel:

$$\text{Având } \left(X_1 \vee X_2\right) \wedge \left(\neg X_2 \vee \neg X_3\right), \text{ putem obține și } \left(X_1 \vee \neg X_3\right)$$

Atenţie, cele 2 clauze iniţiale **nu dispar**, doar o adăugăm pe cea de-a 3-a! Repetând acest algoritm de câte ori este posibil, uneori putem ajunge la o configuraţie în care avem construcţii de tipul $(X_1 \vee X_1) \wedge (\neg X_1 \vee \neg X_1)$. Dacă acest lucru se întâmplă, spunem că formula nu este *consistentă*. Este demonstrat că ***o formulă este satisfiabilă dacă şi numai dacă ea este consistentă***.

Majority-3-SAT

Problema este similară cu 3-SAT, însă întrebarea constă în a determina dacă **formula are cel puţin 2^{n-1}** soluţii, unde *n* este numărul de variabile din formulă. Practic, ne cere să determinăm dacă majoritatea soluţiilor posibile sunt şi valide, de aici numele. Este suficient să reţinem despre această problemă faptul că este **polinomială**.

Integer (Linear) Programming

O problemă de ***integer programming*** (programare cu numere întregi :-?) este o problemă de optimizare matematică/fezabilitate în care majoritatea/toate variabilele sunt restricţionate la a fi numere întregi. Aceste probleme au forme similare cu cea de mai jos:

$$\begin{aligned} \underset{x,y\in\mathbb{Z}}{\text{maximize}} \quad & y \\ \text{subject to} \quad & -x + y \leq 1 \\ & 3x + 2y \leq 12 \\ & 2x + 3y \leq 12 \\ & x, y \geq 0 \end{aligned}$$

Se interpretează "Găsiţi x şi y numere întregi astfel încât y să fie cât mai mare ("maximize") şi să fie respectate condiţiile *-x+y<=1*... ("subject to")". "maximize y" este funcţia obiectiv, iar condiţiile de la secţiunea "subject to"

sunt constrângerile. Dacă funcţia obiectiv şi constrângerile sunt toate liniare, atunci spunem că problema este o problemă de ***integer LINEAR programming*** (programare liniară cu numere întregi).

Toate problemele de acest tip sunt NP-hard!