Temă

TODO-ul

Fie formulele:

- $\varphi = l_1 \vee l_2 \vee \cdots \vee l_k$;
- $\psi = (l_1 \vee l_2 \vee y_1) \wedge (\neg y_1 \vee l_3 \vee y_2) \wedge (\neg y_2 \vee l_4 \vee y_3) \wedge \cdots \wedge (\neg y_{k-3} \vee l_{k-1} \vee l_k)$.

Lema 1. *Dacă $\varphi = 0$, atunci nu există nicio modalitate de a seta valorile variabilelor $y_1, y_2, \ldots, y_{k-3}$ la 0 sau 1 astfel încât $\psi = 1$.*

Demonstrație. Cum $\varphi = 0$, avem că $l_1 = l_2 = \cdots = l_k = 0$. Pentru ca prima clauză din ψ să fie adevărată, trebuie să alegem $y_1 = 1$. Dar atunci, singura variabilă care poate face clauza a doua adevărată este y_2, deci o setăm și pe aceasta la 1. Repetând raționamentul, obținem $y_1 = y_2 = \cdots = y_{k-3} = 1$. Dar astfel, ultima clauză devine $(0 \vee 0 \vee 0) = 0$. Deci, $\psi = 0$. □

Lema 2. *Dacă $\varphi = 1$, atunci $\psi = 1$ indiferent de valorile variabilelor $y_1, y_2, \ldots, y_{k-3}$.*

Demonstrație. Din moment ce $\varphi = 1$, știm că există un $l_i = 1$. Cum fiecare variabilă y_j face ca exact o clauză să fie adevărată (fie cea care-l conține pe y_j, fie cea cu $\neg y_j$), încercăm să setăm y-urile din clauza lui l_i la 0, pentru a face adevărate cât mai multe clauze despre care încă nu știm nimic (care nu-l conțin pe l_i). Formal, avem trei cazuri:

Dacă $3 \leq i \leq k-2$, deci clauza lui l_i este $(\neg y_{i-2} \vee l_i \vee y_{i-1})$, o soluție este să setăm la 1 variabilele $y_1, y_2, \ldots, y_{i-2}$ și la 0 variabilele $y_{i-1}, y_i, \ldots, y_{k-3}$. Obținem $\psi = (l_1 \vee l_2 \vee 1) \wedge (0 \vee l_3 \vee 1) \wedge \cdots \wedge (0 \vee 1 \vee 0) \wedge (1 \vee l_{i+1} \vee 0) \wedge \cdots \wedge (1 \vee l_{k-1} \vee l_k) = 1$.

Dacă $i \leq 2$, deci clauza lui l_i este $(l_1 \vee l_2 \vee y_1)$, o soluție este să setăm toate variabilele $y_1, y_2, \ldots, y_{k-3}$ la 0. Obținem $\psi = (1 \vee 0) \wedge (1 \vee l_3 \vee 0) \wedge \cdots \wedge (1 \vee l_{k-1} \vee l_k) = 1$.

Dacă $i \geq k-1$, deci clauza lui l_i este $(\neg y_{k-3} \vee l_{k-1} \vee l_k)$, o soluție este să setăm toate variabilele $y_1, y_2, \ldots, y_{k-3}$ la 1. Obținem $\psi = (l_1 \vee l_2 \vee 1) \wedge (0 \vee l_3 \vee 1) \wedge \cdots \wedge (0 \vee 1) = 1$. □

În concluzie, formula dată în problema SAT este echisatisfiabilă cu cea obținută aplicând strategia de la seminar. Așadar, problema SAT se reduce polinomial la 3-SAT.

Exercițiul 1.

2-SAT $\in \mathbb{P}$ deoarece poate fi rezolvată de următorul algoritm polinomial:

Plecăm de la observația că $(p \vee q) \equiv (\neg p \to q) \wedge (\neg q \to p)$. Construim un graf orientat G cu $2n$ noduri, etichetate $x_1, x_2, \ldots, x_n, \neg x_1, \neg x_2, \ldots, \neg x_n$. Pentru fiecare clauză de forma $(x \vee y)$ adăugăm în G muchiile $(\neg x, y)$ și $(\neg y, x)$. Trebuie să găsim o atribuire τ pentru care să nu existe în G muchii (x, y) cu $x \not\to y$.

Observăm că toate nodurile dintr-o componentă tare-conexă a lui G trebuie să aibă aceeași valoare. Altfel, știind că orice componentă tare-conexă conține un ciclu ce include toate nodurile din componentă, ar exista două noduri consecutive pe ciclul respectiv, x și y, cu $\tau(x) = 1$ și $\tau(y) = 0$, ceea ce contrazice $x \to y$. Prin urmare, dacă există un x_i cu proprietatea că x_i și $\neg x_i$ se află în aceeași componentă tare-conexă, înseamnă că nu există soluție. Altfel, problema admite întotdeauna o soluție care poate fi găsită în felul următor:

Vom folosi faptul că dacă G conține o muchie (x, y), atunci o conține și pe $(\neg y, \neg x)$. De aici rezultă că oricărei componente tare-conexe C îi corespunde în G o componentă tare-conexă C', care conține toate nodurile din C negate. Construim graful aciclic al componentelor tare-conexe ale lui G și asignăm 0 unei componente cu gradul intern 0, pentru a nu impune restricții componentelor adiacente. Automat, asignăm 1 componentei complementare, care din motive de simetrie are gradul extern 0. Eliminăm cele două componente din graful aciclic și repetăm procesul. □

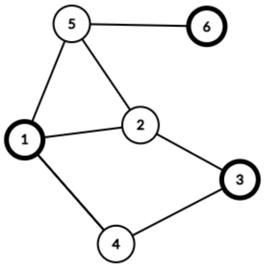
Așadar, problema *de decizie* 2-SAT poate fi rezolvată construind graful G, determinând componentele tare-conexe și verificând dacă există vreun nod x_i care se află în aceeași componentă cu $\neg x_i$. Algoritmul are complexitatea $O(n)$, deci este polinomial.

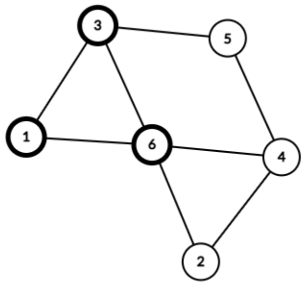
Exercițiul 4.

Trebuie să arătăm că problema de decizie Clique este $\mathbb{NP}$-completă.

În primul rând, Clique $\in \mathbb{NP}$, pentru că poate fi rezolvată în $O(n^2)$ de următorul algoritm: Fie $G = (V, E)$ graful dat. Alegem aleatoriu o submulțime S, de cardinal k, a lui V. Apoi, pentru fiecare două noduri u și v din S, testăm dacă $[u, v] \in E$. Dacă nu, atunci S nu este o clică (de mărime k) a lui G.

În al doilea rând, Clique este $\mathbb{NP}$-hard, pentru că problema Maximum-Independent-Set, care știm că este $\mathbb{NP}$-completă, se reduce la Clique: Orice mulțime independentă S a lui G este o clică în graful complementar al lui G, pentru că $[u, v] \notin G \Leftrightarrow [u, v] \in G', \forall\, u, v \in S, u \neq v$.





Pentru a calcula dimensiunea maximă a unei mulțimi independente a lui G, construim mai întâi, în $O(n^2)$, graful G', după care căutăm răspunsul în $O(n)$ (sau chiar $O(\log n)$, pentru că putem folosi căutare binară), apelând funcția Clique la fiecare pas, pentru a testa dacă dimensiunea curentă este validă. Așadar, Maximum-Independent-Set se reduce polinomial la Clique, deci Clique este o problemă $\mathbb{NP}$-hard.

În concluzie, Clique este $\mathbb{NP}$-completă. $\square$