Rozwiązania zadań ze zbioru zadań edycji 2020

z Języków Formalnych i Złożoności Obliczeniowej

Bartłomiej Grochowski

Część trzecia:

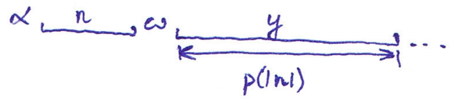
Klasy złożoności czasowej i pamięciowej, redukcje wielomianowe

zadania 139-208

Zamieszczone tu rozwiązania mają służyć
złapaniu pewnych idei i mogą być pomocą przy
rozpoczęciu przygody z JFiZO. Spisywanie rozwiązań,
niestety, nie ułatwi zdania egzaminu.

Uniwersytet Wrocławski
18 czerwca 2023

Zad 139 $A \in NP \Rightarrow \exists$ wielomiany p, q i NMT_A dowodzące:



$A \in NP \Rightarrow$ jeśli damy podpowiedź to umiemy weryfikować

$B = \langle n, y \rangle \in P \Rightarrow$ istnieje deterministyczna MT_B

↑ podpowiedź, maszyna działa w czasie $q(|n|)$.

NMT_A:
* ustaw się za ω // $\log n$ kroków
* niedeterministycznie zgadnij 'y' // $p(|n|)$ kroków
* wróć na początek // $p(|n|)$ kroków $+ c$
* zachowuj się jak MT_B // $q(|n|)$ kroków

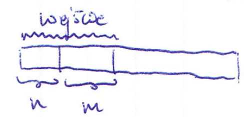
$\longrightarrow NMT_A$ działa w czasie wielomianowym.

JFIZO zad 140

$A \subseteq \mathbb{N}^2$, $A \in P$, $B = \{ n \mid \exists m \ |m| \leq p(|n|) \ \langle n, m \rangle \in A \} \Rightarrow B \in NP$?

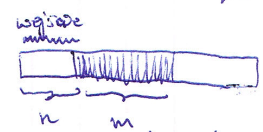
$A \in P$ czyli istnieje MT_A, które spr czy $\langle n, m \rangle \in A$ i zatrzymuje się po max $p(|n| + |m|)$ krokach.

MT_A:



Jeśli $B \in NP$ to istnieje NMT_B, które spr. czy $n \in B$ i zatrzymuje się po max $q(|n|)$ krokach.

NMT_B:



m – zgadywane, maszyna niedeterministycznie zapisuje część taśmy krótszą równą od $p(|n|)$

0 i 1 – koduje m.

→ następnie zachowuje się identycznie jak MT_A, więc w czasie $r(|n| + p(|n|)) = q(|n|)$

↑ czas wielomianowy

Czyli $B \in NP$

program:
* wczytaj n
* zgadnij m → $|m| \leq p(|n|)$
* odpal $MT_A(n, m)$

Istnieje jakiś przebieg akceptujący dla $NMT_B(n) \Leftrightarrow n \in B$.

141

B jest NP $\Rightarrow \exists A \subseteq N^2$, A jest P

$B = \{ n : \exists m \ |m| \le p(|n|),\ \langle n, m \rangle \in A \}$

B → jest jakaś NMT_B, działa w czasie wielomianowym p.

$A = \{ \langle n, \#w \rangle \mid n \in \mathbb{N} \}$ #w - ciąg instrukcji przebiegu akceptującego

w ciąg instrukcji maszyny B o dł $\le p(|n|)$
który jest akceptujący ($NMT_B(n)$)
„działającej na n”

Alfabet w: ciąg instrukcji

A jest w P?

TAK → n ∧ #w

↓

w → robimy ten ciąg instrukcji

na maszynie $NMT_B(n)$

JFIZO zad 142 $\quad 5SAT \le_p 3SAT$

Dają nam φ w postaci 5CNF

Mamy napisać ψ w postaci 3CNF (w wielomianowym czasie)

t. że φ spełnialne $\Leftrightarrow$ ψ spełnialne

$$\varphi: \ \dots \wedge (p_1 \vee p_2 \vee p_3 \vee p_4 \vee p_5) \wedge \dots$$

$$\psi: \ \dots \wedge (p_1 \vee p_2 \vee q_1) \wedge (\neg q_1 \vee p_3 \vee \neg q_2) \wedge (q_2 \vee p_4 \vee p_5) \wedge \dots$$

1° φ spełn. $\Rightarrow$ ψ spełn.

1° $p_1 = T \vee p_2 = T \Rightarrow q_1 = F,\ q_2 = T$

2° $p_3 = T \Rightarrow q_1 = T,\ q_2 = T$

3° $p_4 = T \vee p_5 = T \Rightarrow q_1 = T,\ q_2 = F$ ✓

2° φ sprzeczne $\Rightarrow$ ψ sprzeczne

$p_1 = p_2 = \dots = p_5 = F$

q_1	q_2	ψ	
F	F	F	// 1 klauzula
F	T	F	// 1 klauzula
T	F	F	// 3 klauzula
T	T	F	// 2 klauzula

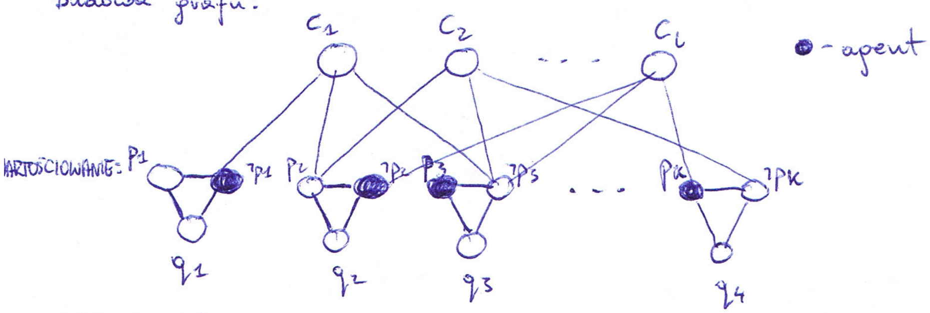
JFIZO zad 143. $3SAT \leq_p STASI$ (DS - dominating set)

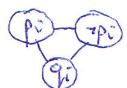
Mamy formułę φ w 3CNF, która ma k różnych zmiennych.

Zbudujemy graf G_φ t.że:

w G_φ da się rozstawić k agentów $\Leftrightarrow$ φ spełnialna

Budowa grafu:



→ Dla każdej zmiennej zdaniowej tworzymy 

→ Dla każdej z l klauzul tworzymy wierzchołek i łączymy po odpowiednio → np. na powyższym rysunku $c_1 = \neg p_1 \vee p_2 \vee \neg p_3$

$c_2 = p_2 \vee \neg p_3 \vee \neg p_K$

Agentów MUSIMY stawiać PO JEDNYM TYLKO na p_i lub $\neg p_i$.

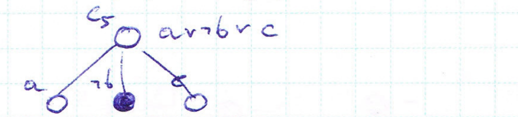
Co jeśli damy gdzie indziej?

1) agent w c_i → brakuje agenta dla q_j, bo mamy k agentów i k q_j.
2) agent na p_i i $\neg p_i$ → —"—
3) agent na q_i → może stać tylko jeśli formuła jest niezależna od zmiennych p_i, czyli żadna klauzula c_j nie jest połączona z p_i ani $\neg p_i$.

Jeśli tak rozstawimy agentów, to oznacza, że mamy wartościowanie, które spełnia formułę φ. (jeśli agenci stoją ok)

($\Leftarrow$) φ spełnialna $\Rightarrow$ istnieje wart. σ, że $\hat{\sigma}(\varphi) = T$. Agent na p_i, jeśli $\hat{\sigma}(p_i) = T$, Agent na $\neg p_i$ jeśli $\hat{\sigma}(p_i) = F$.

* każde $p_i, \neg p_i, q_i$ będzie miało agenta sąsiada - oczywiste
* Każdy wierzch. c_i będzie miał agenta, bo c_i przy tym wart. spełniona! (czyli przynajmniej jeden sąsiad agent)



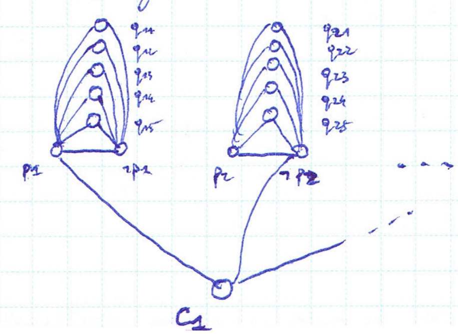
$\nwarrow$ $b = F$, $\neg b = T$, $\sigma(c_5) = T$

($\Rightarrow$) G_φ ma dobre rozstawienie $\Rightarrow$ $\forall_i$ agent stoi na p_i lub $\neg p_i$ lub q_i (jeśli na q_i to p_i nie ma w c_j albo ustawiamy $p_i = T$)

Jeśli agent stoi na p_i to $\hat{\sigma}(p_i) = T$, jeśli na $\neg p_i$ to $\hat{\sigma}(p_i) = F$.

Każda klauzula c_i jest nadzorowana, czyli chociaż jeden kawałek alternatywy jest prawdziwy, czyli $\hat{\sigma}(c_i) = T$ czyli φ spełnialna

INNA REDUKCJA



tu wymuszamy, że agenci są na p_i, gdyby byli na q_{ij} to inne wierzchołki q_{ij} NA PEWNO byłyby bez agenta

JFIZO zad 144

a) $H \le_p H_d$

Dają nam graf G ~~[illegible]~~, mamy zbudować G_d (skierowany) t.że G ma cykl Ham $\Leftrightarrow$ G_d ma skierowany cykl Ham.

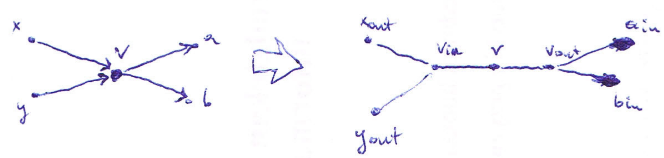
Konstrukcja: $\forall_{u,v} \in E \Rightarrow \{u,v\} \in E_d \wedge \{v,u\} \in E_d$

1° w G jest cykl Ham $\to$ w G_d jest ten sam, bo możemy chodzić po tych samych krawędziach w obie strony.

2° w G nie ma $\to$ w G_d nie ma, bo analogicznie w G_d są te same przejścia po krawędziach co w G.

b) $H_d \le_p H$

Dają nam G_d, budujemy G: $\forall_v \in V_d$ budujemy v_{in}, v, v_{out}
$\forall_{\{u,v\} \in E_d}$ mamy: $\{u_{out}, v_{in}\} \in E$



1° w G_d jest cykl Ham: $v_1 \to v_2 \to \dots \to v_n \to v_1$ wtedy w G jest cykl Ham $v_1 \to v_{1out} \to v_{2in} \to \dots \to v_n \to v_{nout} \to v_{1in} \to v_1$, jest ok bo każde v_k ma st. 2, więc musimy przez niego przechodzić $v_{kin} \to v_k \to v_{kout}$ a reszta to przejścia po krawędziach z cyklu H_d

2° w G_d nie ma cyklu Ham $\to$ w G też nie ma, ten sam argument: st $v_k = 2$, a reszta to przejścia między krawędziami w G_d, nic więcej nam to nie daje

JFIZO zad 145 HORNSAT

hornowska $\to$ 1° wszystkie zmienne zanegowane
2° jeden literał niezanegowany
> w jednej klauzuli.

Obserwacja 1 Jeśli wszystkie klauzule ≥ 2 literały to wartościowanie wszystkich zmiennych na F spełnia formułę.

Czemu? bo w każdej klauzuli jest $\neg p$. $p=F \Rightarrow \neg p = T \Rightarrow C$ spełnialne.

Obserwacja 2 Pojedyncze klauzule WYMUSZAJĄ wartościowanie.

$\neg p \Rightarrow p=F$ oraz $p \Rightarrow p=T$

Algorytm: wczytaj zbiór klauzul
$Y = \emptyset$ // Y zbiór zmiennych = T
dla każdej klauzuli:
* jeśli jest jednoliterałowa:
  > jeśli jest $\neg p$ to musi być $p=F$
    * jeśli $p \in Y$ return NIE //sprzeczność
    * wykreśl zmienną p z każdej klauzuli (bo nie ma znaczenia w spełnialności)
    * wykreśl całą klauzulę z $\neg p$ (bo są one spełnione skoro $p=F \Rightarrow \neg p=T$)
  > jeśli jest p to musi być $p=T$
    * dodaj p do zbioru Y.
    * wykreśl zmienne — symetrycznie
    * wykreśl klauzulę — symetrycznie
* jeśli zbiór klauzul pusty return TAK
* jeśli nie ma żadnej jednoliterałowej return TAK
* else dalej pętla

Algorytm jest wielomianowy $\sim O(n^2)$

JFIZO zad 146

2SAT jest P – pokażemy używając REZOLUCJI

$$\frac{C \vee p \qquad D \vee \neg p}{C \vee D}$$

LOGIKA: zbiór klauzul sprzeczny wtw gdy istnieje rezolucyjny dowód sprzeczności.

Idea: używamy rezolucji i próbujemy dojść do sprzeczności.

n – różne zmienne $x_1 \dots x_n$

Algorytm: wczytaj C // zbiór klauzul, $|C| \le 4n^2 + 2n$

$4n^2$: $2n \cdot 2n$ kombinacje par, 2 bo albo p albo $\neg p$; $2n$: klauzule typu $p \vee \perp$

```
while (true)
{
    for i=1 to |C|
        for j=1 to |C|
            if i≠j && daSięRezolucje(Ci, Cj)
                Cx = res(Ci, Cj)
                if Cx == ⊥
                    return "sprzeczna"
                C = C \ {Ci, Cj} ∪ Cx   // moc zmniejsza się o 1
                udałoSięRezolucje = true
    if udałoSięRezolucje == false
        return "spełnialna"
    else ; // pętla znowu
}
```

$O(n^2)$ $O(n^4)$ $O(n^2)$

Czas $O(n^6) \Rightarrow 2SAT \in P$

Czemu tu działa, a 3SAT jest NP?

1° Jeśli przesłanki mają po 2 literały to rezolwenta też ma 2.

2° Ilość klauzul jest $O(n^2)$, gdzie n to ilość wszystkich zmiennych.

$$\frac{\overset{(2)}{a \vee b} \qquad \overset{(2)}{\neg b \vee c}}{\underset{(2)}{a \vee c}}$$

wielkość klauzul jest stała więc daSięRezolucje i $res(C_i, C_j)$ działają w czasie stałym

→ 3SAT:

$$\frac{\overset{(3)}{a \vee b \vee c} \qquad \overset{(3)}{\neg a \vee d \vee e}}{\underset{(4)}{b \vee c \vee d \vee e}}$$

rozrasta się, więc daSięRezolucje i $res(C_i, C_j)$ nie działają w czasie stałym.

UWAGA: $\frac{A \quad B}{C}$ → A i B NIE ZNIKAJĄ ze zbioru przesłanek!!! ale maksymalnie możemy wyprodukować $4n^2 + 2n + 1$ 2-klauzul. Będzie wolniej, ale się da. Sprawdzić nowo spr czy jest w nim klauzula pusta (czyli $\perp$ – sprzeczna)

C^2_{2n} C^1_{2n} i pusta, pusta

JFiZO zad. 147. $3SAT \leq_p 3SAT_3$ (każda zmienna max 3 razy)

Mamy formułę φ w postaci 3CNF.

Jeśli któraś zmienna występuje więcej niż 3 razy (k razy) to zamieniamy każde jej wystąpienie na różne nowe zmienne.

$p \longrightarrow p_1, p_2, \ldots, p_k$, ale chcemy, żeby miały takie same wartości, czyli:

$$p_1 \Leftrightarrow p_2 \Leftrightarrow p_3 \Leftrightarrow \ldots \Leftrightarrow p_k$$

$$\equiv$$

$$p_1 \Rightarrow p_2 \Rightarrow p_3 \Rightarrow \ldots \Rightarrow p_k \Rightarrow p_1$$

$$\equiv$$

$$(p_1 \Rightarrow p_2) \wedge (p_2 \Rightarrow p_3) \wedge \ldots \wedge (p_k \Rightarrow p_1)$$

$$\equiv$$

$$\psi = (\neg p_1 \vee p_2 \vee \bot) \wedge (\neg p_2 \vee p_3 \vee \bot) \wedge \ldots \wedge (\neg p_k \vee p_1 \vee \bot)$$

Czyli $\varphi' = \varphi_{zamienione} \wedge \psi$

warunek równoważności zapisany jako 3CNF

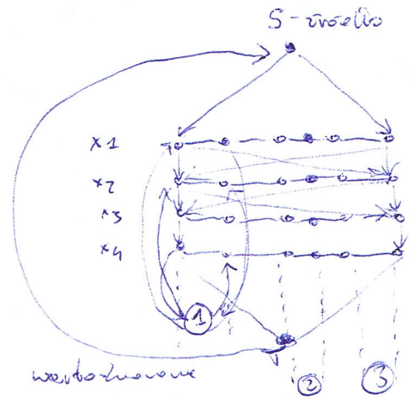
Formuła liniowo większa, więc wielomianowa redukcja.

Przykład: $\varphi = (a \vee b \vee c) \wedge (a \vee d \vee e) \wedge (a \vee f \vee g) \wedge (a \vee h \vee i)$ } 3CNF

$$\left.\begin{array}{l}\varphi' = (a_1 \vee b \vee c) \wedge (a_2 \vee d \vee e) \wedge (a_3 \vee f \vee g) \wedge (a_4 \vee h \vee i) \wedge \\ (\neg a_1 \vee a_2 \vee \bot) \wedge (\neg a_2 \vee a_3 \vee \bot) \wedge (\neg a_3 \vee a_4 \vee \bot) \wedge (\neg a_4 \vee a_1 \vee \bot)\end{array}\right\} 3CNF_3$$

148 $3CNF \leq_p HAM_d$, k liczba klauzul, $2k$ wierzch

mamy φ w postaci 3CNF



T → idziemy w prawo

F → idziemy w lewo

$(x_1 \vee \neg x_2 \vee x_4)$ 1

$\wedge$

(. . .) 2

$\wedge$

(. . .) 3

Graf bez wierzch ① ② ③ ma cykl Ham zawsze.

Ale: φ spełn $\Leftrightarrow$ w G jest cykl Ham.

($\Rightarrow$) da się, przejście przez kolorowy wierzchołek oznacza, że klauzula jest spełniona.

($\Leftarrow$) po tych łańcuszkach przechodzimy w jednym kierunku więc odczytamy sobie wartościowanie

→ nie możemy skoków między poziomami:

nie będzie się dało wejść do wierzch v.

JFIZO zad 149 $\quad HAM \leq_p KOM$

Dając nam graf G. Skonstruujemy G' t.że:

$$G \text{ ma cykl Ham} \iff G' \text{ ma cykl Ham } c' \text{ t.że } w(c') < k$$

G' - graf pełny, $V_{G'} = V_G$, $E_{G'} = \{\langle u,v\rangle \mid u,v \in V_{G'}\}$

$$w(e) = \begin{cases} 1 & \text{gdy } e \in E_G \\ 2 & \text{gdy } e \notin E_G \end{cases}$$

$\cancel{k = |V_G| = |V_{G'}|}$

$k = |V_G| + \frac{1}{2} = |V_{G'}| + \frac{1}{2}$

($\Rightarrow$) w G jest cykl Ham $c \Rightarrow c$ ma $|V_G|$ krawędzi $\Rightarrow$ w G' też będzie ten cykl i ma on wagę $|V_G| < k$ ✓

($\Leftarrow$) w G' jest cykl Ham c' i $w(c') < k \Rightarrow$ wszystkie ~~kra~~ krawędzie w cyklu c' muszą mieć wagę 1 (c' ma $|V_{G'}|$ krawędzi) $\Rightarrow$ z def wszystkie te krawędzie są też w $G \Rightarrow$ w G jest cykl Ham.

JFIZO zad 150

Problem cyklu Hamiltona jest NP-zupełny, czyli:

$\forall X \in NP.\ X \leq_p Ham$

jakiś graf G

Skonstruujemy G': $V_{G'} = V_G$, $E_{G'} \to$ graf pełny

$$w(e) = \begin{cases} 1 & \text{jeśli } e \in G \\ 2 \cdot |V_G| + \varepsilon & \text{jeśli } e \notin G \end{cases}$$

// można więcej wag $\neq 0$, $\neq 1$

<u>Jeśli istnieje algorytm</u>, który wyznacza rozw $\leq 2 \cdot OPT$ to dla G' zwróci dokładnie te krawędzie, które należą do G. Czyli wielomianowo umiemy obliczyć HAM. Czyli też każdy problem $X \in NP$.

$\forall X \in NP.\ X \in P \Rightarrow NP \subseteq P$

wiemy, że $P \subseteq NP$

zatem $P = NP$

JFIZO zad 151

Nierówność trójkąta $\forall_{x,y,z}\ x+y \geq z$

Wielomianowy algorytm rozwiązanie $\leq 2 \cdot OPT$.

1. Zbuduj MST // $O(m \cdot \log n)$ - kruskal

2. Wypisz wierzchołki preorder // $O(m+n)$ - DFS

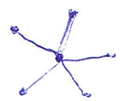
→ Czemu takie rozw jest $\leq 2 \cdot OPT$?

1° Rozw KOMI bez jednej krawędzi to drzewo

czyli $w(MST) \leq w(KOMI)$ $= OPT$

2° Przejście po wszystkich wierzchołkach zajmuje najwięcej:

gdy mamy gwiazdkę

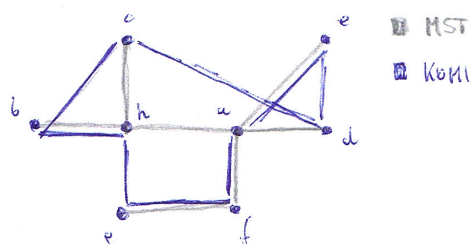


DFS → $2 \cdot w(MST)$
„tam i z powrotem” po każdej krawędzi.

→ a my będziemy chodzić na skróty → będziemy poprawiać

3° Czyli rozwiązanie ogólne nie przekroczy $2 \cdot w(MST) \leq w \cdot OPT$

PRZYKŁAD



DFS: a - f - g - f - a - h - b - h - c - h - a - d - a - e - a → $\leq 2 \cdot OPT$

PREORDER: = = ≤ = ≤ ≤ ≤ = → $\leq DFS \leq 2 \cdot OPT$

152 $d(x,y) < d(x,z) + d(z,y)$

w obu zadaniach 151 i 152 nie ma znaczenia
czy użyjemy
$\cancel{x} < y+z$
$x \leq y+z$

$HAM \leq_p 152$

Mamy graf $G \to G', k$

G': $V_{G'} = V_G$ $E_{G'}$ → graf pełny

$$w(e) = \begin{cases} 2 & e \in E(G) \\ 3 & e \notin E(G) \end{cases} \qquad k = 2|V_G|$$

G zawiera cykl HAM ⇒ G' zawiera cykl HAM po krawędziach o wadze 2 → $< k$

G nie zawiera cyklu HAM ⇒ G' ma którąś z „3” → $\geq k$.

Czemu warunek zachowany?

$\langle x,y \rangle$ $\langle x,z \rangle$ $\langle y,z \rangle$
$\notin \downarrow$ $\in \downarrow$ $\in \downarrow$
3 2 2 → zawsze zachodzi warunek trójkąta

$d(x,y) \leq 3$

$d(z,y) + d(x,z) \geq 4$

JFIZO zad 153. $SAT_2 \in P$

n zmiennych, każda max 2 wystąpienia

max $\rightarrow 2n$ klauzul.

1° Wystąpienia zmiennej p: p ; $\neg p$; $p\ p$; $\neg p\ \neg p$

nadaj wartościowanie: T F T F

i usuń wszystkie spełnione klauzule

2° Zmienne p: $p\ \neg p$; $\neg p\ p$

Algorytm jak w zad 146.

wtedy rezolucja:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

WYRZUCAMY 2 przesłanki i dorzucamy 1 klauzulę, czyli w każdym kroku ~~zwiększamy~~ zmniejszamy liczbę klauzul.

Tutaj nie ma tego problemu co w zad 146, bo nie ma innych wystąpień niż p i $\neg p$ (są tylko 2)

Zauważmy też, że klauzule rosną, ale sumaryczna liczba literałów nie:

$$\frac{\overbrace{p \vee \underbrace{\quad}_{7}}^{8} \qquad \overbrace{\neg p \vee \underbrace{\quad}_{9}}^{10}}{7+9}$$

$= 8+10-2$

Jeśli dojdziemy do $\perp$ to sprzeczny else spełnialny.

Wartościowanie:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D} \text{ - spełnione} \rightarrow$$

C	D	p	$p \vee C$	$\neg p \vee D$
T	T	T	T	T
T	F	F	T	T
F	T	T	T	T

zawsze true

JFIZO zad 154 $\quad$ $CLIQUE_{\frac{n}{2}}$ - NP zupełny

$3SAT \leq_p CLIQUE_{\frac{n}{3}} \leq_p CLIQUE_{\frac{n}{2}}$

1 $3SAT \leq_p CLIQUE_{\frac{n}{3}}$ (Cormen s1028)

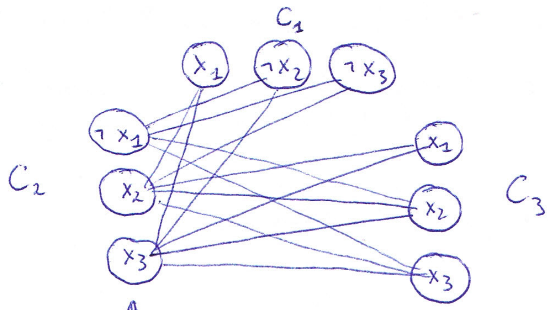
$\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$, $\quad C_i = (l_1^i \vee l_2^i \vee l_3^i)$

Skonstruujemy graf G t.że φ spełn $\Leftrightarrow$ w G jest klika $\frac{n}{3}$

G: wierzchołki l_i^r, l_j^s łączymy gdy:

1° są w różnych klauzulach $(r \neq s)$

i 2° są niesprzeczne czyli nie łączymy p i $\neg p$



wszystko narysowane tylko dla C_2

$(\Rightarrow)$ φ spełn $\Rightarrow$ każda klauzula ma przynajmniej 1 literał z wartością T

$\Rightarrow$ skoro wszystkie wybrane mają wartość T to nie może być tak,

że jeden jest negacją drugiego $(x, \neg x)$ $\Rightarrow$ wszystkie są ze sobą

połączone (wynika z konstrukcji grafu) $\Rightarrow$ jest ich $\frac{n}{3}$, więc klika $\frac{n}{3}$ ✓

$(\Leftarrow)$ w G jest klika $\frac{n}{3}$ $\Rightarrow$ wiemy, że żadne krawędzie nie łączą wierzch

z tej samej trójki $\Rightarrow$ w każdej trójce jest jeden, przypisujemy im T

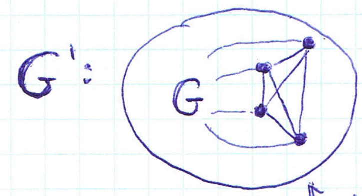
$\Rightarrow$ nie ma możliwości, że przypiszemy $x = T \wedge \neg x = T$, bo nie ma takich

krawędzi $(x - \neg x)$ $\Rightarrow$ każda klauzula spełniona $\Rightarrow$ φ spełn ✓

2 $CLIQUE_{\frac{n}{3}} \leq_p CLIQUE_{\frac{n}{2}}$

Mamy graf G, zbudujemy G' t.że

w G klika $\frac{n}{3}$ $\Leftrightarrow$ w G' klika $\frac{n'}{2}$



dopisujemy $\frac{n}{3}$ wierzch połączonych ze sobą i ze wszystkimi z grafu G.

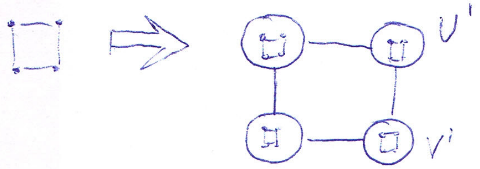
1° G ma klikę $\frac{n}{3}$ $\Rightarrow$ $n' = \frac{4}{3}n$, G' ma klikę $\frac{n}{3} + \frac{n}{3} = \frac{2}{3}n = \frac{n'}{2}$ ✓

2° G nie ma kliki $\frac{n}{3}$ $\Rightarrow$ dopisane $\frac{n}{3}$ połączone wierzch to łącznie mniej niż $\frac{2}{3}n$ czyli $< \frac{n'}{2}$ ✓

ZATEM: $CLIQUE_{\frac{n}{2}}$ jest NP-zupełny.

155

Dostajemy $G \to G^2$



Jeśli w G jest $\langle v, u \rangle$ to w super wierzchołkach wszystko z v' łączymy ze wszystkimi z u'.

Lemat: W G istnieje klika $\geq k \iff G^2$ $\exists$ klika $\geq k^2$

$(\Rightarrow)$ bierzemy super wierzch, tam jest klika k

więc skoro każdy z każdym to klika k^2.

$(\Leftarrow)$ zał, że w G nie ma kliki k

ale wtedy mamy klikę w G' ? ??

$G^2 \to$ klika k^2. Jeśli mamy algorytm to znajdujemy

klikę $\frac{k^2}{2} \Rightarrow$ w G jest $\sqrt{\frac{k^2}{2}} = \frac{k}{\sqrt{2}}$

Czyli $\frac{1}{\sqrt{2}}$ kliki max.

Alg: $\underset{n}{G} \to \underset{n^2}{G^2} \to$ znajduję klikę $\geq \frac{n^2}{2} \to$ w G mamy klikę $\geq \frac{n}{\sqrt{2}}$

UWAGI: * to można iterować jakąś stałą liczbę razy.

* nie znamy algorytmu kliki $\frac{n}{2}$

Zad 156

$\phi = c_1 \vee c_2 \vee \dots \vee c_n$, 3SAT $\frac{9}{10}$

$\to$ 3SAT $\leq_p$ 3SAT $\frac{9}{10}$

Redukcja: dostajemy formułę φ. Skonstruujemy φ' t.że

φ spełn $\iff \varphi'$ $\frac{9}{10}$ spełn

$$\varphi' = \underbrace{\varphi}_{n \text{ klauzul}} \vee \underbrace{y_1 \vee \neg y_1 \vee \dots \vee y_k \vee \neg y_k}_{2k \text{ klauzul}}$$

$\frac{n+k}{n+2k} \geq \frac{9}{10} \to 10k + 10n > 9n + 18n \to n > 8k \to k < \frac{1}{8} n$

k największe

$(\Rightarrow)$ φ spełn $\Rightarrow$ $n+k$ spełn $\Rightarrow$ $>90\%$ klauzul speł

$(\Leftarrow)$ φ nie spełn $\Rightarrow$ max $n-1+k$ spełn $\Rightarrow \frac{n-1+k}{n+2k} < 90\%$

Zad 157 $KLIKA_{\geq \frac{n}{c}} \leq_p KLIKA_{\geq \frac{n'}{c'}}$ $(\forall c,c')$

1° $c' \geq c$, wtedy zostawiamy ten graf. Jeśli istnieje w nim klika co najmniej $\frac{n}{c}$ to również istnieje $\frac{n}{c'}$ bo $\frac{n}{c'} \leq \frac{n}{c}$.

2° $c > c'$, wtedy tworzymy G': $G \oplus K_k$

$$\underbrace{\frac{\overset{n'}{n+k}}{c'}}_{\text{minimalnie ma być klika } \frac{n'}{c'}} \leq \underbrace{\frac{n}{c} + k}_{\text{była klika} \geq \frac{n}{c}, \text{ teraz będzie } \frac{n}{c}+k}$$

$$\frac{n+k}{c'} \leq \frac{n+ck}{c} \Rightarrow ck + cn \leq c'n + cc'k \Rightarrow (c-c')n \leq (cc'-c)k$$

$$\Rightarrow k \geq \frac{n\cdot(c-c')}{cc'-c}$$, do redukcji weźmy $k = \left\lceil \frac{n(c-c')}{cc'-c} \right\rceil$

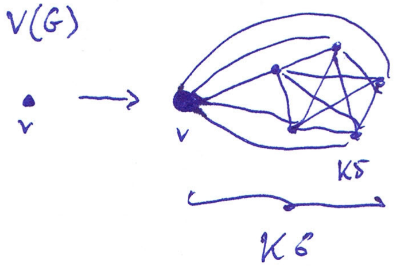
Wtedy w G' będzie klika$_{\geq \frac{n'}{c'}}$ wtedy gdy w G klika$_{\geq \frac{n}{c}}$.

Dowód jak w zad 154 $K_{\frac{n}{3}} \leq K_{\frac{n}{2}}$.

158 $3COL \leq_p$ $3COL+1$

Dany mamy graf G. Stworzymy G':

$\forall v \in V(G)$



Jedyne kolorowanie to takie, że mamy po 2 wierzch. każdego koloru.

$G \in 3COL \iff G' \in 3COL+1$

($\Rightarrow$)

mamy kolorowanie G, dokolorowujemy łatwo kliki

($\Leftarrow$)

~~...~~ jak mamy G' to wyrzucamy te kliki, będzie ok, bo v nie ma sąsiada swojego koloru w tej klice, więc na zewnątrz krawędzie NA PEWNO nie łączą tych samych kolorów.

159 $3SAT_3 \le_p PZPR$

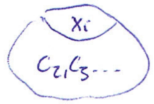
$(\neg x_2 \vee x_3 \vee x_n)$

$x_1 \dots x_k$ – zmienne $\qquad c_1 \dots c_n$ – klauzule

Dla $i \in \{1 \dots k\}$ $\quad P_i = \{ c_j : j \in \{1 \dots n\} \;\; x_i \in c_j \}$

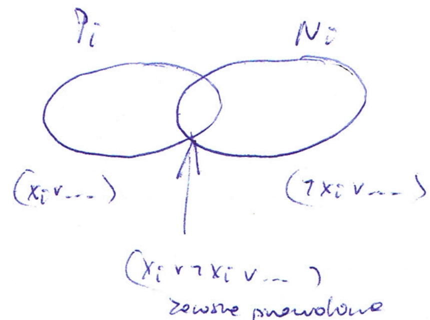
$N_i = \{ c_j : j \in \{1 \dots n\} \;\; \neg x_i \in c_j \}$

$$A = \bigcup_i \left[\{ S \cup \{x_i\} \mid S \subseteq P_i \} \cup \{ S \cup x_0 \mid S \subseteq N_i \} \right]$$



Bez straty ogólności

$P_i \cap N_i = \emptyset$



Tych podzbiorów jest mało

P_i nie stało się więc, max 2^3?

bo mamy $3SAT_3$

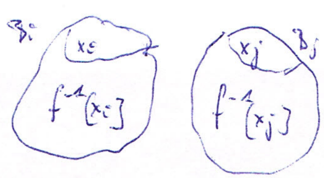
1) $3SAT_3$ ma rozw

$(x_1 \vee \neg x_2 \vee x_5) \wedge (x_2 \vee \dots)$

Możemy zrobić przyporządkowanie, która zmienna sprawia że klauzula jest prawdziwa: $f : \{c_1 \dots c_n\} \to \{x_1 \dots x_k\}$

$$f(c_j) = x_i \Rightarrow \begin{cases} x_i \in c_j \wedge x_i = 1 \\ \text{lub} \\ \neg x_i \in c_j \wedge x_i = 0 \end{cases}$$

$$B = \{ \{x_i\} \cup f^{-1}[x_i] \mid i \in \{1 \dots k\} \}$$



$B_i \cap B_j = \emptyset$

Czemu pokrywające przestrzeń?

1) każde x_i jest pokryte

2) każde c_j jest pokryte, bo f jest f-cją więc $c_j \in f^{-1}[x_i]$ dla pewnego i

2) Załóżmy że PZPR ma rozw, B jest rozw PZPR.

ROZŁĄCZNE

Chcemy odtworzyć wartościowanie

patrzymy, czy w c_i jest x_1 czy $\neg x_1$

jeśli x_1 to $x_1 = 1$ jeśli $\neg x_1$ to $x_1 = 0$

→ jest ok, wynika z definicji

klauzule są spełnione, bo ten x_i sprawia, że są one spełnione

Skoro PZPR ma rozw, to B jest pokryciem zbioru wszystkich, więc wszystkie klauzule muszą być w tych zbiorach B_i czyli spełniony jeśli spełniony x_i.

160 Wszystkie literały to samo: T albo F

160 ∈ PTIME

Dostajemy CNF

FAKT 1 $C_1 \wedge C_2 \wedge \dots \wedge C_n$

Jeśli w C_i mamy p i $\neg p$ to nie służy

$(p_1 \vee p_2) \wedge (p_1 \vee \neg p_2)$ — nie da się

→ mogą być klauzule: $(1 \vee 1 \vee 1) \wedge (0 \vee 0 \vee 0)$ to jest OK

Algorytm:

1 Bierzemy C_1 i ustawiamy tak, żeby były same T (lub F)

2 $(p_1 \vee p_2 \vee p_3) \wedge (\overset{1}{p_4} \vee \overset{1}{p_5}) \wedge (\underset{1}{\neg p_1} \vee \underset{1}{p_4}) \wedge (p_6 \vee p_7)$

potem to

$p_1, p_2, p_3 \to 0, 0, 0$

pomijamy

teraz to, bo jest niespełnione przez p_1

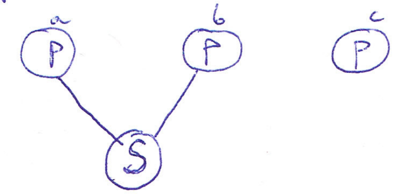
To w następnej iteracji

3 Wracamy do punktu 1 i bierzemy pierwszą klauzulę, która ma niezakonkretyzowane zmienne.

(problem przypomina 2-kolorowanie albo HornSAT)

UWAGA: To nie ma znaczenia, czy dla „pierwszej klauzuli" ustawiamy T czy F bo to symetryczne, dlatego to jest PTIME a nie 2^n.

162 $KOAL_j$



p_a i p_b nie mogą być w koalicji bo ubiegają się o to samo stanowisko

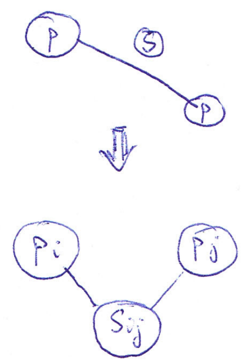
i → partia o i stanowisk

j → o stanowisko j partie

$KOAL_2$

$NIEZ_{\frac{n}{4}} \leq_p KOAL_2^{\infty}$

$NIEZ_{\frac{n}{4}} \leq_p NIEZ_{\frac{n}{2}} \leq KOAL_2$



(⇒) $G \to G'$

$V_i \to PARTIE_i$

$\langle V_i, V_j \rangle \to S_{ij}$

~~potrzebujemy~~ mamy $\frac{n}{2}$, chcemy $\frac{n}{4}$

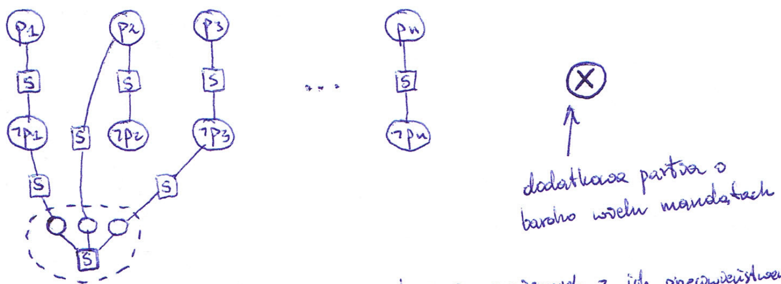
→ dodajemy puste wierzchołki nie połączone żeby mieć $\frac{n}{4}$

(⇐) Mamy $\frac{n}{2}$ wierzch. t.że nie mogą wspólnego stanowiska → czyli w G nie mają wspólnej krawędzi → czyli zbiór niezależne wierzchołków w grafie G.

163 $3SAT_3 \leq_p KOAL_3^3$

Dają nam formułę φ, utworzymy partie i stanowiska t. że

φ spełnialna $\Leftrightarrow$ da się wybrać koalicję większościową



$c_i = p_1 \vee \neg p_2 \vee p_3$ → podzerowe wystąpienie zmiennych z ich przeciwieństwami

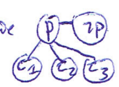
○ - partie

□ - stanowiska

FAKT: Skoro to $3SAT_3$, to każda zmienna występuje max 3 razy

więc każdy literał będzie miał max 3 stanowiska

... CHYBA ŻE występuje w φ 3 razy pozytywnie lub 3 razy negatywnie

(bo wtedy ma krawędzie ) ale wtedy wiemy jak go

zwartościować w φ, więc wykreślamy ten literał z φ otrzymując φ'

i pytamy o spełnialność φ'.

→ Dla każdej klauzuli mamy 3 partie, ale możemy wybrać tylko jedną partię.

→ Dla każdej zmiennej mamy 2 partie, ale możemy wybrać tylko jedną

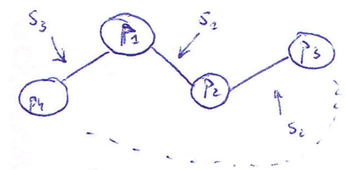
→ Wszystkie partie mają 1 mandat, a ⊗ ma x mandatów

$n+k+x = \frac{2n+3k+x}{2}$

↳ $x = k$, czyli ⊗ ma $k+1$ mandatów

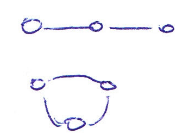
ŻEBY BYŁA WIĘKSZOŚĆ MUSIMY WYBRAĆ ⊗, N - CZYLI WARTOŚCIOWANIE, K - CZYLI KAŻDA KLAUZULA SPEŁNIONA

164 $KOAL_2^2 \in PTIME$



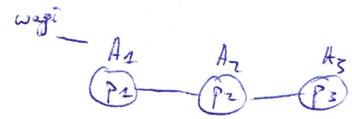
Albo cykl albo ścieżka prosta.

Łączymy dwie partie jeśli są o te same stanowiska

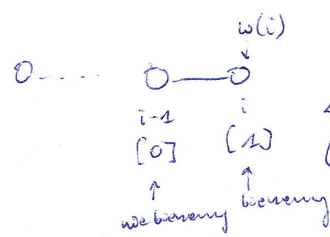


$KOAL_2^2$ to będzie zbiór ścieżek i cykli.

Pytanie: Czy możemy wybrać wierzch. t. żeby waga wynosiła połowę?

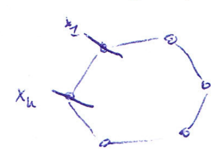


- programowanie dynamiczne działa ✓

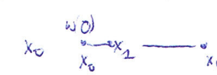


$$\begin{cases} dp[i][1] = dp[i-1][0] + w(i) \\ dp[i][0] = \max(dp[i-1][0], dp[i-1][1]) \end{cases}$$

Jeśli mamy cykl, to

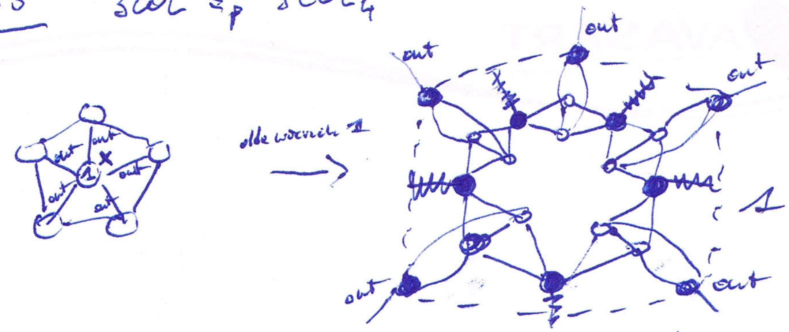


1° $x_1 = 0$ → ścieżka prosta

2° $x_1 = 1$ → dorzucamy x_0  też ścieżka

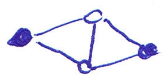
Czyli się da.

165 $3COL \leq_p 3COL_4$



● – te same kolory

st d → $4d$ wierzchołków



wymusza do tego podpięć

Każdy wierzchołek przerabiamy jak na obrazku powyżej. Tworzy się superwierzchołek, ale ~~wierzchołki~~ wierzchołki wyjściowe mają kolory te same. (takie jak przed przerobieniem)

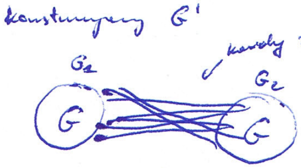
166 Alg $\chi(G)$ lub $\chi(G)+1$ to $P=NP$

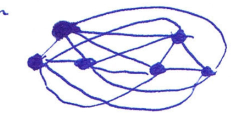
Jeśli istnieje, to skonstruujemy alg rozwiązujący 3COL (A 3-COL jest NP zupełny)

$$G \in 3COL \iff \chi(G) \leq 3$$

Algorytm:

* dostajemy G
* konstruujemy G'





→ Każdy wierzch z lewej musi mieć inny kolor niż te z prawej.

$\forall v \in G_1 \; w \in G_2 \quad l(v) \neq l(w)$ (dwa zestawy kolorów)

$\chi(G') = 2 \cdot \chi(G)$

$\chi(G')$ jest parzyste

* $Alg(G') = \{\chi(G'), \chi(G')+1\}$
  - → jedna z liczb jest parzysta
  - → jeśli $Alg(G)'$ nieparzyste to $\chi(G') = Alg(G') - 1$
  - → jeśli parzyste to $\chi(G') = Alg(G')$
* $\chi(G) = \frac{\chi(G')}{2} = \left\lfloor \frac{Alg(G')}{2} \right\rfloor$ → pozyskujemy liczbę dokładną

→ czyli jeśli $\left\lfloor \frac{Alg(G')}{2} \right\rfloor \leq 3$ to return TAK.

167 cykle się nie pokrywają a zapełniają cały graf.

NP-zupełne.

1. To jest NP – możemy sprawdzić czy cykle są ok.
2. Ham $\leq_p$ 167

Mamy G, dodajemy k izolowanych wierzchołków.

→ każdy z nich to oddzielny cykl.

Niech $k+1 \leq \frac{n+k}{c} < k+2$

- jeśli w G był cykl $\overset{Ham}{\iff}$ $\sigma(G)=1 \Rightarrow \sigma(G')=k+1$
- jeśli nie było, to $\sigma(G)>1 \Rightarrow \sigma(G')>k+1$

Pokażemy, że istnieje takie k (k całkowite) $\sigma(G') \geq k+2 > \frac{n+k}{c}$

$$k+1 \leq \frac{n+k}{c} < k+2$$

$$k \geq \frac{n-c}{c-1} \quad a \quad k > \frac{n-2c}{c-1}$$

$$\frac{n-c}{c-1} - \frac{n-2c}{c-1} = \frac{c}{c-1} > 1$$

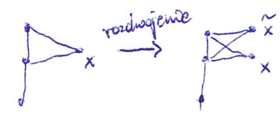
→ czyli zawsze ∃ takie k.

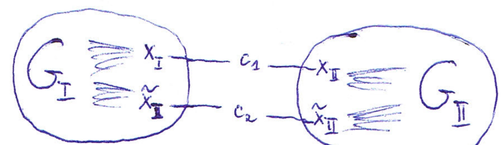
168 Jeśli istnieje taki algorytm to umiemy rozstrzygać cykl Hamiltona.

1. W G jest cykl Ham → $\sigma(G)=1$
2. W G nie ma Ham → $\sigma(G) \geq 2$

Będziemy rozstrzygać cykl Hamiltona:

1. Zróbmy graf G':





gdzie x i $\tilde{x}$ to dowolny wierzchołek w G „rozdwojony”

2. Jeśli w G nie ma cyklu Hamiltona to $\mathcal{A}$ zwróci pokrycie cyklowe większe niż 2 (jeśli nas oszuka to możemy to sprawdzić).
3. Jeśli w G jest cykl Hamiltona to $\mathcal{A}(G')$ zwróci:

1° 1 cykl → wtedy od razu mamy cykl Ham w G, bo mamy ścieżkę Hamiltona $x_I \to \tilde{x}_I$ czyli w G jest cykl Ham.

2° 2 cykle → wtedy wiemy, że któryś z cykli przechodzi przez c_1 i c_2, a skoro są dwa cykle, to BSO mamy:

cykl 1: $x_I \to c_1 \to x_{II} \xrightarrow{\text{cały } G_{II}} \tilde{x}_{II} \to c_2 \to \tilde{x}_I \to \dots \to x_1$

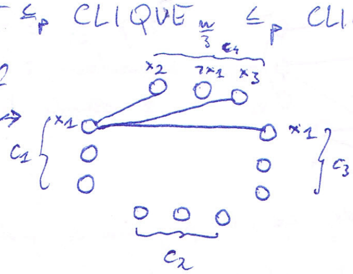
cykl 2: jakiś cykl w G_I

czyli mamy ścieżkę Hamiltona $x_{II} \to \tilde{x}_{II}$ w G_{II} czyli mamy cykl Hamiltona w G.

Zad 172 Oba problemy są NP-zupełne

1. $3SAT \leq_p CLIQUE_{\frac{n}{3}} \leq_p CLIQUE_{\geq \frac{n}{4}}$

$\varphi \xrightarrow{f}$



Łączymy ~~Nie~~ niesprzeczne literały z RÓŻNYCH klauzul

(⇒) φ spełnialna więc w każdej klauzuli jest literał wartościowany na T. ~~Wybi~~ Wybieramy je. Tworzą one klikę $\frac{n}{3}$.

(⇐) Istnieje klika $\frac{n}{3}$ czyli $\frac{n}{3}$ wierzchołków połączone ze sobą. Każdy MUSI być w innej klauzuli. Skoro są niesprzeczne, to możemy je zwartościować tak, żeby wszystkie były T.
⇒ φ spełnialna.

Redukcja $CLIQUE_{\frac{n}{3}} \leq_p CLIQUE_{\geq \frac{n}{4}}$:

→ być może jest tak, że mamy klikę $\frac{n}{3} - 1$ - wtedy φ nie jest spełnialne, a $\frac{n}{3} - 1 \geq \frac{n}{4}$.

Dodajemy więc k niepołączonych wierzchołków, tak żeby

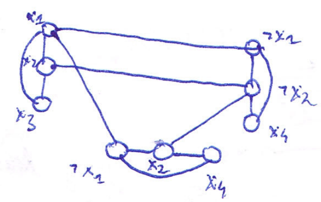
$$\frac{\frac{n}{3}}{n+k} = \frac{1}{4}(n+k)$$, wtedy φ spełn. w $CLIQUE_{\geq \frac{n}{4}}$

Dowód raczej oczywisty

stosunek połączonych do nie połączonych

2. $3SAT \leq_p IN\text{-}SET_{\frac{n}{3}} \leq_p IN\text{-}SET_{\geq \frac{n}{4}}$

$\varphi \xrightarrow{f}$



Łączymy w tych samych klauzulach i sprzeczne literały z różnych klauzul

(⇒) φ spełnialne ⇒ w każdej klauzuli przynajmniej jeden literał to T ⇒ możemy je wybrać, bo NIE MA między nimi krawędzi ⇒ istnieje IS o mocy $\frac{n}{3}$.

(⇐) istnieje IS o mocy $\frac{n}{3}$ ⇒ każdy wierzch. pochodzi z innej klauzuli ⇒ każdy dajemy na T i spełniamy φ.
(niesprzecznie wybieramy który)

Redukcja $IN\text{-}SET_{\frac{n}{3}} \leq_p IN\text{-}SET_{\geq \frac{n}{4}}$

podobnie, dodajemy k wierzchołków połączonych ze sobą i z każdym z G:

$$G \xrightarrow{f} \overset{G'}{\boxed{G \equiv K_k}}$$

wtedy żaden z tych k wierzchołków nie może być w zbiorze niezależnym; wymuszamy, żeby wszystkie były z G. Dowód oczywisty.

JFI20 zad 173 NAE-3SAT $\leq_p$ wesołe kolorowanie

Mamy formułę φ, skonstruujemy w wielomianowym czasie graf t.że

φ NAE spełn $\Leftrightarrow$ G, wesoło kolorowalny

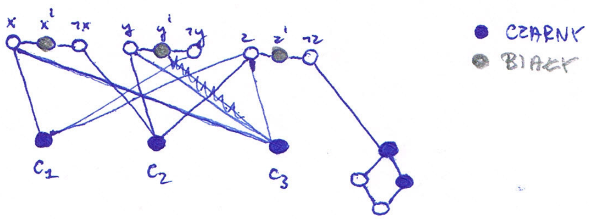
Konstrukcja: * Dla każdej zmiennej stwórz 

* Połącz klauzule z odpowiednimi literałami

* Jeśli któryś literał nie użyty, to połącz z gadżetem 

Przykład:

$$\varphi = \overset{c_1}{(x \vee \neg y \vee z)} \wedge \overset{c_2}{(\neg x \vee y \vee z)} \wedge \overset{c_3}{(x \vee y \vee z)}$$



φ - NAE-spełn $\Rightarrow \exists$, wesołe dwukolorowanie dla G

Klauzule na czarno, p' na biało, p i $\neg p$ wg σ.

φ-NAE-spełn $\Rightarrow$ 1. p' - stopień 2 więc $p=T$ (czarny) $\neg p=F$ (biały) lub na odwrót - wg wartościowania φ

2. p i $\neg p$ mają czarne klauzule i białe p'

3. c_i - skoro NAE-spełn to są sąsiedzi biali i czarni. ✓

$\exists$, wesołe dwukolorowanie $\Rightarrow \varphi$-NAE-spełn

1) p' ma st 2 więc $p=T$ i $\neg p=F$ lub na odwrót

→ czyli mamy wartościowanie.

2) Skoro $\forall i$ c_i ma sąsiada białego lub czarnego to ma T i F

→ czyli φ jest NAE spełn ✓

174

$(M) \wedge (M) \wedge \dots \wedge (T) = T$ ✓

$T \Rightarrow M \rightarrow F$

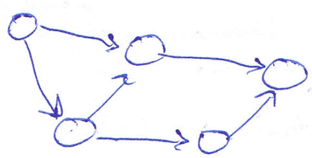
$T \Rightarrow F \rightarrow F$

$M \Rightarrow F \rightarrow F$

podobnie jak w rozw dla normalnego 2SAT ...

$x \vee y \equiv \neg x \Rightarrow y$

$\equiv \neg y \Rightarrow x$, robimy graf i silne składowe



$x \Leftrightarrow \neg x$ – nie da się
else się da

Ale tutaj możemy zwartościować
$x = M$ i $\neg x = M$ i jest ok.
(jedyne możliwe wartościowanie)

Obserwacja:

$F \rightarrow F \rightarrow \dots \rightarrow M \rightarrow M \rightarrow \dots \rightarrow T \rightarrow T$

Tylko tak może być

Algorytm: 1: szukamy SCC (silne spójne składowe) gdzie występują $x, \neg x$ i wartościujemy na M.

2: stawiamy się wszędzie gdzie trzeba M.

3: potem T w tych wierzch co wychodzą z M.???
(szukamy takiego żeby się dało na T ale żeby się z tych nie poszło $M \rightarrow F \rightarrow M$ ...)

Inne rozw:

jak czemuś dajemy M to możemy które usunąć z kolejnego miejsca
i mamy mniejszą klauzulę.

Dowód ???

NAJLEPSZE → Inne rozw 2:

1 T i same M jest ok

Idziemy się po klauzulach → to będzie prawdziwa
i po wszystkich klauzulach próbujemy zwartościować
i resztę zmieniamy na M.

Zgubiliśmy jakoś które że. $l = T$
Reszta będzie na M.

175 3SAT-N NP-zupełny

1) $3SATN \in NP$

$\varphi : p_1 \dots p_k$

wybieramy niedeterministycznie σ i sprawdzamy $\hat{\sigma}(\varphi)$

2) $3SAT \leq_p 3SATN$

$\varphi : p_1 \dots p_k$

dla każdej zmiennej $: (p_i \vee \neg p_i \vee F)$

żeby taka klauzula była spełniona to

p_i nie może być M. wymuszamy $= T$ lub $= F$

$$\overline{\varphi} = \varphi \wedge \bigwedge_i (p_i \vee \neg p_i \vee F)$$

φ spełn $\Rightarrow$ $\overline{\varphi}$ spełn

oczywiste ✓

$\overline{\varphi}$ spełn $\Rightarrow$ φ spełn

w $\overline{\varphi}$ na pewno nie ma $p_i = M$

więc znamy konkretne wartościowanie ✓

176 Tym razem bez stałych T, F, M.

Tutaj analogicznie:

$$\forall x, y \in Var(\varphi) \quad (\neg x \vee x \vee y) \wedge (\neg x \vee x \vee \neg y)$$ // n^2 par

to wymusza że nic nie jest M.

$$x = M \rightarrow (M \vee M \vee y) \wedge (M \vee M \vee \neg y)$$

T F

$$\overline{\varphi} = \varphi \wedge \bigwedge_{x, y \in Var(\varphi)} \left[(\neg x \vee x \vee y) \wedge (\neg x \vee x \vee \neg y) \right]$$

Dowód analogiczny:

($\Rightarrow$) φ spełn $\Rightarrow$ $\overline{\varphi}$ też spełn bo każda klauzula jest T

($\Leftarrow$) $\overline{\varphi}$ spełn $\Rightarrow$ nie może być wszystko na M $\Rightarrow$ ~~któraś~~ klauzula jest T

$\Rightarrow$ to nam określa wartościowanie, bo wtedy NIC nie może

być M $\Rightarrow$ φ spełn przy tym wartościowaniu.

177 $3SAT \le_p 2SAT_{\frac{7}{10}} \le 2SAT_{\frac{3}{4}}$

jako lista, klauzule z powtórzeniami.

3SAT. Każdą klauzulę $(a \vee b \vee c)$ zamieniamy na:

$$a \wedge b \wedge c \wedge d$$
$$\wedge (\neg a \vee \neg b) \wedge (\neg a \vee \neg c) \wedge (\neg b \vee \neg c)$$
$$\wedge (a \vee \neg d) \wedge (b \vee \neg d) \wedge (c \vee \neg d)$$

d - świeże zmienne

Ile możemy maksymalnie spełnić? W zależności od wartościowania a, b, c:

1° $a = b = c = F \rightarrow 6$

2° jedno jest T $\rightarrow 7$

3° dwie są T $\rightarrow 7$

4 wszystkie są T $\rightarrow 7$

$3SAT \le_p 2SAT_{\frac{7}{10}}$

Dowód $\Rightarrow$ $\Leftarrow$ - prosty dowód, wynika z konstrukcji

$2SAT_{\frac{7}{10}} \le 2SAT_{\frac{3}{4}}$

$\frac{7}{11}$ $\frac{7}{14}$

$\rightarrow$ dodajemy do tego pary $(a \vee \neg a) \wedge (b \vee \neg b)$

spełnione $9/12$ klauzul czyli $\frac{3}{4}$ wszystkich

179

NIE, $PTIME^M \not\subseteq NP \cup co\text{-}NP$

$\psi = \{ \langle \varphi, p \rangle \mid p \in \{SAT, \overline{SAT}\}, \varphi \in p \}$

instancja

enum (bool) SAT lub $\overline{SAT}$

$\overline{SAT} \le_p \psi \Rightarrow \psi \notin NP$, bo zał. że $NP \neq co\text{-}NP$

$SAT \le_p \psi \Rightarrow \psi \notin co\text{-}NP$ —"—

$\psi \in PTIME^M$ (trywialny algorytm)

JFIZO zad 180 $3COL \leq_p NAE\text{-}SAT$ ← przypominamy: jedno T i jedno F w każdej klauzuli

Dają nam graf G. Mamy skonstruować formułę φ_G t. że:

$$G \in 3COL \iff \varphi_G \in NAE\text{-}SAT$$

Konstrukcja formuły: $\varphi_G = \bigwedge_{v \in V} (r_v \vee g_v \vee b_v)$ (1)

każdy wierzch ma jeden kolor → $\bigwedge_{v \in V} \left[(r_v \vee g_v \vee t) \wedge (g_v \vee b_v \vee t) \wedge (b_v \vee r_v \vee t) \right]$ (2)

para krawędzi to różne kolory → $\bigwedge_{(u,v) \in E} \left[(r_u \vee r_v \vee t) \wedge (g_u \vee g_v \vee t) \wedge (b_u \vee b_v \vee t) \right]$ (3)

(⇒) Niech $t = true$. Wtedy (1) i (2) są NAE, bo każdy wierzch ma kolor. (3) jest NAE bo $t = true$ i graf jest dobrze pokolorowany, czyli nie ma możliwości, żeby $r_u = r_v = T$ (g, b)

(⇐) $G \notin 3COL \Rightarrow \varphi_G \notin NAE\text{-}SAT$

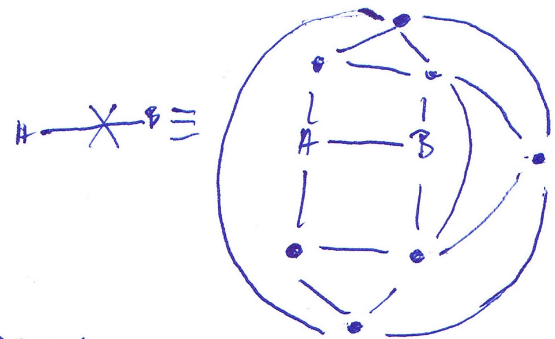
G nie ma kolorowania, czyli któreś dwa sąsiednie wierzchołki mają ten sam kolor. Np $r_u = r_v = T$. Wtedy t musi być F.

Ale w kolejnej klauzuli mamy $g_u = g_v = t = F$ ↯

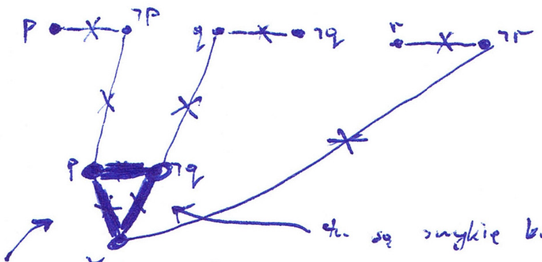
Czyli $\varphi_G \notin NAE\text{-}SAT$

181   $3NAESAT \leq_p$

$$\varphi = x \vee \neg y \vee z$$



→ Gadżet wymusza, że $A \neq B$ (bo inaczej byłby trójkąt w środku tego samego koloru)



← dla każdej zmiennej takie coś

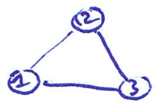
← to są wszystkie krawędzie, a nie gadżety.

dla każdej klauzuli trójkąt, NAE SAT więc ok

182

Weźmy graf 4-kolorowalny

-> dowolny trójkąt



<- 3 różne kolory

kolory

1 2 3 4

nowe kolorowanie -> 5 6

Zawsze będzie tak, że w każdym trójkącie będziemy mieli dwa kolory, czyli nie wszystkie wierzch. tego samego koloru.

-> Jeśli graf 4-kolorowalny to ZAWSZE się da pokolorować dwoma kolorami tak, żeby trójkąty nie miały wszystkich wierzch. w tym samym kolorze.

Odp się zmienia: To już nie jest NP-zupełne tylko trywialne, odp zawsze to TAK.

183

$f: N \to N$, różnowartościowa, $\forall n$ $|n| = |f(n)|$

$\Downarrow$

funkcja jest „na"

czyli to bijekcja.

$A = \{ \langle x, y \rangle \mid \underbrace{f^{-1}(x)} < y \}$

arg. który dający do f, żeby otrzymać x.

f jest wielomianowa, f^{-1} nie jest.

$\Rightarrow$ $NP \cap co\text{-}NP \neq PTIME$.

$A \in NP$

* weźmy x, y
* zgadnij z, $|z| = |x|$
* $f(z) = x \to f^{-1}(x) = z$
* spr czy $z < y$
  - -> TAK ret 1
  - -> NIE ret 0

$\bar{A} \in NP \Rightarrow A \in co\text{-}NP$

| IDENTYCZNIE

* spr czy $z \geq y$

żeby nie wpaść, że $A \in PTIME$

$|x| = n$, $|f^{-1}(x)| = n$, $|z| = n \land f(z) = x$

$z \in \langle 2^n, 2^{n+1})$

$\langle x, z \rangle \in A \to \underset{z}{f^{-1}(x)} < z'$ -> możemy znaleźć w czasie $\log(2^n)$ (bin search)

sprzeczność że $NP \cap coNP \neq PTIME$

JFIZO zad 184 $f:\{0,1\}^* \to \{0,1\}^*$ obliczalna w czasie $p(w) \Rightarrow f^{-1}$ też?

NIE. Pokażemy kontrprzykład. Pomysł: Unarne → binarne ≡ logarytmiczne długości
Binarne → unarne ≡ wykładnicze długości

Niech f:
$$f: \begin{cases} f(\varepsilon) = \varepsilon \\ f(0^{2n}) = 0^n & \leftarrow \text{same zera} \\ f(0^{2n+1}) = bin(n+1) & \leftarrow \text{jakieś kodowanie binarne} \\ f(0^k 1 w) = 0^{k+1} 1 w, \ k \geq 0 & \leftarrow \text{jakieś niepoprawne kodowanie binarne } 0\ldots0\,bin \end{cases}$$

Wtedy f^{-1}:
$$f^{-1}: \begin{cases} f^{-1}(\varepsilon) = \varepsilon \\ f^{-1}(0^n) = 0^{2n} \\ f^{-1}(\overset{n}{bin}) = 0^{2n+1} \\ f^{-1}(0^{k+1} 1 w) = 0^k 1 w, \ k \geq 0 \end{cases}$$

→ Obliczenie $f(w)$ zajmuje wielomianowo dużo czasu w stosunku do $|w|$

→ Obliczenie $f^{-1}(v)$ zajmuje wykładniczo dużo czasu względem $|v|$, bo trzeba wypisać na taśmie MT wykładniczo długi ciąg zer

unarne → binarne logarytmiczne długości
binarne → unarne wykładnicze długości

185

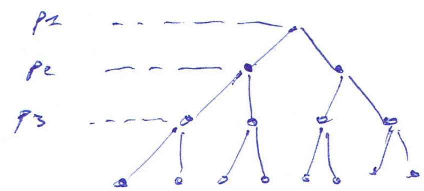
1° Można przechodzić dowolnie dużo razy.

→ Można wiele przemyśleć cały czas
→ czyli można wybrać wartościowanie.

Patroklesowi się uda ⇔ φ jest tautologią.
czyli ten problem jest co-NP-zupełne.

2° Można może przechodzić max 3 razy.

Można może wybrać $\binom{n}{3}$ zmiennych (wielomian)



Można ↙ Patroklos ↘
Ile jest ścieżek w tym drzewie?

Można max 3 w lewo

Głębokość − n, w kolejnym kroku mamy o

Ile jest wartościowań do sprawdzenia? Wielomianowo.

Patroklos jest jak rozwiązywanie SATA, czyli możemy tu sprawdzić te wartościowania (których jest wielomianowo) → czyli to jest NP-zupełne.

$3SAT \leq_p 3Moż$

$3SAT \leq_p 6SAT \to \forall (a \vee b \vee c) \to (a \vee b \vee c \vee \underbrace{x \vee y \vee z}_{}$)

Można mieć to zwartościowanie na F bo mamy przyrzekać.

186 QSPACE - zupełne (TRUDNE)

Mojme $\le \frac{n}{2}$ razy

1 jest w P-SPACE

pełne drzewo ma głębokość $2n$

2. redukcja z QBF

$$\exists p_1 \forall q_1 \exists p_2 \dots \varphi(p_1 \dots q_n)$$

2n zmiennych

Tworzymy formułę 4n zmiennych

$$\varphi' = \left[\bigwedge_{i=1}^{n} (q_i \vee q_i') \wedge \varphi\right] \vee \left[\bigvee_{i=1}^{n} \left((q_i \wedge q_i') \vee p_i'\right)\right]$$

QBF ⇒ Partokles ma strategię wygrywającą

$p_i' - 1$

p_i - tak jak QBF

q_i $q_i' - 1$

¬QBF ⇒ Mojra ma strategię wygrywającą.

φ nieprawdziwe, czyli drugie z alternatywy musi być prawdziwe.

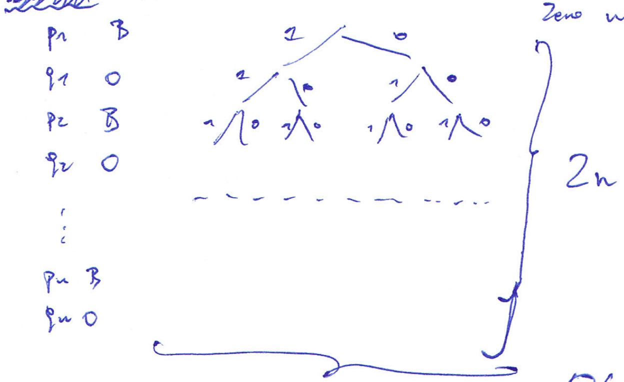
Mojra → 2n ruchów, ok, $q_i' = F$ $p_i' = F$

Przykład: $\varphi = \exists p_1 \forall q_1 \; p_1 \Leftrightarrow q_1$

φ może być spełnione.

187 - ~~[illegible]~~ DRZEWO ROZGRYWKI

Zero może 3 razy.



$O(n^7)$

wrzucić w tym przyp. (bo liniowa wysokość)

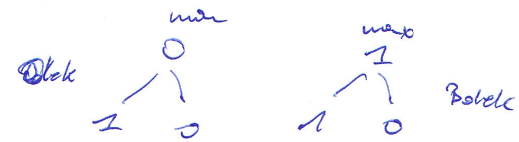
Drzewo jest wielomianowej wielkości.

$$\left(\binom{n}{3} + \binom{n}{2} + n + 1\right)^2 = O(n^6)^*$$

3 zera, 2 zera, 1 zero, zero zer

liczba liści

Algorytm min max



→ backtracking od dołu

i w końcu dowiemy się czy była strategia wygrywająca.

$O(n^7)$ wartościowań do sprawdzenia.

Możemy to zrobić w PTIME.

188 TQBF $\leq_p$ P188

Chcemy skonstruować formułę taką, żeby gracz egzystencjalny miał zawsze wybór (niezależnie od tego jak zagra gracz $\forall$).

$$\begin{array}{cc|cc|}
1 & 0 & & \\
p_1 & p_1' & p_2 & p_2' \\
q_1 & q_1' & q_2 & q_2' \\
0 & 1 & &
\end{array} \rightsquigarrow \dots$$

tyle samo 1 dla $\vec{p}$ i $\vec{q}$ więc gracz $\exists$ ma znów dowolny wybór dla p_2

1| $\begin{array}{cc} 0 & 1 \\ p_1 & p_1' \\ q_1 & q_1' \\ 0 & 1 \end{array}$ 2| $\begin{array}{cc} 0 & 1 \\ p_1 & p_1' \\ q_1 & q_1' \\ 1 & 0 \end{array}$ 3| $\begin{array}{cc} 1 & 0 \\ p_1 & p_1' \\ q_1 & q_1' \\ 0 & 1 \end{array}$ 4| $\begin{array}{cc} 1 & 0 \\ p_1 & p_1' \\ q_1 & q_1' \\ 1 & 0 \end{array}$

Chcemy zmusić gracza $\exists$ żeby zawsze dawał $p_i \neq p_i'$
— " — $\forall$ — " — $q_i \neq q_i'$

Mając formułę φ, skonstruujemy φ' t.że $\exists_{p_1} \dots \forall_{q_n} \varphi$ jest prawdziwe wtw gracz egzystencjalny ma strategię wygrywającą dla φ'.

$$\varphi' = \left(\varphi \wedge \bigwedge_i \neg(p_i \Leftrightarrow p_i')\right) \vee \bigvee_i (q_i \Leftrightarrow q_i')$$

gracz $\exists$ musi dawać RÓŻNE p_i, p_i', żeby były tu same prawdy we wszystkich koniunkcjach

gracz „$\forall$" musi dawać RÓŻNE q_i, q_i' bo inaczej któraś z tych alternatyw zostanie spełniona, czyli φ' będzie prawdziwa

Dowód oczywisty: zdanie $\exists \dots \forall \varphi$ prawdziwe $\Leftrightarrow$ $\exists$ ma strategię wygrywającą.

189 PSPACE-zupełne TQBF $\leq_p$ P189

$$\begin{array}{c|c} B & O \\ \hline 1 & \to 1, 0 \\ 0 & \to 0 \end{array} \qquad \begin{array}{c|c} O & B \\ \hline 1 & \to 1 \\ 0 & \to 0, 1 \end{array}$$

TQBF: $\exists x \forall y_1 \dots \exists x_n \forall y_n \varphi$

$\exists x_1 \to x_1, x_1', x_1''$ | $\forall y_1$
B O B

chcemy, żeby B mógł tutaj dawać 1

$\forall y_1 \to y_1, y_1', y_1''$
O B O

chcemy, żeby O(?) dał 0

$$F = \left(\varphi \vee \bigvee y_i''\right) \wedge \bigwedge x_i''$$

(nie używamy x_i', y_i')

O(?) wszystko musi tutaj na ‚0'

B musi wszystko tutaj na ‚1'

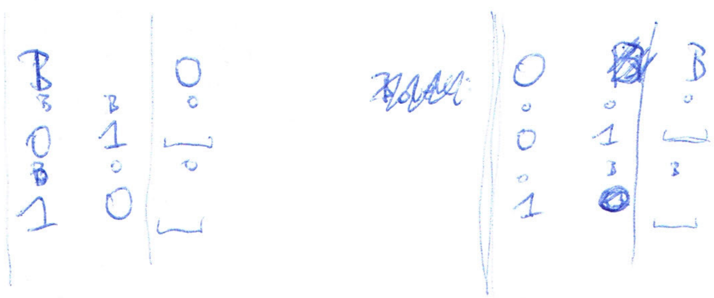
TQBF $\Rightarrow$ Gra (strategia)

strategia: zaczynamy jak w TQBF, wszystkie wektory z x_i'' na 1.
(jeśli któryś y'' będzie ‚1' to nie jak w TQBF tylko wszystkie x_i'' na 1 i też wygrywamy)

$\neg$TQBF $\to$ $\neg$ Gra

tym razem gracz II ma strategię wygrywającą.
$\Rightarrow$ gdy I wszystkie y_i'' na 0.

190



1) Bobek $(p_1 \vee p_2) \wedge ((p_1 \wedge p_2) \vee \boxed{2})$

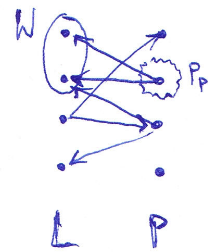
2) Olek $(\neg p_3 \vee \neg p_4) \vee ((\neg p_3 \vee \neg p_4) \wedge \boxed{1})$

$\vdots$

$\varphi(p_1, p_3, \dots, p_{n-1})$

zmienne z TQBF: $p_1, p_3, p_5, \dots$

Zad 191



$P_L = \{ v \in L \mid$ gracz P ma strat. wygrywającą (z wierzch v) $\}$ // ruch L

$P_P = \{ v \in P \mid$ gracz P ma strat. wygrywającą (z wierzch v) $\}$ // ruch P

Algorytm:

* $P_L \leftarrow W$
* do $|V|$ times $\{$

$P_P \leftarrow P_P \cup \{ v \in P \mid \exists w_1, w_2 \in P_L \; E(v,w_1) \wedge E(v,w_2) \}$

$P_L \leftarrow P_L \cup \{ v \in L \mid \forall w_1, w_2 \in P \; E(v,w_1) \wedge E(v,w_2) \Rightarrow w_1 \in P_P \vee w_2 \in P_P \} \}$

* if $c_0 \in P_L$ return P
* else return L

Obserwacje:

1) Dla wierzch w P_P gracz P ma ruch, więc jeśli są takie 2 wierzch, z których ma strat. wygrywającą to je wybiera i gracz L i tak przegra.

2) Dla wierzch w L gracz L ma ruch więc jeśli nie jest w stanie wybrać 2 wierzch z których P nie ma strat wygrywającej to P sobie wybiera ten, z którego ma.

3) Po i-tej iteracji w P_P są wierzch z których P może wygrać po $\leq 2i-1$ ruchach, a w P_L te z których może po $\leq 2i$ ruchach. zatem wystarczy $|V|$ iteracji

192 $A \in PSPACE \Leftrightarrow \overline{A} \in PSPACE \Leftrightarrow \overline{A} \in NPSPACE$

Zbudujemy niedeterministyczną ~~[illegible]~~ maszynę Turinga o dodatkowej wielomianowej pamięci t.że ~~[illegible]~~ zwraca $1 \Leftrightarrow \exists w \notin L_\varphi$.

Lemat: jeśli istnieje słowo „w" dla NFA, które nie jest akceptowane to jest niedłuższe niż $2^{|Q|}$. Gdyby było dłuższe to moglibyśmy je skrócić, bo powtórzyłyby się 2 razy jakieś zbiory stanów.

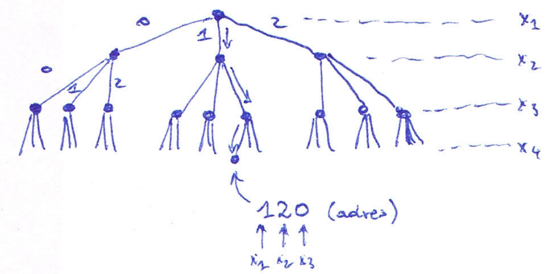
M:
- wczytaj φ
- zbuduj NFA z φ // wiedzieliśmy jak to zrobić w cz. I jest on wielomianowo duży (DFA mógłby być wykładniczo duży)
- $S \leftarrow \{q_0\}$
- próbuj zgadnąć słowo czyli $2^{|Q|}$ razy:
  - $a \leftarrow$ zgadnij literkę
  - $S \leftarrow \{q \in Q \mid \exists r \in S .\ \delta(r, a, q)\}$ // możliwe osiągalne stany automatu
  - if $S \cap F = \phi$: // zgadliśmy słowo t.że nie da się uzyskać stanu akceptującego
    - return 1
- return 0 // nie udało się zgadnąć słowa

Co musi pamiętać M?

1) stany automatu – wielomianowo ✓
2) licznik kroków – $\log(2^{|Q|}) = |Q|$ – wielomianowo ✓

∎

JFI20 zad 193 $\exists! x_1 \exists! x_2 \dots \varphi(x_1 \dots x_n) \in PSPACE$

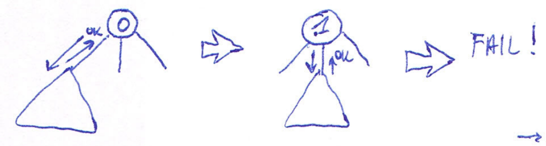


Mamy wykładniczo dużo węzłów $\to 3^n$ a tylko wielomianową pamięć.

Musimy pamiętać:

1 adres wierzchołka gdzie jesteśmy: $O(n)$ pamięci

2 czy już spełniliśmy φ przy poprzednim wartościowaniu dla x_i $\to$ czyli zmienne boolowskie dla każdego wierzchołka na ścieżce w adresie: $O(n)$ pamięci.



FAIL! dwa wartościowania dla x_i spełniają φ.

$\to$ przechodzimy wyżej mówiąc, że nie ok.

Formuła jest prawdziwa, jeśli w korzeniu po przejściu będzie dokładnie jedno wartościowanie dla x_1 spełnione.

Dla $x_i \in \{0,1\}$ się nie zmienia nic. Tylko mamy drzewo binarne.

UWAGA: Warto robić takie zadania ($X \in PSPACE$) pokazując procedurę rekurencyjną o głębokości ~~[illegible]~~ wielomianowej $\to$ na stosie zapiszemy ~~[illegible]~~ wielomianowo pamięci, czyli PSPACE

Zad 196

1° Ten problem $P196 \in PSPACE$

IsTrue (F, σ)
  match F with
  | φ → Compute(φ, σ)
  | ∃::F' → IsTrue(F', 0::σ) ∨ IsTrue(F', 1::σ)
  | ∀::F' → IsTrue(F', 0::σ) ∧ IsTrue(F', 1::σ)
  | ∃! ::F' → [IsTrue(F', 0::σ) ∨ IsTrue(F', 1::σ)]
      ∧ ¬[IsTrue(F', 0::σ) ∧ IsTrue(F', 1::σ)]

→ głębokość rekursji liniowa, pamiętamy tylko wartościowanie.

2° Ten problem jest PSPACE-trudny

$\forall X \in PSPACE \quad X \leq_p TQBF \leq_p P196$

to wiemy

w TQBF mamy ∃∀∃∀...
a tutaj $Q_1 Q_2 Q_3$...
więc to jest ogólniejsze.

Jeśli umielibyśmy rozwiązać P196, to również TQBF.
(ta sama klasa złożoności)

∃ ∀ ∃ ∀ ...
Q_1 Q_2 Q_3 Q_4 ...

197 alt PTIME = PSPACE

$w \# \overbrace{w_1}^{\text{1 gracz}} \# \overbrace{v_1}^{\text{2 gracz}} \# w_2 \# v_2 \# \ldots \# w_{p(n)} \# v_{p(n)} \#$ — $s_{p(n)}$

$n = |w|$ $\quad |w_i|, |v_i| = p(n)$

GI wygrywa gdy $s_{p(n)} \in B$, $B \in PTIME$

1) alt PTIME $\subseteq$ PSPACE

$x \in$ alt PTIME

FAKT 1 $\quad |s_{p(n)}| = n + 2 \cdot p(n) + 2 \cdot p(n) \cdot p(n)$ → wielomian od $|w|$

$|w|$, $\#$, każdy z graczy, słowa długości $p(n)$

FAKT 2 $\quad B \in PTIME$, $q(n)$ wielomian dla B.

$q(|s_{p(n)}|)$ → to jest wielomian (bo złożenie wielomianów)

→ robimy drzewo grafu, rekurencja → głębokość $p(n)$

czyli $\in$ PSPACE

2) PSPACE $\subseteq$ alt PTIME

TQBF $\in$ alt PTIME (a TQBF jest PSPACE zupełny)

$\exists x_1 \forall y_1 \ldots \exists x_n \forall y_n \; \varphi(\ldots)$ → kwantyfikatory na przemian (dwóch graczy)

$\forall a \forall b \; \varphi \to \forall a \exists x \forall b \; \varphi$
$\exists a \exists b \; \varphi \to \exists a \forall x \exists b \; \varphi$ (x nie ma w formule)

B - język prawdziwych formuł (z podstawionymi wartościami za zmienne)

$\underbrace{w}_{\varphi} \# \underbrace{\qquad\qquad}_{\text{wartościowania zmiennych}}$

Dla $\varphi \in$ TQBF ⇒ $\exists_1 p$ t. że Gra(φ, p, B)

Jeśli $\exists x \; \forall y$ to jest strategia wygrywająca.

3) alt PTIME zamknięty na redukcje

$A \in$ alt PTIME $\wedge$ $X \leq_p A$ ⇒ $X \in$ alt PTIME

Gra$(\cdot, p_A, B_A)$ $\quad f, p_f$ $\quad$ chcemy pokazać grę Gra$(\cdot, p_X, B_X)$

B_X: weryfikuj: $w \# \ldots \#$
oblicz $f(w)$
odpal B_A na słowie $f(w) \# \ldots \#$

$p_X = p_A \circ p_f$ → $p_X(w) = p_A(|f(w)|) = p_A(p_f(|w|))$

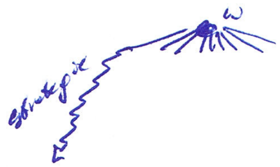
198

$w \to w\#w_1 \to w\#w_1\#v_1 \to w\#w_2\#v_1 \to w\#w_2\#v_2 \to w\#w_3\#v_2 \to \dots$

długość tych słów jest taka sama.

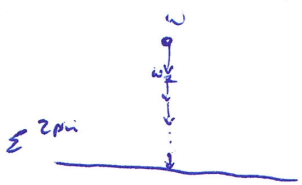
$p(n)$

$\to |\Sigma|^{2 \cdot p(n)}$ → tyle wszystkich stanów.



Strategia to drzewo
→ jak w pompowaniu

Zał., że ma strategię wygrywającą.



Zał., że to drzewo jest skończone więc skończ. stanów

1) w stanie pierwszym g1 nie ruch
→ ale już wtedy jest ten stan, czyli to nie było najkrótsze rozwiązanie wygrywające ↯

2) gracz 2 nie ma ruchu ⇒ jest ok jak się zapętla, czyli stan wygrywający jest nieosiągalny ∞ kroków.

JFIZO zad 202

$P_1(x,y) = R(x,y)$ - - - - - - - - - - - - - - - - 0

$P_2(x,y) = \exists z.\ R(x,z) \wedge R(z,y)$ - - - - - - - - - - $0+1=1$

$P_4(x,y) = \exists z. \forall a. \forall b. (a=x \wedge b=z) \vee (a=z \wedge b=y) \Rightarrow P_2(a,b)$ - - - $1+3=4$

$P_8(x,y) = \exists z. \forall a. \forall b\, (a=x \wedge b=z) \vee (a=z \wedge b=y) \Rightarrow P_4(a,b)$ - - - $4+3=7$

$P_{16}(x,y) = \exists z. \forall a. \forall b\, (a=x \wedge b=z) \vee (a=z \wedge b=y) \Rightarrow P_8(a,b)$ - - - $7+3=10$

$P_k(x,y)$ - da się dojść w k krawędziach

203

$$\exists x_1 \exists x_2 \dots \overset{\exists}{\forall y_1 \dots \forall y_2} \dots \exists x_k \ \varphi$$

Zamieniamy ogólne na egzystencjalne.

1) $\exists x_1 \dots \exists x_{n-2} \forall y_1 \forall y_2 \equiv$

~~$\exists x_1 \exists x_2 \dots \exists x_{n-2} (\vec{y}, 0, 0) \wedge \exists x_1 \dots \exists x_{n-2} (\vec{y}, 1, 0) \wedge \dots$~~

przypisujemy y_1 i y_2 wartościowanie i ma być ok

$$\exists x_1 \dots \exists x_{n-2} \left(\varphi(\vec{x}, \underset{y_1}{0}, \underset{y_2}{0}) \wedge \varphi(\vec{x}, \underset{y_1}{1}, \underset{y_2}{0}) \wedge \dots \right)$$

2) $\forall y_1 \forall y_2 \exists x_1 \dots \exists x_{n-2} \ \varphi \equiv$

$$(\exists x_1 \dots \exists x_{n-2} \varphi(\vec{x}, 0, 0)) \wedge \dots \wedge (\exists x_1' \dots \exists x_{n-2}' \varphi(\vec{x}', 1, 0))$$

3) $\exists x_1 \dots \exists x_i \forall y_1 \exists z_1 \dots \exists z_k \forall y_2 \exists v_1 \dots \exists v_s \equiv$

przekształcenie jak w 1

jak w 2 jak w 1 jak w 2

$$\equiv \exists x_1 \dots \exists x_i \varphi(\vec{x_{1..i}}, 0) \wedge \varphi(\vec{x_{1..i}}, 1)$$

$$\wedge \exists z_1 \dots \exists z_k \varphi(\vec{z_{1..k}}, 0) \wedge \exists z_1' \dots \exists z_k' \varphi(z_1' \dots z_k', 1)$$

$$\wedge \dots$$

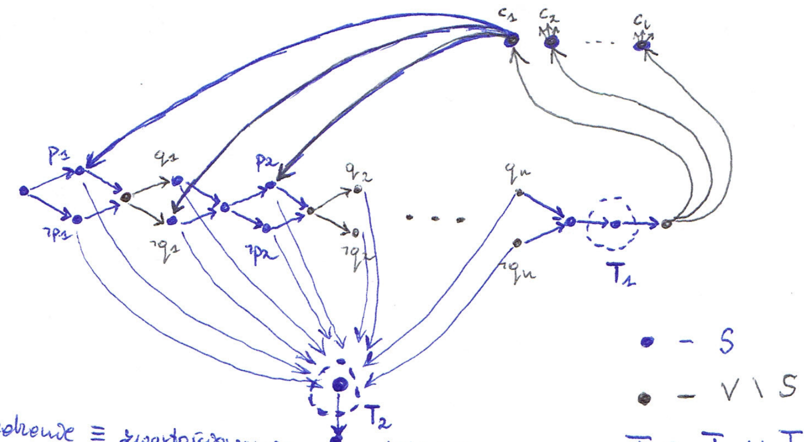
Dostajemy $\exists x_1 \dots \exists x_n \varphi(x_1 \dots x_n)$ czyli pytanie o spełnialność formuły, czyli SAT, czyli NP-zupełne. SAT ≤ 203

Zad 205

$TQBF \le_p GwZ$

Dostajemy instancję problemu TQBF, F. Zbudujemy taką instancję problemu GwZ, $\langle G, S, T, v_0 \rangle$ że:

F prawdziwa $\Leftrightarrow$ gI ma str. wygrywającą.



Odwiedzenie ≡ zwartościowanie na FALSE. $T = T_1 \cup T_2$

($\Rightarrow$) F prawdziwa $\Rightarrow$ GI wartościuje p_i a Zły Przewodnik q_i i da się tak, że φ spełniona $\Rightarrow$ Zły Przewodnik niezależnie którą klauzulę wybierze to nie ma w niej FFF $\Rightarrow$ GI może się przedostać z $c_i \to p_j \to T_2$ i odwiedzi wszystko i wygra.

($\Leftarrow$) F fałszywa $\Rightarrow$ można tak wartościować $\vec{q}$, że φ fałszywa, czyli wybrać klauzulę c_i t.że FFF $\Rightarrow$ gI nie może się stamtąd ruszyć, czyli nie odwiedzi T_2 i przegra.

207

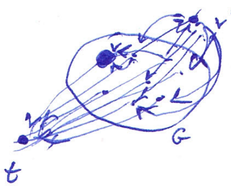
HAMCYC $\le_p$ P207 czyli 207 NP-~~zupełny~~ trudny

czyli $207 \notin P$

$(P \neq NP)$

Dają nam HAMCYC – graf G, zbudujemy $\langle G', S, T, v_0 \rangle$

SKIEROWANY



$T = \{t\}$

$\forall u \in G$ mamy krawędź $\{u, t\}$

$\forall u \in G$ t. że jest krawędź $\{u, v\}$

dodajemy krawędź $\{u, v'\}$ i ~~...~~

$\forall u \in G$ dodajemy $\{v', u\}$

$v_0 = v$ (nie ma krawędzi $v \to v'$)

Zbiór graczy wygrywa $\Leftrightarrow$ istnieje cykl Hamiltona.

($\Leftarrow$) przechodzimy ten cykl i idziemy do v', drugi przez nie nie gubię się, bo wierzchołki były

($\Rightarrow$) gubię może się skończyć wcześniej?

$\to$ nie, gubię w v'

a to oznacza, że byłem już wszędzie

(cykl po wszystkich wierzchołkach)

$\to$ czyli cykl Ham. istnieje.

Zad 208

PKRÓL $\in$ NP – dają nam przypisanie i sprawdzamy, czy nie ma kolizji $\forall v \in V$

3COL $\le_p$ PKRÓL

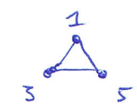
Dają nam graf G, stworzymy kraj (graf G') t. że

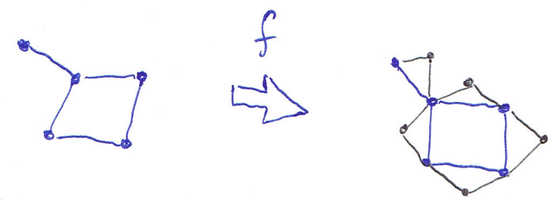
G 3-kolorowalny $\Leftrightarrow$ król może ustalić targi zgodnie z regułami

Gadżet:

musi być pokolorowany kolorami 1, 3, 5

tj. nie można używać 2 i 4





Każdej krawędzi dorabiamy jeden wierzchołek tworząc K_3.

Wtedy G' jest pokolorowany tylko trzema kolorami i każda krawędź ma wierzchołki różnych kolorów.

($\Rightarrow$) G 3-kolorowalny $\Rightarrow$ G' kolorujemy używając 1, 3, 5 i dla dorobionych wierzch. na krawędziach używamy trzeciego koloru.

($\Leftarrow$) Król umie wybrać w G' $\Rightarrow$ musi wybierać 1, 3, 5 przez te kliki K_3 $\Rightarrow$ oryginalny G jest pokolorowany trzema kolorami i sąsiadujące wierz. są różnego koloru $\Rightarrow$ G jest 3-kol. ✓