139

T: Jeżeli A jest w NP, to istnieje NTM_A “zgadująca” rozwiązanie A

Jeżeli A jest w NP, to da się rozstrzygnąć poprawność rozwiązania w czasie wielomianowym. Niech TM_A – maszyna Turinga, która wczytuje n, m, gdzie n – instancja A, m – rozwiązanie, po czym sprawdza poprawność. m musi być wielomianowe, no bo w przeciwnym razie samo wczytywanie zajęłoby zbyt długo więc istnieje p, Działanie musi być wielomianowe, bo problem weryfikacji A jest w P, więc istnieje q, którym da się ograniczyć czas działania TM_A.

Działanie NTM_A:
- niedeterministycznie “zgadnij” m
- wczytaj n
- odpal TM_A dla (n, m)

140

T: A’ jest w NP (A’ – rzut A z zadania)

Niech TM_A – maszyna Turinga rozstrzygająca A w PTIME.
Korzystamy z 139 – skonstruujemy NTM rozpoznającą A’ w PTIME.

Działanie:
- wczytaj n
- niedeterministycznie “zgadnij” $m \leq p(|n|)$.
- odpal TM_A dla (n, m)

141

T: Dla danego B w NP istnieje A, które jest w P
Niech NTM_B – niedeterministyczna maszyna Turinga rozstrzygająca B w PTIME.

A = { (n, m) | $n \in B \land$ m – przebieg NTM_B rozstrzygający n }

Skonstruujemy maszynę Turinga rozstrzygającą A w PTIME.

Działanie:
- wczytaj n, m
- zasymuluj przebieg NTM_B dla n

142

T: 5SAT $\leq_P$ 3SAT

Weźmy instancję problemu 5SAT – formułę w CNF ϕ. Jest ona koniunkcją klauzul postaci:

$$a1 \lor a2 \lor a3 \lor a4 \lor a5$$

Każdą taką klauzulę jesteśmy w stanie zastąpić 3 klauzulami postaci:

$$(a1 \lor a2 \lor z1) \land (\sim z1 \lor a3 \lor z2) \land (\sim z2 \lor a4 \lor a5)$$

Więc dla instancji 5SAT ϕ o n klauzulach tworzymy instancję 3SAT o 3n klauzulach przy pomocy 2n nowych zmiennych.

143

T: 3SAT $\leq_P$ STASI

https://www.nitt.edu/home/academics/departments/cse/faculty/kvi/NPC%20DOMINATING%20SET.pdf

144

T1: H $\leq_P$ Hd

Mamy instancję H G, budujemy instancję Hd G', gdzie zastępujemy każdą krawędź G dwoma krawędziami skierowanymi.

T2: Hd $\leq_P$ H

Mamy instancję Hd G, konstruujemy instancję H G':

- Dla każdego v ∈ V_G dajemy do V_G' v_in, v_mid, v_out
- Dla każdej (v, v') ∈ E_G dajemy do E_G' {v_out, v'_in}

Dowód, że to redukcja – w 1 stronę łatwe, w drugą:
G' ma cykl Hamiltona ⇒ G ma skierowany cykl Hamiltona.
Załóżmy, że G' ma cykl Hamiltona, weźmy ten cykl. Każdy v_mid musi się pojawić w cyklu, a jako że ma on stopień 2, musi być pomiędzy wierzchołkami v_in i v_out. Z kolei v_in ma

krawędzie inne niż {v_mid, v_in} tylko do wierzchołków typu _out (i odwrotnie dla v_out), więc znaleziony cykl musi wyglądać na jeden ze sposobów:

1. v1_in – v1_mid – v1_out – v2_in – … – vn_mid – vn_out
2. v1_out – v1_mid – v1_in – v2_out – … – vn_mid – vn_in

Załóżmy 1., odpowiada to cyklowi w grafie skierowanym G v1 → v2 → … → vn.

145

T: HORNSAT ∈ P

https://www.wikiwand.com/en/Horn-satisfiability

146

T: 2SAT ∈ P

https://www.wikiwand.com/en/2-satisfiability

147

T: 3SAT $\leq_P$ 3SAT_3

Mamy instancję 3SAT ϕ, skonstruujemy instancję 3SAT_3.
Dla każdej zmiennej x, która pojawia się więcej niż 3 razy:

- zastępujemy kolejne wystąpienia nowymi zmiennymi x1, x2, …, xk,
- dodajemy klauzule (~x1 ∨ x2), (~x2 ∨ x3), …, (~xk ∨ x1), co tworzy “krąg implikacji”: x1 ⇒ x2 ⇒ … ⇒ xk ⇒ x1, więc wszystkie muszą być 0 albo 1.

153

T: SAT_2 ∈ P

https://cs.stackexchange.com/questions/86730/show-that-the-sat-problem-for-cnf-formulas-with-at-most-two-occurences-of-each-v

148

T: 3SAT $\leq_P$ HAMCYCLE

http://eaton.math.rpi.edu/faculty/Mitchell/courses/matp6620/notesMATP6620/lecture06/06A_hamiltoniancycle.pdf

149

T: HAMCYCLE $\leq_P$ TSP

Dają nam instancje HAMCYCLE G. Tworzymy instancję TSP – graf pełny o wierzchołkach z G i wagach krawędzi:

w(e) = {
    1 : e ∈ G.E,
    2 : wpp
}

ze stałą k = |V_G|.

150

Pokażemy, że przy pomocy takiego algorytmu dałoby się rozwiązać HAMCYCLE.

Dla danej instancji HAMCYCLE G tworzymy instancję TSP, gdzie:

w(e) = {
    1 : e ∈ E,
    2|V_G| : wpp
}

Jeśli algorytm zwróci cykl długości |V_G|, to G ma cykl Hamiltona.

151

Wielomianowy algorytm aproksymacyjny:

1. Budujemy MST
2. Przechodzimy wierzchołki MST DFS-em
3. Łączymy powtarzające się krawędzie
   (tj. dla $1 \to 2 \to 3 \to 2 \to 4 \to 2 \to 5$ mamy $1 \to 2 \to 3 \to 4 \to 5$)

Dowód optymalności:

1. Rozwiązanie TSP bez jednej krawędzi to drzewo spinające, więc MST nie jest gorsze
2. DFS przechodzi po każdej krawędzi 2 razy, więc mamy 2x gorsze rozwiązanie
3. Z nierówności trójkąta tutaj nadal mamy 2x gorsze rozwiązanie

152

Jak w 149, tylko z wagą 1.5 dla $e \notin E$.

154

T: 3SAT $\leq_P$ KLIKA_2

1. 3SAT ≤p KLIKA_3 (Sipser)
2. KLIKA_3 ≤p KLIKA_2

Problem znalezienia kliki o |V|/3 wierzchołkach jest NP-zupełny. Dla danej instancji dodajemy zbiór X kolejnych |V|/3 wierzchołków połączonych ze wszystkimi innymi. Jeśli znajdziemy klikę o |V|/2 wierzchołkach, to po wyrzuceniu X z rozwiązania zostaje nam klika o |V|/3 wierzchołkach.

158

T: 3COL $\leq_P$ 3COL+1

Niech kolory – {R, G, B}. Dają nam instancję 3COL G. Konstruujemy G’, gdzie do każdego wierzchołka z G “doklejamy” klikę K5.

$L \Rightarrow R$

Mamy 3-kolorowanie G. Zachowujemy wszystko jak w oryginale i dodajemy kolorowanie K5 dla każdego wierzchołka – dla wierzchołka o kolorze R musi być 1x R, 2x G, 2x B.

$R \Rightarrow L$

Mamy 3+1-kolorowanie G’. Każdy wierzchołek z “doklejoną” kliką K5 to razem klika K6, więc musi w niej wystąpić 2x każdy kolor. Co za tym idzie, “oryginalne” wierzchołki muszą być pokolorowane poprawnym “zwykłym” 3-kolorowaniem, bo “doklejone” kliki wyczerpują “tolerancję” max 1 wierzchołka w tym samym kolorze.

159

T: 3SAT_3 $\leq_P$ PZPR

Dają nam instancję 3SAT_3 ϕ. Dla każdej zmiennej p z ϕ usuwamy klauzule z (p ∨ ~p ∨ …), po czym dodajemy do A zbiory:

{p}, {p, C1}, {p, C2}, …, {p, C1, C2, C3},

gdzie C1, C2, C3 – klauzule, w których występuje p w takiej samej postaci,
tj. dla C1 = (p ∨ …), C2 = (p ∨ …), C3 = (~p ∨ …):

- {p}, {p, C1}, {p, C2}, {p, C1, C2}
- {p, C3}

$L \Rightarrow R$

Mamy wartościowanie, które spełnia ϕ. Dla każdej klauzuli Ci, która jest spełniona dzięki wartościowaniu p (T, jeśli Ci zawiera p; F, jeśli ~p), dajemy do rozwiązania {p, C1, …}. Jeśli dana klauzula jest spełniona jednocześnie dzięki wartościowaniu kilku zmiennych, wybieramy jedną z nich, np. jeśli Cx = (p ∨ q ∨ r) i p, q, r = T, to możemy wybrać spośród:

- {p, Cx}, {q}, {r}
- {p}, {q , Cx}, {r}
- {p}, {q}, {r, Cx}

W wyniku mamy B, w którym zbiory są rozłączne, każda zmienna występuje raz, każda klauzula występuje raz.

R ⇒ L

Mamy rodzinę zbiorów rozłącznych B, taką że suma B = A. Dla każdego A_i ∈ B patrzymy, jak wygląda ten zbiór – jeśli jest w nim tylko zmienna p, to jej wartościowanie nie ma znaczenia. Jeśli są w nim jeszcze klauzule, to patrzymy, w jakiej postaci występuje w nich p. Jeśli p, to p = T, jeśli ~p, to p = F.

173

T: NAE-3-SAT $\leq_P$ 173

Redukcja: niech kolorom w grafie odpowiada wartościowanie
Dla każdej zmiennej x mamy wierzchołki x i ~x połączone binarnym wierzchołkiem xalt (żeby nie pokolorować tak samo x i ~x). Dla każdej klauzuli C z 3 literałami dajemy krawędzie do tych literałów i chyba jest ok.

NAE-3-SAT ma rozwiązanie ⇒ 173 ma rozwiązanie

Każdej klauzuli dajmy czarny, każdemu xalt biały, wtedy literały mogą być pokolorowane dowolnie, bo każdy z nich będzie miał sąsiadów w obu kolorach, więc ograniczenia są tylko z klauzul i odwrotności – które da się spełnić, jeśli formuła jest spełnialna.

NAE-3-SAT nie ma rozwiązania ⇒ 173 nie ma rozwiązania

Dla każdego wartościowania istnieje klauzula, której wyjdzie TTT lub FFF, więc nie da się tak pokolorować grafu, żeby wierzchołki klauzulowe miały sąsiadów w obu kolorach.

180

T: 3COL $\leq_P$ NAE-3-SAT

Weźmy instancję problemu 3COL G.
Niech K = {r, g, b}, k(v) – v ∈ G.V jest koloru k ∈ K.
Skonstruujemy r(G) – formułę logiczną w 3-CNF.

Wprowadzamy specjalną zmienną t, która dla poprawnego kolorowania ma być T.
Kodujemy kilka rzeczy:

1. Sąsiadujące wierzchołki nie mogą mieć tego samego koloru – dla każdych {v, v'} ∈ E_G:
   (r(v) ∨ r(v') ∨ t), (g(v) ∨ g(v') ∨ t), (b(v) ∨ b(v') ∨ t).
2. Jeden wierzchołek może mieć tylko jeden kolor – dla każdego v ∈ G.V:
   r(v) ∨ g(v) ∨ b(v).

3. No i prawie dobrze, tylko teraz możemy mieć naraz 2 kolory. Potrzebujemy:
   $(r(v) \lor g(v) \lor t), (g(v) \lor b(v) \lor t), (r(v) \lor b(v) \lor t)$.

3COL G ma rozwiązanie ⇒ NAE-3-SAT r(G) ma rozwiązanie

Z konstrukcji, t = T, a reszta jak w G.

3COL G nie ma rozwiązania ⇒ NAE-3-SAT r(G) nie ma rozwiązania

Jeśli G nie ma 3-kolorowania, to 2 sąsiednie wierzchołki muszą mieć ten sam kolor.
Wtedy istnieją takie $\{v, v'\} \in G.E$, że $r(v) \land r(v')$ (albo g, albo b).
Wtedy żeby $r(v) \lor r(v') \lor t$ była spełniona, to t = F.
Ale wtedy $(g(v) \lor g(v') \lor t), (b(v) \lor b(v') \lor t)$ nie są spełnione.
Więc r(G) nie jest spełnialna.

181

To prawie jest NAE-3-SAT (wierzchołki – literały, kolory – prawda/fałsz, krawędzie – tam gdzie klauzule). Tylko trzeba jeszcze ograniczyć jakoś to, żeby wierzchołki x i ~x nie dostały takiego samego koloru.

183

Zakładamy, że |x| – dł. zapisu binarnego x.
Niech $A = \{ (x, y) \mid f^{-1}(x) < y \}$.

Lemat: f jest bijekcją.
- Z treści jest różnowartościowa
- Z |n| = |f(n)| dla danego n ∈ [2^k, 2^k+1) musi przybrać każdą wartość z tego przedziału

Zał. f istnieje, wtedy pokażemy, że NP ∩ co-NP ≠ PTIME.

1. A ∈ NP – dla danych (x, y) możemy niedeterministycznie zgadnąć n takie że f(n) = x, zwracamy n < y.
2. A ∈ co-NP ⇔ A' ∈ NP – -||-, zwracamy n ≥ y.
3. A ∉ PTIME – załóżmy, że A ∈ PTIME. Wtedy da się obliczyć f^{-1} w PTIME.
   Program obliczający f^{-1}:
   - wczytaj x // niech |x| = k ⇒ x ∈ [2^k, 2^k+1)
   - binarnie przeszukaj i = [k, k + 1) // O(log(2^k)) – O(|x|)
     - jeśli A(i, x), to lewo, wpp prawo czy jakoś tak

192

(Sipser)

Konwertujemy wyrażenie do NFA, odpalamy niedeterministyczną maszynę Turinga symulującą NFA, która spróbuje zgadnąć, czy jakieś słowo NIE należy do języka:

- wczytaj RE
- skonstruuj NFA M ze zbiorem stanów Q
- dla i od 1 do $2^{|Q|}$:
    - niedeterministycznie zgadnij literę $a \in \Sigma$
    - wrzuć a do M
    - jeśli a jest w stanie nieakceptującym:
        - zwróć 1 i zakończ
- zwróć 0 i zakończ

193

Sprawdzamy wszystkie wartościowania w 3-arnym drzewie, z którego pamiętamy tylko obecną ścieżkę. Chcemy ustalić, że wartościowanie jest dokładnie jedno, więc kasujemy podwójne.

202

Korzystamy z tożsamości: $(x \Rightarrow z) \land (y \Rightarrow z)$ wtw $(x \lor y) \Rightarrow z$.

$P_2(x, y) = \exists z\, R(x, z) \land R(z, y)$

$P_k(x, y) = \exists z\, P_{k/2}(x, z) \land P_{k/2}(z, y)$

$= \exists z \forall a \forall b\, ((a = x \land b = z) \Rightarrow P_{k/2}(a, b)) \land ((a = z \land b = y) \Rightarrow P_{k/2}(a, b))$

$= \exists z \forall a \forall b\, ((a = x \land b = z) \lor (a = z \land b = y)) \Rightarrow P_{k/2}(a, b)$