Algorytmy dokładne dla problemów NP-trudnych, część II

Łukasz Kowalik

Instytut Informatyki, Uniwersytet Warszawski

Warszawa, 28-29.05.2010

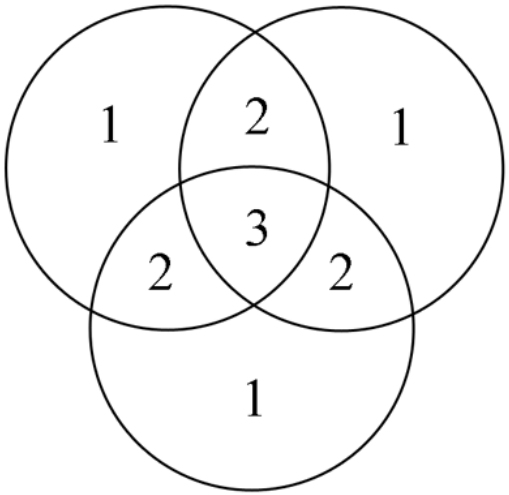
Technika: Zasada włączeń-wyłączeń

Zasada włączeń-wyłączeń

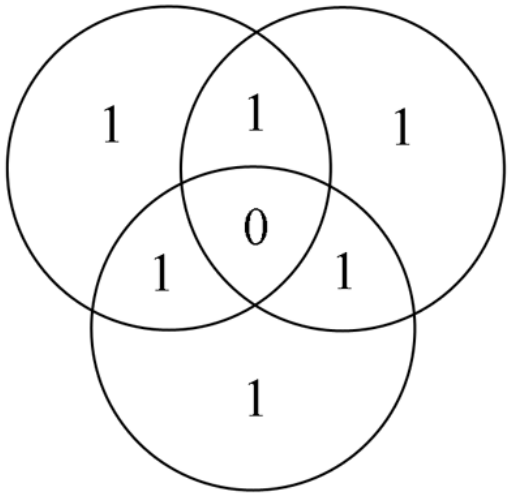
Twierdzenie (Zasada włączeń-wyłączeń, wersja I)

$$|\bigcup_{i\in\{1,\ldots,n\}} A_i| = \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

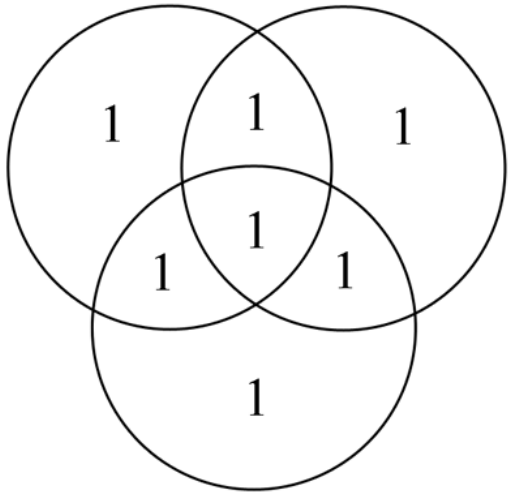
np. $|A \cup B| = |A| + |B| - |A \cap B|$,
$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|$.



$|A| + |B| + |C|$



$|A| + |B| + |C|$
$-(|A \cap B| + |A \cap C| + |B \cap C|)$



$|A| + |B| + |C|$
$-(|A \cap B| + |A \cap C| + |B \cap C|)$
$+|A \cap B \cap C|$

Zasada włączeń-wyłączeń, przepisywanie

Niech $A_1, \ldots, A_n \subseteq U$, gdzie U jest pewnym zbiorem skończonym.

$$|\bigcup_{i \in \{1,\ldots,n\}} A_i| = \sum_{\emptyset \neq X \subseteq \{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i \in X} A_i|$$

Zasada włączeń-wyłączeń, przepisywanie

Niech $A_1, \ldots, A_n \subseteq U$, gdzie U jest pewnym zbiorem skończonym.

$$|\bigcup_{i \in \{1,\ldots,n\}} A_i| = \sum_{\emptyset \neq X \subseteq \{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i \in X} A_i|$$

$$|U| - |\bigcup_{i \in \{1,\ldots,n\}} A_i| = |U| - \sum_{\emptyset \neq X \subseteq \{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i \in X} A_i|$$

Zasada włączeń-wyłączeń, przepisywanie

Niech $A_1, \ldots, A_n \subseteq U$, gdzie U jest pewnym zbiorem skończonym.

$$|\bigcup_{i\in\{1,\ldots,n\}} A_i| = \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

$$|U| - |\bigcup_{i\in\{1,\ldots,n\}} A_i| = |U| - \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

$$|U - \bigcup_{i\in\{1,\ldots,n\}} A_i| = |U| - \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

Zasada włączeń-wyłączeń, przepisywanie

Niech $A_1, \ldots, A_n \subseteq U$, gdzie U jest pewnym zbiorem skończonym.

$$|\bigcup_{i\in\{1,\ldots,n\}} A_i| = \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

$$|U| - |\bigcup_{i\in\{1,\ldots,n\}} A_i| = |U| - \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

$$|U - \bigcup_{i\in\{1,\ldots,n\}} A_i| = |U| - \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

Oznaczmy $\overline{A_i} = U - A_i$ oraz $\bigcap_{i\in\emptyset} \overline{A_i} = U$. Wtedy:

$$|\bigcap_{i\in\{1,\ldots,n\}} \overline{A_i}| = \sum_{X\subseteq\{1,\ldots,n\}} (-1)^{|X|} |\bigcap_{i\in X} A_i|$$

Zasada włączeń-wyłączeń, przepisywanie

Niech $A_1, \ldots, A_n \subseteq U$, gdzie U jest pewnym zbiorem skończonym.

$$|\bigcup_{i\in\{1,\ldots,n\}} A_i| = \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

$$|U| - |\bigcup_{i\in\{1,\ldots,n\}} A_i| = |U| - \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

$$|U - \bigcup_{i\in\{1,\ldots,n\}} A_i| = |U| - \sum_{\emptyset\neq X\subseteq\{1,\ldots,n\}} (-1)^{|X|-1} |\bigcap_{i\in X} A_i|$$

Oznaczmy $\overline{A_i} = U - A_i$ oraz $\bigcap_{i\in\emptyset} \overline{A_i} = U$. Wtedy:

$$|\bigcap_{i\in\{1,\ldots,n\}} \overline{A_i}| = \sum_{X\subseteq\{1,\ldots,n\}} (-1)^{|X|} |\bigcap_{i\in X} A_i|$$

$$|\bigcap_{i\in\{1,\ldots,n\}} A_i| = \sum_{X\subseteq\{1,\ldots,n\}} (-1)^{|X|} |\bigcap_{i\in X} \overline{A_i}|$$

Zasada włączeń-wyłączeń, wersja iloczynowa

Otrzymaliśmy:

Twierdzenie (Zasada włączeń-wyłączeń, wersja iloczynowa)

Niech $A_1, \ldots, A_n \subseteq U$, gdzie U jest pewnym zbiorem skończonym.
Oznaczmy $\overline{A_i} = U - A_i$ oraz $\bigcap_{i \in \emptyset} \overline{A_i} = U$.
Wtedy:

$$\left| \bigcap_{i \in \{1,\ldots,n\}} A_i \right| = \sum_{X \subseteq \{1,\ldots,n\}} (-1)^{|X|} \left| \bigcap_{i \in X} \overline{A_i} \right|$$

Klasyczny przykład: nieporządki

Zadanie

Permutacja $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ jest nieporządkiem, gdy $\pi(i) \neq i$ dla każdego $i = 1, \ldots, n$.
Ile wynosi $d(n)$, liczba nieporządków n-elementowych?

- U jest zbiorem permutacji n-elementowych.

Klasyczny przykład: nieporządki

Zadanie

Permutacja $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ jest nieporządkiem, gdy $\pi(i) \neq i$ dla każdego $i = 1, \ldots, n$.
Ile wynosi $d(n)$, liczba nieporządków n-elementowych?

- U jest zbiorem permutacji n-elementowych.
- Dla $i = 1, \ldots, n$ definiujemy $A_i = \{\pi \in U \ : \ \pi(i) \neq i\}$. „wymagania”

Klasyczny przykład: nieporządki

Zadanie

Permutacja $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ jest nieporządkiem, gdy $\pi(i) \neq i$ dla każdego $i = 1, \ldots, n$.
Ile wynosi $d(n)$, liczba nieporządków n-elementowych?

- U jest zbiorem permutacji n-elementowych.
- Dla $i = 1, \ldots, n$ definiujemy $A_i = \{\pi \in U \ : \ \pi(i) \neq i\}$. „wymagania”
- Wtedy $d(n) = |\bigcap_{i=1,\ldots,n} A_i|$.

Klasyczny przykład: nieporządki

Zadanie

Permutacja $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ jest nieporządkiem, gdy $\pi(i) \neq i$ dla każdego $i = 1, \ldots, n$.
Ile wynosi $d(n)$, liczba nieporządków n-elementowych?

- U jest zbiorem permutacji n-elementowych.
- Dla $i = 1, \ldots, n$ definiujemy $A_i = \{\pi \in U \ : \ \pi(i) \neq i\}$. „wymagania”
- Wtedy $d(n) = |\bigcap_{i=1,\ldots,n} A_i|$.
- $|\bigcap_{i \in X} \overline{A_i}| = (n - |X|)!$. „uproszczony problem”

Klasyczny przykład: nieporządki

Zadanie

Permutacja $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ jest nieporządkiem, gdy $\pi(i) \neq i$ dla każdego $i = 1, \ldots, n$.
Ile wynosi $d(n)$, liczba nieporządków n-elementowych?

- U jest zbiorem permutacji n-elementowych.
- Dla $i = 1, \ldots, n$ definiujemy $A_i = \{\pi \in U \ : \ \pi(i) \neq i\}$. „wymagania"
- Wtedy $d(n) = |\bigcap_{i=1,\ldots,n} A_i|$.
- $|\bigcap_{i \in X} \overline{A_i}| = (n - |X|)!$. „uproszczony problem"

Wniosek

$$d(n) = \sum_{X \subseteq \{1,\ldots,n\}} (-1)^{|X|}(n - |X|)!$$

Zabawkowy przykład algorytmiczny

Problem

Dana formuła w postaci CNF o m klauzulach. Ile jest wartościowań spełniających?

Zabawkowy przykład algorytmiczny

Problem

Dana formuła w postaci CNF o m klauzulach. Ile jest wartościowań spełniających?

- U jest zbiorem wszystkich wartościowań.

Zabawkowy przykład algorytmiczny

Problem

Dana formuła w postaci CNF o m klauzulach. Ile jest wartościowań spełniających?

- U jest zbiorem wszystkich wartościowań.
- A_i = zbiór wartościowań t.ż. klauzula C_i spełniona, $i = 1, \ldots, m$.

Zabawkowy przykład algorytmiczny

Problem

Dana formuła w postaci CNF o m klauzulach. Ile jest wartościowań spełniających?

- U jest zbiorem wszystkich wartościowań.
- A_i = zbiór wartościowań t.ż. klauzula C_i spełniona, $i = 1, \ldots, m$.
- Wtedy rozwiązanie to $|\bigcap_{i=1,\ldots,n} A_i|$.

Zabawkowy przykład algorytmiczny

Problem

Dana formuła w postaci CNF o m klauzulach. Ile jest wartościowań spełniających?

- U jest zbiorem wszystkich wartościowań.
- A_i = zbiór wartościowań t.ż. klauzula C_i spełniona, $i = 1, \ldots, m$.
- Wtedy rozwiązanie to $|\bigcap_{i=1,\ldots,n} A_i|$.
- $|\bigcap_{i \in X} \overline{A_i}| =$

$$\begin{cases} 0 & \text{gdy w } X \text{ są dwie klauzule zawierające przeciwne literały,} \\ 2^v & \text{gdzie } v \text{ jest liczbą zmiennych nie występujących w klauzulach z } X \end{cases}$$

Zabawkowy przykład algorytmiczny

Problem

Dana formuła w postaci CNF o m klauzulach. Ile jest wartościowań spełniających?

- U jest zbiorem wszystkich wartościowań.
- A_i = zbiór wartościowań t.ż. klauzula C_i spełniona, $i = 1, \ldots, m$.
- Wtedy rozwiązanie to $|\bigcap_{i=1,\ldots,n} A_i|$.
- $|\bigcap_{i \in X} \overline{A_i}| =$
  $\begin{cases} 0 & \text{gdy w } X \text{ są dwie klauzule zawierające przeciwne literały,} \\ 2^v & \text{gdzie } v \text{ jest liczbą zmiennych nie występujących w klauzulach z } X \end{cases}$
- Uproszczony problem można rozwiązać w czasie wielomianowym (liniowym), więc otrzymaliśmy algorytm $O^*(2^m)$.

Liczba cykli Hamiltona (Karp 1982)

Problem

Dany n-wierzchołkowy graf nieskierowany $G = (V, E)$. Ile jest cykli Hamiltona?

Liczba cykli Hamiltona (Karp 1982)

Problem

Dany n-wierzchołkowy graf nieskierowany $G = (V, E)$. Ile jest cykli Hamiltona?

- Marszruta długości k w grafie to ciąg wierzchołków $v_0, v_1, \ldots, v_k$ taki, że $v_i v_{i+1} \in E$ dla każdego $i = 0, \ldots, k-1$.
- Marszruta jest zamknięta, gdy $v_0 = v_k$.

Liczba cykli Hamiltona (Karp 1982)

Problem

Dany n-wierzchołkowy graf nieskierowany $G = (V, E)$. Ile jest cykli Hamiltona?

- Marszruta długości k w grafie to ciąg wierzchołków $v_0, v_1, \ldots, v_k$ taki, że $v_i v_{i+1} \in E$ dla każdego $i = 0, \ldots, k-1$.
- Marszruta jest zamknięta, gdy $v_0 = v_k$.
- U jest zbiorem marszrut zamkniętych długości n zawierających wierzchołek 1.

Liczba cykli Hamiltona (Karp 1982)

Problem

Dany n-wierzchołkowy graf nieskierowany $G = (V, E)$. Ile jest cykli Hamiltona?

- Marszruta długości k w grafie to ciąg wierzchołków $v_0, v_1, \ldots, v_k$ taki, że $v_i v_{i+1} \in E$ dla każdego $i = 0, \ldots, k-1$.
- Marszruta jest zamknięta, gdy $v_0 = v_k$.
- U jest zbiorem marszrut zamkniętych długości n zawierających wierzchołek 1.
- A_v = marszruty z U przechodzące przez v, $v \in V$.

Liczba cykli Hamiltona (Karp 1982)

Problem

Dany n-wierzchołkowy graf nieskierowany $G = (V, E)$. Ile jest cykli Hamiltona?

- Marszruta długości k w grafie to ciąg wierzchołków $v_0, v_1, \ldots, v_k$ taki, że $v_i v_{i+1} \in E$ dla każdego $i = 0, \ldots, k-1$.
- Marszruta jest zamknięta, gdy $v_0 = v_k$.
- U jest zbiorem marszrut zamkniętych długości n zawierających wierzchołek 1.
- A_v = marszruty z U przechodzące przez v, $v \in V$.
- Wtedy rozwiązanie to $|\bigcap_{v \in V} A_v|$.

Liczba cykli Hamiltona (Karp 1982)

Problem

Dany n-wierzchołkowy graf nieskierowany $G = (V, E)$. Ile jest cykli Hamiltona?

- Marszruta długości k w grafie to ciąg wierzchołków $v_0, v_1, \ldots, v_k$ taki, że $v_i v_{i+1} \in E$ dla każdego $i = 0, \ldots, k-1$.
- Marszruta jest zamknięta, gdy $v_0 = v_k$.
- U jest zbiorem marszrut zamkniętych długości n zawierających wierzchołek 1.
- A_v = marszruty z U przechodzące przez v, $v \in V$.
- Wtedy rozwiązanie to $|\bigcap_{v \in V} A_v|$.
- Uproszczony problem: $|\bigcap_{v \in X} \overline{A_v}|$ = liczba marszrut zamkniętych z U długości n w $G' = G[V - X]$.

Liczba cykli Hamiltona cd

Uproszczony problem

Ile jest marszrut zamkniętych długości n zawierających wierzchołek 1 w grafie G'?

Programowanie dynamiczne

- $T(d, x) =$ liczba marszrut od 1 do x długości d.
- $T(d, x) = \sum_{y \in V} T(d-1, y) \cdot [yx \in E(G')]$.
- Zwracamy $T(n, 1)$, czas $O(n^3)$.

Liczba cykli Hamiltona cd

Uproszczony problem

Ile jest marszrut zamkniętych długości n zawierających wierzchołek 1 w grafie G'?

Programowanie dynamiczne

- $T(d, x) =$ liczba marszrut od 1 do x długości d.
- $T(d, x) = \sum_{y \in V} T(d-1, y) \cdot [yx \in E(G')]$.
- Zwracamy $T(n, 1)$, czas $O(n^3)$.

Inny sposób: odczytujemy $M^n_{1,1}$, $M =$ macierz sąsiedztwa; czas $O(n^\omega \log n)$.

Liczba cykli Hamiltona cd

Uproszczony problem

Ile jest marszrut zamkniętych długości n zawierających wierzchołek 1 w grafie G'?

Programowanie dynamiczne

- $T(d, x) =$ liczba marszrut od 1 do x długości d.
- $T(d, x) = \sum_{y \in V} T(d-1, y) \cdot [yx \in E(G')]$.
- Zwracamy $T(n, 1)$, czas $O(n^3)$.

Inny sposób: odczytujemy $M^n_{1,1}$, $M =$ macierz sąsiedztwa; czas $O(n^\omega \log n)$.

Wniosek

W czasie $O^*(2^n)$ i **pamięci wielomianowej** można rozwiązać problem cyklu Hamiltona (a nawet zliczyć liczbę takich cykli).

TSP

Problem

Dana macierz wag krawędzi w grafie pełnym $w : V^2 \to \{1, \ldots, C\}$. Ile jest cykli Hamiltona o wadze α, $\alpha = 1, \ldots, nC$?

TSP

Problem

Dana macierz wag krawędzi w grafie pełnym $w : V^2 \to \{1, \ldots, C\}$. Ile jest cykli Hamiltona o wadze α, $\alpha = 1, \ldots, nC$?

Uproszczony problem

Ile jest marszrut zamkniętych długości n **o wadze** α zawierających wierzchołek 1 w grafie G'?

TSP

Problem

Dana macierz wag krawędzi w grafie pełnym $w : V^2 \to \{1, \ldots, C\}$. Ile jest cykli Hamiltona o wadze α, $\alpha = 1, \ldots, nC$?

Uproszczony problem

Ile jest marszrut zamkniętych długości n **o wadze** α zawierających wierzchołek 1 w grafie G'?

Programowanie dynamiczne

- niech $C =$ maksymalna waga krawędzi w G'.
- $T(d, x, \beta) =$ liczba marszrut od 1 do x długości d o wadze β, $\beta = 1, \ldots, \alpha$.
- $T(d, x, \beta) = \sum_{y \in V} T(d-1, y, \beta - w(x, y))$.
- Zwracamy $T(n, 1, \alpha)$, czas $O(n^3 C)$.

TSP

Wniosek

W czasie $O^*(2^n \cdot C)$ i **pamięci wielomianowej** można rozwiązać (decyzyjny) problem TSP.

Wniosek

W czasie $O^*(2^n \cdot C \log C)$ i **pamięci wielomianowej** można rozwiązać (optymalizacyjny) problem TSP.

Kolorowanie w 2^n, Bjorklund, Husfeldt, Koivisto, 2006

k-kolorowanie

k-kolorowaniem grafu $G = (V, E)$ nazywamy dowolną funkcję $c : V \to \{1, \ldots, k\}$ taką, że dla dowolnej krawędzi $xy \in E$, $c(x) \neq c(y)$.

Problem

Dla danego grafu $G = (V, E)$ stwierdzić, czy istnieje jego k-kolorowanie. (Jeśli umiemy to zrobić w czasie $O^*(c^n)$ to dzięki samoredukowalności możemy też w czasie $O^*(c^n)$ **znaleźć** k-kolorowanie gdy takie istnieje).

Kolorowanie w 2^n, Bjorklund, Husfeldt, Koivisto, 2006

k-kolorowanie

k-kolorowaniem grafu $G = (V, E)$ nazywamy dowolną funkcję $c : V \to \{1, \ldots, k\}$ taką, że dla dowolnej krawędzi $xy \in E$, $c(x) \neq c(y)$.

Problem

Dla danego grafu $G = (V, E)$ stwierdzić, czy istnieje jego k-kolorowanie. (Jeśli umiemy to zrobić w czasie $O^*(c^n)$ to dzięki samoredukowalności możemy też w czasie $O^*(c^n)$ **znaleźć** k-kolorowanie gdy takie istnieje).

Obserwacje

- (trywialne) k-kolorowanie to podział V na k zbiorów niezależnych.
- (ciekawsze) Istnieje podział V na k zbiorów niezależnych wtedy i tylko wtedy gdy istnieje **pokrycie** k zbiorami niezależnymi, tzn. k zbiorów niezależnych $I_1, \ldots, I_k$ takich, że $\bigcup_{j=1}^{k} I_j = V$.

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)
- $A_v = \{(I_1, \ldots, I_k) \in U \ : \ v \in \bigcup_{j=1}^{k} I_j\}$

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)
- $A_v = \{(I_1, \ldots, I_k) \in U \ : \ v \in \bigcup_{j=1}^{k} I_j\}$
- Wtedy $|\bigcap_{v \in V} A_v| \neq 0$ wtw gdy G jest k-kolorowalny.

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)
- $A_v = \{(I_1, \ldots, I_k) \in U \ : \ v \in \bigcup_{j=1}^{k} I_j\}$
- Wtedy $|\bigcap_{v \in V} A_v| \neq 0$ wtw gdy G jest k-kolorowalny.
- Uproszczony problem:

$$|\bigcap_{v \in X} \overline{A_v}| =$$

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)
- $A_v = \{(I_1, \ldots, I_k) \in U \ : \ v \in \bigcup_{j=1}^{k} I_j\}$
- Wtedy $|\bigcap_{v \in V} A_v| \neq 0$ wtw gdy G jest k-kolorowalny.
- Uproszczony problem:

$$|\bigcap_{v \in X} \overline{A_v}| = |\{(I_1, \ldots, I_k) \in U \ : \ I_1, \ldots, I_k \subseteq V - X\}|$$

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)
- $A_v = \{(I_1, \ldots, I_k) \in U \ : \ v \in \bigcup_{j=1}^{k} I_j\}$
- Wtedy $|\bigcap_{v \in V} A_v| \neq 0$ wtw gdy G jest k-kolorowalny.
- Uproszczony problem:

$$|\bigcap_{v \in X} \overline{A_v}| = |\{(I_1, \ldots, I_k) \in U \ : \ I_1, \ldots, I_k \subseteq V - X\}| = s(V - X)^k$$

gdzie $s(Y)$ = liczba zbiorów niezależnych które są podzbiorami Y.

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)
- $A_v = \{(I_1, \ldots, I_k) \in U \ : \ v \in \bigcup_{j=1}^{k} I_j\}$
- Wtedy $|\bigcap_{v \in V} A_v| \neq 0$ wtw gdy G jest k-kolorowalny.
- Uproszczony problem:

$$|\bigcap_{v \in X} \overline{A_v}| = |\{(I_1, \ldots, I_k) \in U \ : \ I_1, \ldots, I_k \subseteq V - X\}| = s(V - X)^k$$

  gdzie $s(Y)$ = liczba zbiorów niezależnych które są podzbiorami Y.
- $s(Y)$ możemy obliczyć na początku algorytmu programowaniem dynamicznym **dla wszystkich podzbiorów** $Y \subseteq V$: $s(Y) = s(Y - \{y\}) + s(Y - N[y])$. Zajmie to czas **i pamięć** $O^*(2^n)$, gdyż liczba kolorowań zajmuje $O(n \log k)$ bitów.

Kolorowanie w 2^n, cd.

- U jest zbiorem krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi (nie muszą być nawet parami różne!)
- $A_v = \{(I_1, \ldots, I_k) \in U \ : \ v \in \bigcup_{j=1}^{k} I_j\}$
- Wtedy $|\bigcap_{v \in V} A_v| \neq 0$ wtw gdy G jest k-kolorowalny.
- Uproszczony problem:

$$|\bigcap_{v \in X} \overline{A_v}| = |\{(I_1, \ldots, I_k) \in U \ : \ I_1, \ldots, I_k \subseteq V - X\}| = s(V - X)^k$$

  gdzie $s(Y)$ = liczba zbiorów niezależnych które są podzbiorami Y.
- $s(Y)$ możemy obliczyć na początku algorytmu programowaniem dynamicznym **dla wszystkich podzbiorów** $Y \subseteq V$: $s(Y) = s(Y - \{y\}) + s(Y - N[y])$. Zajmie to czas **i pamięć** $O^*(2^n)$, gdyż liczba kolorowań zajmuje $O(n \log k)$ bitów.
- Następnie $|\bigcap_{v \in X} \overline{A_v}|$ obliczymy w czasie $O^*(1)$, a więc $|\bigcap_{v \in V} A_v|$ dostaniemy w czasie $O^*(2^n)$.

Kolorowanie w 2^n, cd.

Twierdzenie

W czasie $O^*(2^n)$ i pamięci $O^*(2^n)$ możemy dla danego grafu G,

- znaleźć k-kolorowanie, lub stwierdzić, że ono nie istnieje,
- znaleźć liczbę chromatyczną.

Kolorowanie w 2^n, cd.

Twierdzenie

W czasie $O^*(2^n)$ i pamięci $O^*(2^n)$ możemy dla danego grafu G,

- znaleźć k-kolorowanie, lub stwierdzić, że ono nie istnieje,
- znaleźć liczbę chromatyczną.

Twierdzenie

W czasie $O^*(2.25^n)$ i **pamięci wielomianowej** możemy znaleźć k-kolorowanie danego grafu G, lub stwierdzić, że ono nie istnieje.

Dowód

$s(Y)$ obliczamy gotowym algorytmem: w czasie $O(1.2461^n)$ i **pamięci wielomianowej** [Fürer, Kasiviswanathan 2005]. Całkowity czas:

$$\sum_{X \subseteq V} 1.2461^{|X|} = \sum_{k=0}^{n} \binom{n}{k} 1.2461^k = (1 + 1.2461)^n = O(2.25^n).$$

Kilka uwag

Uwaga 1

Używając trochę bardziej skomplikowanego programowania dynamicznego można obliczyć prawdziwą liczbę k-kolorowań (a nie liczbę pokryć z powtórzeniami zbiorami niezależnymi), w takim samym czasie/pamięci.

Kilka uwag

Uwaga 1

Używając trochę bardziej skomplikowanego programowania dynamicznego można obliczyć prawdziwą liczbę k-kolorowań (a nie liczbę pokryć z powtórzeniami zbiorami niezależnymi), w takim samym czasie/pamięci.

Uwaga 2

Przedstawione rozumowanie można przenieść na ogólny problem pokrycia/podziału uniwersum V rodziną zbiorów.

Drzewo Steinera w 2^k, Nerderlof 2009

Wersja bez wag

Dany graf $G = (V, E)$, zbiór terminali $K \subseteq V$ i liczba $c \in \mathbb{N}$. Czy istnieje poddrzewo $T \subseteq G$ takie że $K \subseteq V(T)$ oraz $|E(T)| \leq c$?

Drzewo Steinera w 2^k, Nerderlof 2009

Wersja bez wag

Dany graf $G = (V, E)$, zbiór terminali $K \subseteq V$ i liczba $c \in \mathbb{N}$. Czy istnieje poddrzewo $T \subseteq G$ takie że $K \subseteq V(T)$ oraz $|E(T)| \leq c$?

Wersja ważona

Dodatkowo wagi na krawędziach $w : E \to \mathbb{N}$. Czy istnieje poddrzewo $T \subseteq G$ takie że $K \subseteq V(T)$ oraz $w(E(T)) \leq c$?

Drzewo Steinera w 2^k, Nerderlof 2009

Wersja bez wag

Dany graf $G = (V, E)$, zbiór terminali $K \subseteq V$ i liczba $c \in \mathbb{N}$. Czy istnieje poddrzewo $T \subseteq G$ takie że $K \subseteq V(T)$ oraz $|E(T)| \leq c$?

Wersja ważona

Dodatkowo wagi na krawędziach $w : E \to \mathbb{N}$. Czy istnieje poddrzewo $T \subseteq G$ takie że $K \subseteq V(T)$ oraz $w(E(T)) \leq c$?

Oznaczamy $n = |V|$, $k = |K|$.

Klasyczny algorytm [Dreyfus, Wagner 1972]

Programowanie dynamiczne, działa nawet w wersji ważonej w czasie $O^*(3^k)$ i pamięci $O^*(2^k)$.

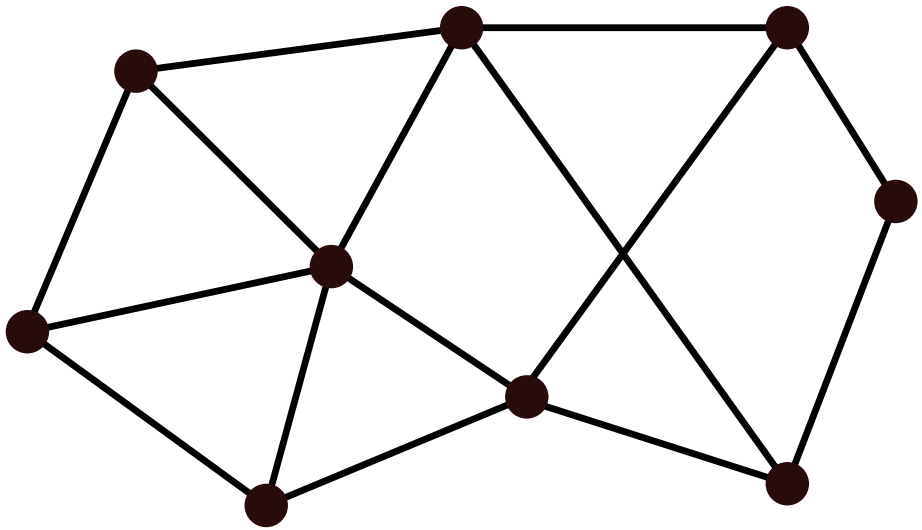
Marszruty drzewowe

Definicja

Niech $G = (V, E)$ będzie grafem nieskierowanym i niech $s \in V$.
Marszrutą drzewową nazwiemy parę $B = (T, h)$, gdzie T jest dowolnym drzewem ukorzenionym oraz $h : V(T) \to V$ jest homomorfizmem, tzn. jeśli $(x, y) \in E(T)$ to $h(x)h(y) \in E(G)$. Powiemy, że B ma początek w s, gdy $h(r) = s$, gdzie r jest korzeniem T. Długością B nazywamy $|E(T)|$.

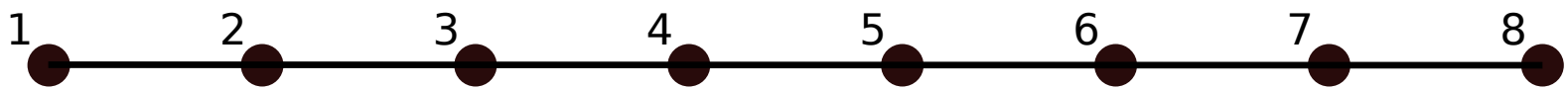
Marszruty drzewowe

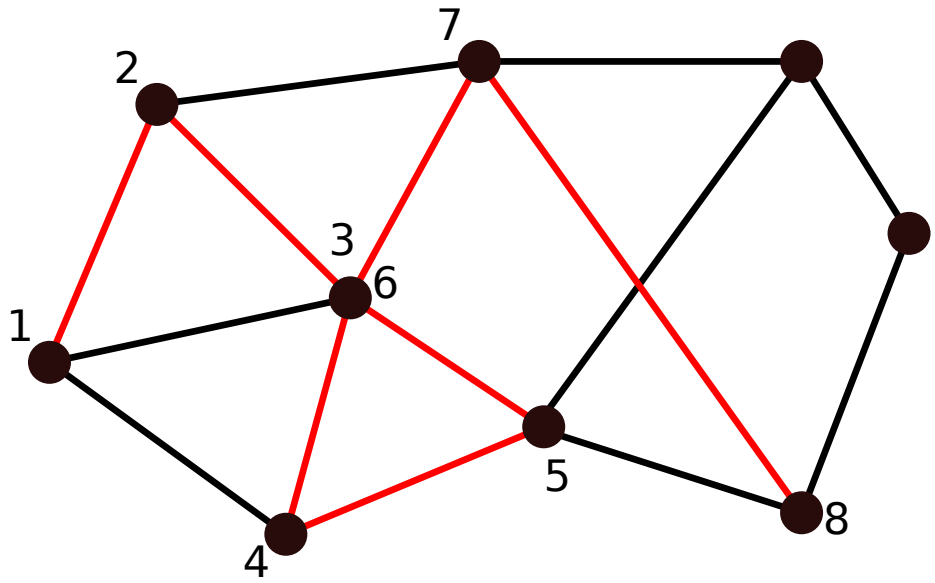
Przykład 1 Każda marszruta (zwykła) jest marszrutą drzewową



Marszruty drzewowe

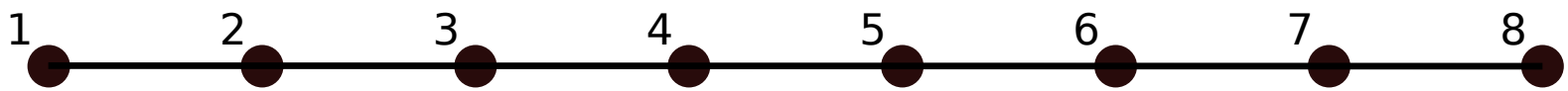
Przykład 1 Każda marszruta (zwykła) jest marszrutą drzewową

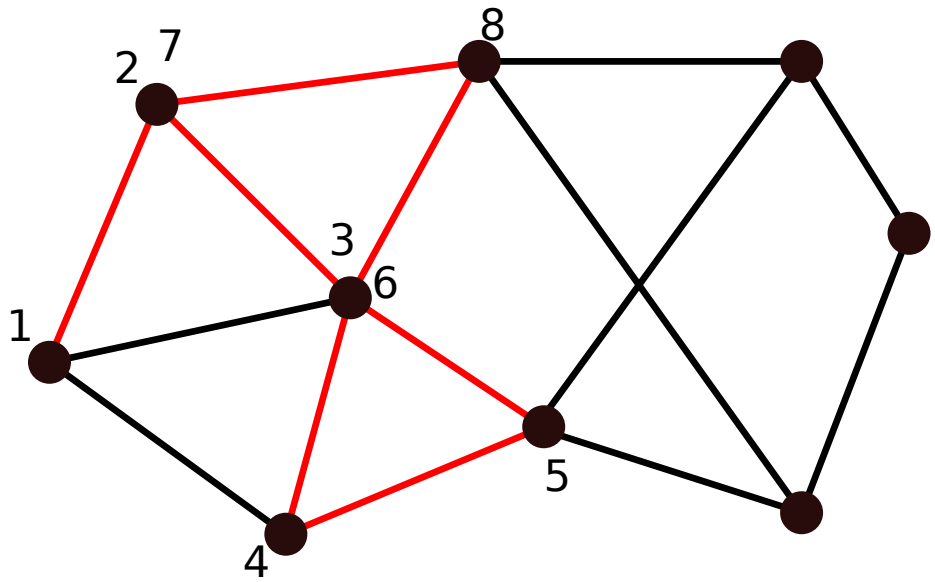




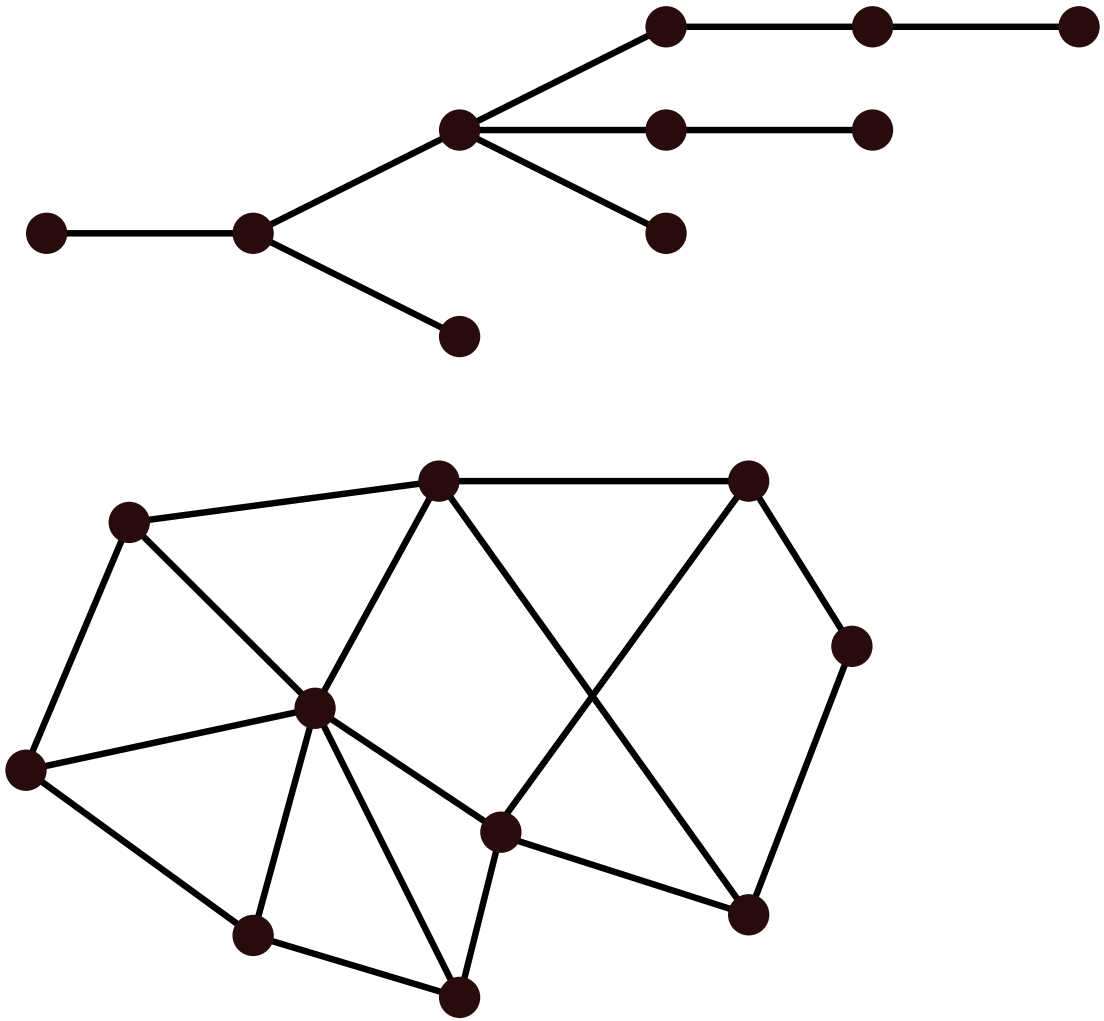
Marszruty drzewowe

Przykład 2 Nawet taka.



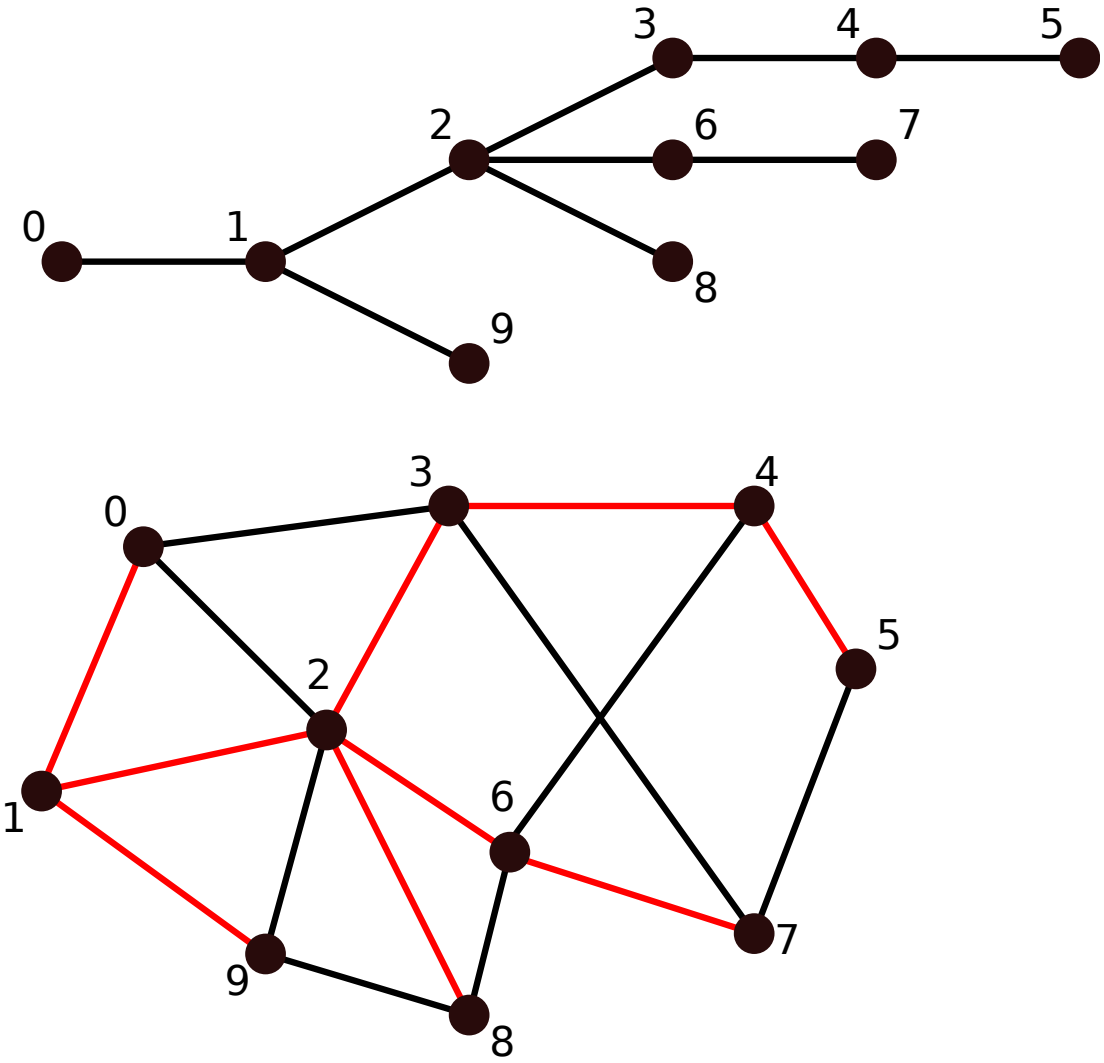


Marszruty drzewowe



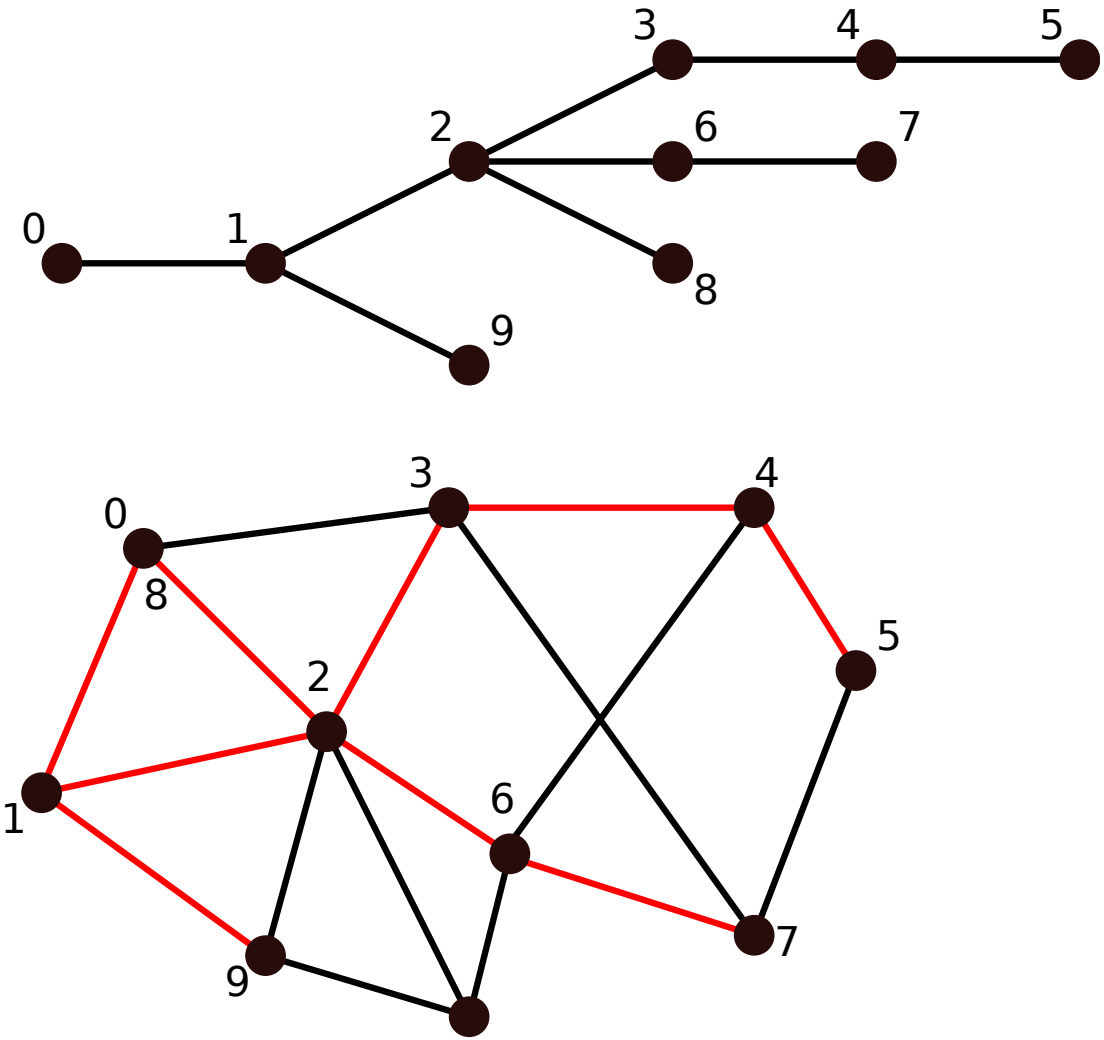
Marszruty drzewowe

Przykład 3 Homomorfizm różnowartościowy.



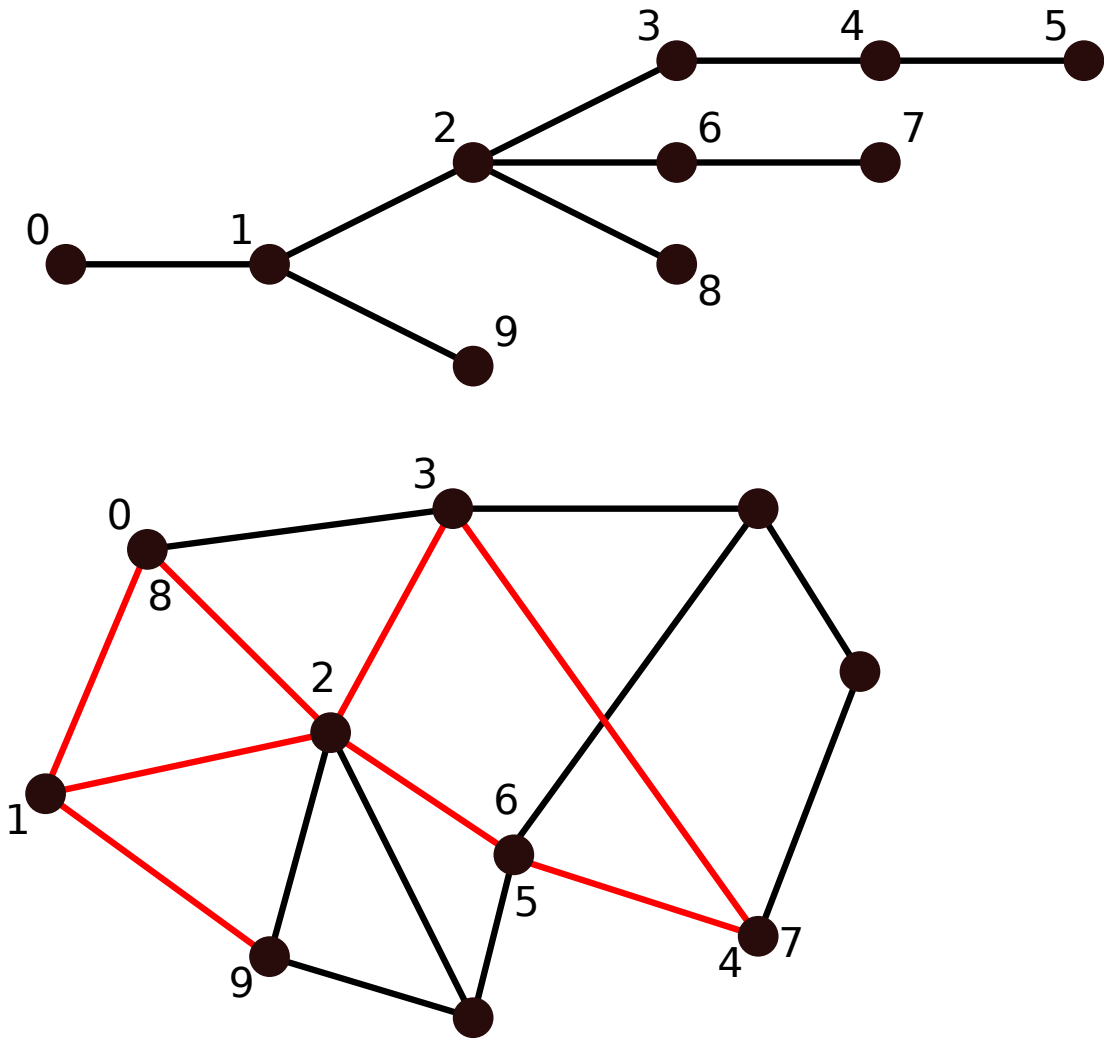
Marszruty drzewowe

Przykład 4 Homomorfizm, który nie jest różnowartościowy.



Marszruty drzewowe

Przykład 5 Homomorfizm, który jeszcze bardziej nie jest różnowartościowy.



Drzewo Steinera, wersja bez wag

Niech $s \in K$ będzie dowolnym terminalem.

Obserwacja

W G istnieje poddrzewo T takie, że $K \subseteq V(T)$ oraz $|E(T)| \leq c$ wtedy i tylko wtedy gdy w G istnieje marszruta drzewowa $B = (T_B, h)$ z s długości $\leq c$ taka, że $K \subseteq h(V(T_B))$.

Drzewo Steinera, wersja bez wag

Niech $s \in K$ będzie dowolnym terminalem.

Obserwacja

W G istnieje poddrzewo T takie, że $K \subseteq V(T)$ oraz $|E(T)| \leq c$ wtedy i tylko wtedy gdy w G istnieje marszruta drzewowa $B = (T_B, h)$ z s długości $\leq c$ taka, że $K \subseteq h(V(T_B))$.

- U jest zbiorem wszystkich marszrut drzewowych z s długości c.

Drzewo Steinera, wersja bez wag

Niech $s \in K$ będzie dowolnym terminalem.

Obserwacja

W G istnieje poddrzewo T takie, że $K \subseteq V(T)$ oraz $|E(T)| \leq c$ wtedy i tylko wtedy gdy w G istnieje marszruta drzewowa $B = (T_B, h)$ z s długości $\leq c$ taka, że $K \subseteq h(V(T_B))$.

- U jest zbiorem wszystkich marszrut drzewowych z s długości c.
- $A_v = \{B \in U \ : \ v \in V(B)\}$ dla $v \in K$.

Drzewo Steinera, wersja bez wag

Niech $s \in K$ będzie dowolnym terminalem.

Obserwacja

W G istnieje poddrzewo T takie, że $K \subseteq V(T)$ oraz $|E(T)| \leq c$ wtedy i tylko wtedy gdy w G istnieje marszruta drzewowa $B = (T_B, h)$ z s długości $\leq c$ taka, że $K \subseteq h(V(T_B))$.

- U jest zbiorem wszystkich marszrut drzewowych z s długości c.
- $A_v = \{B \in U \ : \ v \in V(B)\}$ dla $v \in K$.
- Wtedy $|\bigcap_{v \in K} A_v| \neq 0$ wtw istnieje szukane poddrzewo Steinera.

Drzewo Steinera, wersja bez wag

Niech $s \in K$ będzie dowolnym terminalem.

Obserwacja

W G istnieje poddrzewo T takie, że $K \subseteq V(T)$ oraz $|E(T)| \leq c$ wtedy i tylko wtedy gdy w G istnieje marszruta drzewowa $B = (T_B, h)$ z s długości $\leq c$ taka, że $K \subseteq h(V(T_B))$.

- U jest zbiorem wszystkich marszrut drzewowych z s długości c.
- $A_v = \{B \in U \ : \ v \in V(B)\}$ dla $v \in K$.
- Wtedy $|\bigcap_{v \in K} A_v| \neq 0$ wtw istnieje szukane poddrzewo Steinera.
- Dla $R \subseteq K$ oznaczmy $R' = R \cup (V - K)$.
- Uproszczony problem:

$$|\bigcap_{v \in K} \overline{A_v}| = b_c^{K-X}(s),$$

gdzie $b_j^R(a) =$ liczba marszrut drzewowych z a długości j w $G[R']$.

Drzewo Steinera, uproszczony problem

Dla $R \subseteq K$ oznaczmy $R' = R \cup (V - K)$.

Uproszczony problem

$$|\bigcap_{v \in K} \overline{A_v}| = b_c^{K-X}(s),$$

gdzie $b_j^R(a)$ = liczba marszrut drzewowych z a długości j w $G[R']$.

Drzewo Steinera, uproszczony problem

Dla $R \subseteq K$ oznaczmy $R' = R \cup (V - K)$.

Uproszczony problem

$$|\bigcap_{v \in K} \overline{A_v}| = b_c^{K-X}(s),$$

gdzie $b_j^R(a)$ = liczba marszrut drzewowych z a długości j w $G[R']$.

- $b_j^R(a)$ obliczamy $\forall\ j = 0, \ldots, c$ i $a \in R'$ przez prog. dynamiczne:

$$\begin{cases} 1 & \text{gdy } j = 0, \\ \sum_{t \in N(s) \cap R'} \sum_{j_1 + j_2 = j-1} b_{j_1}^R(s) b_{j_2}^R(t) & \text{wpp.} \end{cases}$$

Drzewo Steinera, uproszczony problem

Dla $R \subseteq K$ oznaczmy $R' = R \cup (V - K)$.

Uproszczony problem

$$|\bigcap_{v \in K} \overline{A_v}| = b_c^{K-X}(s),$$

gdzie $b_j^R(a)$ = liczba marszrut drzewowych z a długości j w $G[R']$.

- $b_j^R(a)$ obliczamy $\forall\ j = 0, \ldots, c$ i $a \in R'$ przez prog. dynamiczne:

$$\begin{cases} 1 & \text{gdy } j = 0, \\ \sum_{t \in N(s) \cap R'} \sum_{j_1 + j_2 = j-1} b_{j_1}^R(s) b_{j_2}^R(t) & \text{wpp.} \end{cases}$$

- Zauważmy, że $b_j^R = O((nj)^j)$ — łatwa indukcja; stąd b_j^R zajmuje $O(n \log n) = O^*(1)$ bitów.
- Czyli uproszczony problem rozwiązujemy w czasie $O(n^4 \cdot n \log n) = O(n^5 \log n)$ i pamięci $O(n^3 \log n)$.

Drzewo Steinera, finisz

Wniosek [Nederlof 2009]

Problem drzewa Steinera bez wag można rozwiązać w czasie $O^*(2^k)$ i pamięci wielomianowej.

Drzewo Steinera, finisz

Wniosek [Nederlof 2009]

Problem drzewa Steinera bez wag można rozwiązać w czasie $O^*(2^k)$ i pamięci wielomianowej.

Twierdzenie [Nederlof 2009]

Problem ważonego drzewa Steinera można rozwiązać w czasie $O^*(C \cdot 2^k)$ i pamięci $O^*(C)$. (Dowód tu pomijamy)

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Szybka transformata Fouriera(FFT)

Iloczyn dwóch wielomianów stopnia n można obliczyć za pomocą $O(n \log n)$ operacji arytmetycznych.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Szybka transformata Fouriera(FFT)

Iloczyn dwóch wielomianów stopnia n można obliczyć za pomocą $O(n \log n)$ operacji arytmetycznych.

- Bez straty ogólności zbiór wierzchołków $V = \{0, \ldots, n-1\}$.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Szybka transformata Fouriera(FFT)

Iloczyn dwóch wielomianów stopnia n można obliczyć za pomocą $O(n \log n)$ operacji arytmetycznych.

- Bez straty ogólności zbiór wierzchołków $V = \{0, \ldots, n-1\}$.
- Dla $X \subseteq V$ wektorem charakterystycznym X nazywamy $\xi(X) : \{0, \ldots, n-1\} \to \{0, 1\}$, gdzie $\xi(X)(i) = [i \in X]$.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Szybka transformata Fouriera(FFT)

Iloczyn dwóch wielomianów stopnia n można obliczyć za pomocą $O(n \log n)$ operacji arytmetycznych.

- Bez straty ogólności zbiór wierzchołków $V = \{0, \ldots, n-1\}$.
- Dla $X \subseteq V$ wektorem charakterystycznym X nazywamy $\xi(X) : \{0, \ldots, n-1\} \to \{0, 1\}$, gdzie $\xi(X)(i) = [i \in X]$.
- Wektorowi $\xi(X)$ odpowiada liczba $\xi(X)_3 = \sum_{i=0}^{n-1} \xi(X)(i) \cdot 3^i$.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Szybka transformata Fouriera(FFT)

Iloczyn dwóch wielomianów stopnia n można obliczyć za pomocą $O(n \log n)$ operacji arytmetycznych.

- Bez straty ogólności zbiór wierzchołków $V = \{0, \ldots, n-1\}$.
- Dla $X \subseteq V$ wektorem charakterystycznym X nazywamy $\xi(X) : \{0, \ldots, n-1\} \to \{0, 1\}$, gdzie $\xi(X)(i) = [i \in X]$.
- Wektorowi $\xi(X)$ odpowiada liczba $\xi(X)_3 = \sum_{i=0}^{n-1} \xi(X)(i) \cdot 3^i$.
- Niech $\mathcal{F}$ będzie rodziną wszystkich zbiorów niezależnych.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Szybka transformata Fouriera(FFT)

Iloczyn dwóch wielomianów stopnia n można obliczyć za pomocą $O(n \log n)$ operacji arytmetycznych.

- Bez straty ogólności zbiór wierzchołków $V = \{0, \ldots, n-1\}$.
- Dla $X \subseteq V$ wektorem charakterystycznym X nazywamy $\xi(X) : \{0, \ldots, n-1\} \to \{0, 1\}$, gdzie $\xi(X)(i) = [i \in X]$.
- Wektorowi $\xi(X)$ odpowiada liczba $\xi(X)_3 = \sum_{i=0}^{n-1} \xi(X)(i) \cdot 3^i$.
- Niech $\mathcal{F}$ będzie rodziną wszystkich zbiorów niezależnych.
- Niech $P(x) = \sum_{S \in \mathcal{F}} x^{\xi(X)_3}$. P jest wielomianem stopnia 3^n.

Kolorowanie przez FFT, cd

- Mnożymy $P(x) \cdot P(x)$ za pomocą FFT w czasie $O^*(2^n)$ i „czyścimy”: w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_3$ nie zawiera dwójek.

Kolorowanie przez FFT, cd

- Mnożymy $P(x) \cdot P(x)$ za pomocą FFT w czasie $O^*(2^n)$ i „czyścimy”: w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_3$ nie zawiera dwójek.
- Otrzymujemy wielomian $\overline{P^2}(x)$ t.ż. dla $X \subseteq V$ podgraf $G[X]$ jest 2-kolorowalny wtw gdy współczynnik w $\overline{P^2}(x)$ przy $x^{\xi(X)_3}$ jest $\neq 0$.

Kolorowanie przez FFT, cd

- Mnożymy $P(x) \cdot P(x)$ za pomocą FFT w czasie $O^*(2^n)$ i „czyścimy”: w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_3$ nie zawiera dwójek.
- Otrzymujemy wielomian $\overline{P^2}(x)$ t.ż. dla $X \subseteq V$ podgraf $G[X]$ jest 2-kolorowalny wtw gdy współczynnik w $\overline{P^2}(x)$ przy $x^{\xi(X)_3}$ jest $\neq 0$.
- $\overline{P^i}(x)$ powstaje z $\overline{P^{i-1}}(x) \cdot P(x)$ przez oczyszczenie.

Kolorowanie przez FFT, cd

- Mnożymy $P(x) \cdot P(x)$ za pomocą FFT w czasie $O^*(2^n)$ i „czyścimy”: w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_3$ nie zawiera dwójek.
- Otrzymujemy wielomian $\overline{P^2}(x)$ t.ż. dla $X \subseteq V$ podgraf $G[X]$ jest 2-kolorowalny wtw gdy współczynnik w $\overline{P^2}(x)$ przy $x^{\xi(X)_3}$ jest $\neq 0$.
- $\overline{P^i}(x)$ powstaje z $\overline{P^{i-1}}(x) \cdot P(x)$ przez oczyszczenie.
- Wtedy G jest k-kolorowalny gdy ...

Kolorowanie przez FFT, cd

- Mnożymy $P(x) \cdot P(x)$ za pomocą FFT w czasie $O^*(2^n)$ i „czyścimy”: w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_3$ nie zawiera dwójek.
- Otrzymujemy wielomian $\overline{P^2}(x)$ t.ż. dla $X \subseteq V$ podgraf $G[X]$ jest 2-kolorowalny wtw gdy współczynnik w $\overline{P^2}(x)$ przy $x^{\xi(X)_3}$ jest $\neq 0$.
- $\overline{P^i}(x)$ powstaje z $\overline{P^{i-1}}(x) \cdot P(x)$ przez oczyszczenie.
- Wtedy G jest k-kolorowalny gdy ... współczynnik w $\overline{P^k}(x)$ przy $x^{\overbrace{(1\cdots1)_3}^{n \text{ razy}}}$ jest $\neq 0$.

Kolorowanie przez FFT, cd

- Mnożymy $P(x) \cdot P(x)$ za pomocą FFT w czasie $O^*(2^n)$ i „czyścimy”: w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_3$ nie zawiera dwójek.
- Otrzymujemy wielomian $\overline{P^2}(x)$ t.ż. dla $X \subseteq V$ podgraf $G[X]$ jest 2-kolorowalny wtw gdy współczynnik w $\overline{P^2}(x)$ przy $x^{\xi(X)_3}$ jest $\neq 0$.
- $\overline{P^i}(x)$ powstaje z $\overline{P^{i-1}}(x) \cdot P(x)$ przez oczyszczenie.
- Wtedy G jest k-kolorowalny gdy ... współczynnik w $\overline{P^k}(x)$ przy $x^{\overbrace{(1\cdots1)_3}^{n \text{ razy}}}$ jest $\neq 0$.

Wniosek (mało impomujący)

Możemy znaleźć kolorowanie w czasie $O^*(3^n)$ i pamięci $O(2^n)$.

Czy to już koniec?

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Trick

$$P(x) = \sum_{S \in \mathcal{F}} x^{\xi(X)_2} = \sum_{i=0}^{n} \overbrace{\sum_{\substack{S \in \mathcal{F} \\ |S|=i}} x^{\xi(X)_2}}^{\text{ozn. } P_i(x)}$$

Wtedy

$$P(x) \cdot P(x) = \sum_{i=0}^{n} \sum_{j=0}^{n} P_i(x) \cdot P_j(x).$$

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Trick

$$P(x) = \sum_{S \in \mathcal{F}} x^{\xi(X)_2} = \sum_{i=0}^{n} \overbrace{\sum_{\substack{S \in \mathcal{F} \\ |S|=i}} x^{\xi(X)_2}}^{\text{ozn. } P_i(x)}$$

Wtedy

$$P(x) \cdot P(x) = \sum_{i=0}^{n} \sum_{j=0}^{n} P_i(x) \cdot P_j(x).$$

- Teraz P jest stopnia 2^n, P_i oczywiście też.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Trick

$$P(x) = \sum_{S \in \mathcal{F}} x^{\xi(X)_2} = \sum_{i=0}^{n} \overbrace{\sum_{\substack{S \in \mathcal{F} \\ |S|=i}} x^{\xi(X)_2}}^{\text{ozn. } P_i(x)}$$

Wtedy

$$P(x) \cdot P(x) = \sum_{i=0}^{n} \sum_{j=0}^{n} P_i(x) \cdot P_j(x).$$

- Teraz P jest stopnia 2^n, P_i oczywiście też.
- Mnożymy $P_i(x) \cdot P_j(x)$ w czasie $O^*(2^n)$ i w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_2$ ma dokładnie $i + j$ jedynek.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Trick

$$P(x) = \sum_{S \in \mathcal{F}} x^{\xi(X)_2} = \sum_{i=0}^{n} \overbrace{\sum_{\substack{S \in \mathcal{F} \\ |S|=i}} x^{\xi(X)_2}}^{\text{ozn. } P_i(x)}$$

Wtedy

$$P(x) \cdot P(x) = \sum_{i=0}^{n} \sum_{j=0}^{n} P_i(x) \cdot P_j(x).$$

- Teraz P jest stopnia 2^n, P_i oczywiście też.
- Mnożymy $P_i(x) \cdot P_j(x)$ w czasie $O^*(2^n)$ i w $P(x) \cdot P(x)$ zostawiamy tylko składniki cx^p, gdzie $(p)_2$ ma dokładnie $i + j$ jedynek.
- W podobny sposób domnażamy P jeszcze $k - 2$ razy i sprawdzamy współczynnik przy $x^{\overbrace{(1 \cdots 1)_2}^{n \text{ razy}}}$.

Kolorowanie przez FFT (Cygan, Pilipczuk 2009)

Zaraz, zaraz, ale dla dowolnego $X \subseteq V$, współczynnik przy $x^{\xi(X)_2}$ jest niezerowy wtw gdy $G[X]$ jest k-kolorowalny.

Wniosek

W czasie i pamięci $O^*(2^n)$ możemy znaleźć **wszystkie** k-kolorowalne indukowane podgrafy G.

Transformata zeta ζ i transformata Möbiusa μ

Będziemy rozważać funkcje określone na podzbiorach pewnego zbioru V i o wartościach w pewnym pierścieniu – dla ustalenia uwagi weźmiemy pierścień $(\mathbb{Z}, +, \cdot)$.

$$f \ : \ 2^V \to \mathbb{Z}$$

Poniższe transformaty zamieniają f na inną funkcję $g \ : \ 2^V \to \mathbb{Z}$.

Transformata zeta

$(\zeta f)(Y) = \sum_{X \subseteq Y} f(X)$.

Transformata Mobiusa

$(\mu f)(Y) = \sum_{X \subseteq Y} (-1)^{|Y-X|} f(X)$.

Dlaczego ζ i μ są fajne?

Wzrór inwersyjny Möbiusa

Dla dowolnego $Y \subseteq V$, mamy $f(Y) = \mu\zeta f(Y)$.

Intuicja

- Powiedzmy, że chcemy szybko policzyć $f(Y)$, ale nie umiemy.
- Powiedzmy, że umiemy natomiast szybko policzyć $(\zeta f)(X)$ dla wszystkich $X \subseteq Y$. Więc liczymy, otrzymując funkcję $g = \zeta f$...
- ... i obliczamy $\mu g(Y)$ w czasie $O^*(2^n)$ — czyli szybko.
- (później zobaczymy, że w $O^*(2^n)$ możemy nawet obliczyć $\mu g(X)$ dla wszystkich $X \subseteq Y$)

Dlaczego ζ i μ są fajne?

Wzrór inwersyjny Möbiusa

Dla dowolnego $Y \subseteq V$, mamy $f(Y) = \mu\zeta f(Y)$.

Dowód

Użyjemy zasady włączeń-wyłączeń.
Dla każdego $X \subseteq Y$ niech $F(X) = \{e_1^X, \ldots, e_{f(X)}^X\}$.

- uniwersum $U = \bigcup_{X \subseteq Y} F(X)$.
- wymagania $A_v = \{e_j^X \ : \ X \ni v\}$

Będziemy korzystać z następujących własności zbiorów F i U:

(W1) $F(X) = \bigcap_{v \in X} A_v \setminus \bigcup_{v \in Y - X} A_v$,

(W2) Dla $X_1 \neq X_2$, $F(X_1) \cap F(X_2) = \emptyset$,

(W3) $f(X) = |F(X)|$.

Dlaczego ζ i μ są fajne?

Wzrór inwersyjny Möbiusa

Dla dowolnego $Y \subseteq V$, mamy $f(Y) = \mu\zeta f(Y)$.

Dowód

(W1) $F(X) = \bigcap_{v \in X} A_v \setminus \bigcup_{v \in Y - X} A_v$,

(W2) Dla $X_1 \neq X_2$, $F(X_1) \cap F(X_2) = \emptyset$,

(W3) $f(X) = |F(X)|$.

Dlaczego ζ i μ są fajne?

Wzrór inwersyjny Möbiusa

Dla dowolnego $Y \subseteq V$, mamy $f(Y) = \mu\zeta f(Y)$.

Dowód

(W1) $F(X) = \bigcap_{v \in X} A_v \setminus \bigcup_{v \in Y-X} A_v$,

(W2) Dla $X_1 \neq X_2$, $F(X_1) \cap F(X_2) = \emptyset$,

(W3) $f(X) = |F(X)|$.

$$\zeta f(X) = \sum_{S \subseteq X} f(S) =^{(W3)} \sum_{S \subseteq X} |F(S)| =^{(W2)} |\bigcup_{S \subseteq X} F(S)|$$

$$=^{(W1)} |\bigcap_{v \in Y-X} \overline{A_v}|.$$

Dlaczego ζ i μ są fajne?

Wzrór inwersyjny Möbiusa

Dla dowolnego $Y \subseteq V$, mamy $f(Y) = \mu\zeta f(Y)$.

Dowód

$$\zeta f(X) = \sum_{S \subseteq X} f(S) =^{(W3)} \sum_{S \subseteq X} |F(S)| =^{(W2)} |\bigcup_{S \subseteq X} F(S)|$$
$$=^{(W1)} |\bigcap_{v \in Y - X} \overline{A_v}|.$$

$$\mu\zeta f(Y) = \sum_{X \subseteq Y} (-1)^{|Y-X|} \zeta f(X) = \sum_{X \subseteq Y} (-1)^{|Y-X|} |\bigcap_{v \in Y-X} \overline{A_v}| =$$
$$= \sum_{X \subseteq Y} (-1)^{|X|} |\bigcap_{v \in X} \overline{A_v}| =^{(\text{ZW-W})} |\bigcap_{v \in Y} A_v| = |F(Y)| = f(Y).$$

Cykl Hamiltona ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) = X$.
Wtedy:

Cykl Hamiltona ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) = X$.
Wtedy:

- $f(V)$ jest liczbą cykli Hamiltona w G.

Cykl Hamiltona ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) = X$.
Wtedy:

- $f(V)$ jest liczbą cykli Hamiltona w G.
- $\zeta f(X) = \sum_{S \subseteq X} f(X)$ jest liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) \subseteq X$.

Cykl Hamiltona ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) = X$.
Wtedy:

- $f(V)$ jest liczbą cykli Hamiltona w G.
- $\zeta f(X) = \sum_{S \subseteq X} f(X)$ jest liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) \subseteq X$.
- Czyli dla dowolnego X, wartość $\zeta f(X)$ możemy obliczyć w czasie $O(n^3)$.

Cykl Hamiltona ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) = X$.
Wtedy:

- $f(V)$ jest liczbą cykli Hamiltona w G.
- $\zeta f(X) = \sum_{S \subseteq X} f(X)$ jest liczbą marszrut zamkniętych W długości n takich, że $1 \in V(W)$ oraz $V(W) \subseteq X$.
- Czyli dla dowolnego X, wartość $\zeta f(X)$ możemy obliczyć w czasie $O(n^3)$.
- A więc $f(V) = \mu\zeta f(V)$ obliczymy w czasie $O^*(2^n)$ i pamięci wielomianowej.

Jeszcze jeden powód dla którego ζ i μ są fajne

Algorytm Yates'a (1937)

Jeśli mamy daną funkcję $f : 2^V \to \mathbb{Z}$, to **wszystkie** 2^n wartości ζf możemy obliczyć w czasie $O^*(2^n)$. Podobnie dla μf.

Jeszcze jeden powód dla którego ζ i μ są fajne

Algorytm Yates'a (1937)

Jeśli mamy daną funkcję $f : 2^V \to \mathbb{Z}$, to **wszystkie** 2^n wartości ζf możemy obliczyć w czasie $O^*(2^n)$. Podobnie dla μf.

Dowód dla ζ. (Dla μ analogicznie.)

Bez straty ogólności $V = \{1, \ldots, n\}$. Oznaczmy:

$$\zeta_j(X) = \sum_{\substack{S \subseteq X \\ (\{j+1,\ldots,n\} \cap X) \subseteq S}} f(S)$$

Wtedy $\zeta_n(X) = \zeta f(X)$.

Jeszcze jeden powód dla którego ζ i μ są fajne

Algorytm Yates'a (1937)

Jeśli mamy daną funkcję $f : 2^V \to \mathbb{Z}$, to **wszystkie** 2^n wartości ζf możemy obliczyć w czasie $O^*(2^n)$. Podobnie dla μf.

Dowód dla ζ. (Dla μ analogicznie.)

Bez straty ogólności $V = \{1, \ldots, n\}$. Oznaczmy:

$$\zeta_j(X) = \sum_{\substack{S \subseteq X \\ (\{j+1,\ldots,n\} \cap X) \subseteq S}} f(S)$$

Wtedy $\zeta_n(X) = \zeta f(X)$.
Obliczymy wszystkie wartości $\zeta_j(X)$, dla $j = 0, \ldots n$, $X \subseteq V$, za pomocą programowania dynamicznego:

$$\zeta_j(X) = \begin{cases} f(X) & \text{gdy } j = 0, \\ \zeta_{j-1}(X) + [j \in X]\zeta_{j-1}(X - \{j\}) & \text{gdy } j > 0. \end{cases}$$

k-kolorowanie ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j = X$.
Wtedy:

k-kolorowanie ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j = X$.
Wtedy:

- $f(X) \neq 0$ wtw $G[X]$ jest k-kolorowalny.

k-kolorowanie ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j = X$.
Wtedy:

- $f(X) \neq 0$ wtw $G[X]$ jest k-kolorowalny.
- $\zeta f(X) = \sum_{S \subseteq X} f(X)$ jest liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j \subseteq X$.

k-kolorowanie ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j = X$.
Wtedy:

- $f(X) \neq 0$ wtw $G[X]$ jest k-kolorowalny.
- $\zeta f(X) = \sum_{S \subseteq X} f(X)$ jest liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j \subseteq X$.
- Jak poprzednio, wszystkie 2^n wartości ζf możemy obliczyć w czasie i pamięci $O^*(2^n)$.

k-kolorowanie ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j = X$.
Wtedy:

- $f(X) \neq 0$ wtw $G[X]$ jest k-kolorowalny.
- $\zeta f(X) = \sum_{S \subseteq X} f(X)$ jest liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j \subseteq X$.
- Jak poprzednio, wszystkie 2^n wartości ζf możemy obliczyć w czasie i pamięci $O^*(2^n)$.
- Za pomocą algorytmu Yatesa znajdujemy $f = \mu\zeta f$.

k-kolorowanie ponownie

Dla $X \subseteq V$, niech $f(X)$ będzie liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j = X$.
Wtedy:

- $f(X) \neq 0$ wtw $G[X]$ jest k-kolorowalny.
- $\zeta f(X) = \sum_{S \subseteq X} f(X)$ jest liczbą krotek $(I_1, \ldots, I_k)$, gdzie I_j są zbiorami niezależnymi w G oraz $\bigcup_{j=1}^{k} I_j \subseteq X$.
- Jak poprzednio, wszystkie 2^n wartości ζf możemy obliczyć w czasie i pamięci $O^*(2^n)$.
- Za pomocą algorytmu Yatesa znajdujemy $f = \mu\zeta f$.
- W ten sposób znaleźliśmy **wszystkie** indukowane k-kolorowalne podgrafy G.

Iloczyn pokryciowy

Iloczyn pokryciowy

Dla dwóch funkcji $f, g : 2^V \to \mathbb{Z}$ definiujemy iloczyn pokryciowy jako funkcję $(f *_c g) : 2^V \to \mathbb{Z}$ taką, że dla dowolnego $Y \subseteq V$,

$$(f *_c g)(Y) = \sum_{A \cup B = Y} f(A)g(B).$$

Iloczyn pokryciowy

Iloczyn pokryciowy

Dla dwóch funkcji $f, g : 2^V \to \mathbb{Z}$ definiujemy iloczyn pokryciowy jako funkcję $(f *_c g) : 2^V \to \mathbb{Z}$ taką, że dla dowolnego $Y \subseteq V$,

$$(f *_c g)(Y) = \sum_{A \cup B = Y} f(A)g(B).$$

Po co nam to? Bo jest naturalne. A poza tym np:

Niech $\mathcal{F}$ będzie rodziną wszystkich zbiorów niezależnych w danym grafie G. Niech $\mathbf{1}_{\mathcal{F}} : 2^V \to \{0, 1\}$ będzie indykatorem $\mathcal{F}$, tzn. $\mathbf{1}_{\mathcal{F}}(X) = [X \in \mathcal{F}]$.

Iloczyn pokryciowy

Iloczyn pokryciowy

Dla dwóch funkcji $f, g : 2^V \to \mathbb{Z}$ definiujemy iloczyn pokryciowy jako funkcję $(f *_c g) : 2^V \to \mathbb{Z}$ taką, że dla dowolnego $Y \subseteq V$,

$$(f *_c g)(Y) = \sum_{A \cup B = Y} f(A)g(B).$$

Po co nam to? Bo jest naturalne. A poza tym np:

Niech $\mathcal{F}$ będzie rodziną wszystkich zbiorów niezależnych w danym grafie G. Niech $\mathbf{1}_\mathcal{F} : 2^V \to \{0, 1\}$ będzie indykatorem $\mathcal{F}$, tzn. $\mathbf{1}_\mathcal{F}(X) = [X \in \mathcal{F}]$. Wówczas

$$\underbrace{\mathbf{1}_\mathcal{F} *_c \mathbf{1}_\mathcal{F} *_c \cdots \mathbf{1}_\mathcal{F}}_{k \text{ razy}}(V) \neq 0 \text{ wtw } G \text{ jest } k\text{-kolorowalny.}$$

Iloczyn pokryciowy, obliczanie

Standardowo: nie umiemy policzyć $(f *_c g)$? To policzmy $\zeta(f *_c g)$.

$$\zeta(f *_c g)(Y) = \sum_{X \subseteq Y} \sum_{A \cup B = X} f(A)g(B) = \sum_{A \cup B \subseteq Y} f(A)g(B) =$$

$$= \left(\sum_{A \subseteq Y} f(A) \right) \left(\sum_{B \subseteq Y} f(B) \right) = (\zeta f(Y))(\zeta g(Y)).$$

Iloczyn pokryciowy, obliczanie

Standardowo: nie umiemy policzyć $(f *_c g)$? To policzmy $\zeta(f *_c g)$.

$$\zeta(f *_c g)(Y) = \sum_{X \subseteq Y} \sum_{A \cup B = X} f(A)g(B) = \sum_{A \cup B \subseteq Y} f(A)g(B) =$$

$$= \left(\sum_{A \subseteq Y} f(A) \right) \left(\sum_{B \subseteq Y} f(B) \right) = (\zeta f(Y))(\zeta g(Y)).$$

Czyli $(f *_c g)(Y) = \mu((\zeta f(Y))(\zeta g(Y)))$. Używamy 3x algorytmu Yates'a i dostajemy czas (i pamięć) $O^*(2^n)$.

Iloczyn pokryciowy, obliczanie

Standardowo: nie umiemy policzyć $(f *_c g)$? To policzmy $\zeta(f *_c g)$.

$$\zeta(f *_c g)(Y) = \sum_{X \subseteq Y} \sum_{A \cup B = X} f(A)g(B) = \sum_{A \cup B \subseteq Y} f(A)g(B) =$$

$$= \left(\sum_{A \subseteq Y} f(A) \right) \left(\sum_{B \subseteq Y} f(B) \right) = (\zeta f(Y))(\zeta g(Y)).$$

Czyli $(f *_c g)(Y) = \mu((\zeta f(Y))(\zeta g(Y)))$. Używamy 3x algorytmu Yates'a i dostajemy czas (i pamięć) $O^*(2^n)$.

Wniosek

Żeby obliczyć $\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V)$ wystarczy $O(\log k)$ takich operacji.

Otrzymaliśmy kolejny algorytm znajdujący wszystkie k-kolorowalne podgrafy indukowane.

Splot (Subset convolution)

Splot

Dla dwóch funkcji $f, g : 2^V \to \mathbb{Z}$ definiujemy splot jako funkcję $(f * g) : 2^V \to \mathbb{Z}$ taką, że dla dowolnego $Y \subseteq V$,

$$(f * g)(Y) = \sum_{X \subseteq Y} f(X)g(Y - X).$$

Równoważnie...

$$(f * g)(Y) = \sum_{\substack{A \cup B = Y \\ A \cap B = \emptyset}} f(A)g(B).$$

Splot (Subset convolution)

Splot

Dla dwóch funkcji $f, g : 2^V \to \mathbb{Z}$ definiujemy splot jako funkcję $(f * g) : 2^V \to \mathbb{Z}$ taką, że dla dowolnego $Y \subseteq V$,

$$(f * g)(Y) = \sum_{X \subseteq Y} f(X)g(Y - X).$$

Równoważnie...

$$(f * g)(Y) = \sum_{\substack{A \cup B = Y \\ A \cap B = \emptyset}} f(A)g(B).$$

Po co nam to? Bo jest naturalne. A poza tym np:

$\pi_G(k) = \underbrace{\mathbf{1}_{\mathcal{F}} * \mathbf{1}_{\mathcal{F}} * \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V)/k!$ jest liczbą podziałów V na k zbiorów niezależnych, natomiast $p_G(k) = \sum_{r=1}^{k} \pi_G(k)k^{\underline{r}}$ liczbą k-kolorowań.

Splot (Subset convolution) – obliczanie

Dla $f : 2^V \to \mathbb{Z}$ niech f_k oznacza f obcięte do zbiorów mocy k, tzn.:

$$f_k(S) = \begin{cases} f(S) & \text{gdy } |S| = k, \\ 0 & \text{w p.p.} \end{cases}$$

Splot (Subset convolution) – obliczanie

Dla $f : 2^V \to \mathbb{Z}$ niech f_k oznacza f obcięte do zbiorów mocy k, tzn.:

$$f_k(S) = \begin{cases} f(S) & \text{gdy } |S| = k, \\ 0 & \text{w p.p.} \end{cases}$$

Wówczas

$$(f * g)(Y) = \sum_{X \subseteq Y} f(X)g(Y - X) = \sum_{i=0}^{n} \sum_{\substack{X \subseteq Y \\ |X|=i}} f(X)g(Y - X) =$$

$$= \sum_{i=0}^{n} (f_i * g_{|Y|-i})(Y).$$

Splot (Subset convolution) – obliczanie

Dla $f : 2^V \to \mathbb{Z}$ niech f_k oznacza f obcięte do zbiorów mocy k, tzn.:

$$f_k(S) = \begin{cases} f(S) & \text{gdy } |S| = k, \\ 0 & \text{w p.p.} \end{cases}$$

Wówczas

$$(f * g)(Y) = \sum_{X \subseteq Y} f(X)g(Y - X) = \sum_{i=0}^{n} \sum_{\substack{X \subseteq Y \\ |X|=i}} f(X)g(Y - X) =$$

$$= \sum_{i=0}^{n} (f_i * g_{|Y|-i})(Y).$$

Wniosek

Wystarczy umieć obliczyć szybko $f_i * g_j$.

Splot (Subset convolution) – obliczanie $f_i * g_j$

Przypomnienie:

$$\zeta(f *_c g)(Y) = \ldots = \sum_{A,B \subseteq Y} f(A)g(B) = \ldots = (\zeta f(Y))(\zeta g(Y)).$$

Wówczas

$$(f_i * g_j)(Y) = \sum_{\substack{A \cup B = Y \\ A \cap B = \emptyset}} f_i(A)g_j(B) =$$

$$= \left(\sum_{A,B \subseteq Y} f_i(A)g_j(B) \right)_{i+j} = [(\zeta f(Y))(\zeta g(Y))]_{i+j}.$$

Wniosek

Można obliczyć liczbę k-kolorowań wszystkich podgrafów indukowanych G w czasie i pamięci $O^*(2^n)$.

Przycinanie transformat (trimmed zeta / Möbius transform)

Dla $f : 2^V \to \mathbb{Z}$ nośnikiem f nazywamy $\text{supp}(f) = \{X \subseteq V \ : \ f(X) \neq 0\}$.
Dla $\mathcal{F} \subseteq 2^V$, niech $\uparrow\mathcal{F} = \{Y \subseteq V \ : \ \text{dla pewnego } X \in \mathcal{F},\ X \subseteq Y\}$.

Algorytm Yates'a

Jeśli mamy daną funkcję $f : 2^V \to \mathbb{Z}$, to **wszystkie** niezerowe wartości ζf możemy obliczyć w czasie $O^*(|\uparrow\text{supp}(f)|)$. Podobnie dla μf.

Przycinanie transformat (trimmed zeta / Möbius transform)

Dla $f : 2^V \to \mathbb{Z}$ nośnikiem f nazywamy $\text{supp}(f) = \{X \subseteq V \ : \ f(X) \neq 0\}$.
Dla $\mathcal{F} \subseteq 2^V$, niech $\uparrow\mathcal{F} = \{Y \subseteq V \ : \ \text{dla pewnego } X \in \mathcal{F},\ X \subseteq Y\}$.

Algorytm Yates'a

Jeśli mamy daną funkcję $f : 2^V \to \mathbb{Z}$, to **wszystkie** niezerowe wartości ζf możemy obliczyć w czasie $O^*(|\uparrow\text{supp}(f)|)$. Podobnie dla μf.

Dowód dla ζ. (Dla μ analogicznie.)

Bez straty ogólności $V = \{1, \ldots, n\}$. Oznaczaliśmy

$$\zeta_j(X) = \sum_{\substack{S \subseteq X \\ (\{j+1,\ldots,n\} \cap X) \subseteq S}} f(S)$$

Zauważmy, że dla $X \notin \uparrow\text{supp}(f)$, $\zeta_j(X) = 0$.

Przycinanie transformat (trimmed zeta / Möbius transform)

Dla $f : 2^V \to \mathbb{Z}$ nośnikiem f nazywamy $\text{supp}(f) = \{X \subseteq V \ : \ f(X) \neq 0\}$.
Dla $\mathcal{F} \subseteq 2^V$, niech $\uparrow\mathcal{F} = \{Y \subseteq V \ : \ \text{dla pewnego } X \in \mathcal{F},\ X \subseteq Y\}$.

Algorytm Yates'a

Jeśli mamy daną funkcję $f : 2^V \to \mathbb{Z}$, to **wszystkie** niezerowe wartości ζf możemy obliczyć w czasie $O^*(|\uparrow\text{supp}(f)|)$. Podobnie dla μf.

Dowód dla ζ. (Dla μ analogicznie.)

Pozostaje obliczyć $\zeta_j(X)$ dla $X \in \uparrow\text{supp}(f)$.
Po prostu zapuszczamy poprzedni dynamik...

$$\zeta_j(X) = \begin{cases} f(X) & \text{gdy } j = 0, \\ \zeta_{j-1}(X) + [j \in X]\zeta_{j-1}(X - \{j\}) & \text{gdy } j > 0. \end{cases}$$

... ale zbiory X bierzemy z kolejki priorytetowej uporządkowanej po mocach zbiorów, a wartości $\zeta_{j-1}(Z)$ ze słownika w czasie $O^*(1)$.

Kolorowanie, ostatni raz

Obserwacja

Niech $\mathcal{F}$ będzie rodziną **maksymalnych** (pod względem zawierania) zbiorów niezależnych.

$$\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V) \neq 0 \text{ wtw } G \text{ jest } k\text{-kolorowalny.}$$

Kolorowanie, ostatni raz

Obserwacja

Niech $\mathcal{F}$ będzie rodziną **maksymalnych** (pod względem zawierania) zbiorów niezależnych.

$$\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V) \neq 0 \text{ wtw } G \text{ jest } k\text{-kolorowalny.}$$

Przypomnienie

$(f *_c g)(Y) = \mu((\zeta f(Y))(\zeta g(Y)))$.

Kolorowanie, ostatni raz

Obserwacja

Niech $\mathcal{F}$ będzie rodziną **maksymalnych** (pod względem zawierania) zbiorów niezależnych.

$$\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V) \neq 0 \text{ wtw } G \text{ jest } k\text{-kolorowalny.}$$

Przypomnienie

$(f *_c g)(Y) = \mu((\zeta f(Y))(\zeta g(Y)))$.

- $\text{supp}\mathbf{1}_{\mathcal{F}} = \mathcal{F}$,

Kolorowanie, ostatni raz

Obserwacja

Niech $\mathcal{F}$ będzie rodziną **maksymalnych** (pod względem zawierania) zbiorów niezależnych.

$$\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V) \neq 0 \text{ wtw } G \text{ jest } k\text{-kolorowalny.}$$

Przypomnienie

$(f *_c g)(Y) = \mu((\zeta f(Y))(\zeta g(Y)))$.

- $\text{supp}\mathbf{1}_{\mathcal{F}} = \mathcal{F}$,
- $\text{supp}\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{r \text{ razy}} \subseteq \uparrow\text{supp}\mathbf{1}_{\mathcal{F}} = \uparrow\mathcal{F}$,

Kolorowanie, ostatni raz

Obserwacja

Niech $\mathcal{F}$ będzie rodziną **maksymalnych** (pod względem zawierania) zbiorów niezależnych.

$$\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V) \neq 0 \text{ wtw } G \text{ jest } k\text{-kolorowalny.}$$

Przypomnienie

$(f *_c g)(Y) = \mu((\zeta f(Y))(\zeta g(Y)))$.

- $\text{supp}\mathbf{1}_{\mathcal{F}} = \mathcal{F}$,
- $\text{supp}\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{r \text{ razy}} \subseteq \uparrow\text{supp}\mathbf{1}_{\mathcal{F}} = \uparrow\mathcal{F}$,
- $\mathcal{F}$ możemy wylistować w czasie $O^*(|\mathcal{F}|)$ (patrz poprzednie wykłady)

Kolorowanie, ostatni raz

Obserwacja

Niech $\mathcal{F}$ będzie rodziną **maksymalnych** (pod względem zawierania) zbiorów niezależnych.

$$\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}(V) \neq 0 \text{ wtw } G \text{ jest } k\text{-kolorowalny.}$$

Przypomnienie

$(f *_c g)(Y) = \mu((\zeta f(Y))(\zeta g(Y)))$.

- $\text{supp}\mathbf{1}_{\mathcal{F}} = \mathcal{F}$,
- $\text{supp}\underbrace{\mathbf{1}_{\mathcal{F}} *_c \mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{r \text{ razy}} \subseteq \uparrow\text{supp}\mathbf{1}_{\mathcal{F}} = \uparrow\mathcal{F}$,
- $\mathcal{F}$ możemy wylistować w czasie $O^*(|\mathcal{F}|)$ (patrz poprzednie wykłady)
- **Wniosek:** Możemy obliczyć $\underbrace{\mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}$ w czasie i pamięci $O^*(|\uparrow\mathcal{F}|)$

Kolorowanie, ostatni raz

Wniosek

Możemy obliczyć $\underbrace{\mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}$ w czasie i pamięci $O^*(|\uparrow\mathcal{F}|)$

Kolorowanie, ostatni raz

Wniosek

Możemy obliczyć $\underbrace{\mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}$ w czasie i pamięci $O^*(|\uparrow\mathcal{F}|)$

Twierdzenie (Bjorklund i inni 2008)

W n-wierzchołkowym grafie o maksymalnym stopniu Δ jest $\leq (2^{\Delta+1} - 1)^{n/(\Delta+1)}$ zbiorów dominujących.

Kolorowanie, ostatni raz

Wniosek

Możemy obliczyć $\underbrace{\mathbf{1}_{\mathcal{F}} *_c \cdots \mathbf{1}_{\mathcal{F}}}_{k \text{ razy}}$ w czasie i pamięci $O^*(|\uparrow\mathcal{F}|)$

Twierdzenie (Bjorklund i inni 2008)

W n-wierzchołkowym grafie o maksymalnym stopniu Δ jest $\leq (2^{\Delta+1} - 1)^{n/(\Delta+1)}$ zbiorów dominujących.

Aaaaha

Ale $\uparrow\mathcal{F}$ zawiera same zbiory dominujące!

Wniosek

Możemy znaleźć k-kolorowanie grafu o maksymalnym stopniu Δ w czasie $O^*(2^{\Delta+1} - 1)^{n/(\Delta+1)}$.

Technika: Szybkie mnożenie macierzy

MAX-SAT (Williams 2004)

Problem MAX-SAT

Dana formuła ϕ w postaci 2-CNF, zawierająca n zmiennych. Znaleźć wartościowanie zmiennych, które maksymalizuje liczbę spełnionych klauzul.

MAX-SAT (Williams 2004)

Problem MAX-SAT

Dana formuła ϕ w postaci 2-CNF, zawierająca n zmiennych. Znaleźć wartościowanie zmiennych, które maksymalizuje liczbę spełnionych klauzul.

Będziemy się zajmować równoważnym (z dokładnością do czynnika log(#klauzul)) problemem:

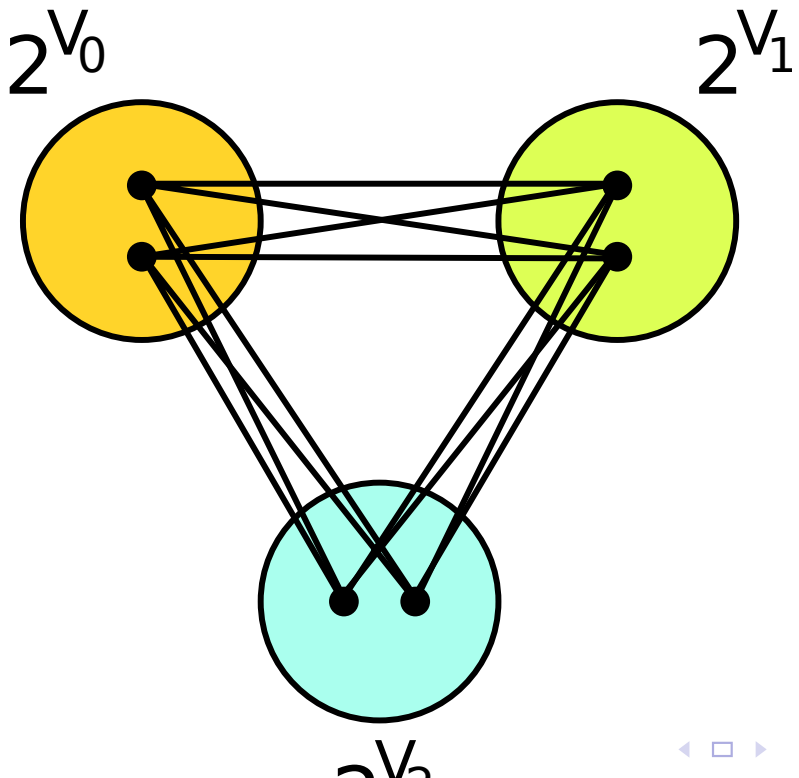
Problem MAX-SAT, wersja testująca

Dana formuła ϕ w postaci 2-CNF, zawierająca n zmiennych oraz $k \in \mathbb{N}$
Czy istnieje wartościowanie zmiennych, dla którego jest dokładnie k spełnionych klauzul.

MAX-SAT (Williams 2004)

Zbudujemy pewien graf G o $O(2^{n/3})$ wierzchołkach.

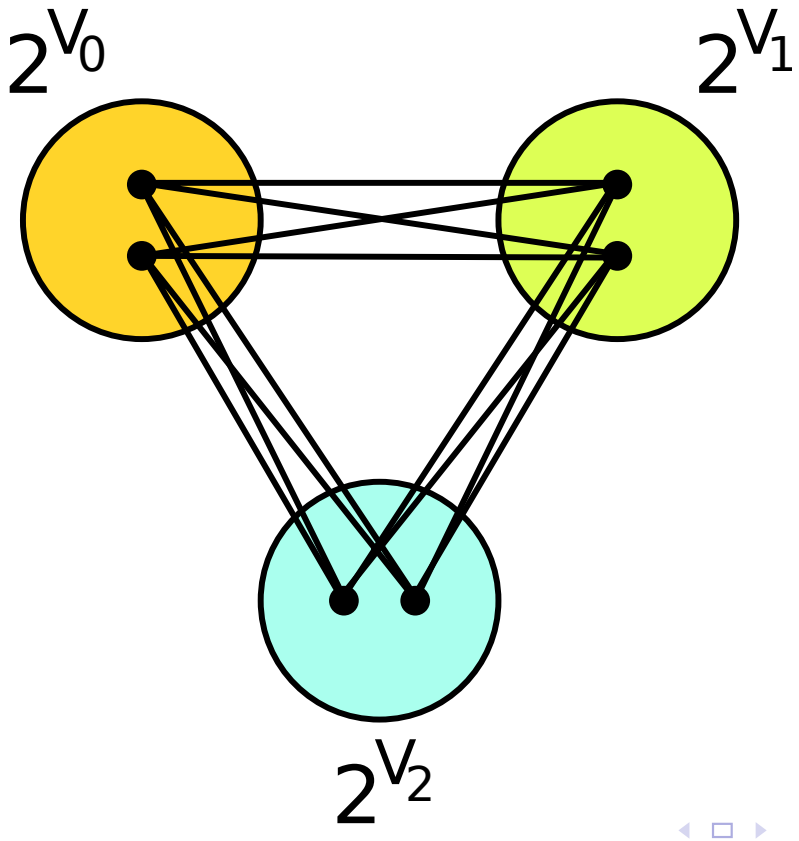
- Ustalmy dowolny podział $V = V_0 \cup V_1 \cup V_2$ na trzy równe części (tak równe jak się da).
- Wierzchołkami G są wszystkie wartościowania $v_i : V_i \to \{0, 1\}$ dla $i = 0, 1, 2$.
- Dla dowolnych $v \in V_i$, $w \in V_{(i+1) \bmod 3}$ graf G zawiera krawędź vw.



MAX-SAT (Williams 2004)

Idea rozwiązania

- Dobierzemy tak wagi na krawędziach, że waga trójkąta vwu w G będzie równa liczbie spełnionych klauzul przy wartościowaniu (v, w, u).
- Wtedy wystarczy sprawdzić, czy istnieje trójkąt o wadze k w G.



MAX-SAT (Williams 2004)

Idea rozwiązania

- Dobierzemy tak wagi na krawędziach, że waga trójkąta vwu w G będzie równa liczbie spełnionych klauzul przy wartościowaniu (v, w, u).
- Wtedy wystarczy sprawdzić, czy istnieje trójkąt o wadze k w G.

Problem 1 Jak dobrać wagi?
Niech $c(v) =$ wszystkie klauzule, które są spełnione przy wartościowaniu v.
Wtedy liczba spełnionych klauzul przy wartościowaniu (v, w, u) wynosi:

$$\begin{aligned}|c(v) \cup c(w) \cup c(u)| = |c(v)| + |c(w)| + |c(u)| \\ - |c(v) \cap c(w)| - |c(v) \cap c(u)| - |c(w) \cap c(u)| \\ + |c(v) \cap c(w) \cap c(u)|.\end{aligned}$$

MAX-SAT (Williams 2004)

Idea rozwiązania

- Dobierzemy tak wagi na krawędziach, że waga trójkąta vwu w G będzie równa liczbie spełnionych klauzul przy wartościowaniu (v, w, u).
- Wtedy wystarczy sprawdzić, czy istnieje trójkąt o wadze k w G.

Problem 1 Jak dobrać wagi?
Niech $c(v) =$ wszystkie klauzule, które są spełnione przy wartościowaniu v.
Wtedy liczba spełnionych klauzul przy wartościowaniu (v, w, u) wynosi:

$$\begin{aligned}|c(v) \cup c(w) \cup c(u)| = {} & |c(v)| + |c(w)| + |c(u)| \\ & - |c(v) \cap c(w)| - |c(v) \cap c(u)| - |c(w) \cap c(u)| \\ & + \underbrace{|c(v) \cap c(w) \cap c(u)|}_{0}.\end{aligned}$$

MAX-SAT (Williams 2004)

Idea rozwiązania

- Dobierzemy tak wagi na krawędziach, że waga trójkąta vwu w G będzie równa liczbie spełnionych klauzul przy wartościowaniu (v, w, u).
- Wtedy wystarczy sprawdzić, czy istnieje trójkąt o wadze k w G.

Problem 1 Jak dobrać wagi?
Niech $c(v) =$ wszystkie klauzule, które są spełnione przy wartościowaniu v.
Wtedy liczba spełnionych klauzul przy wartościowaniu (v, w, u) wynosi:

$$\begin{aligned}|c(v) \cup c(w) \cup c(u)| = {} & |c(v)| + |c(w)| + |c(u)| \\ & - |c(v) \cap c(w)| - |c(w) \cap c(u)| - |c(u) \cap c(v)| \\ & + \underbrace{|c(v) \cap c(w) \cap c(u)|}_{0}.\end{aligned}$$

Czyli dajemy $\text{waga}(xy) = |c(x)| - |c(x) \cap c(y)|$.

MAX-SAT (Williams 2004)

Pozostało sprawdzić, czy istnieje trójkąt o wadze k w G.

Trick

Rozważamy wszystkie $O(m^2)$ podziałów (m = liczba klauzul)
$k = k_0 + k_1 + k_2$. Dla każdego podziału budujemy graf G_{k_0,k_1,k_2} złożony tylko z:

- krawędzi o wadze k_0 między 2^{V_0} a 2^{V_1},
- krawędzi o wadze k_1 między 2^{V_1} a 2^{V_2},
- krawędzi o wadze k_2 między 2^{V_2} a 2^{V_2}.

Wtedy wystarczy...

MAX-SAT (Williams 2004)

Pozostało sprawdzić, czy istnieje trójkąt o wadze k w G.

Trick

Rozważamy wszystkie $O(m^2)$ podziałów (m = liczba klauzul)
$k = k_0 + k_1 + k_2$. Dla każdego podziału budujemy graf G_{k_0,k_1,k_2} złożony tylko z:

- krawędzi o wadze k_0 między 2^{V_0} a 2^{V_1},
- krawędzi o wadze k_1 między 2^{V_1} a 2^{V_2},
- krawędzi o wadze k_2 między 2^{V_2} a 2^{V_2}.

Wtedy wystarczy... sprawdzić, czy istnieje dowolny trójkąt.

Sprawdzanie, czy G_{k_0,k_1,k_2} zawiera trójąt

Uwaga 1

Niech A będzie macierzą sąsiedztwa pewnego grafu G. G zawiera trójkąt wtedy i tylko wtedy gdy na przekątnej macierzy A^3 istnieje element niezerowy.

Twierdzenie (Coppersmith, Winograd 1990)

Niech ω będzie najmniejszą liczbą taką, że można w czasie $O(n^\omega)$ pomnożyć dwie macierze $n \times n$. Wtedy $\omega < 2.376$.

Wniosek

Możemy sprawdzić, czy G_{k_0,k_1,k_2} zawiera trójąt w czasie i pamięci $O(2^{\omega n/3}) = O(1.732^n)$

MAX-SAT (Williams 2004): Podsumowanie

Wniosek

Możemy rozwiązać MAX-SAT w czasie i pamięci $O(1.732^n)$.

MAX-SAT (Williams 2004): Podsumowanie

Wniosek

Możemy rozwiązać MAX-SAT w czasie i pamięci $O(1.732^n)$.

Łatwo przerobić nasz algorytm (jak?) żeby dostać

Wniosek

Możemy zliczyć wszystkie rozwiązania optymalne MAX-SAT w czasie i pamięci $O(1.732^n)$.

Technika: Local-Search

3-SAT, ponownie

Odległość Hamminga $H(v, w)$ dwóch wartościowań v, w to liczba zmiennych na których wartościowania się różnią.

obserwacja

Kula o promieniu d zawiera $\sum_{k=0}^{d} \binom{n}{k}$ wartościowań. ($\Theta(2^n)$ dla $d = n/2$).

obserwacja

Dla danego wartościowania v w czasie $O^*(3^d)$ można sprawdzić, czy istnieje wartościowanie spełniające w takie, że $H(v, w) \leq d$.

3-SAT, ponownie

Odległość Hamminga $H(v, w)$ dwóch wartościowań v, w to liczba zmiennych na których wartościowania się różnią.

obserwacja

Kula o promieniu d zawiera $\sum_{k=0}^{d} \binom{n}{k}$ wartościowań. ($\Theta(2^n)$ dla $d = n/2$).

obserwacja

Dla danego wartościowania v w czasie $O^*(3^d)$ można sprawdzić, czy istnieje wartościowanie spełniające w takie, że $H(v, w) \leq d$.

Dowód

Weź dowolną niespełnioną klauzulę i dla każdego z jej literałów rekurencyjnie sprawdź czy istnieje wartościowanie spełniające, przy którym ten literał ma wartość TRUE.
Dostajemy rekurencję $T(d) = 3T(d-1)$, czyli $T(d) = 3^d$.

3-SAT, ponownie

Odległość Hamminga $H(v, w)$ dwóch wartościowań v, w to liczba zmiennych na których wartościowania się różnią.

obserwacja

Kula o promieniu d zawiera $\sum_{k=0}^{d} \binom{n}{k}$ wartościowań. ($\Theta(2^n)$ dla $d = n/2$).

obserwacja

Dla danego wartościowania v w czasie $O^*(3^d)$ można sprawdzić, czy istnieje wartościowanie spełniające w takie, że $H(v, w) \leq d$.

obserwacja

Dla dowolnego wartościowania v, jest $H(v, 0^n) \leq n/2$ lub $H(v, 1^n) \leq n/2$.

3-SAT, ponownie

Odległość Hamminga $H(v, w)$ dwóch wartościowań v, w to liczba zmiennych na których wartościowania się różnią.

obserwacja

Kula o promieniu d zawiera $\sum_{k=0}^{d} \binom{n}{k}$ wartościowań. ($\Theta(2^n)$ dla $d = n/2$).

obserwacja

Dla danego wartościowania v w czasie $O^*(3^d)$ można sprawdzić, czy istnieje wartościowanie spełniające w takie, że $H(v, w) \leq d$.

obserwacja

Dla dowolnego wartościowania v, jest $H(v, 0^n) \leq n/2$ lub $H(v, 1^n) \leq n/2$.

Wniosek [Schöning 2001]

3-SAT można rozwiązać w czasie $O^*(3^{n/2}) = O^*(1.7321^n)$.
(Przypomnienie: Monien i Speckenmeyer: $O^*(1.62^n)$).

3-SAT, jeszcze raz

Próba: Wylosuj wartościowanie v i sprawdź kulę $B(v, n/4)$ w czasie $O^*(3^{n/4})$.

3-SAT, jeszcze raz

Próba: Wylosuj wartościowanie v i sprawdź kulę $B(v, n/4)$ w czasie $O^*(3^{n/4})$.

$$\Pr[\text{Próba się uda jeśli } \exists \text{ wartościowanie spełniające}] = \frac{\sum_{k=0}^{n/4} \binom{n}{k}}{2^n}.$$

Ze wzoru Stirlinga $n! = \Theta((\frac{n}{e})^n \sqrt{n})$:

$$\sum_{k=0}^{n/4} \binom{n}{k} = \Theta\left(\frac{(\frac{n}{e})^n \sqrt{n}}{(\frac{n}{4e})^{n/4} \sqrt{n/4} (\frac{3n}{4e})^{3n/4} \sqrt{3n/4}}\right)$$

$$= \Theta\left(\frac{4^n}{3^{3/4n} \sqrt{n}}\right) = \Theta\left(\left(\frac{256}{27}\right)^{n/4} n^{-1/2}\right).$$

$$\frac{\sum_{k=0}^{n/4} \binom{n}{k}}{2^n} = \Theta\left(\frac{\left(\frac{256}{27}\right)^{n/4} n^{-1/2}}{16^{n/4}}\right) = \Theta\left(\left(\frac{16}{27}\right)^{n/4} n^{-1/2}\right).$$

3-SAT, jeszcze raz

Próba: Wylosuj wartościowanie v i sprawdź kulę $B(v, n/4)$ w czasie $O^*(3^{n/4})$.

$$\Pr[\text{Próba się uda jeśli } \exists \text{ wartościowanie spełniające}] \geq c \left(\frac{16}{27}\right)^{n/4} n^{-1/2}.$$

3-SAT, jeszcze raz

Próba: Wylosuj wartościowanie v i sprawdź kulę $B(v, n/4)$ w czasie $O^*(3^{n/4})$.

$$\Pr[\text{Próba się uda jeśli } \exists \text{ wartościowanie spełniające}] \geq c \left(\frac{16}{27}\right)^{n/4} n^{-1/2}.$$

Wykonujemy $\frac{100}{c} \cdot \left(\frac{27}{16}\right)^{n/4} n^{1/2}$ prób, co zajmuje czas $O^*(\left(\frac{27}{16}\right)^{n/4} \cdot 3^{n/4}) = O^*(\left(\frac{3}{2}\right)^n)$.

3-SAT, jeszcze raz

Próba: Wylosuj wartościowanie v i sprawdź kulę $B(v, n/4)$ w czasie $O^*(3^{n/4})$.

$$\Pr[\text{Próba się uda jeśli } \exists \text{ wartościowanie spełniające}] \geq c \left(\frac{16}{27}\right)^{n/4} n^{-1/2}.$$

Wykonujemy $\frac{100}{c} \cdot \left(\frac{27}{16}\right)^{n/4} n^{1/2}$ prób, co zajmuje czas $O^*(\left(\frac{27}{16}\right)^{n/4} \cdot 3^{n/4}) = O^*(\left(\frac{3}{2}\right)^n)$.

$\Pr[\text{Wszystkie próby się nie udadzą jeśli } \exists \text{ war. spełn.}]$
$\leq (1 - c \left(\frac{16}{27}\right)^{n/4} n^{-1/2})^{\frac{100}{c} \cdot \left(\frac{27}{16}\right)^{n/4} n^{1/2}} \leq e^{-100}$.

3-SAT, jeszcze raz

Próba: Wylosuj wartościowanie v i sprawdź kulę $B(v, n/4)$ w czasie $O^*(3^{n/4})$.

$$\Pr[\text{Próba się uda jeśli } \exists \text{ wartościowanie spełniające}] \geq c \left(\frac{16}{27}\right)^{n/4} n^{-1/2}.$$

Wykonujemy $\frac{100}{c} \cdot \left(\frac{27}{16}\right)^{n/4} n^{1/2}$ prób, co zajmuje czas $O^*(\left(\frac{27}{16}\right)^{n/4} \cdot 3^{n/4}) = O^*(\left(\frac{3}{2}\right)^n)$.

$\Pr[\text{Wszystkie próby się nie udadzą jeśli } \exists \text{ war. spełn.}]$
$\leq (1 - c \left(\frac{16}{27}\right)^{n/4} n^{-1/2})^{\frac{100}{c} \cdot \left(\frac{27}{16}\right)^{n/4} n^{1/2}} \leq e^{-100}$.

Wniosek [Dantsin, Goerdt, Hirsch, Kannan, Kleinberg, Papadimitriou, Raghavam, Schöning 01]

Istnieje algorytm randomizowany Monte-Carlo dla problemu 3-SAT działający w czasie $O^*(1.5^n)$. Algorytm ten można zderandomizować. (I nie jest najlepszy znany.)

3-SAT, ostatni raz

Próba: Wylosuj wartościowanie v i powtarzaj maksymalnie $3n$ razy: wybierz losową niespełnioną klauzulę i zmień wartość zmiennej losowego literału tej klauzuli.

3-SAT, ostatni raz

Próba: Wylosuj wartościowanie v i powtarzaj maksymalnie $3n$ razy: wybierz losową niespełnioną klauzulę i zmień wartość zmiennej losowego literału tej klauzuli.

Fakt (dowód pomijamy, patrz książka Mitzenmachera i Upfala)

$$\Pr[\text{Próba się uda jeśli } \exists \text{ wart. spełn. } v^*] \geq \frac{c}{\sqrt{n}} \left(\frac{3}{4}\right)^n$$

3-SAT, ostatni raz

Próba: Wylosuj wartościowanie v i powtarzaj maksymalnie $3n$ razy: wybierz losową niespełnioną klauzulę i zmień wartość zmiennej losowego literału tej klauzuli.

Fakt (dowód pomijamy, patrz książka Mitzenmachera i Upfala)

$$\Pr[\text{Próba się uda jeśli } \exists \text{ wart. spełn. } v^*] \geq \frac{c}{\sqrt{n}} \left(\frac{3}{4}\right)^n$$

Jak zwykle, powtarzamy $100 \cdot \frac{\sqrt{n}}{c} \left(\frac{4}{3}\right)^n$ razy i dostajemy prawdopodobieństwo błędu mniejsze niż e^{-100}.

Wniosek [Schöning 2002]

Istnieje algorytm randomizowany Monte-Carlo dla problemu 3-SAT działający w czasie $O^*((4/3)^n)$.

Ćwiczenia

- MAX-CUT w $O(1.732^n)$,
- Liczba doskonałych skojarzeń w $O(1.732^n)$,
- Szeregowanie n zadań danych jako (r_i, d_i, p_i) – znaleźć szeregowanie t_i na 1 procesorze tak, że $[t_i, t_i + p_i] \subseteq [r_i, d_i]$ oraz dla $i \neq j$, $[t_i, t_i + p_i) \cap [t_j, t_j + p_j) = \emptyset$. W czasie $O^*(2^n \max_i d_i)$.
- Samoredukowalność: kolorowanie, cykl Hamiltona, ...