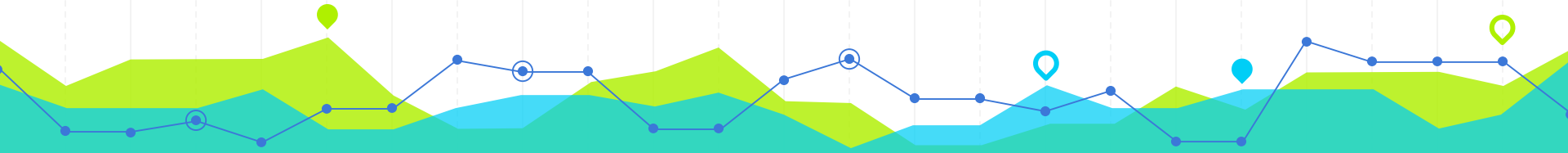
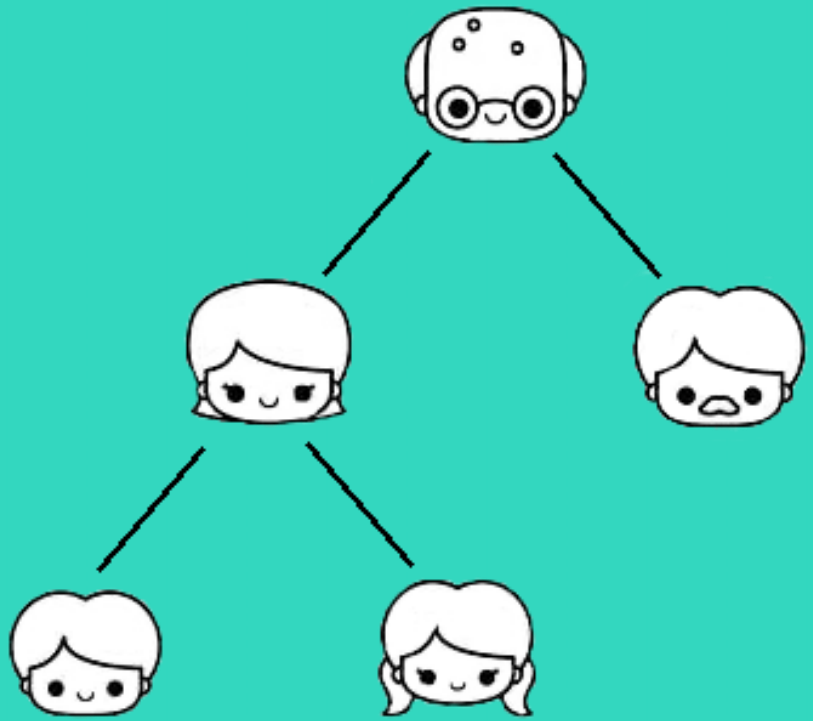
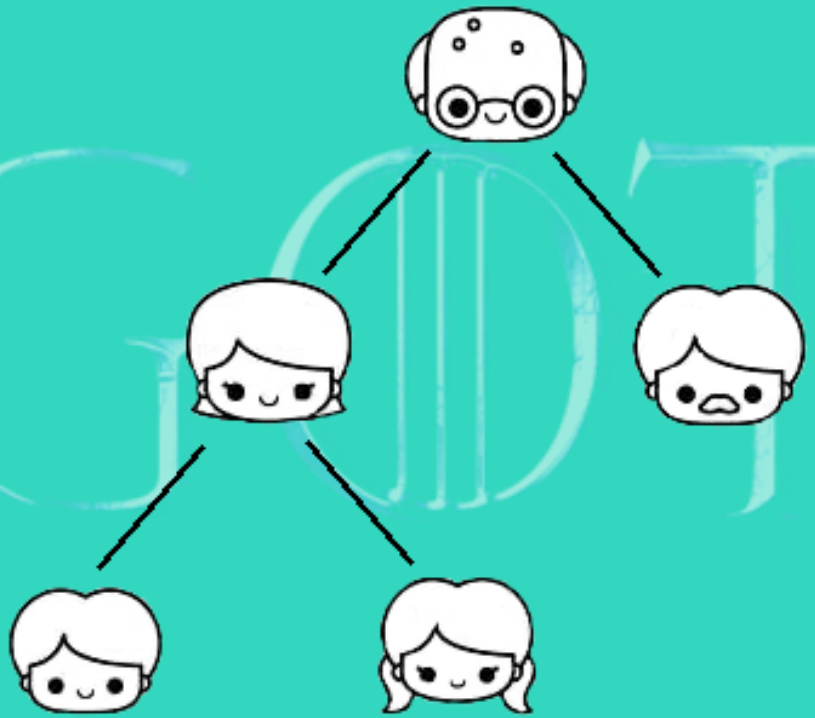
Algorytmika problemów trudnych obliczeniowo



Algorytmy randomizowane
Color coding

Michalina Mizura, Maja Kabus, Kamil Kaznowski
MiNI PW 15.04.2019

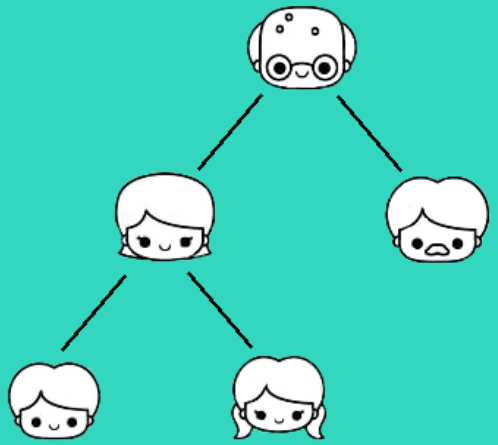


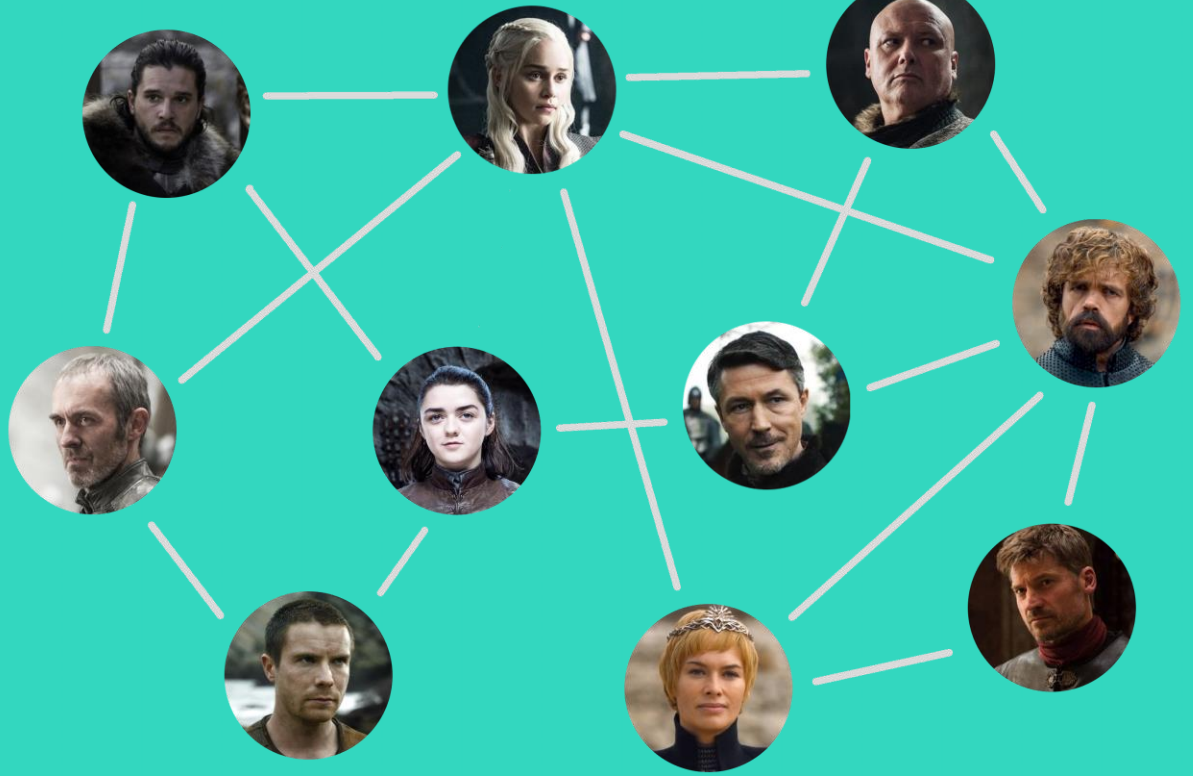


Rodzina jedzie na zlot fanów GOT. Każdy z członków rodziny chce się przebrać za jakąś postać:

- Każdy za inną postać;
- Postaci rodzica i dziecka powinny mieć ze sobą coś wspólnego.

Jaką postać może wybrać każdy z członków rodziny?





Wybór jest problemem (NP) trudnym

Naszym celem jest znalezienie danego drzewa jako podgrafu w innym grafie

- Uogólnienie problemu ścieżki Hamiltona (który jest NP-zupełny)
- Szczególny przypadek szukania dowolnego grafu - możemy próbować wykorzystać strukturę drzewa

Podejście brute-force (1)

Graf G, drzewo T, $|V(G)| = n$, $|V(T)| = k$

Dla każdego podzbioru $V(G)$ o k elementach:
Dla każdej permutacji $V(T)$:
Sprawdź, czy permutacja $V(T)$ dopasowana do wybranych wierzchołków z G jest poprawnym dopasowaniem (czy istnieją takie same krawędzie jak w T)

$$O\left(\binom{n}{k} \cdot k! \cdot k\right) = O(n^k \cdot k)$$

Podejście brute-force (2)

Backtracking:

Ukorzeńmy T w r_T.

Będziemy po kolei dopasowywać kolejne wierzchołki z T do G.
Jeśli nie jesteśmy w stanie, wycofujemy się.

Prosta implementacja osiąga teoretyczną złożoność $O(n^k)$,
ale w praktyce zachowuje się nieco lepiej


CAN YOU PROMISE
THAT I WILL COME BACK?

Algorytmy randomizowane

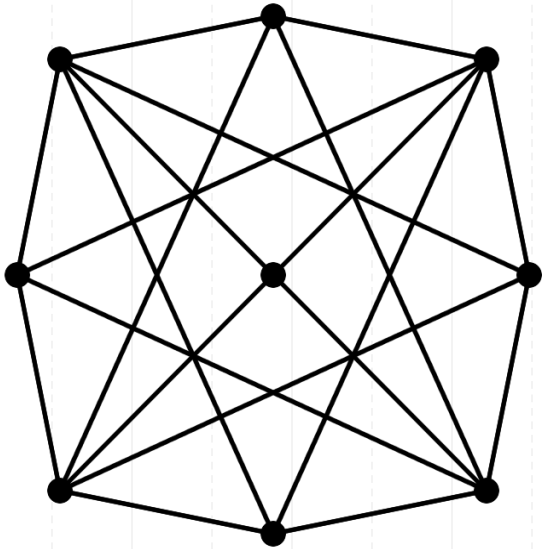
Las Vegas

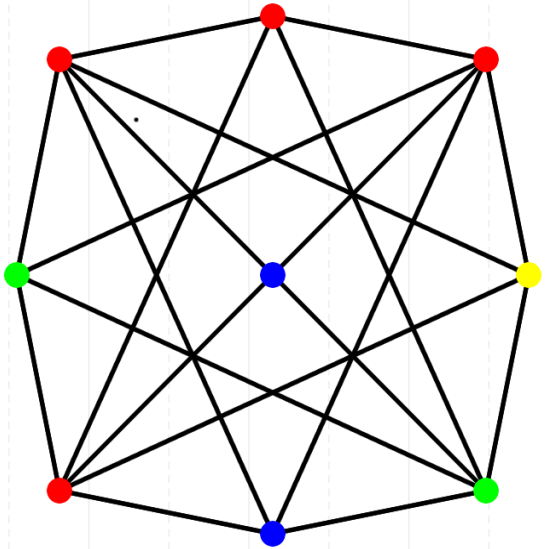
- Zawsze daje dobry wynik
- Czas działania jest losowy
- *QuickSort*

Monte Carlo

- Może zwrócić błędny wynik
- Błąd jedno- lub dwustronny
- *Color coding*

Color coding





Color coding

Jest to algorytm Monte Carlo z jednostronnym błędem (możliwe jedynie fałszywe odpowiedzi negatywne).

- p – prawdopodobieństwo poprawnego wyniku
- $1 - p$ – prawdopodobieństwo błędnego wyniku
- po m powtórzeniach prawdopodobieństwo błędu spada do $(1 - p)^m$

czyli szukanie ścieżki metodą *color coding*

Color coding do szukania ścieżki

Mamy dany graf $G = (V, E)$, $|V| = n$.

Jak sprawdzić, czy G zawiera ścieżkę o długości k?

Color coding do szukania ścieżki

Kolorujemy wierzchołki G na k kolorów; zakładamy, że jeżeli szukana ścieżka istnieje, to będzie kolorowa.

S – niepusty podzbiór zbioru kolorów

u – wierzchołek grafu G

$Path[S][u]$ – prawda wtw. istnieje kolorowa ścieżka o końcu w u wykorzystująca wszystkie kolory z S (i tylko te kolory)

Color coding do szukania ścieżki

Dla każdego S i dla każdego u:

Jeżeli $\chi(u) \notin S$, to $Path[S][u] = \text{false}$

Wpp.:

Jeżeli $|S| = 1$, to $Path[S][u] = \text{true}$

Wpp. $Path[S][u] = \bigvee\{Path[S \setminus \{\chi(u)\}][v] : uv \in E(G)\}$

Color coding do szukania ścieżki

- Złożoność – $2^k n^{O(1)}$ - wielomianowa (względem n)!
- Prawdopodobieństwo dobrego kolorowania – co najmniej e^{-k}
- Dla $k = 5$: $e^{-k} \approx 6{,}7 * 10^{-3}$
- Czyli prawdopodobieństwo błędu $\leq 0{,}9933$

Color coding do szukania ścieżki

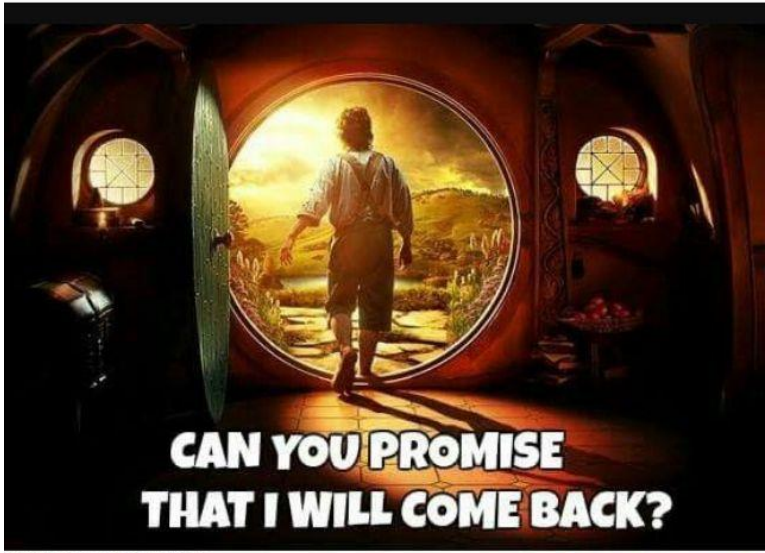
- Po e^k powtórzeniach mamy stałe ograniczenie górne prawdopodobieństwa błędu: e^{-1}
- Dalsze powtórzenia jeszcze bardziej zmniejszą to prawdopodobieństwo

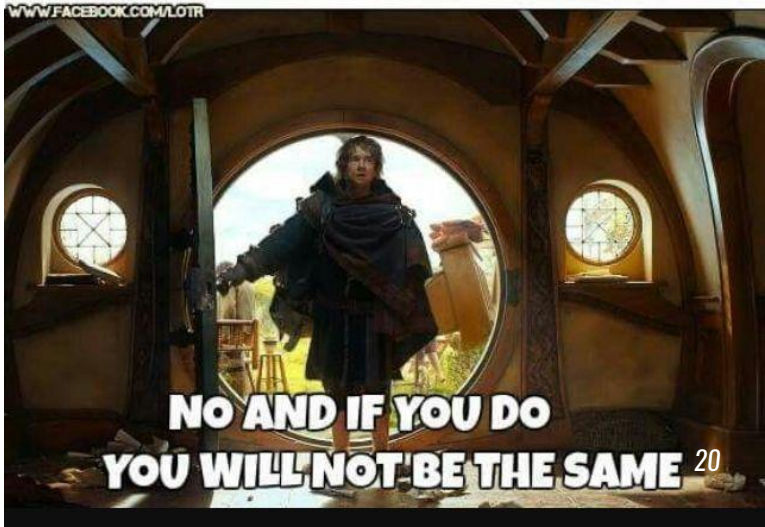
Color coding do szukania ścieżki

Dla $k = 5$:

- 149 powtórzeń – p. błędu $\leq 0{,}37$
- 1000 powtórzeń – p. błędu $\leq 0{,}0012$
- 10000 powtórzeń – p. błędu $\leq 4{,}3 * 10^{-30}$

Wracając do oryginalnego problemu...





Tam...

$G = (V, E)$ - dowolny graf

$T = (V_T, E_T),\ |V_T| = k$
– dane drzewo o k wierzchołkach

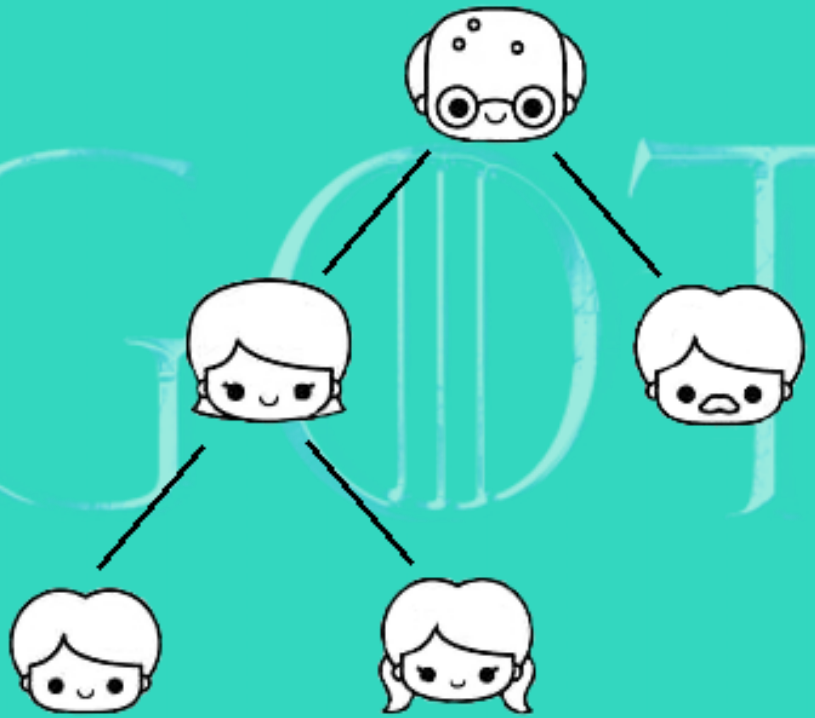
Znaleźć, jeśli istnieje, podgraf G izomorficzny z T.

...i z powrotem

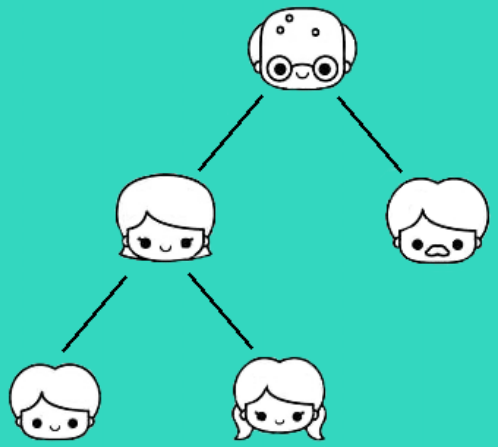
$G = (V, E)$ - graf **z wierzchołkami w k kolorach**

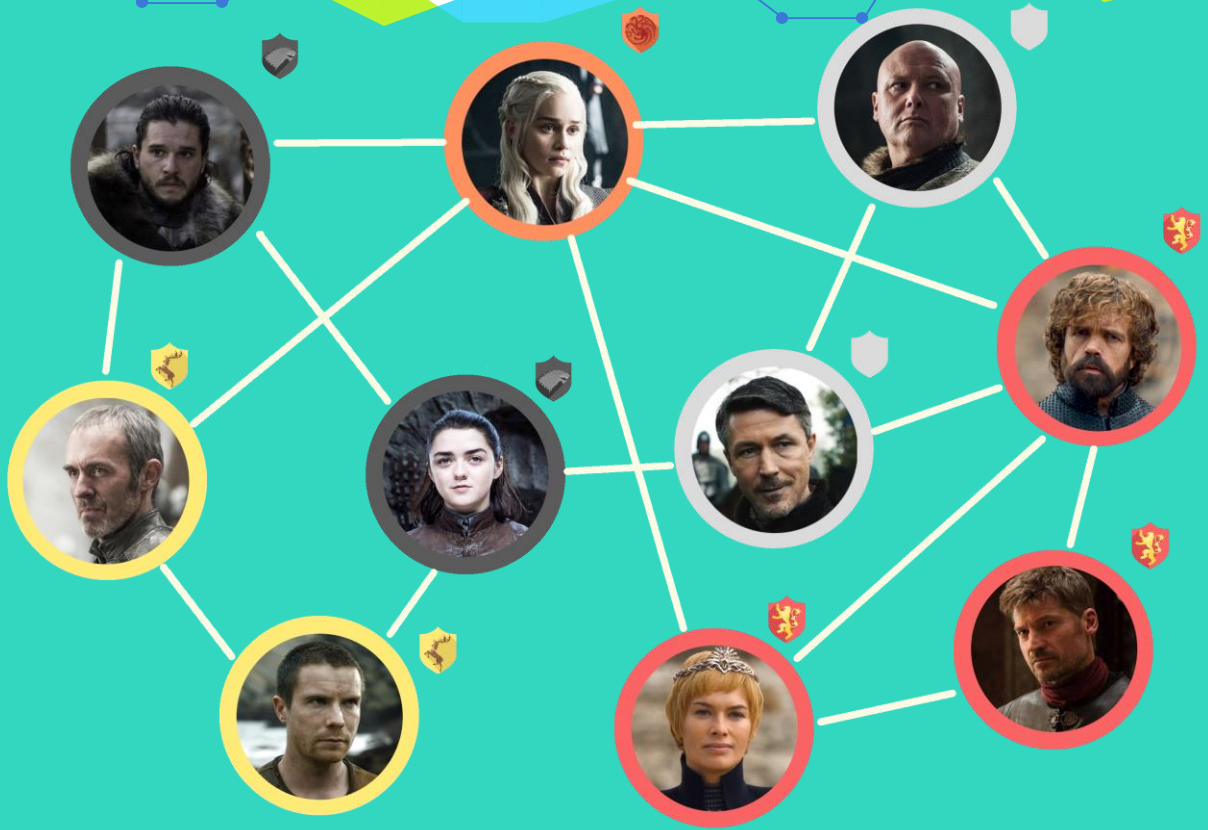
$T = (V_T, E_T),\ |V_T| = k$
– dane drzewo o k wierzchołkach

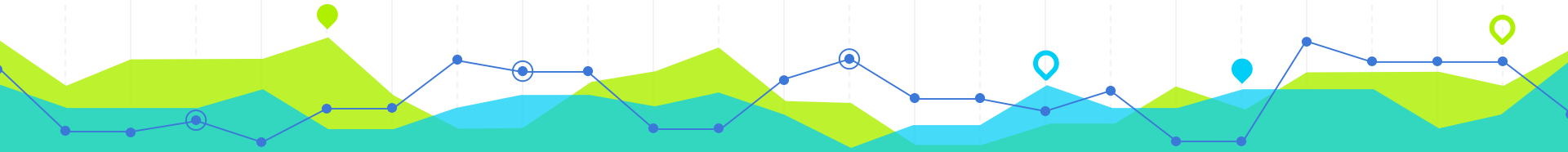
Znaleźć, jeśli istnieje, **kolorowy** podgraf G izomorficzny z T.



- Rozpatrujemy tylko postaci z 5 rodów (tyle rodów - kolorów, ile członków rodziny)
- Każdy członek rodziny wybiera postać z innego rodu







Idźmy w głąb!

Depth-First Search

Programowanie dynamiczne

Algorytm 1 Main

```
function Main( )
    Pokoloruj G na k kolorów
    K ← Kolekcja[k][n]    // Struktura nk kolekcji zbiorów kolorów
    x ← dowolny wierzchołek z V(T)
    DFS(x)
    if  w K dla pewnej pary wierzchołków z G i T istnieje zbiór
     kolorów liczności k  then
        return true
    return false
end function
```

Jak aktualizować zbiory kolorów dla wierzchołka r?

Łączymy zbiory: $Kolekcja[r][u]$ i $Kolekcja[c][v]$, gdzie: $< u, v > \in E(G)$, $< r, c > \in E(T)$, c – następnik r .

Dopasowujemy:

- krawędzie $< r, c > \in E(T)$ i $< u, v > \in E(G)$,
- poddrzewo T o korzeniu w c i jego kolorową kopię o korzeniu w v,
- poddrzewo T składające się z r i drzew o korzeniach w uprzednio przerobionych następnikach r oraz jego kolorową kopię o korzeniu w u .

Algorytm 2 Aktualizacja kolekcji kolorów wierzchołków w G

procedure UpdateCollections(r)
  for all c - następnik r w T **do**
    $K_t \leftarrow$ pusta kolekcja par
      (wierzchołek z G, zbiór zbiorów kolorów)
    for all $< u, v > \in E(G)$ **do**
      for all zbiór kolorów $C_u \in$ kolekcja $K[r][u]$ **do**
        for all zbiór kolorów $C_v \in$ kolekcja $K[c][v]$ **do**
          if $C_u \bigcap C_v = \emptyset$ **then**
            Dodaj $C_u \bigcup C_v$ do kolekcji $K_t[u]$
    $K[r][:] \leftarrow K_t[:]$
end procedure

Złożoność obliczeniowa

Szukanie kolorowego podgrafu izomorficznego z drzewem.

Dla każdego wierzchołka T:

Podłącz do rodzica - dla każdej krawędzi (u, v) w $E(G)$:

Połącz odpowiadające kolekcje rodzica i dziecka

Złożoność: $\boldsymbol{O(k \cdot E(G) \cdot f(k))}$,

gdzie $f(k)$ to koszt połączenia dwóch kolekcji.

Złożoność obliczeniowa – naiwne łączenie kolekcji

Kolekcja kolorów dla poddrzewa wielkości t ma rozmiar:
$\binom{k}{t} < 2^k$

Naiwne łączenie kolekcji, iloczyn zbiorów:
Dla każdej pary sprawdzamy, czy jest rozłączna.

$f(k)$ jest $O(2^k \cdot 2^k) = O(2^{2k}) = \boldsymbol{O(4^k)}$

Złożoność obliczeniowa – sprytne łączenie kolekcji

Szukamy tylko par zbiorów wzajemnie rozłącznych!

Łączymy kolekcje K_1 i K_2 o rozmiarach t_1 i t_2.

Dla każdego elementu c_1 z kolekcji K_1:
Dla każdej istniejącego zbioru c_2 o t_2 kolorach
(tylko spośród pozostałych kolorów, nieużytych w c_1):
Sprawdzamy, czy zbiór c_2 jest w kolekcji K_2,
jeśli tak → to dodajemy sumę zbiorów c_1 i c_2

Złożoność obliczeniowa – sprytne łączenie kolekcji

Dla każdego elementu c_1 z kolekcji K_1:
  Dla każdej istniejącego zbioru c_2 o t_2 kolorach
  (tylko spośród pozostałych kolorów, nieużytych w c_1):
    Sprawdzamy, czy zbiór c_2 jest w kolekcji K_2,
    jeśli tak → to dodajemy sumę zbiorów c_1 i c_2

$\binom{k}{t_1} \cdot \binom{k-t_1}{t_2}$ - podział zbioru k kolorów na 3 części: te z c_1, te z c_2 i te spoza tych dwóch zbiorów.

$\binom{k}{t_1} \cdot \binom{k-t_1}{t_2} < 3^k$, więc $f(k)$ jest $\boldsymbol{O(3^k \cdot k)}$ (hurra!)

Złożoność obliczeniowa

Osiągnięta złożoność:

$$O(k^2 \cdot E(G) \cdot 3^k)$$

Czy możemy zejść jeszcze niżej?

Złożoność obliczeniowa – ograniczenie na liczbę krawędzi

Fakt: Dla dowolnego grafu G, jeśli $E(G) > k \cdot V(G)$, to w G znajduje się dowolne drzewo o k wierzchołkach, i możemy je znaleźć w czasie wielomianowym względem zarówno k, jak i $V(G)$.

Daje nam to górne oszacowanie na liczbę krawędzi:

$$E(G) \leq k \cdot V(G)$$

Złożoność obliczeniowa znajdowania kolorowego drzewa w grafie

Ostateczna złożoność:

$$O(k^3 \cdot V(G) \cdot 3^k)$$

Jest to jednak złożoność znajdowania kolorowego drzewa. Najpierw musimy trafić na dobre kolorowanie grafu G.

Złożoność obliczeniowa znajdowania drzewa w grafie

Opisany algorytm znalezienia drzewa o k wierzchołkach jako podgrafu G, z pewnością $(1 - \delta)$, ma złożoność obliczeniową:

$$O\left(k^3 \cdot V(G) \cdot (3e)^k \cdot log\left(\frac{1}{\delta}\right)\right)$$

Podsumowując...

Na szaro

- $O(n^k)$
- Niezawodność
- Można poszczególne przypadki rozpatrywać jednocześnie
- Ogólny problem izomorfizmu podgrafu

Na kolorowo

- $O(e^k 2^{O(k)} n)$
- Losowość – false negatives
- Niezależne powtórzenia → możliwość zrównoleglenia
- Można rozszerzyć do znajdowania podgrafu o ograniczonej szerokości drzewowej

...tylko po co to wszystko?

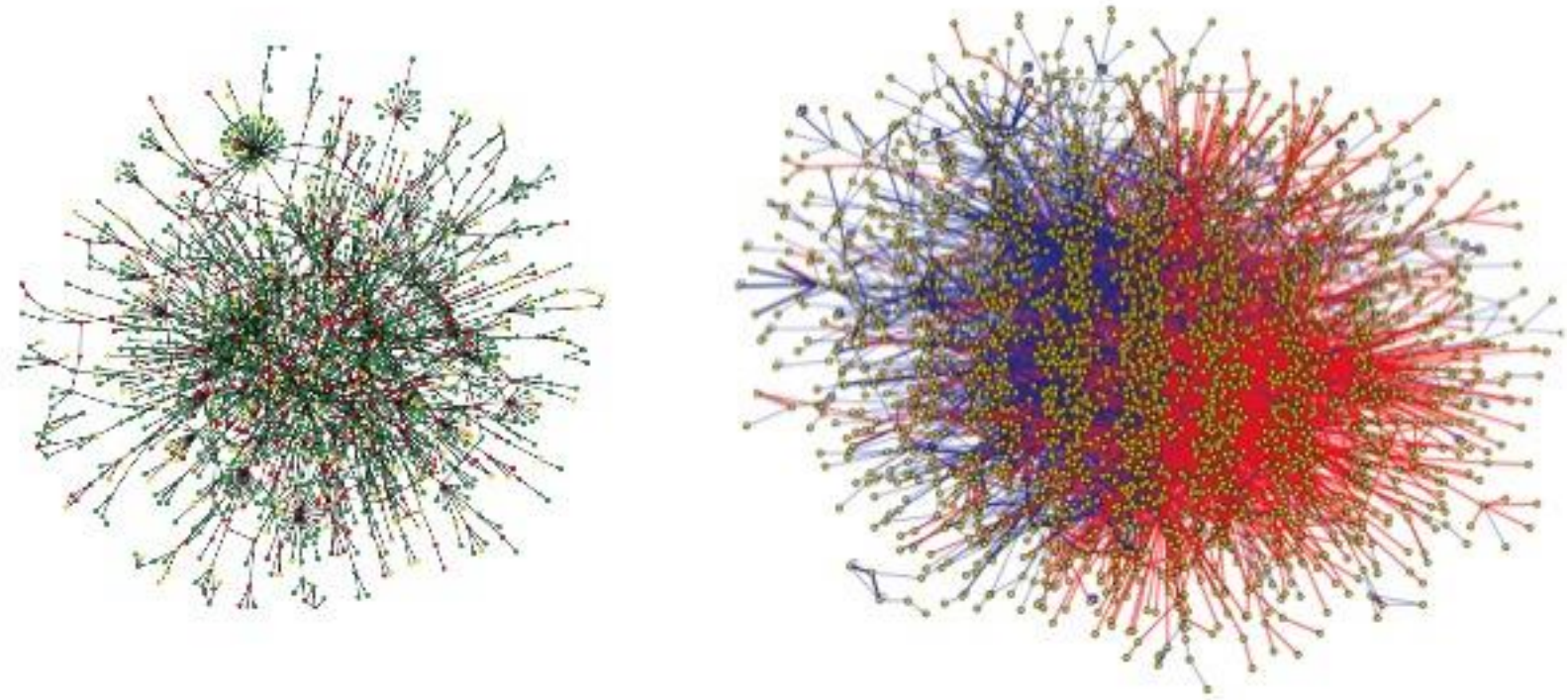


Figure 16 Yeast (left) and human (right) interactomes obtained using the yeast-two hybrid method. Images reprinted by permission from Macmillan Publishers Ltd: Jeong et al. Nature 2001. 411 (3) and Rual et al. Nature 2005: 437 (4).

Źródło: [3]

Literatura

[1] M. Cygan *et al.*, *Parameterized Algorithms*. Cham Springer International Publishing, 2015.

[2] F. Hormozdiari, I. Hajirasouliha, N. Alon, P. Dao, and S. C.Sahinalp, “Biomolecular network motif counting and discovery by color coding”, *Bioinformatics*, t. 24, nr 13, s. 241–249, 2008.

[3] EMBL-EBI Train online. (2016). *Protein-protein interaction networks.* [online] Dostęp pod: https://www.ebi.ac.uk/training/online/course/network-analysis-protein-interaction-data-introduction/protein-protein-interaction-networks [Dostęp 5 Apr. 2019].

[4] EMBL-EBI Train online. (2016). *The importance of molecular interactions.* [online] Dostęp pod: https://www.ebi.ac.uk/training/online/course/protein-interactions-and-their-importance/protein-protein-interactions/importance-molecular-i [Dostęp 5 Apr. 2019].

„(...) albowiem błąd się błędem odciska, błędem obraca, błąd tworzy, aż losowość zamienia się w Los Świata."