Sprawozdanie

Imię i nazwisko - wykonane zadanie :
Arkadiusz Pituła - porównanie algorytmów, wnioski
Jakub Pranica - implementacja algorytmów
Robert Trojan - sprawozdanie

Kierunek / grupa : Informatyka / 13i
Przedmiot : Metody programowania
Prowadzący : dr. inż. Zbigniew Kokosiński

Sprawozdanie na zajęcia z dnia 22.05.17r

1.Temat : Algorytmy zachłanne dla problemu kolorowania grafu

2. Opis problemu : **Kolorowanie grafu** polega w ogólności na przypisaniu określonym elementom składowym grafu (najczęściej wierzchołkom, rzadziej krawędziom lub ścianom) wybranych kolorów według ściśle określonych reguł. Klasyczne (czyli wierzchołkowe) kolorowanie grafu jest związane z przypisaniem wszystkim wierzchołkom w grafie jednej z wybranych barw w ten sposób, aby żadne dwa sąsiednie wierzchołki nie miały tego samego koloru. Innymi słowy, pewne pokolorowanie wierzchołkowe jest poprawne (legalne, dozwolone) wtedy, gdy końcom żadnej krawędzi nie przypisano tego samego koloru.

Klasyczne (wierzchołkowe) kolorowanie grafu – przyporządkowywanie wierzchołkom grafu liczb naturalnych w taki sposób, aby końce żadnej krawędzi nie miały przypisanej tej samej liczby. Ze względów historycznych oraz dla lepszego zobrazowania problemu mówi się o kolorowaniu, przy czym różnym kolorom odpowiadają różne liczby.

Pokolorowaniem wierzchołków grafu nazywamy jedno konkretne przyporządkowanie kolorów wierzchołkom. Pokolorowanie jest **legalne (dozwolone)**, gdy końcom żadnej krawędzi nie przyporządkowano tego samego koloru.

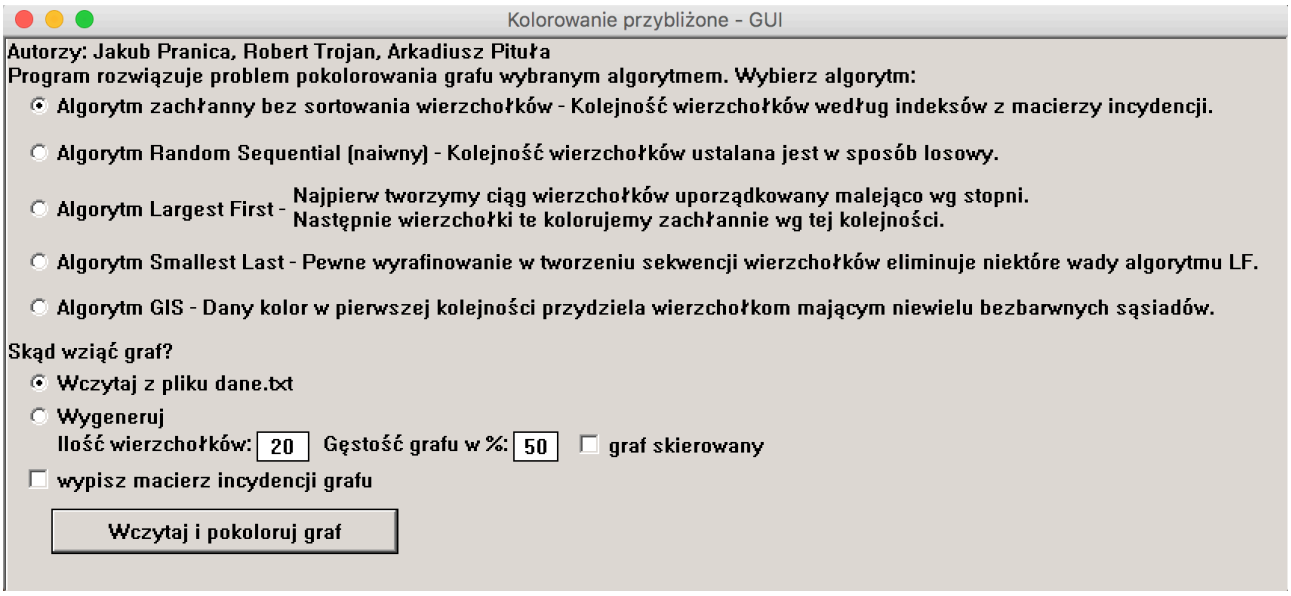
Optymalnym pokolorowaniem danego grafu nazywamy legalne pokolorowanie zawierające najmniejszą możliwą liczbę kolorów.

Liczbą chromatyczną grafu nazywamy liczbę równą najmniejszej możliwej liczbie kolorów potrzebnych do legalnego pokolorowania wierzchołków grafu.

Ze względu na bardzo szerokie zastosowania, kolorowanie grafów jest przedmiotem rozległych badań. Problem znalezienia optymalnego pokolorowania, a także znalezienia liczby chromatycznej jest NP-trudny. Z tego względu do kolorowania dużych grafów nadają się jedynie algorytmy przybliżone.

Algorytm zachłanny - algorytm, który w celu wyznaczenia rozwiązania w każdym kroku dokonuje zachłannego, tj. najlepiej rokującego w danym momencie wyboru rozwiązania częściowego. Innymi słowy algorytm zachłanny nie dokonuje oceny czy w kolejnych krokach jest sens wykonywać dane działanie, dokonuje decyzji lokalnie optymalnej, dokonuje on wyboru wydającego się w danej chwili najlepszym, kontynuując rozwiązanie podproblemu wynikającego z podjętej decyzji.

3. Opis programu :



Okno główne programu

Do wyboru w programie mamy pięć zaimplementowanych algorytmów kolorowania grafów **Algorytm zachłanny bez sortowania wierzchołków**, **Algorytm Random Sequential**, **Algorytm Largest First**, **Algorytm Smallest Last**, **Algorytm GIS**. Graf, który chcemy pokolorować możemy wczytać z pliku lub wygenerować. Generując graf mamy do wyboru ilość jego wierzchołków, jego gęstość oraz czy ma on być skierowany lub nieskierowany. W programie mamy również możliwość wypisania macierzy incydencji. Po wybraniu odpowiednich opcji możemy wczytać lub wygenerować (zależnie od wybranej opcji) i pokolorować graf za pomocą przycisku zatwierdzającego.

Poniżej zostaną krótko omówione wykorzystane w programie algorytmy :

1. Algorytm zachłanny bez sortowania wierzchołków - kolejność wierzchołków ustalana jest w na podstawie indeksów w macierzy incydencji.

2. Algorytm Random Sequential - jest to algorytm naiwny. Kolejność wierzchołków ustalana jest w sposób losowy.

3. Algorytm LF (largest first)
Kolorowanie grafu za pomocą algorytmu **LF** można opisać następująco:

1. Uporządkuj wierzchołki grafu malejąco według ich stopni (liczby krawędzi z nich wychodzących).
2. Koloruj wierzchołki zachłannie, zgodnie z ustaloną wcześniej kolejnością (zaczynając od wierzchołka o największym stopniu).

Algorytm **LF** jest algorytmem statycznym, gdyż raz ustalona kolejność wierzchołków nie zmienia się w trakcie jego działania. Najmniejszym dość trudnym grafem jest ścieżka P_6.

4. Algorytm SL (smallest last)
Algorytm **SL** wygląda następująco:

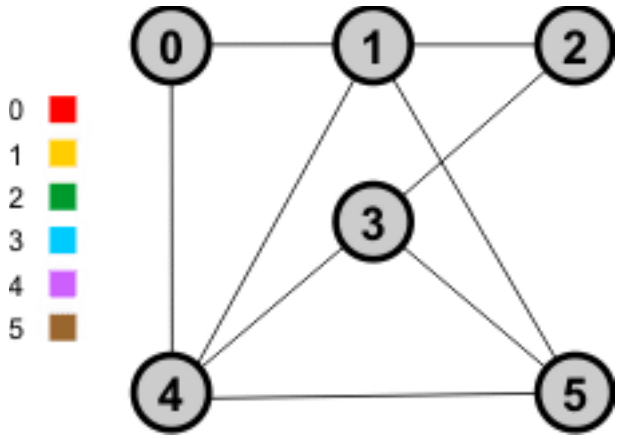
1. Znajdź wierzchołek o minimalnym stopniu i usuń go z grafu.
2. Powtarzaj krok pierwszy tak długo, aż graf będzie pusty (zapamiętaj kolejność usuwanych wierzchołków).
3. Koloruj wierzchołki zachłannie, zgodnie z ustaloną wcześniej kolejnością (zaczynając od wierzchołków usuniętych później).

Algorytm **SL** jest statyczny, jego złożoność wynosi $O(n + m)$, gdzie n - liczba wierzchołków, m - liczba krawędzi.

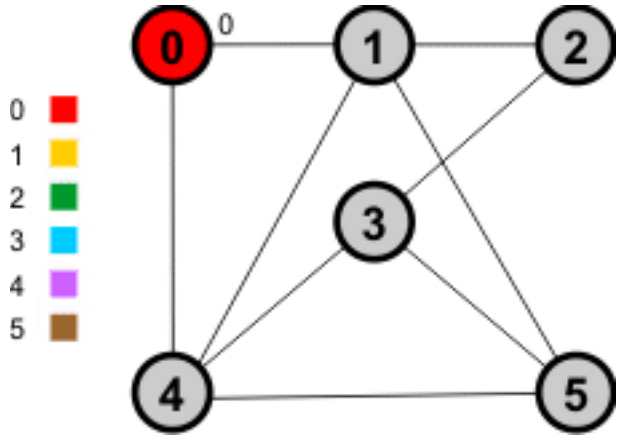
5. Algorytm GIS - Metoda ta została zaproponowana przez Johnsona. Jest ona realizacją algorytmu maksymalnych zbiorów niezależnych (stąd jej nazwa **Greedy Independent Sets**), w której wierzchołki grafu G rozważa się w pewnym porządku i wierzchołkowi v_i przypisuje się kolor c, jeżeli v_i nie jest sąsiedni z żadnym wierzchołkiem pomalowanym dotąd kolorem c. Po wyczerpaniu możliwości kolorowania kolorem c usuwamy pokolorowane wierzchołki z grafu i kolorujemy pozostały podgraf kolorem następnym. Podejście takie wynika z obserwacji, że kolor np. pierwszy należy w pierwszej kolejności przydzielać wierzchołkom mającym niewielu bezbarwnych sąsiadów (w ten sposób blokujemy możliwość użycia koloru dla jak najmniejszej liczby niepomalowanych dotąd wierzchołków).

Zobaczmy na przykładzie, jak działa jeden z algorytmów (Algorytm zachłanny bez sortowania wierzchołków) :

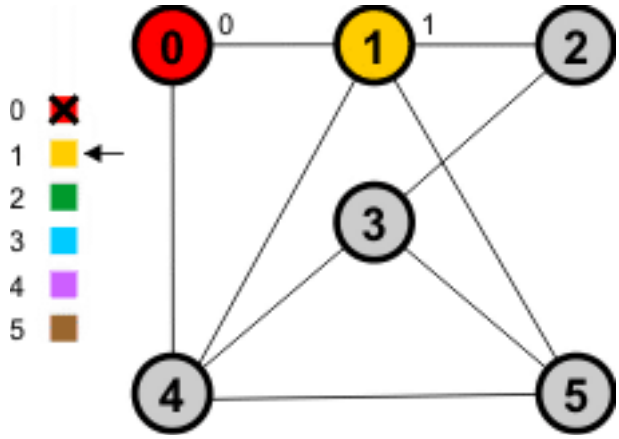
Kolory są reprezentowane przez kolejne liczby 0,1,2,...
Wybieramy w grafie pierwszy wierzchołek i kolorujemy go pierwszym kolorem nr 0.
Dla każdego z pozostałych wierzchołków grafu kolor ustalamy jako najniższy kolor, który nie jest użyty przez żadnego z jego sąsiadów.



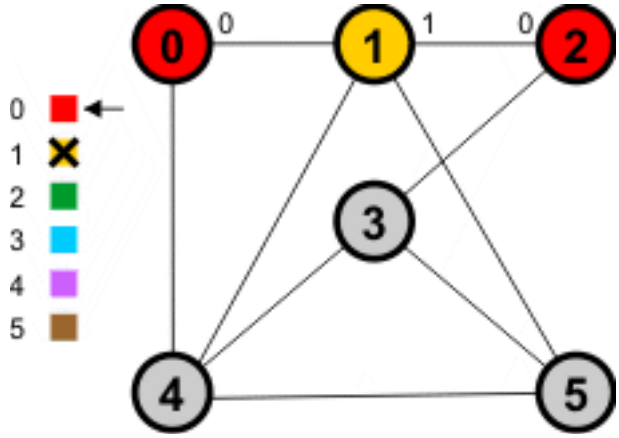
Oto graf do pokolorowania. Początkowo rezerwujemy tyle kolorów, ile graf posiada wierzchołków. Kolory numerujemy odpowiednio od 0 do 5.



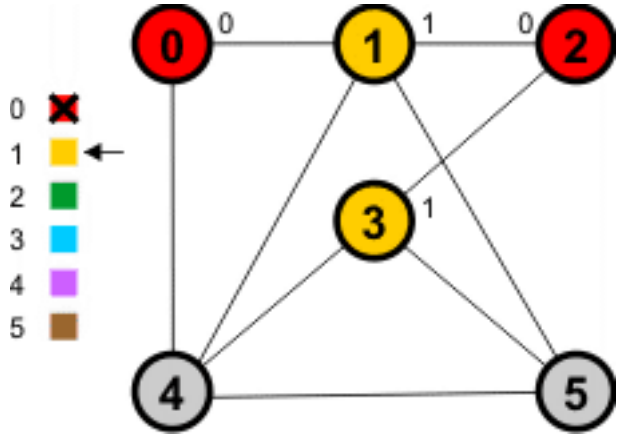
W grafie wybieramy wierzchołek nr 0 (może być to dowolny inny wierzchołek) i kolorujemy go pierwszym kolorem nr 0, czyli kolorem czerwonym.



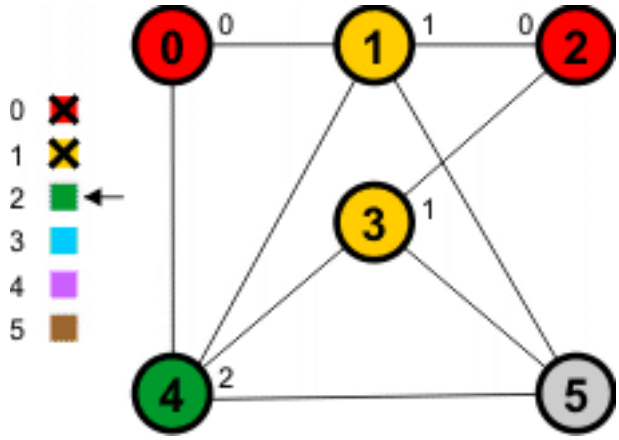
Wybieramy wierzchołek nr 1. Przeglądamy wszystkich jego sąsiadów (0, 2, 4, 5) i wykreślamy z tabeli kolory zużyte. W tym przypadku będzie to kolor czerwony nr 0 wierzchołka nr 0. Z pozostałych kolorów wybieramy kolor o najniższym numerze 1 i przypisujemy go wierzchołkowi nr 1.



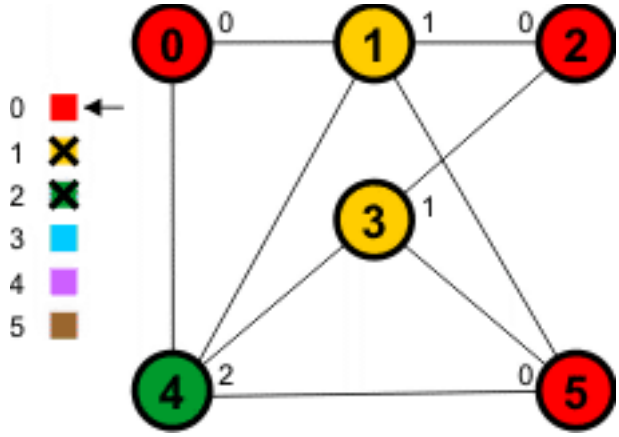
Wybieramy wierzchołek nr 2. Eliminujemy z listy kolory jego sąsiadów. Tutaj będzie to kolor nr 1 wierzchołka nr 1. Z pozostałych kolorów wybieramy najniższy, czyli kolor nr 0. Kolor ten przypisujemy wierzchołkowi nr 2.



Wybieramy wierzchołek nr 3. Na liście kolorów eliminujemy kolor czerwony nr 2, którym jest pokolorowany sąsiad nr 2. Z pozostałych na liście kolorów wybieramy kolor żółty nr 1 i kolorujemy nim wierzchołek nr 3.



Wybieramy wierzchołek nr 4. Z listy kolorów wykreślamy kolor czerwony nr 0 (sąsiad 0) oraz kolor żółty nr 1 (sąsiedzi nr 1 i 3). Z pozostałych kolorów wybieramy kolor zielony nr 2 (kolor o najniższym numerze) i nim kolorujemy wierzchołek nr 4.



Wybieramy ostatni wierzchołek nr 5. Jego sąsiedzi mają kolory nr 1 i 2, które wykreślamy z listy. Jako kolor wierzchołka nr 5 wybieramy kolor czerwony nr 0, ponieważ posiada on najniższy numer.

Graf jest pokolorowany. Zużyliśmy 3 kolory: czerwony (numer 0), żółty (numer 1) i zielony (numer 2).

4. Lista kroków algorytmów :

Algorytm zachłanny :

Lista kroków:

K01: Wypełnij tablicę *CT* wartościami -1 ; *-1 oznacza brak przypisanego koloru*
K02: *CT*[0] ← 0 ; *pierwszemu wierzchołkowi przypisujemy najniższy kolor.*
K03: **Dla** *v* = 1,2,...,*n* - 1 **wykonuj** K04...K08 ; *przechodzimy przez pozostałe wierzchołki grafu*
K04: Wypełnij tablicę *C* wartościami false ; *oznaczamy wszystkie kolory jako niezajęte*
K05 **Dla** każdego sąsiada *u* wierzchołka *v* **wykonuj:** ; *przeglądamy sąsiadów wierzchołka v*
Jeśli *CT*[*u*] > - 1, **to** *C*[*CT*[*u*]] ← true ; *kolor sąsiada oznaczamy jako zajęty*
K06: *i* = 0
K07: **Dopóki** *C*[*i*] = true, **wykonuj** *i* ← *i* + 1 ; *szukamy najniższego z dostępnych kolorów*
K08: *CT*[*v*] ← *i* ; *znaleziony kolor przypisujemy wierzchołkowi v*
K09: **Zakończ z wynikiem** *CT*

Algorytm LF :

Lista kroków:

K01: **Dla** *v* = 0,1,...,*n* - 1 **wykonuj** K02...K13 ; *przeglądamy kolejne wierzchołki grafu*
K02: *VT*[*v*] ← *v* ; *umieszczamy numer wierzchołka w tablicy VT*
K03: *DT*[*v*] ← 0 ; *zerujemy stopień wyjściowy wierzchołka v*
K04: **Dla** każdego sąsiada *u* wierzchołka *v* wykonuj:
DT[*v*] ← *DT*[*v*] + 1 ; *obliczamy stopień wyjściowy wierzchołka v*
K05: *d* ← *DT*[*v*] ; *sortujemy tablice VT i DT malejąco względem stopni wychodzących*
K06: *i* ← *v*
K07: **Dopóki** (*i* > 0) ∧ (*DT*[*i* - 1] < *d*) **wykonuj** K08...K10 ; *szukamy pozycji wierzchołka w posortowanych tablicach*
K08: *DT*[*i*] ← *DT*[*i* - 1] ; *przesuwamy elementy w DT, aby zrobić miejsce dla d*
K09: *VT*[*i*] ← *VT*[*i* - 1] ; *to samo w tablicy VT*
K10: *i* ← *i* - 1 ; *cofamy się o jedną pozycję*
K11: *DT*[*i*] ← *d* ; *element wstawiamy na właściwe miejsce w obu tablicach*
K12: *VT*[*i*] ← *v*
K13: Wypełnij tablicę *CT* wartościami -1 ; *-1 oznacza brak przypisanego koloru*
K14: *CT*[*VT*[0]] ← 0 ; *pierwszemu wierzchołkowi wg VT przypisujemy najniższy kolor.*
K15: **Dla** *v* = 1,2,...,*n* - 1 **wykonuj** K16...K20 ; *przechodzimy przez pozostałe wierzchołki grafu*
K16: Wypełnij tablicę *C* wartościami false ; *oznaczamy wszystkie kolory jako niezajęte*
K17 **Dla** każdego sąsiada *u* wierzchołka *VT*[*v*] **wykonuj:** ; *przeglądamy sąsiadów wierzchołka VT[v]*
Jeśli *CT*[*u*] > - 1, **to** *C*[*CT*[*u*]] ← true ; *kolor sąsiada oznaczamy jako zajęty*
K18: *i* = 0
K19: **Dopóki** *C*[*i*] = true, **wykonuj** *i* ← *i* + 1 ; *szukamy najniższego z dostępnych kolorów*
K20: *CT*[*VT*[*v*]] ← *i* ; *znaleziony kolor przypisujemy wierzchołkowi VT[v]*
K21: **Zakończ z wynikiem** *CT*

5. Porównanie algorytmów :

1. Dane wejściowe dla obu algorytmów :
Graf testowy „anna”

Wynik algorytmów :
Algorytm zachłanny (bez sortowania wierzchołków) : ilość użytych kolorów **12**
Algorytm RS : ilość użytych kolorów **11**
Algorytm LF : ilość użytych kolorów **11**
Algorytm SL : ilość użytych kolorów **11**
Algorytm GIS : ilość użytych kolorów **13**

2. Dane wejściowe dla obu algorytmów :
Wygenerowany graf o ilości wierzchołków **20**,
gęstość grafu w % **50**,
Graf jest nieskierowany

Wynik algorytmów :
Algorytm zachłanny (bez sortowania wierzchołków) : ilość użytych kolorów **7**
Algorytm RS : ilość użytych kolorów **7**
Algorytm LF : ilość użytych kolorów **6**
Algorytm SL : ilość użytych kolorów **6**
Algorytm GIS : ilość użytych kolorów **7**

3. Dane wejściowe dla obu algorytmów :
Wygenerowany graf o ilości wierzchołków : **888**,
gęstość grafu w % : **88**,
Graf jest skierowany

Wynik algorytmów :
Algorytm zachłanny (bez sortowania wierzchołków) : ilość użytych kolorów **270**
Algorytm RS : ilość użytych kolorów **271**
Algorytm LF : ilość użytych kolorów **267**
Algorytm SL : ilość użytych kolorów **264**
Algorytm GIS : ilość użytych kolorów **212**

6. Wnioski : Stworzony program był przez nas wielokrotnie testowany. Program ten spełnia wymagania zadania ponieważ po wprowadzeniu odpowiednich danych wejściowych za każdym razem dane wyjściowe zgadzały się z założeniami. W wielu testach (około *40*, w tym *3* opisane wyżej) porównywana była ilość użytych kolorów. Graf testowy „anna” można pokolorować przy użyciu conajmniej *11* kolorów. Z optymalnym pokolorowaniem go poradziły sobie algorytmy **RS** (co wynika raczej z pomyślnie wylosowanej kolejności), **LF** oraz **SL**. Przeprowadzone testy pozwalają stwierdzić, że najlepsze wyniki w kolorowaniu krawędzi dla małych grafów dają algorytmy **LF** i **SF**, natomiast do kolorowana dużych grafów najlepiej jest użyć algorytmu **GIS**. Wynika to z faktu, że w tych algorytmach korzystamy z pewnego rodzaju uporządkowania. Jest to niewielka zmiana jednak w przypadku dużych grafów może okazać się znacząca. Przy niewielkim skomplikowaniu grafów praktycznie niewidoczna jest różnica w czasie oczekiwania na wynik.

7. Literatura :

- *http://eduinf.waw.pl/inf/alg/001_search/0142.php#P2*
- *http://eduinf.waw.pl/inf/alg/001_search/0142.php#P3*
- *https://pl.wikipedia.org/wiki/Heurystyka_(informatyka)*
- *http://riad.pk.edu.pl/~zk/MP_HP.html*
- *https://pl.wikipedia.org/wiki/Kolorowanie_grafu*
- *http://users.uj.edu.pl/~ufkapano/algorytmy/lekcja14/color.html*
- *Symfonia C++, J.Grębosz, Oficyna Kallimach, Kraków 1999*
- *Język C++ bardziej efektywny, S.Meyers, WNT 1998*
- *Algorytmy + Struktury danych = Programy, N.Wirth, WNT 2001*
- *Wprowadzenie do algorytmów, T.H.Cormen, C.E.Leiserson, R.L.Rivest, WNT 1997,2004*
- *Optymalizacja dyskretna modele i metody kolorowania grafów, Marek Kubale*