

# Grafy i Sieci. Sprawozdanie 2.

SK11 Kolorowanie grafu za pomocą przeszukiwania z tabu.

Michał Aniserowicz, Jakub Turek

## 1 Temat projektu

SK11 Kolorowanie grafu za pomocą przeszukiwania z tabu.

## 2 Opis algorytmu

Zadaniem programu jest pokolorowanie wierzchołków zadanego grafu z użyciem jak najmniejszej liczby kolorów. Kolorowanie odbywa się z wykorzystaniem heurystycznego algorytmu przeszukiwania z tabu. Węzłem przestrzeni przeszukiwań jest pokolorowany (legalnie bądź nie) graf.

### 2.1 Sąsiedztwo

Sąsiadami w przestrzeni przeszukiwań są takie dwa pokolorowane grafy  $G$  i  $H$ , że graf  $H$  można osiągnąć poprzez zmianę koloru jednego z wierzchołków grafu  $G$ .

### 2.2 Funckja celu

Algorytm dąży do minimalizacji funkcji celu<sup>1</sup>:

$$f(G) = - \sum_{i=1}^k C_i^2 + \sum_{i=1}^k 2C_i E_i \quad (1)$$

gdzie:

- $G$  - graf, dla którego liczona jest funkcja celu,
- $k$  - liczba kolorów użytych do pokolorowania grafu  $G$ ,
- $C_i$  - liczba wierzchołków grafu  $G$  pokolorowanych na  $i$ -ty kolor,
- $E_i$  - liczba krawędzi grafu  $G$ , których oba końce pokolorowane są na  $i$ -ty kolor.

Definicję funkcji należy rozumieć następująco:

1. z jednej strony, faworyzowane są pokolorowania z użyciem jak najmniejszej liczby kolorów,
2. z drugiej strony, dyskryminowane są pokolorowania nielegalne.

---

<sup>1</sup>Definicja funkcji celu zaczerpnięta z: D. S. Johnson, C. R. Aragon, L. A. McGeoch, C. Schevon, Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, Operations Research, Vol. 39, No. 3, May-June 1991, pp. 378-406.

### 2.3 Lista tabu

Lista tabu zawiera ograniczoną liczbę ostatnich akcji podjętych przez algorytm. Pojedynczą akcją jest wybór wierzchołka, który zostanie pokolorowany na inny kolor.

## 3 Struktury danych

Podstawową strukturą danych użytą w programie jest graf, na który składa się zbiór wierzchołków. Pojedynczy wierzchołek zawiera:

- identyfikator wierzchołka (ciąg znaków),
- identyfikator przypisanego koloru (liczba całkowita),
- zbiór wskazań na sąsiadujące wierzchołki (kolekcja wskazań).

## 4 Projekty testów

## 5 Założenia programu

### 5.1 Złożoność obliczeniowa

Złożoność obliczeniową można oszacować wyrażeniem:

$$i * (nk * m + nk * n^2 + (nk * \log(nk))) \quad (2)$$

gdzie:

- $i$  - liczba iteracji,
- $n$  - liczba wierzchołków grafu,
- $m$  - rozmiar pamięci tabu.

Każda z  $i$  iteracji składa się z następujących kroków:

1. Wyznaczenie wszystkich możliwych sąsiadów aktualnego grafu, wraz ze sprawdzeniem dopuszczalności (lista tabu) -  $nk * m$ .
2. Obliczenie funkcji celu dla każdego dopuszczalnego sąsiada -  $nk * n^2$ . Obliczenie funkcji celu wymaga odwiedzenia wszystkich wierzchołków oraz wszystkich wierzchołków z nimi połączonych -  $n^2$ .
3. Sortowanie wyznaczonych sąsiadów ze względu na wartość funkcji celu -  $nk * \log(nk)$ .

Zatem algorytm ma złożoność wielomianową.

```
3
A,B
C,D
A,D
C,E
```

Rysunek 1: Przykładowe dane wejściowe.

## 5.2 Dane wejściowe

Na rysunku 1 przedstawiono przykładowe dane wejściowe. Dane wejściowe składają się z:

**Liczebności klas kolorów** specyfikowanej jako pierwszy parametr.

**Połączeń pomiędzy wierzchołkami** specyfikowanych w kolejnych wierszach jako para identyfikatorów wierzchołków oddzielonych przecinkami.

## 5.3 Dane wyjściowe

Dane wyjściowe zależą od wybranego trybu programu (opisane w sekcji 5.5).

**Tryb standardowy** Na wyjściu programu wypisywane są:

- Nazwa pliku z analizowanym grafem.
- Wynikowe przyporządkowanie *wierzchołków*  $\leftrightarrow$  *kolor*.
- Sprawdzenie legalności wynikowego przyporządkowania (tak / nie).

**Tryb „rozmowny”** Na wyjściu programu, poza informacjami z trybu standardowego, wypisywane są:

- Dane początkowe - nazwy klas kolorów, wylosowane przyporządkowanie, wartość funkcji celu dla wylosowanego przyporządkowania.
- Dane związane z każdą iteracją - numer iteracji, liczba iteracji bez zmiany wyniku, wartość funkcji celu dla najlepszej permutacji w iteracji, wartość funkcji celu dla najlepszego wyniku, zawartość pamięci krótkotrwałej dla danej iteracji, pokolorowanie (permutację) wybrane w danej iteracji.

## 5.4 Parametry algorytmu

**Wielkość pamięci** Rozmiar tablicy tabu jest wymaganym parametrem aplikacji. Rozmiar tablicy tabu jest specyfikowany opcją `-m`.

Przykład: `<nazwa_programu> -m 5` uruchamia algorytm z pięcioelementową tablicą tabu.

**Maksymalna liczba iteracji** Maksymalna liczba iteracji określa liczbę przejść algorytmu, po której aplikacja zakończy działanie (opisane w sekcji 5.6). Maksymalną liczbę iteracji specyfikuje się opcją `-i`.

Przykład: `<nazwa_programu> -i 500` uruchamia algorytm dla maksymalnie 500 iteracji.

**Maksymalna liczba iteracji bez zmiany rezultatu** Maksymalna liczba iteracji bez zmiany rezultatu określa liczbę przejść algorytmu, po której aplikacja wyłączy się, jeżeli wartość funkcji celu dla najlepszego dotychczas znalezionej pokolorowania nie zmieni się (opisane w sekcji 5.6). Maksymalną liczbę iteracji bez zmiany wyniku specyfikuje się opcją `-s`.

Przykład: `<nazwa_programu> -s 25` uruchamia algorytm dla maksymalnie 25 iteracji bez zmiany wyniku.

## 5.5 Opcje programu

Poza parametrami algorytmu, aplikacja udostępnia opisane poniżej opcje.

**Plik(i) wejściowe** W opcjach programu można wyspecyfikować jeden lub więcej plików wejściowych. Nazwę pliku wejściowego specyfikuje się bez dodatkowych opcji, zaraz po nazwie programu.

Przykład: `<nazwa_programu> graf1.txt graf2.txt` wykona algorytm dla grafów opisanych w plikach *graf1.txt* oraz *graf2.txt*.

**Plik wyjściowy** W opcjach programu można wyspecyfikować nazwę pliku, do którego zostanie zapisane wyjście programu. Nazwę pliku wyjściowego specyfikuje się opcją `-o`. Nazwa pliku wyjściowego jest parametrem opcjonalnym. Domyślnie wyjście przekierowywane jest na standardowy strumień (konsolę).

Przykład: `<nazwa_programu> -o wyjście1.txt` zapisze wyjście algorytmu do pliku *wyjście1.txt*.

**Tryb „rozmowny”** W opcjach programu można włączyć tryb „rozmowny” (*verbose*), który wyprowadza dodatkowe informacje diagnostyczne na wyjście w trakcie działania algorytmu. W trybie domyślnym na wyjście wyprowadzany jest tylko wynik działania algorytmu. Tryb „rozmowny” specyfikuje się opcją `-v`.

Przykład: `<nazwa_programu> -v` uruchamia aplikację w trybie „rozmownym”.

## 5.6 Kryteria stopu

**Maksymalna liczba iteracji** Wykonywanie programu zakończy się, gdy algorytm przekroczy maksymalną liczbę iteracji. Maksymalna liczba iteracji jest podana jako parametr aplikacji.

**Maksymalna liczba iteracji bez zmiany rezultatu** Wykonywanie programu zakończy się, gdy algorytm przekroczy maksymalną liczbę iteracji, w których nie zmieniła się wartość funkcji celu dla najlepszego pokolorowania. Maksymalna liczba iteracji bez zmiany wyniku jest podawana w parametrach aplikacji.

W obu przypadkach jako wynik działania programu zostanie podane najlepsze dotychczas znalezione pokolorowanie badanego grafu.

## 5.7 Sytuacje wyjątkowe

**Brak któregośkolwiek z wymaganych parametrów programu** W tym przypadku działanie programu kończy się niepowodzeniem, a na wyjście podawana jest informacja o poprawnym sposobie wywołania.

**Niepoprawny format pliku wejściowego** W tym przypadku działanie programu kończy się niepowodzeniem, a na wyjście podawana jest informacja o błędnym formacie pliku.

**Niespójny graf wejściowy** W tym przypadku działanie programu kończy się niepowodzeniem, a na wyjście podawana jest informacja o błędnej konstrukcji grafu.

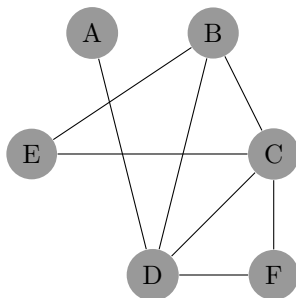
**Brak możliwości wykonania jakiejkolwiek akcji** Ta sytuacja może zdarzyć się w przypadku, gdy lista tabu ma zbyt duży rozmiar i zawiera wszystkie możliwe w danym kroku akcje (tzn. każda możliwa permutacja aktualnego grafu jest zabroniona). Aby umożliwić dalsze działanie, program usuwa najstarsze elementy listy, dopóki nie osiągnie stanu, w którym dozwolona będzie przynajmniej jedna z możliwych akcji.


## 6 Przykład działania

Poniżej przedstawiono przykładowy przebieg działania programu.

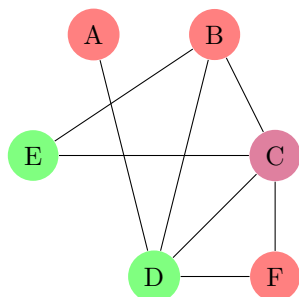
Dane wejściowe:

- Graf:



- Zestaw kolorów: 
- Wielkość pamięci: 3
- Maksymalna liczba iteracji: 100
- Maksymalna liczba iteracji bez zmiany rezultatu: 2

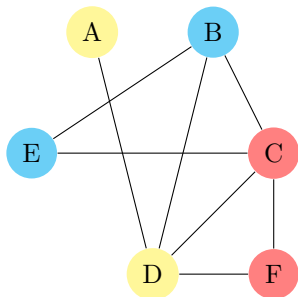
Przykładowy wynik:



## 6.1 Przygotowanie grafu

Przed rozpoczęciem właściwego działania algorytmu, program buduje graf i dokonuje jego losowego pokolorowania.

- Graf:



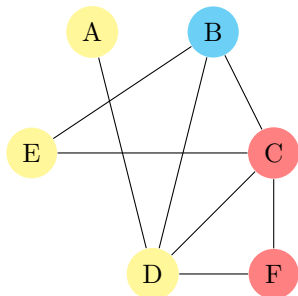
- Lista tabu: pusta


## 6.2 Przeszukiwanie

W każdym kroku algorytm sprawdza wszystkich sąsiadów aktualnego grafu. Jako aktualny graf wybiera tego sąsiada, dla którego wartość funkcji celu jest najmniejsza.



### 6.2.1 Krok 1

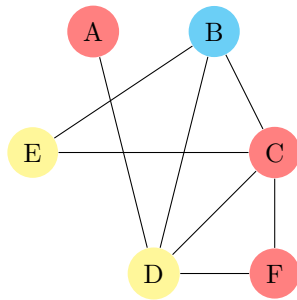
- Akcja:  $E \rightarrow E$
- Wartość funkcji celu:  $-4$
- Graf:





- Lista tabu: 



### 6.2.2 Krok 2

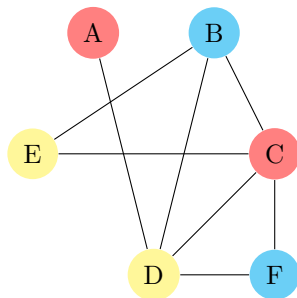
- Akcja:   $\rightarrow$  
- Wartość funkcji celu:  $-8$
- Graf:






- Lista tabu:  



### 6.2.3 Krok 3

- Akcja:   $\rightarrow$  
- Wartość funkcji celu:  $-12$
- Graf:

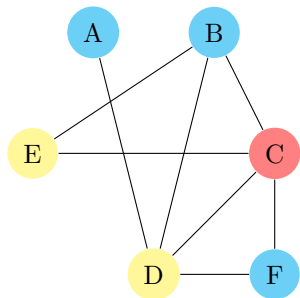


- Lista tabu:   

### 6.2.4 Krok 4

- Akcja:   $\rightarrow$  
- Wartość funkcji celu:  $-14$

- Graf:



- Lista tabu:



W tym momencie znalezione zostało optymalne pokolorowanie grafu. Program nie sprawdza jednak poprawności pokolorowania w każdym kroku, dlatego kontynuuje działanie.

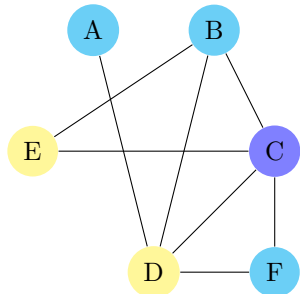
#### 6.2.5 Krok 5

- Akcja:



- Wartość funkcji celu:  $-14$

- Graf:



- Lista tabu:



W tym momencie osiągnięto maksymalną liczbę iteracji bez zmiany rezultatu. Program kończy działanie, jako wynik podając graf otrzymany w kroku 4 (jako że graf otrzymany w kroku 5 nie jest od niego lepszy).