

Analiza Algorytmów, Lista 5 Raport

Tomasz Krent

5 czerwca 2020

1 Zadanie 12

Program *zad12.py* dla każdej możliwej konfiguracji początkowej przechodzi do konfiguracji legalnej w momencie gdy wszystkie argumenty mają tę samą wartość. Program działa w taki sposób, że dla każdej początkowej konfiguracji dodaje odpowiednie konfiguracje od niej pochodzące do kolejki i zdejmuję je wykonując odpowiednie operacje do momentu, w którym kolejka będzie pusta. W momencie gdy ze zdjętej z kolejki konfiguracji można przejść do kolejnej możliwej poprzez wykonanie działania na określonym procesie konfiguracja ta jest dodawana na koniec kolejki. Wyniki działania programu dla określonego n , gdzie t oznacza czas działania na maszynie w sekundach, k ilość możliwych konfiguracji, tab w postaci tabeli pokazujący przykład najdłuższej konfiguracji w postaci zmieniających się rejestrów:

- $n = 2, t = 0.001, k = 9, tab = [[0, 1] \rightarrow [0, 0]]$
- $n = 3, t = 0.006, k = 120, tab = [[0, 2, 0] \rightarrow [1, 2, 0] \rightarrow [1, 2, 2] \rightarrow [1, 1, 2] \rightarrow [1, 1, 1]]$
- $n = 4, t = 0.305, k = 7535, tab = [[0, 2, 1, 0] \rightarrow [1, 2, 1, 0] \rightarrow [1, 2, 1, 1] \rightarrow [1, 2, 2, 1] \rightarrow [1, 1, 2, 1] \rightarrow [2, 1, 2, 1] \rightarrow [2, 1, 2, 2] \rightarrow [2, 1, 1, 2] \rightarrow [2, 2, 1, 2] \rightarrow [3, 2, 1, 2] \rightarrow [3, 2, 1, 1] \rightarrow [3, 2, 2, 1] \rightarrow [3, 2, 2, 2] \rightarrow [3, 3, 2, 2] \rightarrow [3, 3, 3, 2] \rightarrow [3, 3, 3, 3]]$
- $n = 5, t = 403.400, k = 7822266, tab = [[0, 3, 2, 1, 0] \rightarrow [1, 3, 2, 1, 0] \rightarrow [1, 3, 2, 1, 1] \rightarrow [1, 3, 2, 2, 1] \rightarrow [1, 3, 3, 2, 1] \rightarrow [1, 1, 3, 2, 1] \rightarrow [2, 1, 3, 2, 1] \rightarrow [2, 1, 3, 2, 2] \rightarrow [2, 1, 3, 3, 2] \rightarrow [2, 1, 1, 3, 2] \rightarrow [2, 2, 1, 3, 2] \rightarrow [3, 2, 1, 3, 2] \rightarrow [3, 2, 1, 3, 3] \rightarrow [3, 2, 1, 1, 3] \rightarrow [3, 2, 2, 1, 3] \rightarrow [3, 3, 2, 1, 3] \rightarrow [4, 3, 2, 1, 3] \rightarrow [4, 3, 2, 1, 1] \rightarrow [4, 3, 2, 2, 1] \rightarrow [4, 3, 2, 2, 2] \rightarrow [4, 3, 3, 2, 2] \rightarrow [4, 3, 3, 3, 2] \rightarrow [4, 3, 3, 3, 3] \rightarrow [4, 4, 3, 3, 3] \rightarrow [4, 4, 4, 3, 3] \rightarrow [4, 4, 4, 4, 3] \rightarrow [4, 4, 4, 4, 4]]$

2 Zadanie 13

1. Algorytm:

Algorithm 1 Maximal Independent

```
if  $s(i) = 0 \wedge (\forall j \in N(i))(s(j) = 0)$  then ▷ część R1
     $s(i) \leftarrow 1$ 
end if
if  $s(i) = 1 \wedge (\exists j \in N(i))(s(j) = 1)$  then ▷ część R2
     $s(i) \leftarrow 0$ 
end if
```

Legalna konfiguracja dla problemu maksymalnego niezależnego zbioru występuje gdy każdy węzeł jest niezależny ($s(i) = 1 \wedge (\forall j \in N(i))(s(j) = 0)$) lub zdominowany ($s(i) = 0 \wedge (\exists j \in N(i))(s(j) = 1)$).

W momencie gdy konfiguracja jest legalna zbiór $S = \{i | s(i) = 1\}$ jest maksymalnym zbiorem niezależnym. W tej legalnej konfiguracji nie występuje sytuacja, że dwa sąsiadujące ze sobą węzły są niezależne oraz nie występuje taki węzeł który ma wszystkich swoich sąsiadów zdominowanych a nie jest niezależny, co spełnia definicję maksymalnego zbioru niezależnego.

W czasie działania algorytmu każdy węzeł, który w danym momencie nie jest ani niezależny ani zdominowany, dzięki wykonaniu części *R1* algorytmu stanie się niezależny, a dzięki wykonaniu części *R2* stanie się zdominowany. Dodatkowo w momencie, gdy jakiś węzeł s stał się niezależny nie może zmienić już swojego stanu, ponieważ każdy z jego sąsiadów jest zdominowany przez niego i przez to nie mogą one również zmienić swojego stanu w tym przypadku na niezależny. Natomiast każdy węzeł który stał się zdominowany nie może ponownie zostać zdominowany, a może jedynie stać się niezależny w momencie, gdy wszyscy jego sąsiedzi zostaną w jakiś sposób zdominowani przez któregoś ze swoich sąsiadów. Algorytm więc przywraca legalną konfigurację w czasie mniejszym niż $2n$ zmian konfiguracji, ponieważ każdy węzeł maksymalnie dwa razy może zmienić swój stan względem stanu początkowego.

2. Wyniki w postaci zmieniających się konfiguracji grafów obrazujące przykładowe działanie algorytmów przy konkretnych konfiguracjach początkowych:



