Analiza algorytmów. Lista 4

Piotr Berezowski, 236749

4 czerwca 2020

1 Zadanie 12

1.1 Opis zadania

Zaimplementuj symulator algorytmu Mutual Exclusion Dijkstry. Dla ustalonego n oznaczającego liczbę procesów w pierścieniu, zweryfikuj, że startując z dowolnej konfiguracji początkowej algorytm przejdzie do legalnej konfiguracji. Jeśli z pewnej konfiguracji można przejść do kilku możliwych konfiguracji w zależności od tego, który proces wykona krok jako pierwszy, każde wykonanie powinno zostać zweryfikowane. Jaka jest największa liczba kroków do czasu osiągnięcia legalnej konfiguracji dla ustalonego n? Dla jakich wartości n możesz uzyskać odpowiedź w sensownym czasie? Za zadanie możesz otrzymać $3 \times N$ punktów, gdzie N oznacza największą wartość n, dla której uda Ci się zweryfikować algorytm.

1.2 Rozwiązanie

Implementacja zadania znajduje się w pliku *zad1.py*.

Największa wartość n dla jakiej algorytm udało się zweryfikować jest $n = 7$, gdzie ilość wszystkich możliwych konfiguracji jest równa 2097152. Największa liczba kroków do legalnej konfiguracji dla takiego n jest równa 57.

Poniżej w tabeli przedstawiono wyniki dla kolejnych wartości n, dla których udało się zweryfikować algorytm.

n	**Ilość wszystkich konfiguracji**	**Max liczba kroków**
1	2	0
2	9	1
3	64	4
4	625	15
5	7776	26
6	117649	40
7	2097152	57

2 Zadanie 13

2.1 Opis zadania

Rozważmy graf $G = (V, E)$. Dwa wierzchołki $v, w \in V$ nazywamy niezależnymi, jeśli $\{v, w\} \notin E$. Podzbiór $S \subseteq V$ wierzchołków nazywamy niezależnym, jeśli wszystkie jego elementy są parami niezależne. Wzorując się na algorytmie Maximal Matching podanym na wykładzie zaprojektuj, zaimplementuj i przetestuj samostabilizujący algorytm znajdujący maksymalny zbiór niezależny (ang. Maximal Independent Set) w nieskierowanym grafie spójnym. Podaj

przekonywujące uzasadnienie poprawności algorytmu (formalny dowód - zadanie na ćwiczenia). Algorytmy znajdowania maksymalnego zbioru niezależnego mają wiele zastosowań, możesz np. myśleć o problemie przydziału częstotliwości w sieciach bezprzewodowych.

2.2 Rozwiązanie

Implementacja zadania znajduje się w pliku *zad2.py*.

Każdy proces odpowiada pojedynczemu wierzchołkowi w grafie. Każdy proces kontroluje jeden rejestr $r_p \in \{0, 1\}$ który przechowuje informacje o tym, czy dany wierzchołek należy do zbioru niezależnego. $N(p)$ oznacza zbiór sąsiadów p. W każdym kroku algorytmu, dla wierzchołka p możemy znajdować się w jednej z następujących sytuacji:

1. Wszystkie wierzchołki $q \in N(p)$ spełniają $r_q = 0$.
2. Przynajmniej jeden wierzchołek $q \in N(p)$ spełnia $r_q = 1$.

W przypadku sytuacji 1, jeśli $r_p = 0$, to ustawiamy $r_p \leftarrow 1$, w przeciwnym przypadku nie robimy nic.

W przypadku sytuacji 2, jeśli $r_p = 1$, to ustawiamy $r_p \leftarrow 0$, w przeciwnym przypadku nie robimy nic.

Po ustabilizowaniu się algorytmu (wyznaczeniu maksymalnego zbioru niezależnego) żaden jego krok nie zmieni obecnej konfiguracji. Wszyscy sąsiedzi wierzchołków oetykietowanych numerem 1 będą miały ustawioną wartość rejestru na 0 (sytuacja 2), a wszyscy sąsiedzi wierzchołków oetykietowanych numerem 0 będą miały tylko jednego sąsiada z wartością rejestru równą 1. Widzimy, że w tym przypadku wierzchołki, które zostały oetykietowane numerem 1 tworzą pewien maksymalny zbiór niezależny.

Poniżej przedstawiono pseudokod pętli wykonywanej przez każdy z procesów.

1: Każdy proces p wykonuje pętle

```
1 while True do
2     if r_p = 0 ∧ (∀_{q∈N(p)})(r_q = 0) then
3         r_p ← 1;
4     if r_p = 1 ∧ (∃_{q∈N(p)})(r_q = 1) then
5         r_p ← 0;
```