

Algorytmy Równoległe i Rozproszone

Część IX - Rozproszone algorytmy probabilistyczne.

Łukasz Kuszner
pokój 209, WETI

<http://www.sphere.pl/~kuszner/>
kuszner@sphere.pl

Oficjalna strona wykładu

<http://www.sphere.pl/~kuszner/ARiR/>

2005/06

Algorytmy losowe

Rzut monetą

Złożoność

Przykład

Przykład II

Strona główna

Strona tytułowa

◀◀

▶▶

◀

▶

Strona 1 z 22

Powrót

Full Screen

Zamknij

Koniec

1. Rozproszone algorytmy probabilistyczne.

Prezentowane do tej pory algorytmy nie zawierały instrukcji generujących liczby losowe lub pseudolosowe, dlatego nazywamy je deterministycznymi.

Rozproszonych algorytmów niedeterministycznych można używać do łamania symetrii. Istnieje również nieudowodniona hipoteza mówiąca, że korzystając z elementów losowości można konstruować efektywne algorytmy prościej.

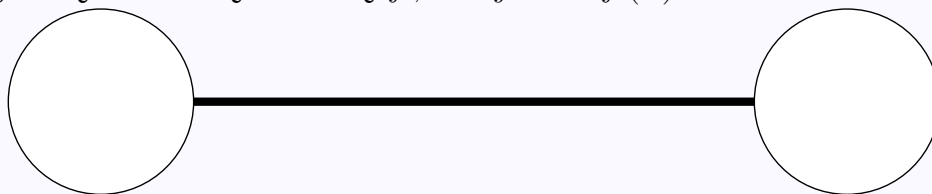
[Strona główna](#)[Strona tytułowa](#)[◀](#)[▶](#)[◀](#)[▶](#)

Strona 2 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

2. Przykład

Rozpatrzmy graf jak na rysunku, w którym każdy wierzchołek v ma jedną zmienną lokalną f , inicjalnie $f(v) = 0$.



Zmienna f może przyjmować dwie wartości 0 i 1. Celem jest doprowadzenie do sytuacji, w której oba wierzchołki mają różne wartości.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)

Strona 3 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Algorytm 1: color

R1: **if** $\exists_{u \in N(v)} f(u) = f(v)$
then $f(v) = \text{random}(2)$

Rozpatrzmy działanie tego algorytmu w modelu synchronicznym. Jeśli mamy „pecha”, to wykonanie tego algorytmu może trwać dowolnie długo.

[Strona główna](#)[Strona tytułowa](#)[◀◀](#)[▶▶](#)[◀](#)[▶](#)[Strona 4 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

3. Złożoność algorytmów losowych

Nie można więc brać miary złożoności algorytmu jako czasu działania w najgorszym przypadku.

Zwykle przyjmujemy następującą definicję:

Definicja 1 Niech \mathcal{A} będzie algorytmem probabilistycznym, a n rozmiarem problemu. Mówimy, że \mathcal{A} jest $O(f(n))$, jeśli istnieje stała $c > 0$ i $q \in (0..1)$ oraz prawdopodobieństwo, że czas działania \mathcal{A} przekroczy $cf(n) + t$ jest mniejsze niż q^t , dokładniej:

$$\Pr[T_{\mathcal{A}}(n) > cf(n) + t] \leq q^t. \quad (1)$$

[Strona główna](#)[Strona tytułowa](#)[◀◀](#)[▶▶](#)[◀](#)[▶](#)[Strona 5 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Lub w trochę słabszej wersji

Definicja 2 *Mówimy, że \mathcal{A} jest $O(f(n))$, jeśli istnieje stała $c > 0$ i $q(n) \in (0..1)$ takie, że $1/(1 - q(n))$ jest $O(f(n))$ oraz prawdopodobieństwo, że czas działania \mathcal{A} przekroczy $cf(n) + t$ jest mniejsze niż $q(n)^t$, dokładniej:*

$$\Pr[T_{\mathcal{A}}(n) > cf(n) + t] \leq q(n)^t. \quad (2)$$

Zauważmy, że jeśli \mathcal{A} jest $O(F(n))$, to również wartość oczekiwana $\mathbb{E}[T_{\mathcal{A}}(n)]$ jest $O(F(n))$.

[Strona główna](#)[Strona tytułowa](#)[◀◀](#)[▶▶](#)[◀](#)[▶](#)[Strona 6 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Niech

$$T(n) = a(n) + T(h(n)), \quad (3)$$

gdzie:

$$a(n) \geq 0 \quad (4)$$

$$h(n) \in [0, n] \quad (5)$$

$$0 \leq \mathbb{E}[h(n)] \leq m(n) \leq n \quad (6)$$

$$m(n), m(n)/n \text{ są niemalejące.} \quad (7)$$

Niech $u(n)$ będzie nieujemnym rozwiązaniem

$$x(n) = a(n) + x(m(n)) \quad (8)$$

[Strona główna](#)
[Strona tytułowa](#)
[Strona 7 z 22](#)
[Powrót](#)
[Full Screen](#)
[Zamknij](#)
[Koniec](#)

W zależności od $a(n)$ mamy dwie nierówności (Karp 1994).

Twierdzenie 1 *Jeśli $a(n) = 0$ dla $n < d$ i $a(n) = 1$ dla $n \geq d$ i $c_t = \min\{x | u(x) \geq t\}$, to*

$$\Pr[T(n) > u(n) + n] \leq \left(\frac{m(n)}{n}\right)^{n-1} \frac{m(n)}{c_u(n)} \quad (9)$$

Twierdzenie 2 *Jeśli $a(n)$ jest ściśle rosnąca, to*

$$\Pr[T(n) > u(n) + na(n)] \leq \left(\frac{m(n)}{n}\right)^n \quad (10)$$

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 8 z 22

Powrót

Full Screen

Zamknij

Koniec

4. Przykład – kolorowanie grafów

Każdy wierzchołek u przechowuje paletę barw nie przypisanych jeszcze przez żadnego z sąsiadów — inicjalnie rozmiaru $\deg(u)+1$. Każdy wierzchołek losuje kolor ze swojej palety i wysyła komunikat do swoich sąsiadów. Jeśli żaden z nich nie wybrał tej barwy, u jest pokolorowany i informuje o tym swoich sąsiadów. Jeśli jakiś sąsiad wylosował taki sam kolor, u nie zatrzymuje go. Na początku następnej rundy u usuwa ze swojej palety kolory użyte przez jego sąsiadów.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 9 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Johansson (1992) wykazał, że w tym algorytmie każdy wierzchołek w każdej rundzie otrzyma kolor z prawdopodobieństwem $p \geq 1/4$.

Wstawiając $a(n) = 1$, $m(n) = 3n/4$ otrzymamy:

$$\Pr\left\{T(n) \geq \left\lfloor \log_{4/3} n \right\rfloor + t + 1\right\} \leq \left(\frac{3}{4}\right)^{t-1} \quad (11)$$

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 10 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

5. Przykład – kolorowanie grafów II

Algorytm DLF

Każdy wierzchołek v posiada trzy parametry:

- swój stopień: $d(v)$,
- losowy parametr: $rv(v)$,
- inicjalnie pustą paletę zabronionych kolorów, które zostały już użyte przez jego sąsiadów: $uc(v)$.

Parametry $d(v)$ i $rv(v)$ decydują o kolejności kolorowania.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)

Strona 11 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Priorytet

Niech v_1, v_2 będą wierzchołkami grafu. Mówimy, że v_1 ma *wyższy priorytet* niż v_2 , jeśli:

$$d(v_1) > d(v_2)$$

lub

$$(d(v_1) = d(v_2)) \quad \wedge \quad (rv(v_1) > rv(v_2))$$

[Strona główna](#)[Strona tytułowa](#)[◀◀](#)[▶▶](#)[◀](#)[▶](#)

Strona 12 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

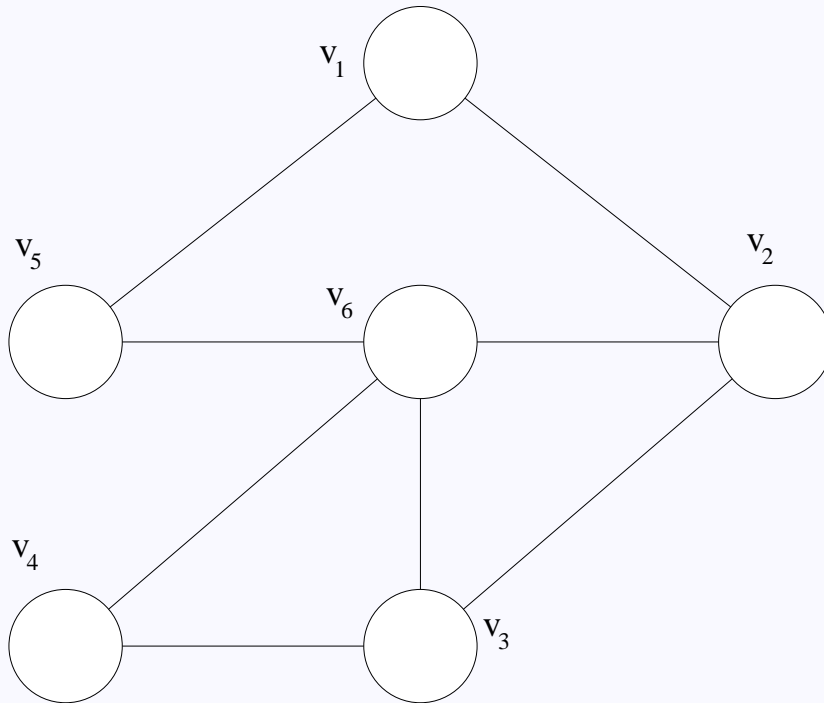
Algorytm 2: DLF

1. Wylosuj parametr $rv(v)$ z rozkładem równomiernym na przedziale $[0, 1]$.
2. Wyślij wszystkim sąsiadom swoje parametry: stopień $d(v)$, losową liczbę $rv(v)$ oraz kolor c – pierwszy, którego nie ma na liście $uc(v)$.
3. Odbierz wiadomości od sąsiadów o ich parametrach.
4. Jeśli c nie koliduje z kolorami jego sąsiadów lub v ma najwyższy priorytet spośród tych, które z nim kolidują przydziel sobie kolor c , poinformuj o tym sąsiadów i zakończ.
5. Jeśli nie: odbierz informację o przydzielonych kolorach od sąsiadów i zaktualizuj listę $uc(v)$.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)

Strona 13 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)



Sekwencja kolorów zielony, czerwony, żółty,...

Algoritmy losowe

Rzut monetą

Złożoność

Przykład

Przykład II

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

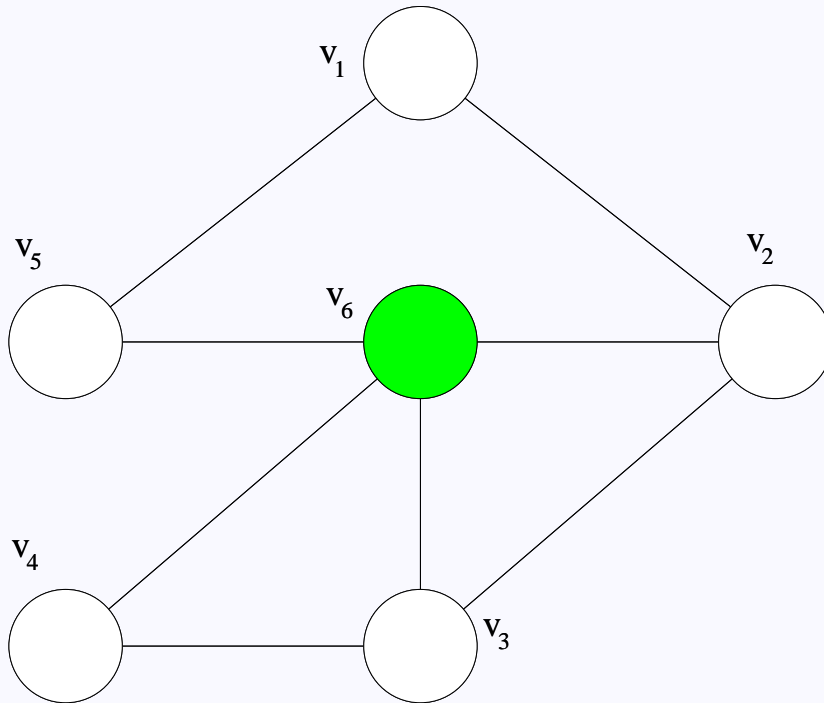
Strona 14 z 22

Powrót

Full Screen

Zamknij

Koniec



Sekwencja kolorów zielony, czerwony, żółty,...

Algoritmy losowe

Rzut monetą

Złożoność

Przykład

Przykład II

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

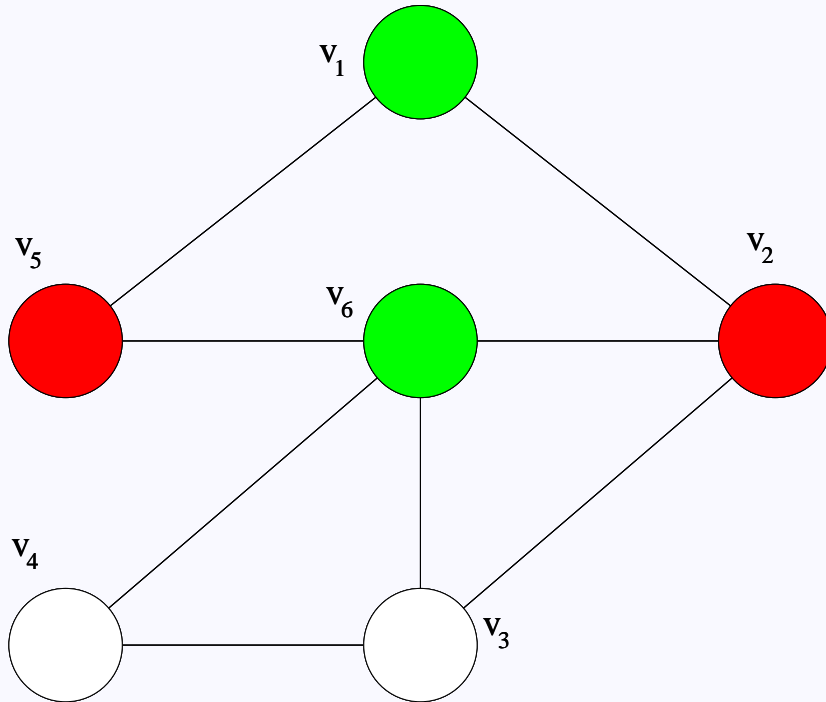
Strona 14 z 22

Powrót

Full Screen

Zamknij

Koniec



Sekwencja kolorów zielony, czerwony, żółty,...

Algoritmy losowe

Rzut monetą

Złożoność

Przykład

Przykład II

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

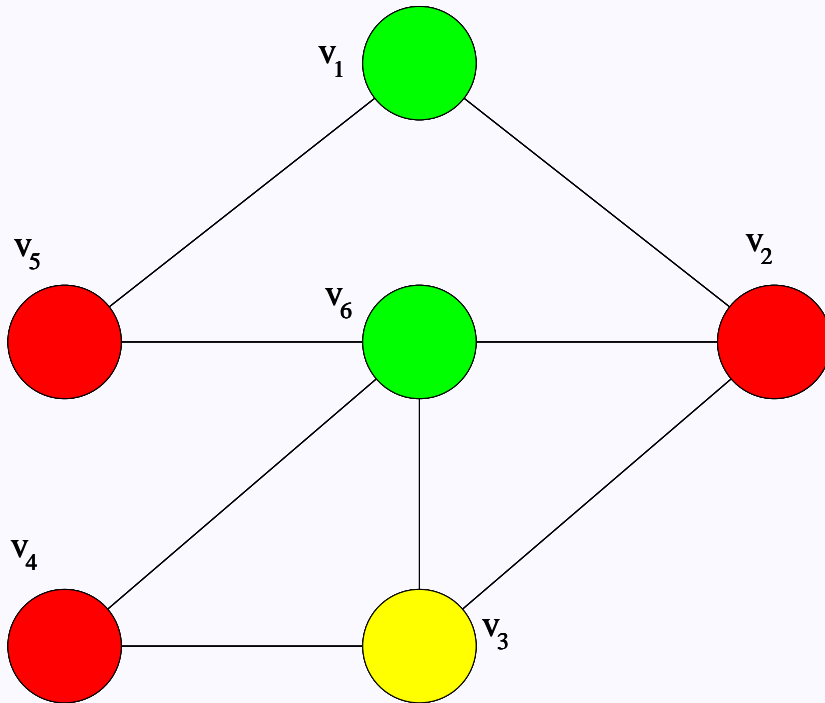
Strona 14 z 22

Powrót

Full Screen

Zamknij

Koniec



Sekwencja kolorów zielony, czerwony, żółty,...

Algoritmy losowe

Rzut monetą

Złożoność

Przykład

Przykład II

Strona główna

Strona tytułowa



Strona 14 z 22

Powrót

Full Screen

Zamknij

Koniec

Twierdzenie 3 Dla dowolnego n -wierzchołkowego grafu G algorytm DLF działa w czasie $O(\Delta^2 \log n)$ rund.

Niech G będzie grafem n -wierzchołkowym. Podzielimy cały proces kolorowania G na fazy i oszacujemy liczbę rund w każdej fazie procesu. Pierwsza faza zaczyna się wraz z początkiem działania algorytmu i kończy w rundzie, w której wszystkie wierzchołki o stopniu Δ są pokolorowane. Druga faza zaczyna się w kolejnej rundzie i kończy, gdy pokolorowane zostaną wszystkie wierzchołki o stopniu $\Delta - 1$, itd.

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Strona 15 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Dla $i = 1, \dots, \Delta$ niech V_i oznacza zbiór wierzchołków w G o stopniu i a $n_i = |V_i|$. Dla każdego $i = 1, \dots, \Delta$ zdefiniujemy zmienne losowe T_i równe numerowi pierwszej rundy, w której wszystkie wierzchołki o stopniach co najmniej i są pokolorowane, czyli T_i jest czasem zakończenia $(\Delta + 1 - i)$ -tej fazy procesu kolorowania. Dla uproszczenia notacji niech $T_{\Delta+1} = 0$.

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Strona 16 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Niech teraz k będzie dodatnią liczbą całkowitą taką, że $1 \leq k \leq \Delta$. Oszacujemy $\Pr[T_k > s_2 \mid T_{k+1} \leq s_1]$, dla pewnych liczb całkowitych $s_1 < s_2$, to jest prawdopodobieństwo, że faza o numerze $(\Delta + 1 - k)$ skończy się po rundzie o numerze s_2 pod warunkiem, że poprzednia faza zakończyła się do rundy o numerze s_1 .

[Strona główna](#)[Strona tytułowa](#)[Strona 17 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Weźmy pod uwagę dowolny wierzchołek v o stopniu k w czasie rundy l , $s_1 < l \leq s_2$. Ponieważ $T_{k+1} \leq s_1$, więc v nie ma niepokolorowanych sąsiadów o wyższym stopniu niż k . Co więcej, ma też co najwyżej k sąsiadów o stopniu k żądających tego samego koloru co v , więc prawdopodobieństwo, że v zostanie pokolorowany w trakcie rundy l jest nie mniejsze niż $1/(k+1)$. Jako że liczba niepokolorowanych wierzchołków stopnia k jest nie większa niż n_k , Rozumując jak dla algorytmu trywialnego otrzymujemy poniższe oszacowanie:

$$\Pr[T_k > s_2 \mid T_{k+1} \leq s_1] \leq \left(\frac{k}{k+1}\right)^{t-1}, \quad (12)$$

gdzie $s_2 = s_1 + \lfloor \log_{(k+1)/k} n_k \rfloor + 1 + t$ a liczba całkowita $t \geq 1$.

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 18 z 22

Powrót

Full Screen

Zamknij

Koniec

Teraz niech $t_1 > \dots > t_\Delta > t_{\Delta+1} = 0$ będzie ciągiem liczb całkowitych. Używając nierówności Bayesa otrzymujemy:

$$\Pr[T_k \leq t_k] \geq \Pr[T_k \leq t_k \mid T_{k+1} \leq t_{k+1}] \cdot \Pr[T_{k+1} \leq t_{k+1}], \quad (13)$$

dla $k = 1, \dots, \Delta$. Iterując (13) i zauważając, że $T_{\text{DLF}}(G) = T_1$ mamy:

$$\Pr[T_{\text{DLF}}(G) \leq t_1] \geq \prod_{k=1}^{\Delta} \Pr[T_k \leq t_k \mid T_{k+1} \leq t_{k+1}]. \quad (14)$$

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 19 z 22

Powrót

Full Screen

Zamknij

Koniec

Teraz ustalmy $t \geq 1$ i dla $k = 1, \dots, \Delta$ niech

$$t_k = \sum_{i=k}^{\Delta} \left(\lfloor \log_{1+1/i} n_i \rfloor + t + 1 \right).$$

Z nierówności (12) i (14) mamy, że

$$\begin{aligned} \Pr[T_{\text{DLF}}(G) \leq t_1] &\geq \prod_{k=1}^{\Delta} \left(1 - \left(\frac{1}{1+1/k} \right)^{t-1} \right) \geq \\ &\left(1 - \left(\frac{1}{1+1/\Delta} \right)^{t-1} \right)^{\Delta} \geq 1 - \Delta \left(\frac{1}{1+1/\Delta} \right)^{t-1}. \end{aligned} \quad (15)$$

Ostatnie szacowanie wynika z nierówności Bernoulliego.

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 20 z 22

Powrót

Full Screen

Zamknij

Koniec

Oczywista nierówność $n_i \leq n$ implikuje, że $t_1 \leq \Delta \log_{1+1/\Delta}(n) + (t+1)\Delta$. Więc z (15) mamy dla każdego rzeczywistego $t \geq 1$,

$$\Pr[T_{\text{DLF}}(G) > \Delta \log_{1+1/\Delta}(n) + (t+1)\Delta] \leq \Delta \left(\frac{1}{1+1/\Delta} \right)^{t-2}. \quad (16)$$

Podstawiając $u = t\Delta$ do (16) otrzymujemy

$$\Pr[T_{\text{DLF}}(G) > \Delta(\log(n)/\log(1+1/\Delta) + 1) + u] \leq 2(\Delta+1) \left(\frac{\Delta}{1+\Delta} \right)^{u/\Delta}. \quad (17)$$

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 21 z 22

Powrót

Full Screen

Zamknij

Koniec

Teraz dla $s = u + \log_q(2(\Delta + 1))$, gdzie $q = (\Delta/(\Delta + 1))^{1/\Delta}$, otrzymujemy nierówność

$$\Pr[T_{\text{DLF}}(G) > \Delta(\log(2(\Delta + 1)n)/\log(1 + 1/\Delta) + 1) + s] \leq q^s, \quad (18)$$

Z nierówności (18) wynika, że algorytm działa w czasie $O(\Delta^2 \log n)$.

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Strona 22 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)