$RL \subseteq SC$

13 grudnia 2004 roku

Plan

Znane wyniki

- Osiągalność w grafach skierowanych:
  - liniowa przestrzeń i czas
  - przestrzeń $O(\log^2 n)$, ale czas nie jest wielomianowy (Savitch)
- Osiągalność w grafach nieskierowanych:
  - przestrzeń n^ϵ i wielomianowy czas
  - logarytmiczna przestrzeń i wielomianowy czas, ale zrandomizowany

Znane wyniki

- Osiągalność w grafach skierowanych:
  - liniowa przestrzeń i czas
  - przestrzeń $O(\log^2 n)$, ale czas nie jest wielomianowy (Savitch)
- Osiągalność w grafach nieskierowanych:
  - przestrzeń n^ϵ i wielomianowy czas
  - logarytmiczna przestrzeń i wielomianowy czas, ale zrandomizowany

Znane wyniki

- Osiągalność w grafach skierowanych:
  - liniowa przestrzeń i czas
  - przestrzeń $O(\log^2 n)$, ale czas nie jest wielomianowy (Savitch)
- Osiągalność w grafach nieskierowanych:
  - przestrzeń n^ϵ i wielomianowy czas
  - logarytmiczna przestrzeń i wielomianowy czas, ale zrandomizowany

Znane wyniki

- Osiągalność w grafach skierowanych:
  - liniowa przestrzeń i czas
  - przestrzeń $O(\log^2 n)$, ale czas nie jest wielomianowy (Savitch)
- Osiągalność w grafach nieskierowanych:
  - przestrzeń n^ϵ i wielomianowy czas
  - logarytmiczna przestrzeń i wielomianowy czas, ale zrandomizowany

Znane wyniki

- Osiągalność w grafach skierowanych:
  - liniowa przestrzeń i czas
  - przestrzeń $O(\log^2 n)$, ale czas nie jest wielomianowy (Savitch)
- Osiągalność w grafach nieskierowanych:
  - przestrzeń n^ϵ i wielomianowy czas
  - logarytmiczna przestrzeń i wielomianowy czas, ale zrandomizowany

Znane wyniki

- Osiągalność w grafach skierowanych:
  - liniowa przestrzeń i czas
  - przestrzeń $O(\log^2 n)$, ale czas nie jest wielomianowy (Savitch)
- Osiągalność w grafach nieskierowanych:
  - przestrzeń n^ϵ i wielomianowy czas
  - logarytmiczna przestrzeń i wielomianowy czas, ale zrandomizowany

Co chcemy pokazać

Twierdzenie (OSIĄGALNOŚĆ $\in$ SC)

Istnieje deterministyczny algorytm rozwiązujący problem osiągalności w grafach nieskierowanych w czasie wielomianowym i w przestrzeni $O(\log^2 n)$.

Twierdzenie (BPL $\subseteq$ SC)

$$BPL \subseteq SC.$$

a dokładniej dla wszystkich konstruowalnych $S(n) = \Omega(\log n)$

$$BPSPACE(S(n)) \subseteq DTISP(2^{O(S(n))}, S^2(n)).$$

Co chcemy pokazać

Twierdzenie (OSIĄGALNOŚĆ $\in$ SC)

Istnieje deterministyczny algorytm rozwiązujący problem osiągalności w grafach nieskierowanych w czasie wielomianowym i w przestrzeni $O(log^2 n)$.

Twierdzenie (BPL $\subseteq$ SC)

$$BPL \subseteq SC.$$

a dokładniej dla wszystkich konstruowalnych $S(n) = \Omega(logn)$

$$BPSPACE(S(n)) \subseteq DTISP(2^{O(S(n))}, S^2(n)).$$

Co chcemy pokazać

Twierdzenie (OSIĄGALNOŚĆ $\in$ SC)

Istnieje deterministyczny algorytm rozwiązujący problem osiągalności w grafach nieskierowanych w czasie wielomianowym i w przestrzeni $O(log^2 n)$.

Twierdzenie (BPL $\subseteq$ SC)

$$BPL \subseteq SC.$$

a dokładniej dla wszystkich konstruowalnych $S(n) = \Omega(logn)$

$$BPSPACE(S(n)) \subseteq DTISP(2^{O(S(n))}, S^2(n)).$$

Co chcemy pokazać

Twierdzenie (APROKS. OSIĄGALNOŚĆ)

Istnieje deterministyczny algorytm rozwiązujący następujący problem:
Wejście:*Macierz prawdopodobieństw sąsiedztwa M rozmiaru n na n, liczba całkowita t i ułamek ϵ (t i ϵ są podane w postaci unarnej)*
Wyjście:*Macierz A taka że $\|A - M^t\| \leqslant \epsilon$.*
Algorytm działa w czasie poly(N) i przestrzeni $O(log^2 N)$, gdzie $N = n^2 + t + \epsilon^{-1}$ jest rozmiarem wejścia.

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Pomysły

- Symulacja maszyny BPL przez automat skończony
  - Określone wejście
  - Prawdopodobieństwo osiągnięcia stanu akceptującego
- Uniwersalne funkcje haszujące
- Standardowo określona norma macierzy
- Generatory pseudolosowe
  - Opierają się na uniwersalnych funkcjach haszujących
  - Zdefiniowane rekurencyjnie — łatwe wyznaczanie wyniku porcjami
  - Dobra aproksymacja osiągalności w DFA

Notacja

- Q — automat skończony o n stanach, i, j — stany Q, $x \in \{0,1\}^*$, D — rozkład prawdopodobieństwa na $\{0,1\}^*$, t — liczba całkowita
  - x przechodzi z i do j w Q jeśli Q zaczynając od stanu i na słowie x dojdzie do stanu j
  - $Q(D)$ — macierz n na n taka, że

$$Q(D)[i,j] = Pr_{x \in D}[x \text{ przechodzi z } i \text{ do } j \text{ w } Q]$$

  - U_t — rozkład jednostajny na $\{0,1\}^t$

Notacja

- Q — automat skończony o n stanach, i, j — stany Q, $x \in \{0,1\}^*$, D — rozkład prawdopodobieństwa na $\{0,1\}^*$, t — liczba całkowita
  - x przechodzi z i do j w Q jeśli Q zaczynając od stanu i na słowie x dojdzie do stanu j
  - $Q(D)$ — macierz n na n taka, że

$$Q(D)[i,j] = Pr_{x \in D}[x \text{ przechodzi z } i \text{ do } j \text{ w } Q]$$

  - U_t — rozkład jednostajny na $\{0,1\}^t$

Notacja

- Q — automat skończony o n stanach, i, j — stany Q, $x \in \{0,1\}^*$, D — rozkład prawdopodobieństwa na $\{0,1\}^*$, t — liczba całkowita
  - x przechodzi z i do j w Q jeśli Q zaczynając od stanu i na słowie x dojdzie do stanu j
  - $Q(D)$ — macierz n na n taka, że

$$Q(D)[i,j] = Pr_{x \in D}[x \text{ przechodzi z } i \text{ do } j \text{ w } Q]$$

  - U_t — rozkład jednostajny na $\{0,1\}^t$

Notacja

- Q — automat skończony o n stanach, i, j — stany Q, $x \in \{0,1\}^*$, D — rozkład prawdopodobieństwa na $\{0,1\}^*$, t — liczba całkowita
  - x przechodzi z i do j w Q jeśli Q zaczynając od stanu i na słowie x dojdzie do stanu j
  - $Q(D)$ — macierz n na n taka, że

$$Q(D)[i,j] = Pr_{x \in D}[x \text{ przechodzi z } i \text{ do } j \text{ w } Q]$$

  - U_t — rozkład jednostajny na $\{0,1\}^t$

Notacja

- m, k — liczby całkowite $\geqslant 0$, H — rodzina uniwersalnych funkcji haszujących $h : \{0,1\}^m \to \{0,1\}^m$, $\circ$ — operacja konkatenacji
  - $G_0(x) = x$
  - $G_k(x, h_1, \ldots, h_k) = G_{k-1}(x, h_1, \ldots, h_{k-1}) \circ G_{k-1}(h_k(x), h_1, \ldots h_{k-1})$
- $h_1, \ldots, h_k \in H$, $D_{h_1,\ldots,h_k}$ — rozkład prawdopodobieństwa na $G_k(x, h_1, \ldots, h_k)$ przy x z rozkładu jednostajnego na $\{0,1\}^m$, $M_{h_1,\ldots,h_k} = Q(D_{h_1,\ldots,h_k})$
  - $h_k \in H$ jest ϵ-dobra dla $h_1, \ldots, h_{k-1}$ jeśli $\|M^2_{h_1,\ldots,h_{k-1}} - M_{h_1,\ldots,h_k}\| \leqslant \epsilon$ (dla $k = 1$ mamy $M_{null} = Q(U_m)$)

Notacja

- m, k — liczby całkowite $\geqslant 0$, H — rodzina uniwersalnych funkcji haszujących $h : \{0,1\}^m \to \{0,1\}^m$, $\circ$ — operacja konkatenacji
  - $G_0(x) = x$
  - $G_k(x, h_1, \ldots, h_k) = G_{k-1}(x, h_1, \ldots, h_{k-1}) \circ G_{k-1}(h_k(x), h_1, \ldots h_{k-1})$
- $h_1, \ldots, h_k \in H$, $D_{h_1,\ldots,h_k}$ — rozkład prawdopodobieństwa na $G_k(x, h_1, \ldots, h_k)$ przy x z rozkładu jednostajnego na $\{0,1\}^m$, $M_{h_1,\ldots,h_k} = Q(D_{h_1,\ldots,h_k})$
  - $h_k \in H$ jest ϵ-dobra dla $h_1, \ldots, h_{k-1}$ jeśli $\|M^2_{h_1,\ldots,h_{k-1}} - M_{h_1,\ldots,h_k}\| \leqslant \epsilon$ (dla $k = 1$ mamy $M_{null} = Q(U_m)$)

Notacja

- m, k — liczby całkowite $\geqslant 0$, H — rodzina uniwersalnych funkcji haszujących $h : \{0,1\}^m \to \{0,1\}^m$, $\circ$ — operacja konkatenacji
  - $G_0(x) = x$
  - $G_k(x, h_1, \ldots, h_k) = G_{k-1}(x, h_1, \ldots, h_{k-1}) \circ G_{k-1}(h_k(x), h_1, \ldots h_{k-1})$
- $h_1, \ldots, h_k \in H$, $D_{h_1,\ldots,h_k}$ — rozkład prawdopodobieństwa na $G_k(x, h_1, \ldots, h_k)$ przy x z rozkładu jednostajnego na $\{0,1\}^m$, $M_{h_1,\ldots,h_k} = Q(D_{h_1,\ldots,h_k})$
  - $h_k \in H$ jest ϵ-dobra dla $h_1, \ldots, h_{k-1}$ jeśli $\|M^2_{h_1,\ldots,h_{k-1}} - M_{h_1,\ldots,h_k}\| \leqslant \epsilon$ (dla $k = 1$ mamy $M_{null} = Q(U_m)$)

Notacja

- m, k — liczby całkowite $\geqslant 0$, H — rodzina uniwersalnych funkcji haszujących $h : \{0,1\}^m \to \{0,1\}^m$, $\circ$ — operacja konkatenacji
  - $G_0(x) = x$
  - $G_k(x, h_1, \ldots, h_k) = G_{k-1}(x, h_1, \ldots, h_{k-1}) \circ G_{k-1}(h_k(x), h_1, \ldots h_{k-1})$
- $h_1, \ldots, h_k \in H$, $D_{h_1,\ldots,h_k}$ — rozkład prawdopodobieństwa na $G_k(x, h_1, \ldots, h_k)$ przy x z rozkładu jednostajnego na $\{0,1\}^m$, $M_{h_1,\ldots,h_k} = Q(D_{h_1,\ldots,h_k})$
  - $h_k \in H$ jest ϵ-dobra dla $h_1, \ldots, h_{k-1}$ jeśli $\|M^2_{h_1,\ldots h_{k-1}} - M_{h_1,\ldots,h_k}\| \leqslant \epsilon$ (dla $k = 1$ mamy $M_{null} = Q(U_m)$)

Notacja

- m, k — liczby całkowite $\geqslant 0$, H — rodzina uniwersalnych funkcji haszujących $h : \{0,1\}^m \to \{0,1\}^m$, $\circ$ — operacja konkatenacji
  - $G_0(x) = x$
  - $G_k(x, h_1, \ldots, h_k) = G_{k-1}(x, h_1, \ldots, h_{k-1}) \circ G_{k-1}(h_k(x), h_1, \ldots h_{k-1})$
- $h_1, \ldots, h_k \in H$, $D_{h_1,\ldots,h_k}$ — rozkład prawdopodobieństwa na $G_k(x, h_1, \ldots, h_k)$ przy x z rozkładu jednostajnego na $\{0,1\}^m$, $M_{h_1,\ldots,h_k} = Q(D_{h_1,\ldots,h_k})$
  - $h_k \in H$ jest ϵ-dobra dla $h_1, \ldots, h_{k-1}$ jeśli $\|M^2_{h_1,\ldots h_{k-1}} - M_{h_1,\ldots,h_k}\| \leqslant \epsilon$ (dla $k = 1$ mamy $M_{null} = Q(U_m)$)

Główny algorytm

Załóżmy, że mamy dany algorytm rozwiązujący następujący problem:
Wejście: n-stanowy DFA, liczba całkowita t i ułamek ϵ (t i ϵ są podane w postaci unarnej)
Wyjście: Liczba całkowita $t' \geqslant t$ i macierz A rozmiaru n na n taka, że $\|A - Q(U_{t'})\| \leqslant \epsilon$.
Algorytm działa w czasie $O(n^{24}\epsilon^{-6}t^7\log^3 N)$, gdzie N jest całkowitym rozmiarem wejścia.

BPL $\subseteq$ SC.

- Zbuduj DFA Q reprezentujący działanie danej maszyny T na danym wejściu x
- Niech $t = n$ będzie liczbą stanów Q i niech $\epsilon = 0.1$
- Uruchom główny algorytm
- Wynik działania pozwala odpowiedzieć, czy T akceptuje x ($\sum_{stany\ \ akceptujace\ j} A[1, j]$)

□

OSIĄGALNOŚĆ $\in$ SC.

Wynika z BPL $\subseteq$ SC i z OSIĄGALNOŚĆ $\in$ RL □

BPL $\subseteq$ SC.

- Zbuduj DFA Q reprezentujący działanie danej maszyny T na danym wejściu x
- Niech $t = n$ będzie liczbą stanów Q i niech $\epsilon = 0.1$
- Uruchom główny algorytm
- Wynik działania pozwala odpowiedzieć, czy T akceptuje x ($\sum_{stany\ \ akceptujace\ j} A[1, j]$)

□

OSIĄGALNOŚĆ $\in$ SC.

Wynika z BPL $\subseteq$ SC i z OSIĄGALNOŚĆ $\in$ RL □

BPL $\subseteq$ SC.

- Zbuduj DFA Q reprezentujący działanie danej maszyny T na danym wejściu x
- Niech $t = n$ będzie liczbą stanów Q i niech $\epsilon = 0.1$
- Uruchom główny algorytm
- Wynik działania pozwala odpowiedzieć, czy T akceptuje x ($\sum_{stany\ \ akceptujace\ j} A[1, j]$)

□

OSIĄGALNOŚĆ $\in$ SC.

Wynika z BPL $\subseteq$ SC i z OSIĄGALNOŚĆ $\in$ RL □

BPL $\subseteq$ SC.

- Zbuduj DFA Q reprezentujący działanie danej maszyny T na danym wejściu x
- Niech $t = n$ będzie liczbą stanów Q i niech $\epsilon = 0.1$
- Uruchom główny algorytm
- Wynik działania pozwala odpowiedzieć, czy T akceptuje x ($\sum_{stany\ akceptujace\ j} A[1,j]$)

□

OSIĄGALNOŚĆ $\in$ SC.

Wynika z BPL $\subseteq$ SC i z OSIĄGALNOŚĆ $\in$ RL □

BPL $\subseteq$ SC.

- Zbuduj DFA Q reprezentujący działanie danej maszyny T na danym wejściu x
- Niech $t = n$ będzie liczbą stanów Q i niech $\epsilon = 0.1$
- Uruchom główny algorytm
- Wynik działania pozwala odpowiedzieć, czy T akceptuje x ($\sum_{stany\ \ akceptujace\ j} A[1, j]$)

□

OSIĄGALNOŚĆ $\in$ SC.

Wynika z BPL $\subseteq$ SC i z OSIĄGALNOŚĆ $\in$ RL □

APROKS. OSIĄGALNOŚĆ.

- Zbuduj DFA Q:
  - Stany główne: $\{< j, i >: 1 \leqslant j \leqslant n, 0 \leqslant i \leqslant t\}$
  - Stany dodatkowe: binarne drzewo stanów prowadzące od stanu postaci $< j, i >$ do wszystkich stanów $< j', i + 1 >$ (dla $0 \leqslant i < t$)
  - Ze stanu $< j, i >$ da się przejśc do stanu $< j', i + 1 >$ z prawdopodobieństwem $M[j, j']$ (z dokładnością do ϵ/nt)
- Uruchom główny algorytm
- Wybierz z wyniku podmacierz wyznaczoną przez wiersze postaci $< j, 0 >$ i kolumny postaci $< j, t >$

APROKS. OSIĄGALNOŚĆ.

- Zbuduj DFA Q:
  - Stany główne: $\{< j, i >: 1 \leqslant j \leqslant n, 0 \leqslant i \leqslant t\}$
  - Stany dodatkowe: binarne drzewo stanów prowadzące od stanu postaci $< j, i >$ do wszystkich stanów $< j', i + 1 >$ (dla $0 \leqslant i < t$)
  - Ze stanu $< j, i >$ da się przejśc do stanu $< j', i + 1 >$ z prawdopodobieństwem $M[j, j']$ (z dokładnością do ϵ/nt)
- Uruchom główny algorytm
- Wybierz z wyniku podmacierz wyznaczoną przez wiersze postaci $< j, 0 >$ i kolumny postaci $< j, t >$

APROKS. OSIĄGALNOŚĆ.

- Zbuduj DFA Q:
  - Stany główne: $\{< j, i >: 1 \leqslant j \leqslant n, 0 \leqslant i \leqslant t\}$
  - Stany dodatkowe: binarne drzewo stanów prowadzące od stanu postaci $< j, i >$ do wszystkich stanów $< j', i + 1 >$ (dla $0 \leqslant i < t$)
  - Ze stanu $< j, i >$ da się przejśc do stanu $< j', i + 1 >$ z prawdopodobieństwem $M[j, j']$ (z dokładnością do ϵ/nt)
- Uruchom główny algorytm
- Wybierz z wyniku podmacierz wyznaczoną przez wiersze postaci $< j, 0 >$ i kolumny postaci $< j, t >$

□

APROKS. OSIĄGALNOŚĆ.

- Zbuduj DFA Q:
  - Stany główne: $\{< j, i >: 1 \leqslant j \leqslant n, 0 \leqslant i \leqslant t\}$
  - Stany dodatkowe: binarne drzewo stanów prowadzące od stanu postaci $< j, i >$ do wszystkich stanów $< j', i + 1 >$ (dla $0 \leqslant i < t$)
  - Ze stanu $< j, i >$ da się przejśc do stanu $< j', i + 1 >$ z prawdopodobieństwem $M[j, j']$ (z dokładnością do ϵ/nt)
- Uruchom główny algorytm
- Wybierz z wyniku podmacierz wyznaczoną przez wiersze postaci $< j, 0 >$ i kolumny postaci $< j, t >$

□

APROKS. OSIĄGALNOŚĆ.

- Zbuduj DFA Q:
  - Stany główne: $\{< j, i >: 1 \leqslant j \leqslant n, 0 \leqslant i \leqslant t\}$
  - Stany dodatkowe: binarne drzewo stanów prowadzące od stanu postaci $< j, i >$ do wszystkich stanów $< j', i + 1 >$ (dla $0 \leqslant i < t$)
  - Ze stanu $< j, i >$ da się przejśc do stanu $< j', i + 1 >$ z prawdopodobieństwem $M[j, j']$ (z dokładnością do ϵ/nt)
- Uruchom główny algorytm
- Wybierz z wyniku podmacierz wyznaczoną przez wiersze postaci $< j, 0 >$ i kolumny postaci $< j, t >$

APROKS. OSIĄGALNOŚĆ.

- Zbuduj DFA Q:
  - Stany główne: $\{< j, i >: 1 \leqslant j \leqslant n, 0 \leqslant i \leqslant t\}$
  - Stany dodatkowe: binarne drzewo stanów prowadzące od stanu postaci $< j, i >$ do wszystkich stanów $< j', i + 1 >$ (dla $0 \leqslant i < t$)
  - Ze stanu $< j, i >$ da się przejśc do stanu $< j', i + 1 >$ z prawdopodobieństwem $M[j, j']$ (z dokładnością do ϵ/nt)
- Uruchom główny algorytm
- Wybierz z wyniku podmacierz wyznaczoną przez wiersze postaci $< j, 0 >$ i kolumny postaci $< j, t >$

□

Główny algorytm

- Niech $K = \log_2 t$, $\delta = \epsilon/(2^K)$, $m = 1 + 7\log_2 n - 2\log_2 \delta$, $t' = m2^K$.
- Dla $k = 1, \ldots, K$ oblicz h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$ (procedura FIND)
- Dla wszystkich i, j — stany Q oblicz $A[i, j] = M_{h_1,\ldots,h_k}[i, j]$ (procedura COMPUTE)

Główny algorytm

- Niech $K = \log_2 t$, $\delta = \epsilon/(2^K)$, $m = 1 + 7\log_2 n - 2\log_2 \delta$, $t' = m2^K$.
- Dla $k = 1, \ldots, K$ oblicz h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$ (procedura FIND)
- Dla wszystkich i, j — stany Q oblicz $A[i,j] = M_{h_1,\ldots,h_k}[i,j]$ (procedura COMPUTE)

Główny algorytm

- Niech $K = \log_2 t$, $\delta = \epsilon/(2^K)$, $m = 1 + 7\log_2 n - 2\log_2 \delta$, $t' = m2^K$.
- Dla $k = 1, \ldots, K$ oblicz h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$ (procedura FIND)
- Dla wszystkich i, j — stany Q oblicz $A[i,j] = M_{h_1,\ldots,h_k}[i,j]$ (procedura COMPUTE)

Procedura FIND

Wejście: Funkcje haszujące $h_1, \ldots, h_{k-1}$, ułamek $\delta > 0$
Wyjście: Funkcja haszująca h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$
Algorytm:

- Dla każdego $h \in H$:
  - Dla wszystkich i, j — stany Q:
    - Policz $p_1 = M_{h_1,\ldots,h_{k-1},h}[i,j]$ (procedura COMPUTE)
    - Policz $p_2 = \sum_{\text{stany } l} M_{h_1,\ldots,h_{k-1}}[i,l] M_{h_1,\ldots,h_{k-1}}[l,j]$ ($2n$ wywołań COMPUTE)
    - Jeśli $|p_1 - p_2| > \delta/n$ przejdź do następnego h
  - Zwróć $h_k = h$

Procedura FIND

Wejście: Funkcje haszujące $h_1, \ldots, h_{k-1}$, ułamek $\delta > 0$
Wyjście: Funkcja haszująca h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$
Algorytm:

- Dla każdego $h \in H$:
  - Dla wszystkich i, j — stany Q:
    - Policz $p_1 = M_{h_1,\ldots,h_{k-1},h}[i,j]$ (procedura COMPUTE)
    - Policz $p_2 = \sum_{\text{stany } l} M_{h_1,\ldots,h_{k-1}}[i,l] M_{h_1,\ldots,h_{k-1}}[l,j]$ ($2n$ wywołań COMPUTE)
    - Jeśli $|p_1 - p_2| > \delta/n$ przejdź do następnego h
  - Zwróć $h_k = h$

Procedura FIND

Wejście: Funkcje haszujące $h_1, \ldots, h_{k-1}$, ułamek $\delta > 0$
Wyjście: Funkcja haszująca h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$
Algorytm:

- Dla każdego $h \in H$:
  - Dla wszystkich i, j — stany Q:
    - Policz $p_1 = M_{h_1,\ldots,h_{k-1},h}[i,j]$ (procedura COMPUTE)
    - Policz $p_2 = \sum_{\text{stany } l} M_{h_1,\ldots,h_{k-1}}[i,l] M_{h_1,\ldots,h_{k-1}}[l,j]$ ($2n$ wywołań COMPUTE)
    - Jeśli $|p_1 - p_2| > \delta/n$ przejdź do następnego h
  - Zwróć $h_k = h$

Procedura FIND

Wejście: Funkcje haszujące $h_1, \ldots, h_{k-1}$, ułamek $\delta > 0$
Wyjście: Funkcja haszująca h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$
Algorytm:

- Dla każdego $h \in H$:
  - Dla wszystkich i, j — stany Q:
    - Policz $p_1 = M_{h_1,\ldots,h_{k-1},h}[i,j]$ (procedura COMPUTE)
    - Policz $p_2 = \sum_{\text{stany } l} M_{h_1,\ldots,h_{k-1}}[i,l] M_{h_1,\ldots,h_{k-1}}[l,j]$ ($2n$ wywołań COMPUTE)
    - Jeśli $|p_1 - p_2| > \delta/n$ przejdź do następnego h
  - Zwróć $h_k = h$

Procedura FIND

Wejście: Funkcje haszujące $h_1, \ldots, h_{k-1}$, ułamek $\delta > 0$
Wyjście: Funkcja haszująca h_k, która jest δ-dobra dla $h_1, \ldots, h_{k-1}$
Algorytm:

- Dla każdego $h \in H$:
  - Dla wszystkich i, j — stany Q:
    - Policz $p_1 = M_{h_1,\ldots,h_{k-1},h}[i,j]$ (procedura COMPUTE)
    - Policz $p_2 = \sum_{\text{stany } l} M_{h_1,\ldots,h_{k-1}}[i,l] M_{h_1,\ldots,h_{k-1}}[l,j]$ ($2n$ wywołań COMPUTE)
    - Jeśli $|p_1 - p_2| > \delta/n$ przejdź do następnego h
  - Zwróć $h_k = h$

Procedura COMPUTE

Wejście: Funkcje haszujące $h_1, \ldots, h_k$, stany i, j
Wyjście: Wartość $M_{h_1,\ldots,h_k}[i,j]$

- count := 0
- Dla wszystkich $x \in \{0,1\}^m$:
  - Symuluj Q zaczynając od i na wejściu $G_k(x, h_1, \ldots, h_k)$
  - Jeśli Q skończył pracę w stanie j to count := count $+1$
- Zwróć $M_{h_1,\ldots,h_k}[i,j] =$count$/2^m$

Procedura COMPUTE

Wejście: Funkcje haszujące $h_1, \ldots, h_k$, stany i, j
Wyjście: Wartość $M_{h_1,\ldots,h_k}[i,j]$

- count := 0
- Dla wszystkich $x \in \{0,1\}^m$:
  - Symuluj Q zaczynając od i na wejściu $G_k(x, h_1, \ldots, h_k)$
  - Jeśli Q skończył pracę w stanie j to count := count $+1$
- Zwróć $M_{h_1,\ldots,h_k}[i,j] =$count$/2^m$

Procedura COMPUTE

Wejście: Funkcje haszujące $h_1, \ldots, h_k$, stany i, j
Wyjście: Wartość $M_{h_1,\ldots,h_k}[i,j]$

- count := 0
- Dla wszystkich $x \in \{0,1\}^m$:
  - Symuluj Q zaczynając od i na wejściu $G_k(x, h_1, \ldots, h_k)$
  - Jeśli Q skończył pracę w stanie j to count := count $+1$
- Zwróć $M_{h_1,\ldots,h_k}[i,j] =$count$/2^m$

Procedura COMPUTE

Wejście: Funkcje haszujące $h_1, \ldots, h_k$, stany i, j
Wyjście: Wartość $M_{h_1,\ldots,h_k}[i,j]$

- count := 0
- Dla wszystkich $x \in \{0,1\}^m$:
  - Symuluj Q zaczynając od i na wejściu $G_k(x, h_1, \ldots, h_k)$
  - Jeśli Q skończył pracę w stanie j to count := count $+1$
- Zwróć $M_{h_1,\ldots,h_k}[i,j] =$count$/2^m$

Procedura COMPUTE

Wejście: Funkcje haszujące $h_1, \ldots, h_k$, stany i, j
Wyjście: Wartość $M_{h_1,\ldots,h_k}[i,j]$

- count := 0
- Dla wszystkich $x \in \{0,1\}^m$:
  - Symuluj Q zaczynając od i na wejściu $G_k(x, h_1, \ldots, h_k)$
  - Jeśli Q skończył pracę w stanie j to count := count $+1$
- Zwróć $M_{h_1,\ldots,h_k}[i,j] =$count$/2^m$