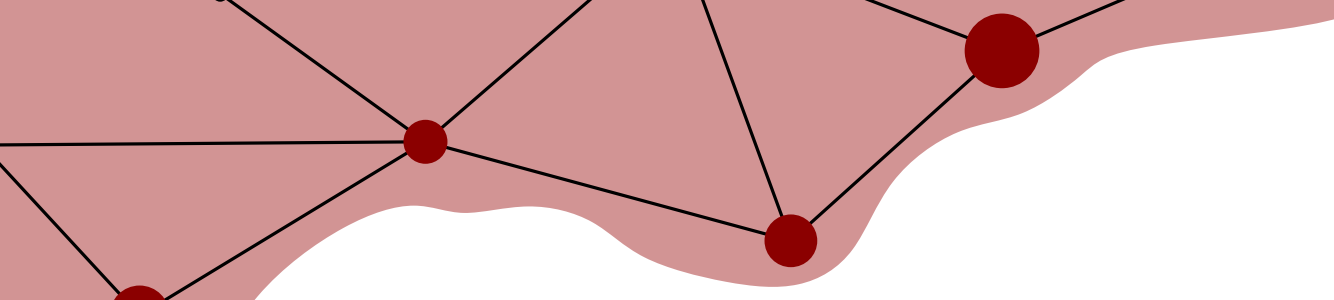


Dynamiczny PageRank

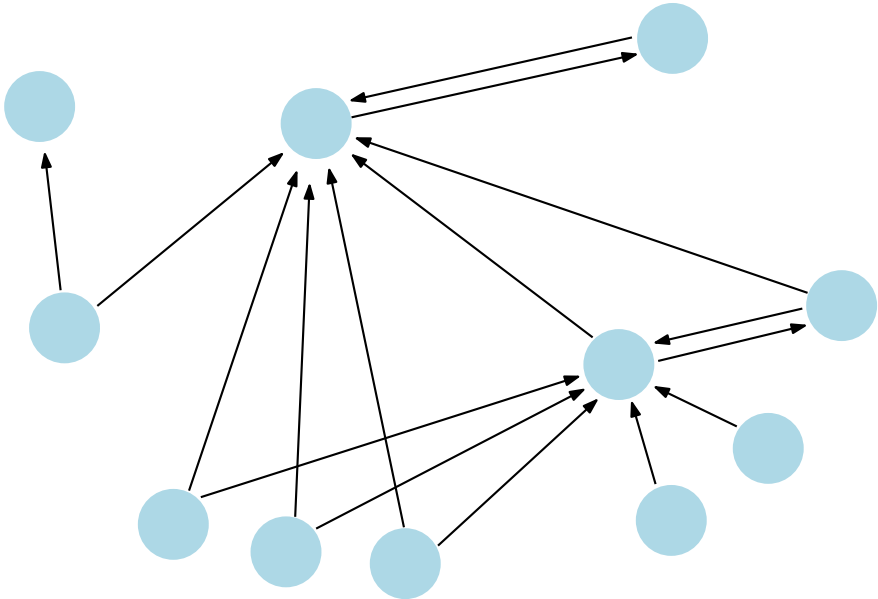
Dynamic PageRank: Algorithms and Lower Bounds 2024

- Rajesh Jayaram
- Jakub Łącki
- Slobodan Mitrović
- Krzysztof Onak
- Piotr Sankowski

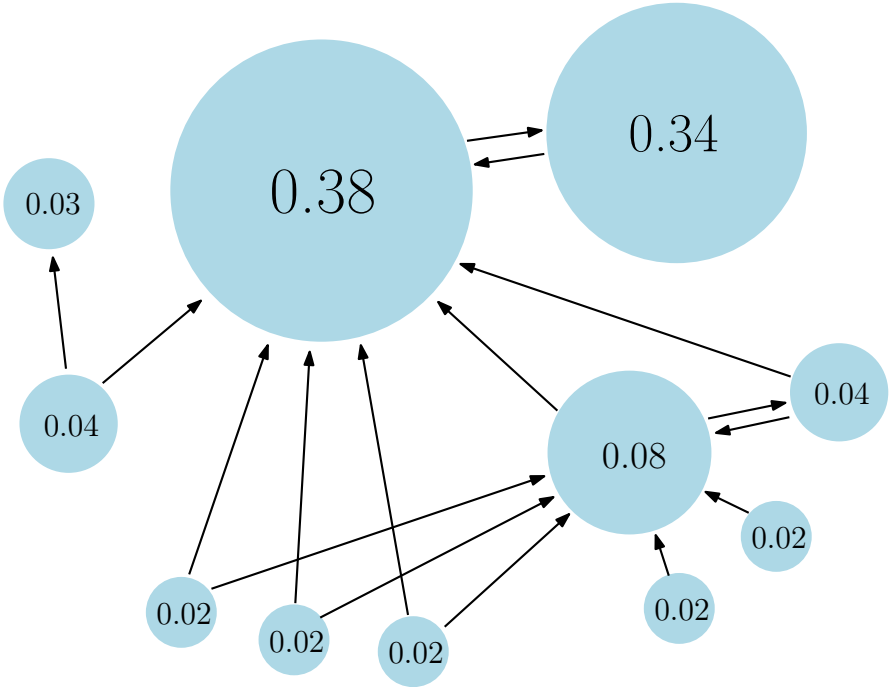
Zuzanna Szewczyk
Piotr Kubicki

PageRank

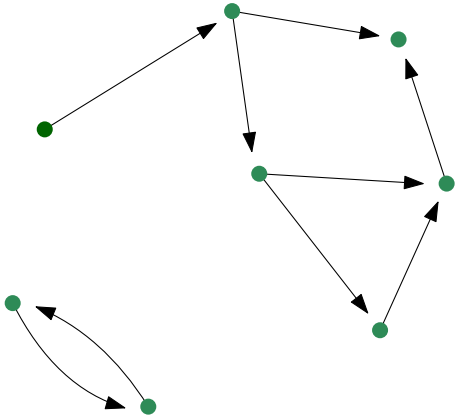
Cel: stworzyć ranking oceniający wierzchołki grafu (strony internetowe)



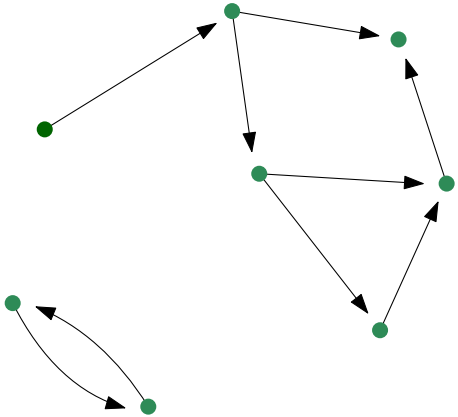
PageRank



Rozkład stacjonarny

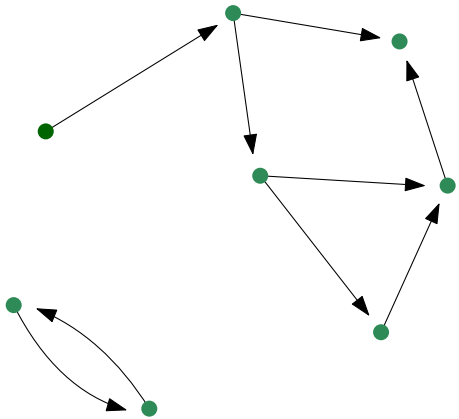


Rozkład stacjonarny



Proces stochastyczny ma rozkład stacjonarny jeżeli łańcuch markowa jest ergodyczny:

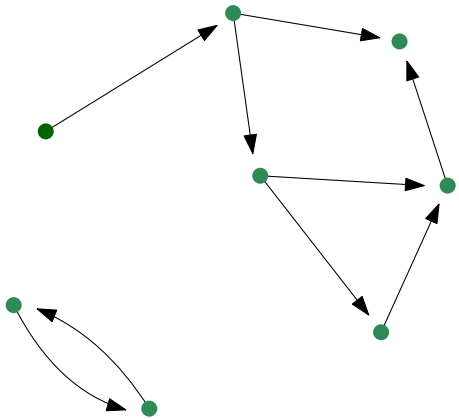
Rozkład stacjonarny



Proces stochastyczny ma rozkład stacjonarny jeżeli łańcuch markowa jest ergodyczny:

- nieredukowalny
  - jedna spójna składowa

Rozkład stacjonarny



Proces stochastyczny ma rozkład stacjonarny jeżeli łańcuch markowa jest ergodyczny:

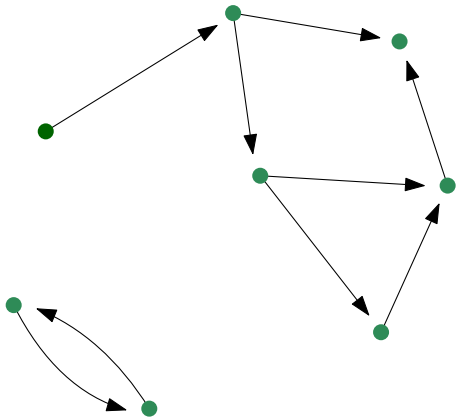
- nieredukowalny

  jedna spójna składowa

- pozytywnie rekurencyjny

  Wszystkie stany pozytywnie rekurencyjne

Rozkład stacjonarny



Proces stochastyczny ma rozkład stacjonarny jeżeli łańcuch markowa jest ergodyczny:

- nieredukowalny

  jedna spójna składowa

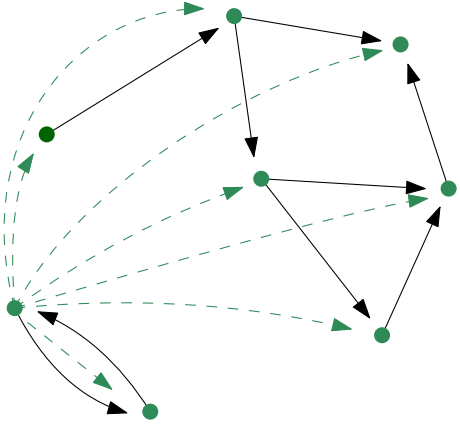
- pozytywnie rekurencyjny

  Wszystkie stany pozytywnie rekurencyjne

- aperiodyczny

  brak okresowości

Rozkład stacjonarny



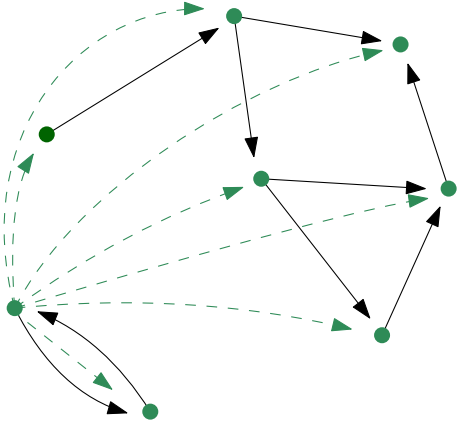
Proces stochastyczny ma rozkład stacjonarny jeżeli łańcuch markowa jest ergodyczny:

- nieredukowalny
  jedna spójna składowa
- pozytywnie rekurencyjny
  Wszystkie stany pozytywnie rekurencyjne
- aperiodyczny
  brak okresowości

ϵ prawdopodobieństwo losowego skoku

$1 - \epsilon$ prawdopodobieństwo przejścia krawędzią grafu (damping factor)

Rozkład stacjonarny



Proces stochastyczny ma rozkład stacjonarny jeżeli łańcuch markowa jest ergodyczny:

- ✓ nieredukowalny
  jedna spójna składowa
- ✓ pozytywnie rekurencyjny
  Wszystkie stany pozytywnie rekurencyjne
- ✓ aperiodyczny
  brak okresowości

ϵ prawdopodobieństwo losowego skoku

$1 - \epsilon$ prawdopodobieństwo przejścia krawędzią grafu (damping factor)

Definicja wektora PageRank

Dla grafu $G = (V, E)$ wektor PageRank $\pi \in \mathbb{R}^n$ to rozkład stacjonarny spaceru losowego, który w każdym kroku z prawdopodobieństwej ϵ skacze do dowolnego wierzchołka.

Definicja wektora PageRank

Dla grafu $G = (V, E)$ wektor PageRank $\pi \in \mathbb{R}^n$ to rozkład stacjonarny spaceru losowego, który w każdym kroku z prawdopodobieństwej ϵ skacze do dowolnego wierzchołka.

Macierz przejścia M ma więc postać:

$$M_{i,j} = \begin{cases} \frac{\epsilon}{n} + (1-\epsilon)\frac{1}{deg(i)}, & \text{if } (i,j) \in E \\ \frac{\epsilon}{n}, & \text{else} \end{cases}$$

Definicja wektora PageRank

Dla grafu $G = (V, E)$ wektor PageRank $\pi \in \mathbb{R}^n$ to rozkład stacjonarny spaceru losowego, który w każdym kroku z prawdopodobieństwej ϵ skacze do dowolnego wierzchołka.

Macierz przejścia M ma więc postać:

$$M_{i,j} = \begin{cases} \frac{\epsilon}{n} + (1-\epsilon)\frac{1}{deg(i)}, & \text{if } (i,j) \in E \\ \frac{\epsilon}{n}, & \text{else} \end{cases}$$

Obliczanie przybliżonej wartości π

Definicja wektora PageRank

Dla grafu $G = (V, E)$ wektor PageRank $\pi \in \mathbb{R}^n$ to rozkład stacjonarny spaceru losowego, który w każdym kroku z prawdopodobieństwej ϵ skacze do dowolnego wierzchołka.

Macierz przejścia M ma więc postać:

$$M_{i,j} = \begin{cases} \frac{\epsilon}{n} + (1-\epsilon)\frac{1}{deg(i)}, & \text{if } (i,j) \in E \\ \frac{\epsilon}{n}, & \text{else} \end{cases}$$

Obliczanie przybliżonej wartości π

- Metoda potęgowa

Definicja wektora PageRank

Dla grafu $G = (V, E)$ wektor PageRank $\pi \in \mathbb{R}^n$ to rozkład stacjonarny spaceru losowego, który w każdym kroku z prawdopodobieństwej ϵ skacze do dowolnego wierzchołka.

Macierz przejścia M ma więc postać:

$$M_{i,j} = \begin{cases} \frac{\epsilon}{n} + (1-\epsilon)\frac{1}{deg(i)}, & \text{if } (i,j) \in E \\ \frac{\epsilon}{n}, & \text{else} \end{cases}$$

Obliczanie przybliżonej wartości π

- Metoda potęgowa
- Samplowanie losowych spacerów

Rodzaje aproksymacji

Rodzaje aproksymacji

Aproksymacja addytywna α

$$\|\tilde{\pi} - \pi\|_1 \leq \alpha$$

Rodzaje aproksymacji

Aproksymacja addytywna α

$$\|\tilde{\pi} - \pi\|_1 \leq \alpha$$

Aproksymacja multiplikatywna $1 + \alpha$

$$\tilde{\pi}_v \in [(1-\alpha)\pi_v, (1+\alpha)\pi_v]$$

dla każdego wierzchołka v

Rodzaje aproksymacji

Aproksymacja addytywna α

$$\|\tilde{\pi} - \pi\|_1 \leq \alpha$$

Aproksymacja multiplikatywna $1 + \alpha$

$$\tilde{\pi_v} \in [(1-\alpha)\pi_v, (1+\alpha)\pi_v]$$

dla każdego wierzchołka v

$$(1-\alpha)\pi_v \leq \tilde{\pi_v} \leq (1+\alpha)\pi_v$$

$$-\alpha\pi_v \leq \tilde{\pi_v} - \pi_v \leq \alpha\pi_v$$

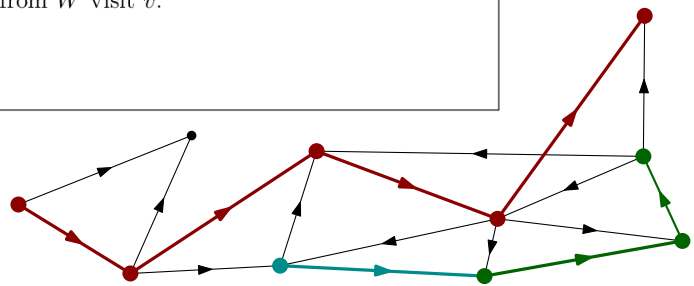
$$|\tilde{\pi_v} - \pi_v| \leq \alpha\pi_v$$

$$\sum_{v \in V} |\tilde{\pi_v} - \pi_v| \leq \alpha \sum_{v \in V} \pi_v = \alpha$$

Przybliżanie π za pomocą losowych spacerów

Algorytm 1

1: Sample a set W of $R = \lceil \frac{9 \ln n}{\epsilon \alpha^2} \rceil$ random walks starting from each vertex of G.
Each walk length is chosen from geometric distribution with parameter $1 - \epsilon$.
2: Remove from W all walks longer than ℓ.
3: **for** $v \in V$ **do**
4: $\quad \mathbf{X}_v \leftarrow$ the number of times the walks from W visit v.
5: $\quad \tilde{\pi}(v) \leftarrow \frac{\mathbf{X}_v}{|W|/\epsilon}$.
6: **end for**
7: Return $\tilde{\pi}$



Proposition 1 ([BCG10, LMOS20]). *Let π be the PageRank vector of a graph G. The estimate $\tilde{\pi}$ computed by Algorithm 1 (with $\ell = \infty$) satisfies (a) for all $v \in V$ we have $\mathbb{E}[\tilde{\pi}_v] = \pi_v$, and (b) with probability $1 - 1/\operatorname{poly}(n)$, simultaneously for all $v \in V$, we have $\tilde{\pi}_v = (1 \pm \alpha)\pi_v$.*

Dynamiczny PageRank

Dynamiczny PageRank

Dany jest graf G wraz ze wszystkimi jego wierzchołkam, a jego krawędzie są wstawiane i usuwane. Celem jest utrzymanie przybliżonego wektora PageRank $\tilde{\pi}$.

Rozważamy głównie grafy skierowane

Dynamiczny PageRank

Dany jest graf G wraz ze wszystkimi jego wierzchołkam, a jego krawędzie są wstawiane i usuwane. Celem jest utrzymanie przybliżonego wektora PageRank $\tilde{\pi}$.

Rozważamy głównie grafy skierowane

W pracy rozważane jest jedynie jawne utrzymywanie wartości $\tilde{\pi}$ po każdej operacji wstawienia/usunięcia krawędzi

Dynamiczny PageRank

Dany jest graf G wraz ze wszystkimi jego wierzchołkam, a jego krawędzie są wstawiane i usuwane. Celem jest utrzymanie przybliżonego wektora PageRank $\tilde{\pi}$.

Rozważamy głównie grafy skierowane

W pracy rozważane jest jedynie jawne utrzymywanie wartości $\tilde{\pi}$ po każdej operacji wstawienia/usunięcia krawędzi

Rozróżniamy też trzy ustawienia ciągu operacji:

- inkrementacyjne (ciąg wstawień)
- dekrementacyjne (ciąg usunięć)
- w pełni dynamiczne

Dynamiczny PageRank

Dany jest graf G wraz ze wszystkimi jego wierzchołkam, a jego krawędzie są wstawiane i usuwane. Celem jest utrzymanie przybliżonego wektora PageRank $\tilde{\pi}$.

Rozważamy głównie grafy skierowane

W pracy rozważane jest jedynie jawne utrzymywanie wartości $\tilde{\pi}$ po każdej operacji wstawienia/usunięcia krawędzi

Rozróżniamy też trzy ustawienia ciągu operacji:

- inkrementacyjne (ciąg wstawień)
- dekrementacyjne (ciąg usunięć)
- w pełni dynamiczne

Stosowane podejścia:

- obliczanie PageRank za każdym razem od nowa
- utrzymywanie zbioru losowych spacerów i aktualizowanie go!

dynamiczna wersja Algorytmu 1 [Bahmani 2010]

Co chcemy wiedzieć o dynamicznym PageRank

Co chcemy wiedzieć o dynamicznym PageRank

Jesteśmy w stanie utrzymywać $\alpha + 1$ multiplikatywną aproksymację dla ciągu losowych wstawień (bądź ciągu losowych usunięć) w czasie $poly\ log(n)$.
[Bahmani 2010]

Co chcemy wiedzieć o dynamicznym PageRank

Jesteśmy w stanie utrzymywać $\alpha + 1$ multiplikatywną aproksymację dla ciągu losowych wstawień (bądź ciągu losowych usunięć) w czasie $poly\ log(n)$.
[Bahmani 2010]

Czy zachodzi też w ogólnym przypadku?

Co chcemy wiedzieć o dynamicznym PageRank

Jesteśmy w stanie utrzymywać $\alpha + 1$ multiplikatywną aproksymację dla ciągu losowych wstawień (bądź ciągu losowych usunięć) w czasie $poly\ log(n)$.
[Bahmani 2010]

Czy zachodzi też w ogólnym przypadku?

Jakie jest dolne ograniczenie na czas działania algorytmu utrzymującego α addytywną aproksymację dla ciągu wstawień krawędzi?

Co chcemy wiedzieć o dynamicznym PageRank

Jesteśmy w stanie utrzymywać $\alpha + 1$ multiplikatywną aproksymację dla ciągu losowych wstawień (bądź ciągu losowych usunięć) w czasie $poly\ log(n)$.
[Bahmani 2010]

Czy zachodzi też w ogólnym przypadku?

Jakie jest dolne ograniczenie na czas działania algorytmu utrzymującego α addytywną aproksymację dla ciągu wstawień krawędzi?

A jakie gdy utrzymujemy $\alpha + 1$ multiplikatywą aproksymacją

Wkład pracy

Wkład pracy

Ograniczenie dolne

dla utrzymywania α addytywnej aproksymacji

Zakładając, że $(1/\alpha) = n^{o(1/log\ log\ n)}$

Koszt ciągu operacji: $n \cdot (\frac{1}{\alpha})^{\Omega(loglogn)}$

Wkład pracy

Ograniczenie dolne

dla utrzymywania α addytywnej aproksymacji

Zakładając, że $(1/\alpha) = n^{o(1/log\ log\ n)}$

Koszt ciągu operacji: $n \cdot (\frac{1}{\alpha})^{\Omega(loglogn)}$

Ograniczenie dolne

dla utrzymywania $\alpha + 1$ multiplikatywnej aproksymacji

Amortyzowany czas operacji jest rzędu $\Omega(n^{1-\delta}), \delta > 0$

Wkład pracy

Ograniczenie dolne

dla utrzymywania α addytywnej aproksymacji

Zakładając, że $(1/\alpha) = n^{o(1/log\ log\ n)}$

Koszt ciągu operacji: $n \cdot (\frac{1}{\alpha})^{\Omega(loglogn)}$

Modyfikacja algorytmu utrzymującego losowe spacery
Analiza poprawności i złożoności dla α addytywnej aproksymacji w ciągu wstawień bądź ciągu usunięć

Czas działania dla ciągu operacji: $O(m) + n \cdot (\frac{1}{\alpha})^{O_\epsilon(loglogn)}$

Ograniczenie dolne

dla utrzymywania $\alpha + 1$ multiplikatywnej aproksymacji

Amortyzowany czas operacji jest rzędu $\Omega(n^{1-\delta}), \delta > 0$

Wkład pracy

Ograniczenie dolne

dla utrzymywania α addytywnej aproksymacji

Zakładając, że $(1/\alpha) = n^{o(1/log\ log\ n)}$

Koszt ciągu operacji: $n \cdot (\frac{1}{\alpha})^{\Omega(loglogn)}$

Ograniczenie dolne

dla utrzymywania $\alpha + 1$ multiplikatywnej aproksymacji

Amortyzowany czas operacji jest rzędu $\Omega(n^{1-\delta}), \delta > 0$

Modyfikacja algorytmu utrzymującego losowe spacery
Analiza poprawności i złożoności dla α addytywnej aproksymacji w ciągu wstawień bądź ciągu usunięć

Czas działania dla ciągu operacji: $O(m) + n \cdot (\frac{1}{\alpha})^{O_\epsilon(loglogn)}$

Analiza działania algorytmu dla grafów nieskierowanych w pełni dynamicznym ustawieniu

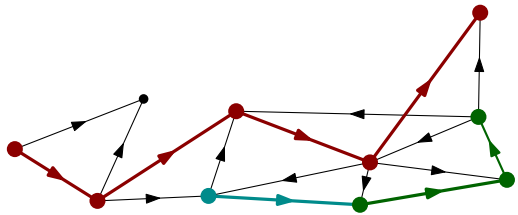
Utrzymanie $\alpha+1$ multiplikatywnej aproksymacji w czasie $O(log^5 n/(\epsilon^2\alpha^2))$ dla jednej operacji

Masa prawdopodobieństwa

Masa prawdopodobieństwa

Będziemy interpretowali PageRank w myśl propozycji 1:

$\tilde{\pi}$ computed by Algorithm 1 (with $\ell = \infty$) satisfies (a) for all $v \in V$ we have $\mathbb{E}[\tilde{\pi}_v] = \pi_v$

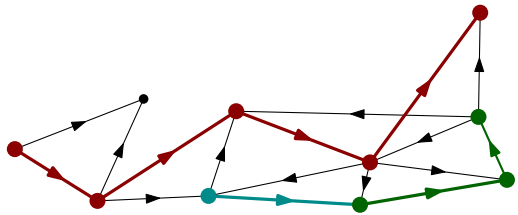


Masa prawdopodobieństwa

Będziemy interpretowali PageRank w myśl propozycji 1:

$\tilde{\pi}$ computed by Algorithm 1 (with $\ell = \infty$) satisfies (a) for all $v \in V$ we have $\mathbb{E}[\tilde{\pi}_v] = \pi_v$

Każdy wierzchołek ma pewną masę probabilistyczną, która generuje, bądź otrzymuje od innych wierzchołków.



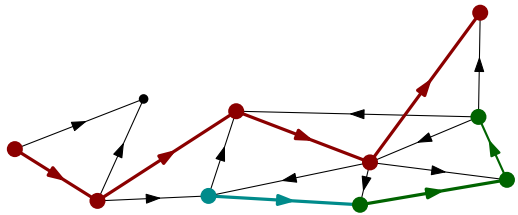
Masa prawdopodobieństwa

Będziemy interpretowali PageRank w myśl propozycji 1:

$\tilde{\pi}$ computed by Algorithm 1 (with $\ell = \infty$) satisfies (a) for all $v \in V$ we have $\mathbb{E}[\tilde{\pi}_v] = \pi_v$

Każdy wierzchołek ma pewną masę probabilistyczną, która generuje, bądź otrzymuje od innych wierzchołków.

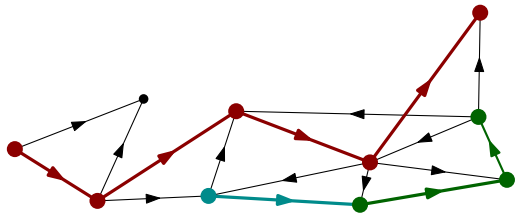
Każdy wierzchołek generuje $1/n$ masy



Masa prawdopodobieństwa

Będziemy interpretowali PageRank w myśl propozycji 1:

$\tilde{\pi}$ computed by Algorithm 1 (with $\ell = \infty$) satisfies (a) for all $v \in V$ we have $\mathbb{E}[\tilde{\pi}_v] = \pi_v$



Każdy wierzchołek ma pewną masę probabilistyczną, która generuje, bądź otrzymuje od innych wierzchołków.

Każdy wierzchołek generuje $1/n$ masy

$(1 - \epsilon)$ części masy każdego wierzchołka jest rozdzielona równomiernie między jego sąsiadów.

PageRank wierzchołka do ϵ'owa część jego masy prawdopodobieństwa.

Masa prawdopodobieństwa

Będziemy interpretowali PageRank w myśl propozycji 1:

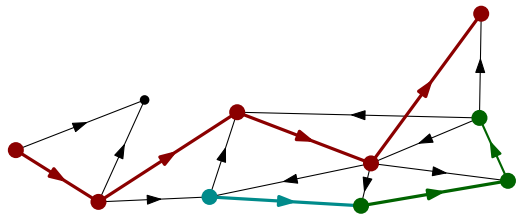
Każdy wierzchołek ma pewną masę probabilistyczną, która generuje, bądź otrzymuje od innych wierzchołków.

Każdy wierzchołek generuje $1/n$ masy

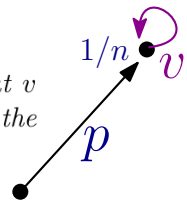
$(1-\epsilon)$ części masy każdego wierzchołka jest rozdzielona równomiernie między jego sąsiadów.

PageRank wierzchołka do ϵ'owa część jego masy prawdopodobieństwa.

$\tilde{\pi}$ computed by Algorithm 1 (with $\ell = \infty$) satisfies (a) for all $v \in V$ we have $\mathbb{E}[\tilde{\pi}_v] = \pi_v$

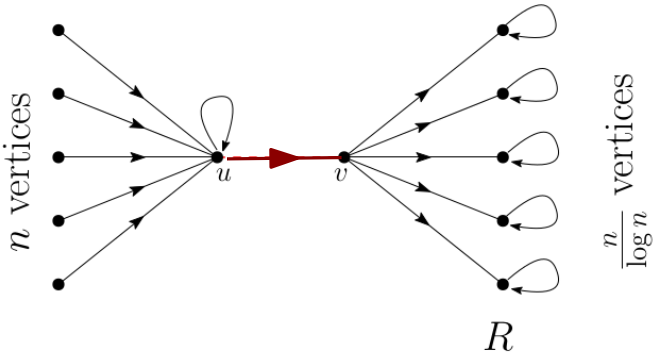


Observation 3.1. *Let v be a vertex, whose only outgoing edge is a self loop. Assume that v receives a probability mass of p along its incoming edges other than the self-loop. Then, the PageRank of v is $p + 1/n$.*

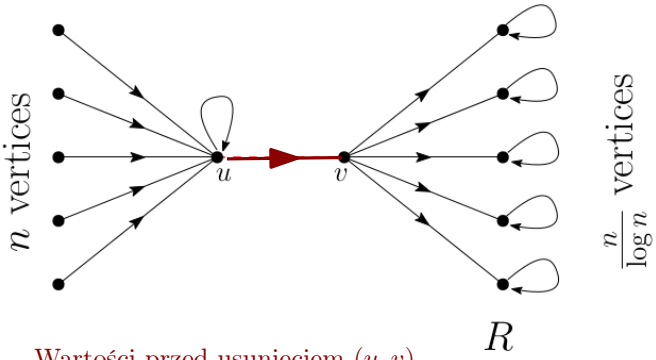


Trudny przypadek pojedynczego update'u

Trudny przypadek pojedynczego update'u



Trudny przypadek pojedynczego update'u

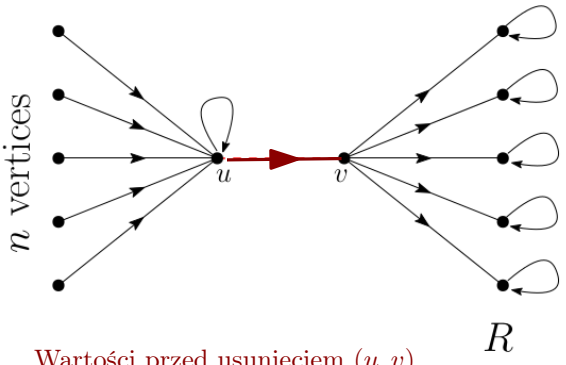


Wartości przed usunięciem (u, v)

$$\pi_u, \pi_v \in \Omega(\epsilon)$$

$$\pi_x \in \Omega((\log n)/n) \quad x \in R$$

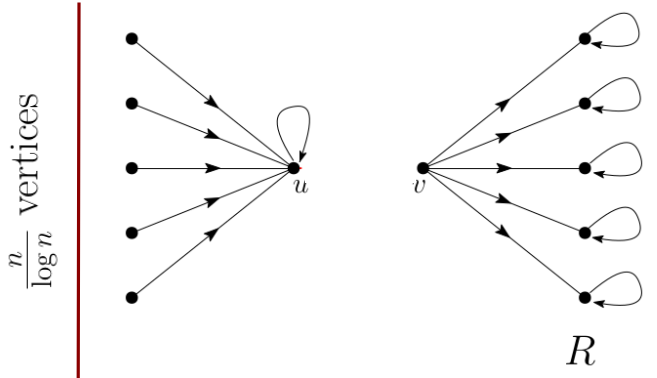
Trudny przypadek pojedynczego update'u



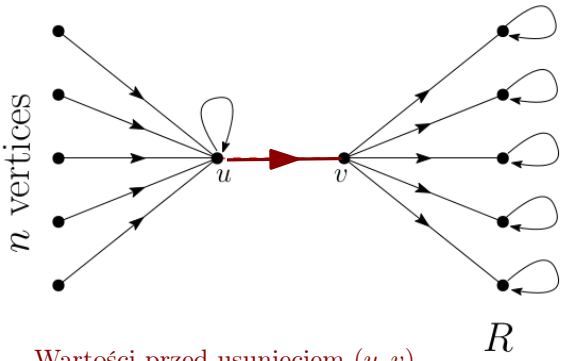
Wartości przed usunięciem (u, v)

$$\pi_u, \pi_v \in \Omega(\epsilon)$$

$$\pi_x \in \Omega((\log n)/n) \quad x \in R$$

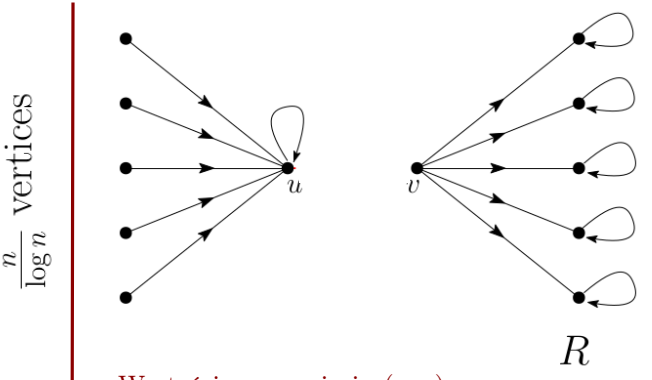


Trudny przypadek pojedynczego update'u



Wartości przed usunięciem (u, v)

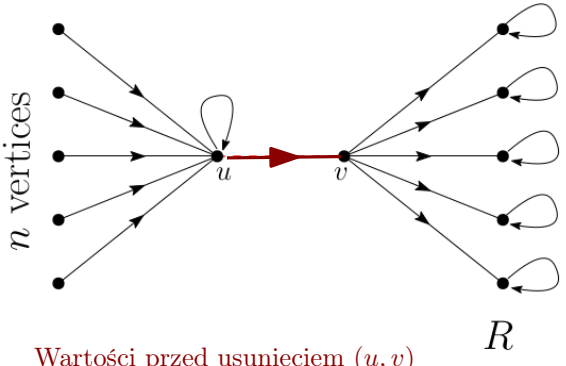
$$\pi_u, \pi_v \in \Omega(\epsilon)$$
$$\pi_x \in \Omega((\log n)/n) \quad x \in R$$



Wartości po usunięciu (u, v)

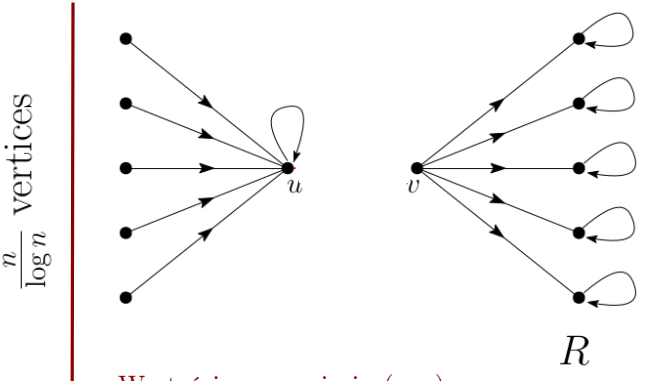
$$\pi'_u \in \Omega(1)$$
$$\pi'_v \in O(\epsilon/n)$$
$$\pi'_x \in O(1/n) \quad x \in R$$

Trudny przypadek pojedynczego update'u



Wartości przed usunięciem (u, v)

$$\pi_u, \pi_v \in \Omega(\epsilon)$$
$$\pi_x \in \Omega((\log n)/n) \quad x \in R$$

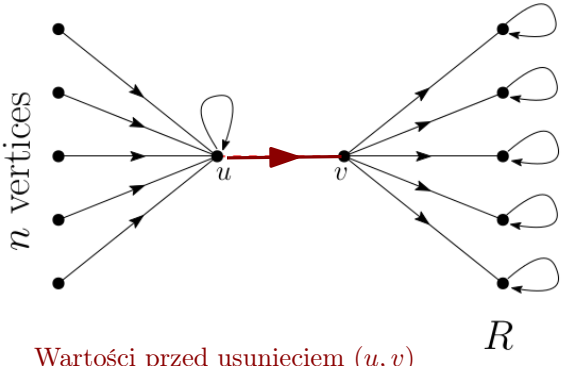


Wartości po usunięciu (u, v)

$$\pi'_u \in \Omega(1)$$
$$\pi'_v \in O(\epsilon/n)$$
$$\pi'_x \in O(1/n) \quad x \in R$$

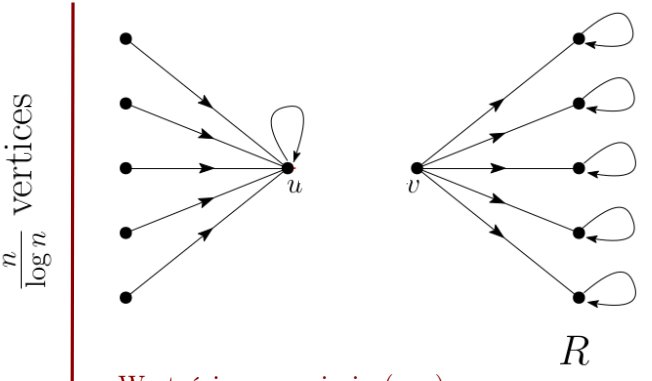
Dla $\Omega(\frac{n}{log\ n})$ wierzchołków wartości różnią się $\Omega(log\ n)$ krotnie po jednej operacji usunięcia.

Trudny przypadek pojedynczego update'u



Wartości przed usunięciem (u, v)

$$\pi_u, \pi_v \in \Omega(\epsilon)$$
$$\pi_x \in \Omega((\log n)/n) \quad x \in R$$



Wartości po usunięciu (u, v)

$$\pi'_u \in \Omega(1)$$
$$\pi'_v \in O(\epsilon/n)$$
$$\pi'_x \in O(1/n) \quad x \in R$$

Dla $\Omega(\frac{n}{log\ n})$ wierzchołków wartości różnią się $\Omega(log\ n)$ krotnie po jednej operacji usunięcia.

Co ze złożonością amortyzowaną?

Twierdzenie 1

Twierdzenie 1

Theorem 1.1. *Fix $\epsilon \in (0.01, 0.99)$. For any sufficiently large $n \geq 1$ and any α such that $1/\alpha = n^{o(1/\log\log n)}$, any algorithm which explicitly maintains α-additive approximation of PageRank must run in $n \cdot (1/\alpha)^{\Omega(\log\log n)}$ total time.*

Twierdzenie 1

Theorem 1.1. *Fix $\epsilon \in (0.01, 0.99)$. For any sufficiently large $n \geq 1$ and any α such that $1/\alpha = n^{o(1/\log\log n)}$, any algorithm which explicitly maintains α-additive approximation of PageRank must run in $n \cdot (1/\alpha)^{\Omega(\log\log n)}$ total time.*

Zarys dowodu:

Pokażemy konstrukcję grafu, do którego będziemy wstawiać krawędzie.
Wiele wstawień będzie wymagało aktualizacji liniowej liczby wartości $\tilde{\pi}$.
Po drodze potrzebne będą dwa lematy.

Konstrukcja (multi)grafu

Konstrukcja (multi)grafu

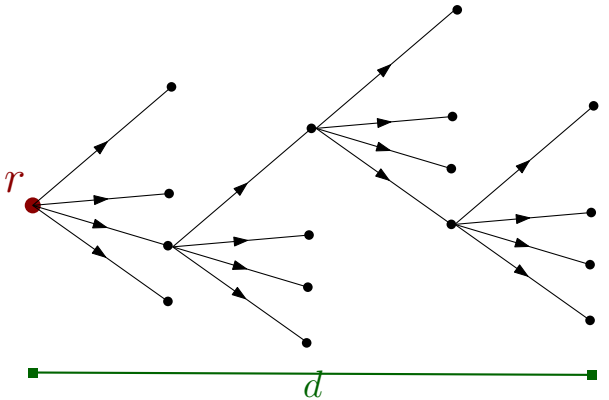
Graf G będzie składał się z 4 części: R, H, S_0, S_1

Konstrukcja (multi)grafu

Graf G będzie składał się z 4 części: R, H, S_0, S_1

Podgraf H będzie drzewem o korzeniu r i głębokości d.

Każdy wierzchołek wewnętrzny będzie miał t dzieci numerowanych od 0 do $t-1$



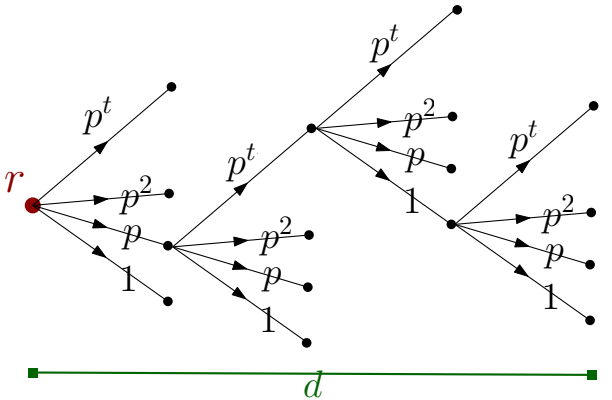
Konstrukcja (multi)grafu

Graf G będzie składał się z 4 części: R, H, S_0, S_1

Podgraf H będzie drzewem o korzeniu r i głębokości d.

Każdy wierzchołek wewnętrzny będzie miał t dzieci numerowanych od 0 do $t-1$

Do dziecka i-tego będzie prowadziło p^i równoległych krawędzi, dla $p = (1/\epsilon) \geq 2$.



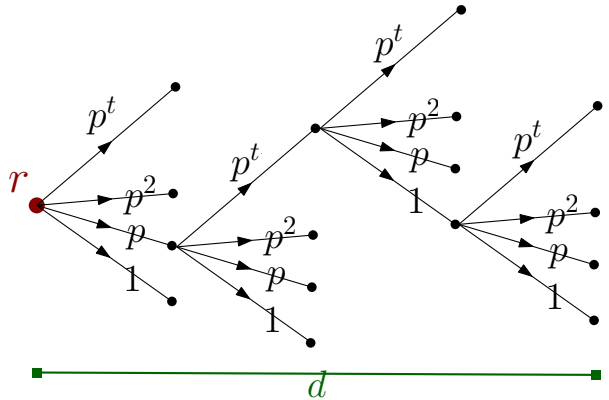
Konstrukcja (multi)grafu

Graf G będzie składał się z 4 części: R, H, S_0, S_1

Podgraf H będzie drzewem o korzeniu r i głębokości d.

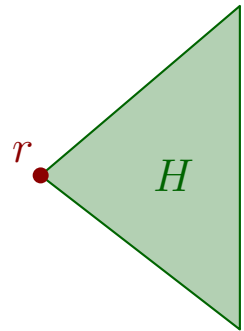
Każdy wierzchołek wewnętrzny będzie miał t dzieci numerowanych od 0 do $t-1$

Do dziecka i-tego będzie prowadziło p^i równoległych krawędzi, dla $p = (1/\epsilon) \geq 2$.



Z każdego wierzchołka wewnętrznego H wychodzi $O(p^t)$ krawędzi.

W H jest $\Theta(t^d)$ wierzchołków oraz $\Theta(t^d \cdot p^t)$ krawędzi.

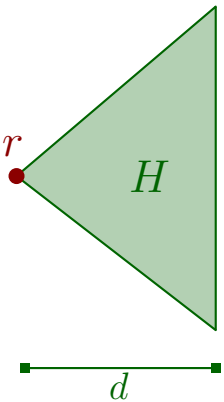


Konstrukcja (multi)grafu

Graf G będzie składał się z 4 części: R, H, S_0, S_1

Z każdego wierzchołka wewnętrznego H wychodzi $O(p^t)$ krawędzi.

W H jest $\Theta(t^d)$ wierzchołków oraz $\Theta(t^d \cdot p^t)$ krawędzi.



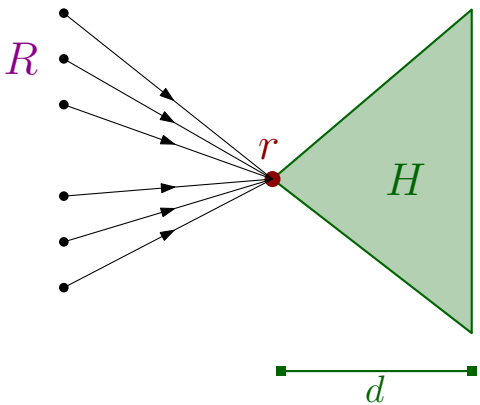
Konstrukcja (multi)grafu

Graf G będzie składał się z 4 części: R, H, S_0, S_1

Podgraf R to $n/4$ wierzchołków. Z każdego z nich wychodzi jedna krawędź do r.

Z każdego wierzchołka wewnętrznego H wychodzi $O(p^t)$ krawędzi.

W H jest $\Theta(t^d)$ wierzchołków oraz $\Theta(t^d \cdot p^t)$ krawędzi.



Konstrukcja (multi)grafu

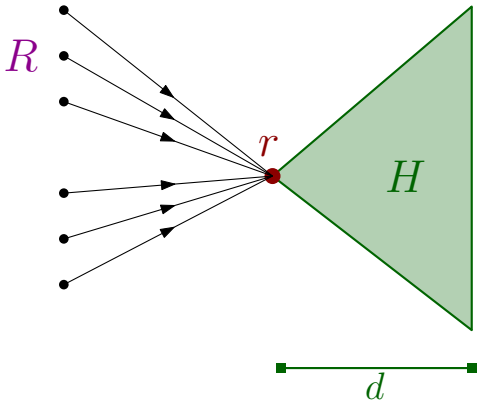
Graf G będzie składał się z 4 części: R, H, S_0, S_1

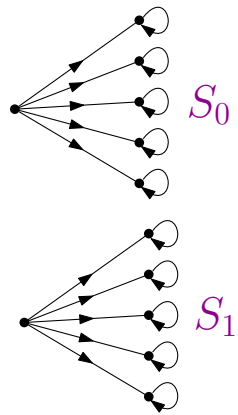
Podgraf R to $n/4$ wierzchołków. Z każdego z nich wychodzi jedna krawędź do r.

S_0 i S_1 to gwiazdy o $n/4$ wierzchołkach, których centra to c_0 i c_1, a liście mają pętle do samych siebie.

Z każdego wierzchołka wewnętrznego H wychodzi $O(p^t)$ krawędzi.

W H jest $\Theta(t^d)$ wierzchołków oraz $\Theta(t^d \cdot p^t)$ krawędzi.





Konstrukcja (multi)grafu

Graf G będzie składał się z 4 części: R, H, S_0, S_1

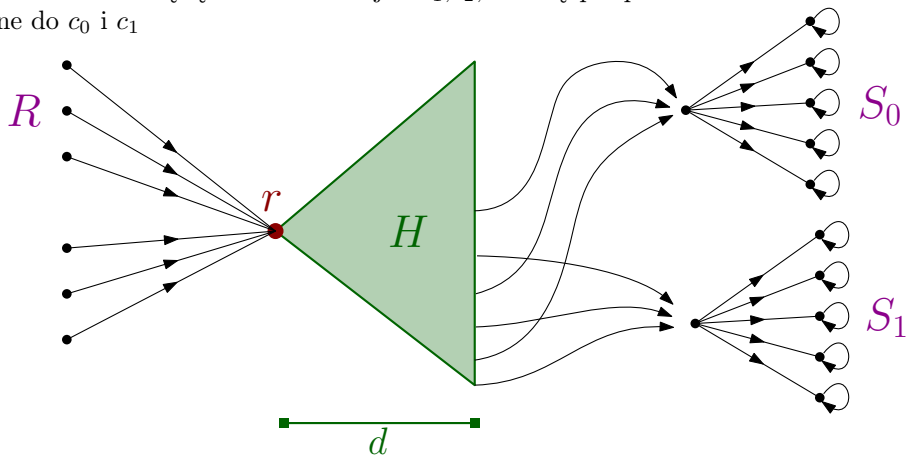
Podgraf R to $n/4$ wierzchołków. Z każdego z nich wychodzi jedna krawędź do r.

S_0 i S_1 to gwiazdy o $n/4$ wierzchołkach, których centra to c_0 i c_1, a liście mają pętle do samych siebie.

Liście drzewa H będą nazwane kolejno $l_1, l_2, ...$ i są podpinane na zmiane do c_0 i c_1

Z każdego wierzchołka wewnętrznego H wychodzi $O(p^t)$ krawędzi.

W H jest $\Theta(t^d)$ wierzchołków oraz $\Theta(t^d \cdot p^t)$ krawędzi.



Konstrukcja (multi)grafu

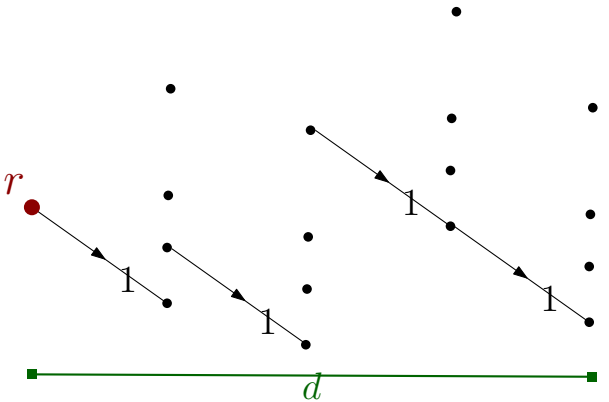
Konstrukcja (multi)grafu

Ciąg wstawień będzie odbywał się na podgrafie H.

Konstrukcja (multi)grafu

Ciąg wstawień będzie odbywał się na podgrafie H.

Początkowo tylko dzieci z indeksem 0 będą podpięte do rodzica.

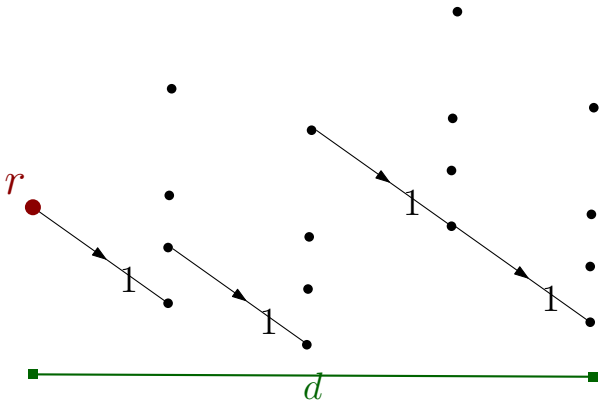


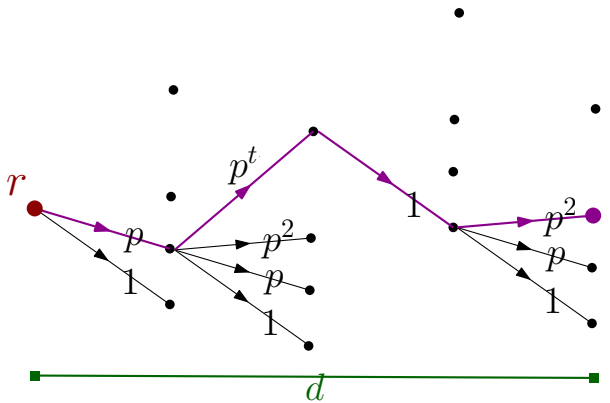
Konstrukcja (multi)grafu

Ciąg wstawień będzie odbywał się na podgrafie H.

Początkowo tylko dzieci z indeksem 0 będą podpięte do rodzica.

Zgodnie z porządkiem pre-order będziemy chodzić po drzewie i podpinać wszystkie krawędzie od rodzica do aktualnego wierzchołka





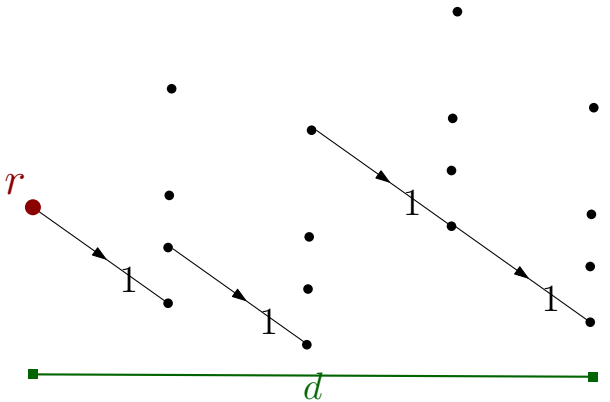
Konstrukcja (multi)grafu

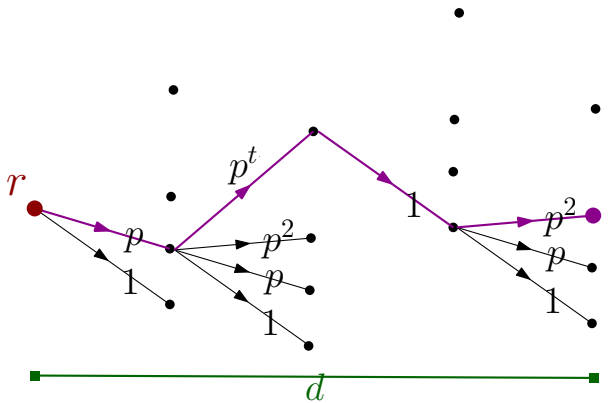
Ciąg wstawień będzie odbywał się na podgrafie H.

Początkowo tylko dzieci z indeksem 0 będą podpięte do rodzica.

Zgodnie z porządkiem pre-order będziemy chodzić po drzewie i podpinać wszystkie krawędzie od rodzica do aktualnego wierzchołka

Za każdym razem gdy dojdziemy do liścia l_i połączymy go ze wszystkimi wierzchołkami z R





Plan i potrzebne lematy

Plan i potrzebne lematy

Intuicja: Łącząc dany liść z grafem R przekażemy mu znaczną część masy prawdopodobieństwa (duża liczba krawędzi prowadzi do nowo połączonego liścia), co wpłynie na wartość w liściach jednej z gwiazd S.

Plan i potrzebne lematy

Intuicja: Łącząc dany liść z grafem R przekażemy mu znaczną część masy prawdopodobieństwa (duża liczba krawędzi prowadzi do nowo połączonego liścia), co wpłynie na wartość w liściach jednej z gwiazd S.

Lemat 1(3.2) *Consider the graph G^τ obtained right after inserting all edges on the path from R to ℓ_i. Let m_i be the probability mass that reaches ℓ_i from R in G^τ. Then $m_i \geq (1-\epsilon)^{2d+2}/4$.*

Moreover, out of the probability mass that reaches the leaves of H from R, at least $(1-1/p)^d$ fraction reaches ℓ_i.

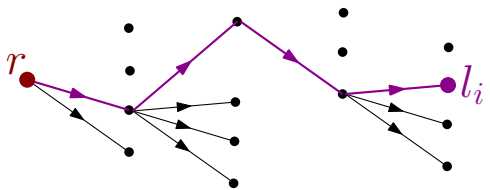
Plan i potrzebne lematy

Intuicja: Łącząc dany liść z grafem R przekażemy mu znaczną część masy prawdopodobieństwa (duża liczba krawędzi prowadzi do nowo połączonego liścia), co wpłynie na wartość w liściach jednej z gwiazd S.

Lemat 1(3.2) *Consider the graph G^τ obtained right after inserting all edges on the path from R to ℓ_i. Let m_i be the probability mass that reaches ℓ_i from R in G^τ. Then $m_i \geq (1-\epsilon)^{2d+2}/4$.*

Moreover, out of the probability mass that reaches the leaves of H from R, at least $(1-1/p)^d$ fraction reaches ℓ_i.

Docierając do wierzchołka l_i wszytkie krawędzie z lewej strony ścieżki są wstawione, a wszystkie z prawej puste.



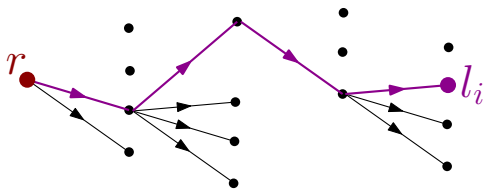
Plan i potrzebne lematy

Intuicja: Łącząc dany liść z grafem R przekażemy mu znaczną część masy prawdopodobieństwa (duża liczba krawędzi prowadzi do nowo połączonego liścia), co wpłynie na wartość w liściach jednej z gwiazd S.

Lemat 1(3.2) *Consider the graph G^τ obtained right after inserting all edges on the path from R to ℓ_i. Let m_i be the probability mass that reaches ℓ_i from R in G^τ. Then $m_i \geq (1-\epsilon)^{2d+2}/4$.*

Moreover, out of the probability mass that reaches the leaves of H from R, at least $(1-1/p)^d$ fraction reaches ℓ_i.

Docierając do wierzchołka l_i wszytkie krawędzie z lewej strony ścieżki są wstawione, a wszystkie z prawej puste.



Dla każdego wierzchołka na ścieżce do l_i możemy obliczyć jaką część krawędzi kieruje w stronę l_i:

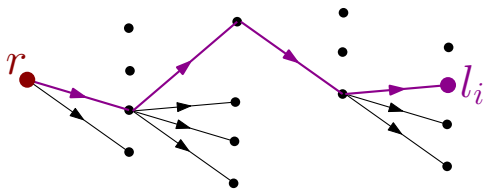
Plan i potrzebne lematy

Intuicja: Łącząc dany liść z grafem R przekażemy mu znaczną część masy prawdopodobieństwa (duża liczba krawędzi prowadzi do nowo połączonego liścia), co wpłynie na wartość w liściach jednej z gwiazd S.

Lemat 1(3.2) *Consider the graph G^τ obtained right after inserting all edges on the path from R to ℓ_i. Let m_i be the probability mass that reaches ℓ_i from R in G^τ. Then $m_i \geq (1-\epsilon)^{2d+2}/4$.*

Moreover, out of the probability mass that reaches the leaves of H from R, at least $(1-1/p)^d$ fraction reaches ℓ_i.

Docierając do wierzchołka l_i wszytkie krawędzie z lewej strony ścieżki są wstawione, a wszystkie z prawej puste.



Dla każdego wierzchołka na ścieżce do l_i możemy obliczyć jaką część krawędzi kieruje w stronę l_i:

$$p^{j-1} / \left(\sum_{k=0}^{j-1} p^k \right) = p^{j-1} \cdot \frac{p-1}{p^j - 1} \geq p^{j-1} \cdot \frac{p-1}{p^j} = 1 - 1/p.$$

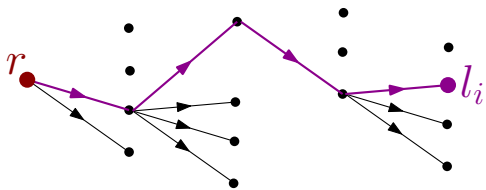
Plan i potrzebne lematy

Intuicja: Łącząc dany liść z grafem R przekażemy mu znaczną część masy prawdopodobieństwa (duża liczba krawędzi prowadzi do nowo połączonego liścia), co wpłynie na wartość w liściach jednej z gwiazd S.

Lemat 1(3.2) *Consider the graph G^τ obtained right after inserting all edges on the path from R to ℓ_i. Let m_i be the probability mass that reaches ℓ_i from R in G^τ. Then $m_i \geq (1-\epsilon)^{2d+2}/4$.*

Moreover, out of the probability mass that reaches the leaves of H from R, at least $(1-1/p)^d$ fraction reaches ℓ_i.

Docierając do wierzchołka l_i wszytkie krawędzie z lewej strony ścieżki są wstawione, a wszystkie z prawej puste.



Dla każdego wierzchołka na ścieżce do l_i możemy obliczyć jaką część krawędzi kieruje w stronę l_i:

$$p^{j-1}/\left(\sum_{k=0}^{j-1} p^k\right) = p^{j-1}\cdot\frac{p-1}{p^j-1} \geq p^{j-1}\cdot\frac{p-1}{p^j} = 1-1/p.$$

Ścieżka od R do l_i ma długość $d+1$. Na każdym jej kroku $(1-\epsilon)$ masy jest przekazywane dalej, z czego przynajmniej $(1-1/p) = (1-\epsilon)$ przekazywane jest w stronę l_i. R generuje masę $1/4$ zatem teza zachodzi. □

Plan i potrzebne lematy

Plan i potrzebne lematy

Lemat 2(3.3) *Consider four vectors* $v^1, \tilde{v}^1, v^2, \tilde{v}^2 \in \mathbb{R}^n$, *such that* $\|v^1 - \tilde{v}^1\|_1 \leq \alpha$, $\|v^2 - \tilde{v}^2\|_1 \leq \alpha$ *and* v^1 *and* v^2 *differ by at least* $100 \cdot \alpha/n$ *on at least* $n/4$ *coordinates. Then* $\tilde{v}^1$ *and* $\tilde{v}^2$ *differ on* $\Omega(n)$ *coordinates.*

Plan i potrzebne lematy

Lemat 2(3.3) *Consider four vectors* $v^1, \tilde{v}^1, v^2, \tilde{v}^2 \in \mathbb{R}^n$, *such that* $\|v^1 - \tilde{v}^1\|_1 \leq \alpha$, $\|v^2 - \tilde{v}^2\|_1 \leq \alpha$ *and* v^1 *and* v^2 *differ by at least* $100 \cdot \alpha/n$ *on at least* $n/4$ *coordinates. Then* $\tilde{v}^1$ *and* $\tilde{v}^2$ *differ on* $\Omega(n)$ *coordinates.*

Dowód nie wprost. Zakładamy, że $\tilde{v^1}$ i $\tilde{v^2}$ różnią się na najwyżej $n/1000$ miejscach (promil).

Plan i potrzebne lematy

Lemat 2(3.3) *Consider four vectors* $v^1, \tilde{v}^1, v^2, \tilde{v}^2 \in \mathbb{R}^n$, *such that* $\|v^1 - \tilde{v}^1\|_1 \leq \alpha$, $\|v^2 - \tilde{v}^2\|_1 \leq \alpha$ *and* v^1 *and* v^2 *differ by at least* $100 \cdot \alpha/n$ *on at least* $n/4$ *coordinates. Then* $\tilde{v}^1$ *and* $\tilde{v}^2$ *differ on* $\Omega(n)$ *coordinates.*

Dowód nie wprost. Zakładamy, że $\tilde{v^1}$ i $\tilde{v^2}$ różnią się na najwyżej $n/1000$ miejscach (promil).

Stworzymy zbiór I z takich współrzędnych, które spełnią wszystkie warunki:

1) $\tilde{v^1}$ i $\tilde{v^2}$ są takie same (przynajmniej $0.999n$)
2) v^1 i v^2 różne o przynajmniej $100\alpha/n$ (przynajmniej $n/4$)
3) v^1 i $\tilde{v^1}$ różne o najwyżej $10\alpha/n$ (przynajmniej $0.9n$)
4) v^2 i $\tilde{v^2}$ różne o najwyżej $10\alpha/n$ (przynajmniej $0.9n$)

Plan i potrzebne lematy

Lemat 2(3.3) *Consider four vectors* $v^1, \tilde{v}^1, v^2, \tilde{v}^2 \in \mathbb{R}^n$, *such that* $\|v^1 - \tilde{v}^1\|_1 \leq \alpha$, $\|v^2 - \tilde{v}^2\|_1 \leq \alpha$ *and* v^1 *and* v^2 *differ by at least* $100 \cdot \alpha/n$ *on at least* $n/4$ *coordinates. Then* $\tilde{v}^1$ *and* $\tilde{v}^2$ *differ on* $\Omega(n)$ *coordinates.*

Dowód nie wprost. Zakładamy, że $\tilde{v^1}$ i $\tilde{v^2}$ różnią się na najwyżej $n/1000$ miejscach (promil).

Stworzymy zbiór I z takich współrzędnych, które spełnią wszystkie warunki:

1) $\tilde{v^1}$ i $\tilde{v^2}$ są takie same (przynajmniej $0.999n$)
2) v^1 i v^2 różne o przynajmniej $100\alpha/n$ (przynajmniej $n/4$)
3) v^1 i $\tilde{v^1}$ różne o najwyżej $10\alpha/n$ (przynajmniej $0.9n$)
4) v^2 i $\tilde{v^2}$ różne o najwyżej $10\alpha/n$ (przynajmniej $0.9n$)

Zbiór I jest niepusty.
Możemy zauważyć, że:

$$|\tilde{v^1} - \tilde{v^2}| \geq |v^1 - v^2| - |v^1 - \tilde{v^1}| - |v^2 - \tilde{v^2}|$$

Plan i potrzebne lematy

Lemat 2(3.3) *Consider four vectors* $v^1, \tilde{v}^1, v^2, \tilde{v}^2 \in \mathbb{R}^n$, *such that* $\|v^1 - \tilde{v}^1\|_1 \leq \alpha$, $\|v^2 - \tilde{v}^2\|_1 \leq \alpha$ *and* v^1 *and* v^2 *differ by at least* $100 \cdot \alpha/n$ *on at least* $n/4$ *coordinates. Then* $\tilde{v}^1$ *and* $\tilde{v}^2$ *differ on* $\Omega(n)$ *coordinates.*

Dowód nie wprost. Zakładamy, że $\tilde{v^1}$ i $\tilde{v^2}$ różnią się na najwyżej $n/1000$ miejscach (promil).

Stworzymy zbiór I z takich współrzędnych, które spełnią wszystkie warunki:

1) $\tilde{v^1}$ i $\tilde{v^2}$ są takie same (przynajmniej $0.999n$)
2) v^1 i v^2 różne o przynajmniej $100\alpha/n$ (przynajmniej $n/4$)
3) v^1 i $\tilde{v^1}$ różne o najwyżej $10\alpha/n$ (przynajmniej $0.9n$)
4) v^2 i $\tilde{v^2}$ różne o najwyżej $10\alpha/n$ (przynajmniej $0.9n$)

Zbiór I jest niepusty.
Możemy zauważyć, że:

$$\begin{aligned} |\tilde{v^1} - \tilde{v^2}| &\geq |v^1 - v^2| - |v^1 - \tilde{v^1}| - |v^2 - \tilde{v^2}| \\ &\geq 100\alpha/n - 10\alpha/n - 10\alpha/n \\ &= 80\alpha/n \end{aligned}$$

Dostajemy sprzeczność z 1). □

Ustalenie wartosci d i t

Ustalenie wartosci d i t

ϵ jest ustalone
$p = 1/\epsilon$

Ustalamy, że:

$$t = 1/2 \cdot \log_p n$$

$$d = \frac{\log(101\alpha)}{2\log(1-\epsilon)} - 2 \geq 1 \qquad d = \Theta(\log(1/\alpha))$$

Ustalenie wartosci d i t

ϵ jest ustalone
$p = 1/\epsilon$

Ustalamy, że:

$$t = 1/2 \cdot \log_p n$$

$$d = \frac{\log(101\alpha)}{2\log(1-\epsilon)} - 2 \geq 1 \qquad d = \Theta(\log(1/\alpha))$$

Liczba liści podgrafu H:

$$t^d = (1/2 \cdot \log_{1/\epsilon} n)^d = \left(\frac{\log n}{2 \log 1/\epsilon}\right)^{\Theta(\log(1/\alpha))} = \log^{\Theta(\log(1/\alpha))} n = (1/\alpha)^{\Theta(\log\log n)}$$

Ustalenie wartosci d i t

ϵ jest ustalone
$p = 1/\epsilon$

Ustalamy, że:

$$t = 1/2 \cdot \log_p n$$

$$d = \frac{\log(101\alpha)}{2\log(1-\epsilon)} - 2 \geq 1 \qquad d = \Theta(\log(1/\alpha))$$

Liczba liści podgrafu H:

$$t^d = (1/2 \cdot \log_{1/\epsilon} n)^d = \left(\frac{\log n}{2 \log 1/\epsilon}\right)^{\Theta(\log(1/\alpha))} = \log^{\Theta(\log(1/\alpha))} n = (1/\alpha)^{\Theta(\log\log n)}$$

H ma $\Theta(t^d) = o(n)$ wierzchołków

Ustalenie wartosci d i t

ϵ jest ustalone
$p = 1/\epsilon$

Ustalamy, że:

$$t = 1/2 \cdot \log_p n$$

$$d = \frac{\log(101\alpha)}{2\log(1-\epsilon)} - 2 \geq 1 \qquad d = \Theta(\log(1/\alpha))$$

Liczba liści podgrafu H:

$$t^d = (1/2 \cdot \log_{1/\epsilon} n)^d = \left(\frac{\log n}{2 \log 1/\epsilon}\right)^{\Theta(\log(1/\alpha))} = \log^{\Theta(\log(1/\alpha))} n = (1/\alpha)^{\Theta(\log\log n)}$$

H ma $\Theta(t^d) = o(n)$ wierzchołków

Liczba dzieci wierzchołka H

$$\Theta(p^t) = \Theta(p^{1/2 \cdot \log_p n}) = \Theta(n^{1/2})$$

Ustalenie wartosci d i t

ϵ jest ustalone
$p = 1/\epsilon$

Ustalamy, że:

$$t = 1/2 \cdot \log_p n$$

$$d = \frac{\log(101\alpha)}{2\log(1-\epsilon)} - 2 \geq 1 \qquad d = \Theta(\log(1/\alpha))$$

Liczba liści podgrafu H:

$$t^d = (1/2 \cdot \log_{1/\epsilon} n)^d = \left(\frac{\log n}{2 \log 1/\epsilon}\right)^{\Theta(\log(1/\alpha))} = \log^{\Theta(\log(1/\alpha))} n = (1/\alpha)^{\Theta(\log\log n)}$$

H ma $\Theta(t^d) = o(n)$ wierzchołków

Liczba dzieci wierzchołka H

$$\Theta(p^t) = \Theta(p^{1/2 \cdot \log_p n}) = \Theta(n^{1/2})$$

Graf G ma n wierzchołków i $O(n)$ krawędzi

Liczba krawędzi H

$$(1/\alpha)^{\Theta(\log\log n)} \cdot n^{1/2} = n^{o(1)} \cdot \Theta(n^{1/2}) = O(n)$$

Końcówka dowodu Tw1

Końcówka dowodu Tw1

Ustalmy, że właśnie l_j został osiągnięty. π^b będzie wartością sprzed chwili, a π^a po osiągnięciu l_j.

Końcówka dowodu Tw1

Ustalmy, że właśnie l_j został osiągnięty. π^b będzie wartością sprzed chwili, a π^a po osiągnięciu l_j.

Masa prawdopodobieństwa trafia do $\pi^b_{l_j}$ tylko z grafu H.

$$\pi^b_{\ell_j} \leq (d+1)/n = \Theta(\log(1/\alpha))/n = o(\log n)/n.$$

Końcówka dowodu Tw1

Ustalmy, że właśnie l_j został osiągnięty. π^b będzie wartością sprzed chwili, a π^a po osiągnięciu l_j.

Masa prawdopodobieństwa trafia do $\pi^b_{l_j}$ tylko z grafu H.

$$\pi^b_{\ell_j} \leq (d+1)/n = \Theta(\log(1/\alpha))/n = o(\log n)/n.$$

Z lematu 1(3.2):

$$\pi^a_{\ell_j} \geq \epsilon \cdot (1-\epsilon)^{2d+2}/4 = \epsilon \cdot (1-\epsilon)^{\frac{\log(101\alpha)}{\log(1-\epsilon)}-2}/4 = 101 \cdot \epsilon \cdot \alpha \cdot (1-\epsilon)^{-2}$$

$$\pi^a_{\ell_j} - \pi^b_{\ell_j} \geq 100 \cdot \epsilon \cdot \alpha (1-\epsilon)^{-2}$$

Końcówka dowodu Tw1

Ustalmy, że właśnie l_j został osiągnięty. π^b będzie wartością sprzed chwili, a π^a po osiągnięciu l_j.

Masa prawdopodobieństwa trafia do $\pi^b_{l_j}$ tylko z grafu H.

$$\pi^b_{\ell_j} \leq (d+1)/n = \Theta(\log(1/\alpha))/n = o(\log n)/n.$$

Z lematu 1(3.2):

$$\pi^a_{\ell_j} \geq \epsilon \cdot (1-\epsilon)^{2d+2}/4 = \epsilon \cdot (1-\epsilon)^{\frac{\log(101\alpha)}{\log(1-\epsilon)}-2}/4 = 101 \cdot \epsilon \cdot \alpha \cdot (1-\epsilon)^{-2}$$

$$\pi^a_{\ell_j} - \pi^b_{\ell_j} \geq 100 \cdot \epsilon \cdot \alpha (1-\epsilon)^{-2}$$

Zatem każdy z liści jednej z gwiazd otrzyma przynajmniej tyle nowej masy prawdopodobieństwa: $s = |S| = n/4$

$$100 \cdot \alpha \cdot (1-\epsilon)^{-2}/4 \cdot (1-\epsilon)^2/s = 100 \cdot \alpha/(4s) = 100 \cdot \alpha/n$$

Końcówka dowodu Tw1

Ustalmy, że właśnie l_j został osiągnięty. π^b będzie wartością sprzed chwili, a π^a po osiągnięciu l_j.

Masa prawdopodobieństwa trafia do $\pi^b_{l_j}$ tylko z grafu H.

$$\pi^b_{\ell_j} \leq (d+1)/n = \Theta(\log(1/\alpha))/n = o(\log n)/n.$$

Z lematu 1(3.2):

$$\pi^a_{\ell_j} \geq \epsilon \cdot (1-\epsilon)^{2d+2}/4 = \epsilon \cdot (1-\epsilon)^{\frac{\log(101\alpha)}{\log(1-\epsilon)}-2}/4 = 101 \cdot \epsilon \cdot \alpha \cdot (1-\epsilon)^{-2}$$

$$\pi^a_{\ell_j} - \pi^b_{\ell_j} \geq 100 \cdot \epsilon \cdot \alpha (1-\epsilon)^{-2}$$

Zatem każdy z liści jednej z gwiazd otrzyma przynajmniej tyle nowej masy prawdopodobieństwa: $s = |S| = n/4$

$$100 \cdot \alpha \cdot (1-\epsilon)^{-2}/4 \cdot (1-\epsilon)^2/s = 100 \cdot \alpha/(4s) = 100 \cdot \alpha/n$$

Na podstawie lematu 2(3.3) wiemy, że wektor PageRank będzie musiał być updateowany na $\Omega(n)$ żeby zachować α addytywną aproksymację.

Końcówka dowodu Tw1

Liczba liści H to $(1/\alpha)^{\Theta(\log\log n)}$

Końcówka dowodu Tw1

Liczba liści H to $(1/\alpha)^{\Theta(\log\log n)}$

Po osiągnięciu każdego z nich należy zupdateować $\Omega(n)$ wierzchołków, co prowadzi nas do tezy.

Theorem 1.1. *Fix $\epsilon \in (0.01, 0.99)$. For any sufficiently large $n \geq 1$ and any α such that $1/\alpha = n^{o(1/\log\log n)}$, any algorithm which explicitly maintains α-additive approximation of PageRank must run in $n \cdot (1/\alpha)^{\Omega(\log\log n)}$ total time.*

Twierdzenie 3

Theorem 1.3. *There exists a sequence of $\Theta(n)$ edge insertions applied to an initially empty graph on n vertices for which the following holds. For any constant $\delta > 0$, any algorithm that maintains a vector $\tilde{\pi} \in \mathbb{R}^n$ such that $(1/2)\pi_v < \tilde{\pi}_v \leq 2\pi_v$ at all time steps, must take time $\Omega(n^{2-\delta})$ to process the sequence. In particular, the amortized update time of any such algorithm is $\Omega(n^{1-\delta})$.*

Twierdzenie 3

Theorem 1.3. *There exists a sequence of $\Theta(n)$ edge insertions applied to an initially empty graph on n vertices for which the following holds. For any constant $\delta > 0$, any algorithm that maintains a vector $\tilde{\pi} \in \mathbb{R}^n$ such that $(1/2)\pi_v < \tilde{\pi}_v \leq 2\pi_v$ at all time steps, must take time $\Omega(n^{2-\delta})$ to process the sequence. In particular, the amortized update time of any such algorithm is $\Omega(n^{1-\delta})$.*

Dla tej samej konstrukcji ustalamy odpowiednie parametry:

$$t = \delta/2 \log n / \log\log n$$
$$d = \log_t(n^{1-2\delta}) = \Theta(\log n / \log\log n)$$

Twierdzenie 3

Theorem 1.3. *There exists a sequence of $\Theta(n)$ edge insertions applied to an initially empty graph on n vertices for which the following holds. For any constant $\delta > 0$, any algorithm that maintains a vector $\tilde{\pi} \in \mathbb{R}^n$ such that $(1/2)\pi_v < \tilde{\pi}_v \leq 2\pi_v$ at all time steps, must take time $\Omega(n^{2-\delta})$ to process the sequence. In particular, the amortized update time of any such algorithm is $\Omega(n^{1-\delta})$.*

Dla tej samej konstrukcji ustalamy odpowiednie parametry:

$$t = \delta/2 \log n / \log\log n$$
$$d = \log_t(n^{1-2\delta}) = \Theta(\log n / \log\log n)$$

Liczba krawędzi z i-tego dziecka $\quad (\log^2 n)^i \quad (p = \log^2 n)$

Twierdzenie 3

Theorem 1.3. *There exists a sequence of $\Theta(n)$ edge insertions applied to an initially empty graph on n vertices for which the following holds. For any constant $\delta > 0$, any algorithm that maintains a vector $\tilde{\pi} \in \mathbb{R}^n$ such that $(1/2)\pi_v < \tilde{\pi}_v \leq 2\pi_v$ at all time steps, must take time $\Omega(n^{2-\delta})$ to process the sequence. In particular, the amortized update time of any such algorithm is $\Omega(n^{1-\delta})$.*

Dla tej samej konstrukcji ustalamy odpowiednie parametry:

$$t = \delta/2 \log n / \log\log n$$
$$d = \log_t(n^{1-2\delta}) = \Theta(\log n / \log\log n)$$

Liczba krawędzi z i-tego dziecka $\quad (\log^2 n)^i \quad (p = \log^2 n)$

Podgraf H otrzyma $n^{1-2\delta}$ wierzchołków

Z lematu 1(3.2) wynika, że w liściu jest następująca część masy z R

$$(1 - 1/p)^d = (1 - 1/\log^2 n)^{\Theta(\log n/\log\log n)} \geq 1 - 1/\log n.$$

Twierdzenie 3

Theorem 1.3. *There exists a sequence of $\Theta(n)$ edge insertions applied to an initially empty graph on n vertices for which the following holds. For any constant $\delta > 0$, any algorithm that maintains a vector $\tilde{\pi} \in \mathbb{R}^n$ such that $(1/2)\pi_v < \tilde{\pi}_v \leq 2\pi_v$ at all time steps, must take time $\Omega(n^{2-\delta})$ to process the sequence. In particular, the amortized update time of any such algorithm is $\Omega(n^{1-\delta})$.*

Dla tej samej konstrukcji ustalamy odpowiednie parametry:

$$t = \delta/2 \log n / \log\log n$$
$$d = \log_t(n^{1-2\delta}) = \Theta(\log n / \log\log n)$$

Liczba krawędzi z i-tego dziecka $\quad (\log^2 n)^i \quad (p = \log^2 n)$

Podgraf H otrzyma $n^{1-2\delta}$ wierzchołków

Z lematu 1(3.2) wynika, że w liściu jest następująca część masy z R

$$(1-1/p)^d = (1-1/\log^2 n)^{\Theta(\log n/\log\log n)} \geq 1 - 1/\log n.$$

Stosunek mas w obu gwiazdach wyniesie

$$\frac{1-1/\log n}{1/\log n} = \Theta(\log n)$$

Twierdzenie 3

Theorem 1.3. *There exists a sequence of $\Theta(n)$ edge insertions applied to an initially empty graph on n vertices for which the following holds. For any constant $\delta > 0$, any algorithm that maintains a vector $\tilde{\pi} \in \mathbb{R}^n$ such that $(1/2)\pi_v < \tilde{\pi}_v \leq 2\pi_v$ at all time steps, must take time $\Omega(n^{2-\delta})$ to process the sequence. In particular, the amortized update time of any such algorithm is $\Omega(n^{1-\delta})$.*

Dla tej samej konstrukcji ustalamy odpowiednie parametry:

$$t = \delta/2 \log n / \log\log n$$
$$d = \log_t(n^{1-2\delta}) = \Theta(\log n / \log\log n)$$

Liczba liści gwiazd:

$$n^{1-2\delta}$$

Liczba krawędzi z i-tego dziecka $\quad (\log^2 n)^i \quad (p = \log^2 n)$

Podgraf H otrzyma $n^{1-2\delta}$ wierzchołków

Z lematu 1(3.2) wynika, że w liściu jest następująca część masy z R

$$(1 - 1/p)^d = (1 - 1/\log^2 n)^{\Theta(\log n/\log\log n)} \geq 1 - 1/\log n.$$

Stosunek mas w obu gwiazdach wyniesie

$$\frac{1 - 1/\log n}{1/\log n} = \Theta(\log n)$$

Twierdzenie 3

Theorem 1.3. *There exists a sequence of $\Theta(n)$ edge insertions applied to an initially empty graph on n vertices for which the following holds. For any constant $\delta > 0$, any algorithm that maintains a vector $\tilde{\pi} \in \mathbb{R}^n$ such that $(1/2)\pi_v < \tilde{\pi}_v \leq 2\pi_v$ at all time steps, must take time $\Omega(n^{2-\delta})$ to process the sequence. In particular, the amortized update time of any such algorithm is $\Omega(n^{1-\delta})$.*

Dla tej samej konstrukcji ustalamy odpowiednie parametry:

$$t = \delta/2 \log n / \log\log n$$
$$d = \log_t(n^{1-2\delta}) = \Theta(\log n / \log\log n)$$

Liczba krawędzi z i-tego dziecka $\quad (\log^2 n)^i \quad (p = \log^2 n)$

Podgraf H otrzyma $n^{1-2\delta}$ wierzchołków

Z lematu 1(3.2) wynika, że w liściu jest następująca część masy z R

$$(1-1/p)^d = (1-1/\log^2 n)^{\Theta(\log n/\log\log n)} \geq 1 - 1/\log n.$$

Stosunek mas w obu gwiazdach wyniesie

$$\frac{1-1/\log n}{1/\log n} = \Theta(\log n)$$

Liczba liści gwiazd:

$$n^{1-2\delta}$$

Zatem przy każdym z $\Omega(n^{1-\delta})$ podłączeń liści, potrzeba updateować $\Omega(n^{1-2\delta})$ wierzchołków z jednej z gwiazd.

Koszt ciągu operacji będzie rzędu $\Omega(n^{2-3\delta})$

□

Algorytm dynamiczny

1: $W \leftarrow \emptyset$
2: Let ℓ be the length of longest generated walk.
3: **for** $t = 1 \dots \ell$ **do**
4: Sample each walk from $S_{u,t}$ with probability $1/d_u$ in the following way. First, select an integer $r_{u,t}$ from the binomial distribution with parameters $|S_{u,t}|$ and $1/d_u$. Second, select $r_{u,t}$ integers uniformly at random and without repetition from $[1, |S_{u,t}|]$. Then, for each of those integers i select the i-th walk from $S_{u,t}$. If e is an undirected edge, apply the same steps for $S_{v,t}$.
5: For each walk w selected in the last step such that $w \notin W$, add w to W and label w by t.
6: **end for**
7: **for** each $w \in W$ **do**
8: Let j be the label remembered for w on Line 5.
9: Generate walk w' with the following properties:

- The walks w and w' have the same length.
- The vertex-prefixes of length j of w and w' are the same.
- After that prefix, if w has more than j vertices, w' walks along e.
- The remaining edges of w' are chosen randomly, i.e., the rest of w' is a newly generated random walk.

10: Update the data structures by removing w and inserting w'.
11: **end for**

Algorytm dynamiczny

1: $W \leftarrow \emptyset$
2: Let ℓ be the length of longest generated walk.
3: **for** $t = 1 \dots \ell$ **do**
4: Sample each walk from $S_{u,t}$ with probability $1/d_u$ in the following way. First, select an integer $r_{u,t}$ from the binomial distribution with parameters $|S_{u,t}|$ and $1/d_u$. Second, select $r_{u,t}$ integers uniformly at random and without repetition from $[1, |S_{u,t}|]$. Then, for each of those integers i select the i-th walk from $S_{u,t}$. If e is an undirected edge, apply the same steps for $S_{v,t}$.
5: For each walk w selected in the last step such that $w \notin W$, add w to W and label w by t.
6: **end for**
7: **for** each $w \in W$ **do**
8: Let j be the label remembered for w on Line 5.
9: Generate walk w' with the following properties:

- The walks w and w' have the same length.
- The vertex-prefixes of length j of w and w' are the same.
- After that prefix, if w has more than j vertices, w' walks along e.
- The remaining edges of w' are chosen randomly, i.e., the rest of w' is a newly generated random walk.

10: Update the data structures by removing w and inserting w'.
11: **end for**

W - zbiór ścieżek, które poprawimy

$S_{u,t}$ - drzewo binarne z krawędziami, których t-ty wierzchołek to u.

Wstawienie krawędzi (u, v)

- rejection sampling
- ponowne generowanie odrzuconych spacerów

Algorytm dynamiczny

1: Let $W_e \subseteq W$ be the list of walks passing through e.
2: **for** $w \in W_e$ **do**
3: Let w_p be the longest prefix of w not containing e.
4: Let w' be a walk of length $|w|$ such that w' has w_p as its prefix, and the remainder of w' is a random walk.
5: To update W, remove w from W and the corresponding data structures, and insert w'.
6: **end for**

Dla każdej ścieżki z $W_{(u,v)}$ generujemy w' tak aby:

- miały ten sam prefiks do wieżchołka u
- dalsza część w' była spacerem losowym

Twierdzenie 2

Theorem 1.2. *For any $\epsilon \in (0,1)$, there is an algorithm that with high probability explicitly maintains an α additive approximation of PageRank of any graph G in either incremental or decremental setting. The algorithm processes the entire sequence of updates in $O(m) + n \cdot (1/\alpha)^{O_\epsilon(\log \log n)}$ total time and works correctly against an oblivious adversary.*

Twierdzenie 2

Theorem 1.2. *For any $\epsilon \in (0,1)$, there is an algorithm that with high probability explicitly maintains an α additive approximation of PageRank of any graph G in either incremental or decremental setting. The algorithm processes the entire sequence of updates in $O(m) + n \cdot (1/\alpha)^{O_\epsilon(\log \log n)}$ total time and works correctly against an oblivious adversary.*

Najpierw pokazane jest, że ograniczając maksymalną długość spacerów l otrzymujemy α addytywną aproksymację.

Lemma 5.1. *Let π be the PageRank of a directed graph G. Then, with high probability, Algorithm 1 for $\ell = \lceil 2/\epsilon \cdot \log(2/(\alpha\epsilon)) \rceil$ outputs a vector π_{ADD} such that $\|\pi - \pi_{ADD}\|_1 \leq 5\frac{\alpha}{1-\epsilon}$.*

Twierdzenie 2

Theorem 1.2. *For any $\epsilon \in (0,1)$, there is an algorithm that with high probability explicitly maintains an α additive approximation of PageRank of any graph G in either incremental or decremental setting. The algorithm processes the entire sequence of updates in $O(m) + n \cdot (1/\alpha)^{O_\epsilon(\log \log n)}$ total time and works correctly against an oblivious adversary.*

Najpierw pokazane jest, że ograniczając maksymalną długość spacerów l otrzymujemy α addytywną aproksymację.

Lemma 5.1. *Let π be the PageRank of a directed graph G. Then, with high probability, Algorithm 1 for $\ell = \lceil 2/\epsilon \cdot \log(2/(\alpha\epsilon)) \rceil$ outputs a vector π_{ADD} such that $\|\pi - \pi_{ADD}\|_1 \leq 5\frac{\alpha}{1-\epsilon}$.*

Kolejny lemat bazuje na obserwacji jak dużo razy możemy wpłynąć na daną krawędź, której wierzchołek bierze udział w dokładaniu kolejnych krawędzi do grafu

Lemma 5.2. *Let G be a directed graph undergoing edge insertions (or deletions). The total number of times a random walk of length ℓ is being regenerated is bounded by $O(\log^\ell n)$ in expectation.*

Twierdzenie 4

Twierdzenie 1.4

Dla dowolnego $\epsilon \in (0,1)$ istnieje algorytm, który z dużym prawdopodobieństwem jawnie utrzymuje $(1+\alpha)$-multiplikatywne przybliżenie PageRanku dowolnego nieskierowanego grafu G w w pełni dynamicznym ustawieniu.
Algorytm przetwarza każdą aktualizację w czasie $O\left(\frac{\log^5 n}{\epsilon^2\alpha^2}\right)$ i działa poprawnie przeciwko przeciwnikowi nieświadomemu.

Dziękujemy