Distributed Selection: A missing Piece of Data Aggregation

Paweł Kacprzak

Instytut Informatyki Uniwersytetu Wrocławskiego

4 grudnia 2012

Plan na dziś

1. Model obliczeń rozproszonych
2. Użyteczny przykład algorytmu rozproszonego
3. Funkcje agregujące
4. Obliczanie 'łatwych' funkcji agregujących
5. 'Trudniejsze' funkcje - poprzednie wyniki; specjalne przypadki
6. Właściwe algorytmy
7. Algorytm RAND
8. Krótko o DET i Lower Bound

Plan na dziś

1. Model obliczeń rozproszonych
2. Użyteczny przykład algorytmu rozproszonego
3. Funkcje agregujące
4. Obliczanie 'łatwych' funkcji agregujących
5. 'Trudniejsze' funkcje - poprzednie wyniki; specjalne przypadki
6. Właściwe algorytmy
7. Algorytm RAND
8. Krótko o DET i Lower Bound

Plan na dziś

1. Model obliczeń rozproszonych
2. Użyteczny przykład algorytmu rozproszonego
3. Funkcje agregujące
4. Obliczanie 'łatwych' funkcji agregujących
5. 'Trudniejsze' funkcje - poprzednie wyniki; specjalne przypadki
6. Właściwe algorytmy
7. Algorytm RAND
8. Krótko o DET i Lower Bound

Plan na dziś

1. Model obliczeń rozproszonych
2. Użyteczny przykład algorytmu rozproszonego
3. Funkcje agregujące
4. Obliczanie 'łatwych' funkcji agregujących
5. 'Trudniejsze' funkcje - poprzednie wyniki; specjalne przypadki
6. Właściwe algorytmy
7. Algorytm RAND
8. Krótko o DET i Lower Bound

Modelujemy jako graf

- spójny graf $G = (V, E)$
- $\{v, u\} \in E$ iff. u może się bezpośrednio komunikować z v
- dwustronne kanały komunikacyjne
- tylko lokalna wiedza
- kiedy algorytm się kończy - specjalny stan procesora; słaby, mocny warunek stopu
- nie ma błędów stopu; nie ma awarii kanałów
- wszystkie procesory działają zgodnie z protokołem

Modelujemy jako graf

- spójny graf $G = (V, E)$
- $\{v, u\} \in E$ iff. u może się bezpośrednio komunikować z v
- dwustronne kanały komunikacyjne
- tylko lokalna wiedza
- kiedy algorytm się kończy - specjalny stan procesora; słaby, mocny warunek stopu
- nie ma błędów stopu; nie ma awarii kanałów
- wszystkie procesory działają zgodnie z protokołem

Modelujemy jako graf

- spójny graf $G = (V, E)$
- $\{v, u\} \in E$ iff. u może się bezpośrednio komunikować z v
- dwustronne kanały komunikacyjne
- tylko lokalna wiedza
- kiedy algorytm się kończy - specjalny stan procesora; słaby, mocny warunek stopu
- nie ma błędów stopu; nie ma awarii kanałów
- wszystkie procesory działają zgodnie z protokołem

Modelujemy jako graf

- spójny graf $G = (V, E)$
- $\{v, u\} \in E$ iff. u może się bezpośrednio komunikować z v
- dwustronne kanały komunikacyjne
- tylko lokalna wiedza
- kiedy algorytm się kończy - specjalny stan procesora; słaby, mocny warunek stopu
- nie ma błędów stopu; nie ma awarii kanałów
- wszystkie procesory działają zgodnie z protokołem

Modelujemy jako graf

- spójny graf $G = (V, E)$
- $\{v, u\} \in E$ iff. u może się bezpośrednio komunikować z v
- dwustronne kanały komunikacyjne
- tylko lokalna wiedza
- kiedy algorytm się kończy - specjalny stan procesora; słaby, mocny warunek stopu
- nie ma błędów stopu; nie ma awarii kanałów
- wszystkie procesory działają zgodnie z protokołem

Modelujemy jako graf

- spójny graf $G = (V, E)$
- $\{v, u\} \in E$ iff. u może się bezpośrednio komunikować z v
- dwustronne kanały komunikacyjne
- tylko lokalna wiedza
- kiedy algorytm się kończy - specjalny stan procesora; słaby, mocny warunek stopu
- nie ma błędów stopu; nie ma awarii kanałów
- wszystkie procesory działają zgodnie z protokołem

Jak mierzyć złożoność?

- złożoność komunikacyjna
- złożoność czasowa? - 2 różne modele

Jak mierzyć złożoność?

- złożoność komunikacyjna
- złożoność czasowa? - 2 różne modele

Jak mierzyć złożoność?

- złożoność komunikacyjna
- złożoność czasowa? - 2 różne modele

Jak mierzyć złożoność?

- złożoność komunikacyjna
- złożoność czasowa? - 2 różne modele

Model synchroniczny

- pojęcie rundy
- złożoność czasowa

Model synchroniczny

- pojęcie rundy
- złożoność czasowa

Model synchroniczny

- pojęcie rundy
- złożoność czasowa

Model asynchroniczny

- nie ma rund
- bardziej realistyczny
- znormalizowana złożoność czasowa
- czasami używa się 'synchronizatorów'

Model asynchroniczny

- nie ma rund
- bardziej realistyczny
- znormalizowana złożoność czasowa
- czasami używa się 'synchronizatorów'

Model asynchroniczny

- nie ma rund
- bardziej realistyczny
- znormalizowana złożoność czasowa
- czasami używa się 'synchronizatorów'

Model asynchroniczny

- nie ma rund
- bardziej realistyczny
- znormalizowana złożoność czasowa
- czasami używa się 'synchronizatorów'

Dodatkowe założenia

- unikalne id
- ograniczony rozmiar komunikatu
- graf ma lidera r

Dodatkowe założenia

- unikalne id
- ograniczony rozmiar komunikatu
- graf ma lidera r

Dodatkowe założenia

- unikalne id
- ograniczony rozmiar komunikatu
- graf ma lidera r

Dodatkowe założenia

- unikalne id
- ograniczony rozmiar komunikatu
- graf ma lidera r

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- $convergecast$ - czas $O(n)$, wiadomości $O(n)$

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- $convergecast$ - czas $O(n)$, wiadomości $O(n)$

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- $convergecast$ - czas $O(n)$, wiadomości $O(n)$

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- $convergecast$ - czas $O(n)$, wiadomości $O(n)$

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- *convergecast* - czas $O(n)$, wiadomości $O(n)$

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- *convergecast* - czas $O(n)$, wiadomości $O(n)$

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- *convergecast* - czas $O(n)$, wiadomości $O(n)$

Drzewo rozpinające grafu ukorzenione w r

1. r 'pyta' wszystkich sąsiadów: 'czy jesteś moim synem w drzewie rozpinającym?'
2. $v \neq r$ odpowiada 'tak' jeśli jest pytany po raz pierwszy i wtedy zadaje pytanie wszystkim sąsiadom oprócz ojca; inaczej odpowiada 'nie'; r zawsze odpowiada 'nie'
3. v kończy algorytm jeśli dostał odpowiedzi od wszystkich sąsiadów, do których wysłał zapytanie.

- zł. czasowa $O(D)$
- zł. komunikacyjna $O(m)$
- tylko lokalne zatrzymanie - jak poprawić?
- $convergecast$ - czas $O(n)$, wiadomości $O(n)$

Funkcje agregujące

Zapominamy na chwilę o modelu rozproszonym.

Podział funkcji agregujących

- distributive - np. max, min, sum, $count$
- algebraic - np. avg, var
- holistic - k^{th} element, mediana

Wierzy się, że kombinacje powyższych klas funkcji mogą wyrazić wszystkie sensowne zapytania agregujące

Funkcje agregujące

Zapominamy na chwilę o modelu rozproszonym.

Podział funkcji agregujących

- distributive - np. max, min, sum, $count$
- algebraic - np. avg, var
- holistic - k^{th} element, mediana

Wierzy się, że kombinacje powyższych klas funkcji mogą wyrazić wszystkie sensowne zapytania agregujące

Funkcje agregujące

Zapominamy na chwilę o modelu rozproszonym.

Podział funkcji agregujących

- distributive - np. max, min, sum, $count$
- algebraic - np. avg, var
- holistic - k^{th} element, mediana

Wierzy się, że kombinacje powyższych klas funkcji mogą wyrazić wszystkie sensowne zapytania agregujące

Funkcje agregujące

Zapominamy na chwilę o modelu rozproszonym.

Podział funkcji agregujących

- distributive - np. max, min, sum, $count$
- algebraic - np. avg, var
- holistic - k^{th} element, mediana

Wierzy się, że kombinacje powyższych klas funkcji mogą wyrazić wszystkie sensowne zapytania agregujące

Funkcje agregujące

Zapominamy na chwilę o modelu rozproszonym.

Podział funkcji agregujących

- distributive - np. max, min, sum, $count$
- algebraic - np. avg, var
- holistic - k^{th} element, mediana

Wierzy się, że kombinacje powyższych klas funkcji mogą wyrazić wszystkie sensowne zapytania agregujące

Funkcje agregujące

Zapominamy na chwilę o modelu rozproszonym.

Podział funkcji agregujących

- distributive - np. max, min, sum, $count$
- algebraic - np. avg, var
- holistic - k^{th} element, mediana

Wierzy się, że kombinacje powyższych klas funkcji mogą wyrazić wszystkie sensowne zapytania agregujące

Jak to zrobić?

Convergecast na ukorzenionym drzewie BFS

- rekurencja na drzewie
- r nakazuje swoim dzieciom obliczenie funkcji w swoich poddrzewach
- wierzchołek wewnętrzny czeka aż wszystkie dzieci mu odpowiedzą, aplikuje funkcje i przesyła do ojca

Analiza

- szybkie
- globalne zatrzymanie po czasie $O(D)$
- problem?

Jak to zrobić?

Convergecast na ukorzenionym drzewie BFS

- rekurencja na drzewie
- r nakazuje swoim dzieciom obliczenie funkcji w swoich poddrzewach
- wierzchołek wewnętrzny czeka aż wszystkie dzieci mu odpowiedzą, aplikuje funkcje i przesyła do ojca

Analiza

- szybkie
- globalne zatrzymanie po czasie $O(D)$
- problem?

Jak to zrobić?

Convergecast na ukorzenionym drzewie BFS

- rekurencja na drzewie
- r nakazuje swoim dzieciom obliczenie funkcji w swoich poddrzewach
- wierzchołek wewnętrzny czeka aż wszystkie dzieci mu odpowiedzą, aplikuje funkcje i przesyła do ojca

Analiza

- szybkie
- globalne zatrzymanie po czasie $O(D)$
- problem?

Jak to zrobić?

Convergecast na ukorzenionym drzewie BFS

- rekurencja na drzewie
- r nakazuje swoim dzieciom obliczenie funkcji w swoich poddrzewach
- wierzchołek wewnętrzny czeka aż wszystkie dzieci mu odpowiedzą, aplikuje funkcje i przesyła do ojca

Analiza

- szybkie
- globalne zatrzymanie po czasie $O(D)$
- problem?

Jak to zrobić?

Convergecast na ukorzenionym drzewie BFS

- rekurencja na drzewie
- r nakazuje swoim dzieciom obliczenie funkcji w swoich poddrzewach
- wierzchołek wewnętrzny czeka aż wszystkie dzieci mu odpowiedzą, aplikuje funkcje i przesyła do ojca

Analiza

- szybkie
- globalne zatrzymanie po czasie $O(D)$
- problem?

Jak to zrobić?

Convergecast na ukorzenionym drzewie BFS

- rekurencja na drzewie
- r nakazuje swoim dzieciom obliczenie funkcji w swoich poddrzewach
- wierzchołek wewnętrzny czeka aż wszystkie dzieci mu odpowiedzą, aplikuje funkcje i przesyła do ojca

Analiza

- szybkie
- globalne zatrzymanie po czasie $O(D)$
- problem?

Jak to zrobić?

Convergecast na ukorzenionym drzewie BFS

- rekurencja na drzewie
- r nakazuje swoim dzieciom obliczenie funkcji w swoich poddrzewach
- wierzchołek wewnętrzny czeka aż wszystkie dzieci mu odpowiedzą, aplikuje funkcje i przesyła do ojca

Analiza

- szybkie
- globalne zatrzymanie po czasie $O(D)$
- problem?

Jak obliczać drzewo BFS?

- model synchroniczny - proste
- co z asynchronicznym?
- albo czas, albo komunikacja
- Bellman-Ford

Jak obliczać drzewo BFS?

- model synchroniczny - proste
- co z asynchronicznym?
- albo czas, albo komunikacja
- Bellman-Ford

Jak obliczać drzewo BFS?

- model synchroniczny - proste
- co z asynchronicznym?
- albo czas, albo komunikacja
- Bellman-Ford

Jak obliczać drzewo BFS?

- model synchroniczny - proste
- co z asynchronicznym?
- albo czas, albo komunikacja
- Bellman-Ford

Jak obliczać drzewo BFS?

- model synchroniczny - proste
- co z asynchronicznym?
- albo czas, albo komunikacja
- Bellman-Ford

Od teraz skupimy się tylko na problemie wyboru k^{th} elementu

Główne wyniki

- ściśle trudniejsze niż $convergecast$ (distributive, algebraic) - dolna granica $\Omega(Dlog_D n)$
- randomizowany algorytm typu Las Vegas, który z wysokim ppb. osiąga lower bound
- deterministyczny algorytm - $O(Dlog_D^2 n)$

Od teraz skupimy się tylko na problemie wyboru k^{th} elementu

Główne wyniki

- ściśle trudniejsze niż $convergecast$ (distributive, algebraic) - dolna granica $\Omega(Dlog_D n)$
- randomizowany algorytm typu Las Vegas, który z wysokim ppb. osiąga lower bound
- deterministyczny algorytm - $O(Dlog_D^2 n)$

Od teraz skupimy się tylko na problemie wyboru k^{th} elementu

Główne wyniki

- ściśle trudniejsze niż $convergecast$ (distributive, algebraic) - dolna granica $\Omega(Dlog_D n)$
- randomizowany algorytm typu Las Vegas, który z wysokim ppb. osiąga lower bound
- deterministyczny algorytm - $O(Dlog_D^2 n)$

Od teraz skupimy się tylko na problemie wyboru k^{th} elementu

Główne wyniki

- ściśle trudniejsze niż $convergecast$ (distributive, algebraic) - dolna granica $\Omega(Dlog_D n)$
- randomizowany algorytm typu Las Vegas, który z wysokim ppb. osiąga lower bound
- deterministyczny algorytm - $O(Dlog_D^2 n)$

Od teraz skupimy się tylko na problemie wyboru k^{th} elementu

Główne wyniki

- ściśle trudniejsze niż $convergecast$ (distributive, algebraic) - dolna granica $\Omega(Dlog_D n)$
- randomizowany algorytm typu Las Vegas, który z wysokim ppb. osiąga lower bound
- deterministyczny algorytm - $O(Dlog_D^2 n)$

Model RAM

- randomizowany - oczekiwany $O(n)$
- deterministyczny $O(n)$ - magiczne piątki

Model RAM

- randomizowany - oczekiwany $O(n)$
- deterministyczny $O(n)$ - magiczne piątki

Model RAM

- randomizowany - oczekiwany $O(n)$
- deterministyczny $O(n)$ - magiczne piątki

Model rozproszony - specjalne przypadki

- gwiazdy, kliki
- dwa wierzchołki - każdy zna $n/2$ wartości
- pierścienie $O(n)$
- siatki $O(\sqrt{n})$
- pełne drzewa binarne $O(log^3 n)$

Model rozproszony - specjalne przypadki

- gwiazdy, kliki
- dwa wierzchołki - każdy zna $n/2$ wartości
- pierścienie $O(n)$
- siatki $O(\sqrt{n})$
- pełne drzewa binarne $O(log^3 n)$

Model rozproszony - specjalne przypadki

- gwiazdy, kliki
- dwa wierzchołki - każdy zna $n/2$ wartości
- pierścienie $O(n)$
- siatki $O(\sqrt{n})$
- pełne drzewa binarne $O(log^3 n)$

Model rozproszony - specjalne przypadki

- gwiazdy, kliki
- dwa wierzchołki - każdy zna $n/2$ wartości
- pierścienie $O(n)$
- siatki $O(\sqrt{n})$
- pełne drzewa binarne $O(log^3 n)$

Model rozproszony - specjalne przypadki

- gwiazdy, kliki
- dwa wierzchołki - każdy zna $n/2$ wartości
- pierścienie $O(n)$
- siatki $O(\sqrt{n})$
- pełne drzewa binarne $O(log^3 n)$

Model rozproszony - specjalne przypadki

- gwiazdy, kliki
- dwa wierzchołki - każdy zna $n/2$ wartości
- pierścienie $O(n)$
- siatki $O(\sqrt{n})$
- pełne drzewa binarne $O(log^3 n)$

Model rozproszony - graf ogólny

- det. $O(Dn^{0.9114})$
- rand. oczekiwane $O(Dlogn)$

A co jeśli mamy ograniczenia na wartości?

- $x_{max} = O(n^{O(1)})$
- $O(Dlogn)$
- $O(Dlogx_k)$

Model rozproszony - graf ogólny

- det. $O(Dn^{0.9114})$
- rand. oczekiwane $O(Dlogn)$

A co jeśli mamy ograniczenia na wartości?

- $x_{max} = O(n^{O(1)})$
- $O(Dlogn)$
- $O(Dlogx_k)$

Model rozproszony - graf ogólny

- det. $O(Dn^{0.9114})$
- rand. oczekiwane $O(Dlogn)$

A co jeśli mamy ograniczenia na wartości?

- $x_{max} = O(n^{O(1)})$
- $O(Dlogn)$
- $O(Dlogx_k)$

Model rozproszony - graf ogólny

- det. $O(Dn^{0.9114})$
- rand. oczekiwane $O(Dlogn)$

A co jeśli mamy ograniczenia na wartości?

- $x_{max} = O(n^{O(1)})$
- $O(Dlogn)$
- $O(Dlogx_k)$

Model rozproszony - graf ogólny

- det. $O(Dn^{0.9114})$
- rand. oczekiwane $O(Dlogn)$

A co jeśli mamy ograniczenia na wartości?

- $x_{max} = O(n^{O(1)})$
- $O(Dlogn)$
- $O(Dlogx_k)$

Model rozproszony - graf ogólny

- det. $O(Dn^{0.9114})$
- rand. oczekiwane $O(Dlogn)$

A co jeśli mamy ograniczenia na wartości?

- $x_{max} = O(n^{O(1)})$
- $O(Dlogn)$
- $O(Dlogx_k)$

Model rozproszony - graf ogólny

- det. $O(Dn^{0.9114})$
- rand. oczekiwane $O(Dlogn)$

A co jeśli mamy ograniczenia na wartości?

- $x_{max} = O(n^{O(1)})$
- $O(Dlogn)$
- $O(Dlogx_k)$

Definicje

- iteracje, kandydaci, fazy
- $n^{(i)}$ - liczba kandydatów w i^{th} fazie

Założenia

- drzewo BFS jest obliczone
- wierzchołki znają D

Definicje

- iteracje, kandydaci, fazy
- $n^{(i)}$ - liczba kandydatów w i^{th} fazie

Założenia

- drzewo BFS jest obliczone
- wierzchołki znają D

Definicje

- iteracje, kandydaci, fazy
- $n^{(i)}$ - liczba kandydatów w i^{th} fazie

Założenia

- drzewo BFS jest obliczone
- wierzchołki znają D

Definicje

- iteracje, kandydaci, fazy
- $n^{(i)}$ - liczba kandydatów w i^{th} fazie

Założenia

- drzewo BFS jest obliczone
- wierzchołki znają D

Co i jak mierzymy?

- jaka złożoność nas interesuje
- jak mierzymy złożoność czasową algorytmu randomizowanego?

Dla przypomnienia

- normalizowany czas
- lokalne obliczenia nic nie kosztują

Co i jak mierzymy?

- jaka złożoność nas interesuje
- jak mierzymy złożoność czasową algorytmu randomizowanego?

Dla przypomnienia

- normalizowany czas
- lokalne obliczenia nic nie kosztują

Co i jak mierzymy?

- jaka złożoność nas interesuje
- jak mierzymy złożoność czasową algorytmu randomizowanego?

Dla przypomnienia

- normalizowany czas
- lokalne obliczenia nic nie kosztują

Co i jak mierzymy?

- jaka złożoność nas interesuje
- jak mierzymy złożoność czasową algorytmu randomizowanego?

Dla przypomnienia

- normalizowany czas
- lokalne obliczenia nic nie kosztują

SIMPLE RAND

1. wybierz losowy element p ze zbioru kandydatów $\approx$ losowa ścieżka w drzewie, zaczynająca się w r
2. oblicz frakcję elementów $< p$ i $> p$
3. wyznacz nowy zbiór kandydatów (wierzchołki wewnętrzne muszą znać liczbę kandydatów w poddrzewach swoich synów)

Analiza

- Oczekiwany czas $O(Dlogn)$ - dlaczego?
- Można nawet pokazać, że czas $O(Dlogn)$ jest osiągany w.h.p.

SIMPLE RAND

1. wybierz losowy element p ze zbioru kandydatów $\approx$ losowa ścieżka w drzewie, zaczynająca się w r
2. oblicz frakcję elementów $< p$ i $> p$
3. wyznacz nowy zbiór kandydatów (wierzchołki wewnętrzne muszą znać liczbę kandydatów w poddrzewach swoich synów)

Analiza

- Oczekiwany czas $O(Dlogn)$ - dlaczego?
- Można nawet pokazać, że czas $O(Dlogn)$ jest osiągany w.h.p.

SIMPLE RAND

1. wybierz losowy element p ze zbioru kandydatów $\approx$ losowa ścieżka w drzewie, zaczynająca się w r
2. oblicz frakcję elementów $< p$ i $> p$
3. wyznacz nowy zbiór kandydatów (wierzchołki wewnętrzne muszą znać liczbę kandydatów w poddrzewach swoich synów)

Analiza

- Oczekiwany czas $O(Dlogn)$ - dlaczego?
- Można nawet pokazać, że czas $O(Dlogn)$ jest osiągany w.h.p.

SIMPLE RAND

1. wybierz losowy element p ze zbioru kandydatów $\approx$ losowa ścieżka w drzewie, zaczynająca się w r
2. oblicz frakcję elementów $< p$ i $> p$
3. wyznacz nowy zbiór kandydatów (wierzchołki wewnętrzne muszą znać liczbę kandydatów w poddrzewach swoich synów)

Analiza

- Oczekiwany czas $O(Dlogn)$ - dlaczego?
- Można nawet pokazać, że czas $O(Dlogn)$ jest osiągany w.h.p.

SIMPLE RAND

1. wybierz losowy element p ze zbioru kandydatów $\approx$ losowa ścieżka w drzewie, zaczynająca się w r
2. oblicz frakcję elementów $< p$ i $> p$
3. wyznacz nowy zbiór kandydatów (wierzchołki wewnętrzne muszą znać liczbę kandydatów w poddrzewach swoich synów)

Analiza

- Oczekiwany czas $O(Dlogn)$ - dlaczego?
- Można nawet pokazać, że czas $O(Dlogn)$ jest osiągany w.h.p.

SIMPLE RAND

1. wybierz losowy element p ze zbioru kandydatów $\approx$ losowa ścieżka w drzewie, zaczynająca się w r
2. oblicz frakcję elementów $< p$ i $> p$
3. wyznacz nowy zbiór kandydatów (wierzchołki wewnętrzne muszą znać liczbę kandydatów w poddrzewach swoich synów)

Analiza

- Oczekiwany czas $O(Dlogn)$ - dlaczego?
- Można nawet pokazać, że czas $O(Dlogn)$ jest osiągany w.h.p.

SIMPLE RAND

1. wybierz losowy element p ze zbioru kandydatów $\approx$ losowa ścieżka w drzewie, zaczynająca się w r
2. oblicz frakcję elementów $< p$ i $> p$
3. wyznacz nowy zbiór kandydatów (wierzchołki wewnętrzne muszą znać liczbę kandydatów w poddrzewach swoich synów)

Analiza

- Oczekiwany czas $O(Dlogn)$ - dlaczego?
- Można nawet pokazać, że czas $O(Dlogn)$ jest osiągany w.h.p.

Idea

- gdzie możemy coś poprawić
- wybór jednego losowego elementu trwa $O(D)$
- ile losowych elementów możemy wybrać w czasie $O(D)$?
- czy to coś daje?

Idea

- gdzie możemy coś poprawić
- wybór jednego losowego elementu trwa $O(D)$
- ile losowych elementów możemy wybrać w czasie $O(D)$?
- czy to coś daje?

Idea

- gdzie możemy coś poprawić
- wybór jednego losowego elementu trwa $O(D)$
- ile losowych elementów możemy wybrać w czasie $O(D)$?
- czy to coś daje?

Idea

- gdzie możemy coś poprawić
- wybór jednego losowego elementu trwa $O(D)$
- ile losowych elementów możemy wybrać w czasie $O(D)$?
- czy to coś daje?

Idea

- gdzie możemy coś poprawić
- wybór jednego losowego elementu trwa $O(D)$
- ile losowych elementów możemy wybrać w czasie $O(D)$?
- czy to coś daje?

Algorithm 1 $\mathcal{A}^{rand}(t,k)$

1: $x_{j-1} := -\infty$; $x_j := \infty$
2: **repeat**
3: $x_0 := x_{j-1}$; $x_{t+1} := x_j$
4: $\{x_1, \ldots, x_t\} :=$ getRndElementsInRange$(t, (x_{j-1}, x_j))$
5: **for** $i = 1, \ldots, t$ **in parallel do**
6: $r_i :=$ countElementsInRange$((x_{i-1}, x_i])$
7: **od**
8: **if** $x_0 \neq -\infty$ **then** $r_1 := r_1 + 1$ **fi**
9: $j := \min_{l \in \{1,\ldots,t+1\}} \sum_{i=1}^{l} r_i > k$
10: $k := k - \sum_{i=1}^{j-1} r_i$
11: **if** $k \neq 0$ **and** $j \neq 1$ **then** $k := k + 1$ **fi**
12: **until** $r_j \leq t$ **or** $k = 0$
13: **if** $k = 0$ **then**
14: **return** x_j
15: **else**
16: $\{x_1, \ldots, x_s\} :=$ getElementsInRange$([x_{j-1}, x_j])$
17: **return** x_k
18: **fi**

Twierdzenie 1

$RAND(8\lambda D, k)$ wyznacza k^{th} element w mniej niż $3log_D n$ fazach z ppb. $\geqslant 1 - 1/n^{\lambda}$

Dowód

...

Twierdzenie 1

$RAND(8\lambda D, k)$ wyznacza k^{th} element w mniej niż $3log_D n$ fazach z ppb. $\geqslant 1 - 1/n^{\lambda}$

Dowód

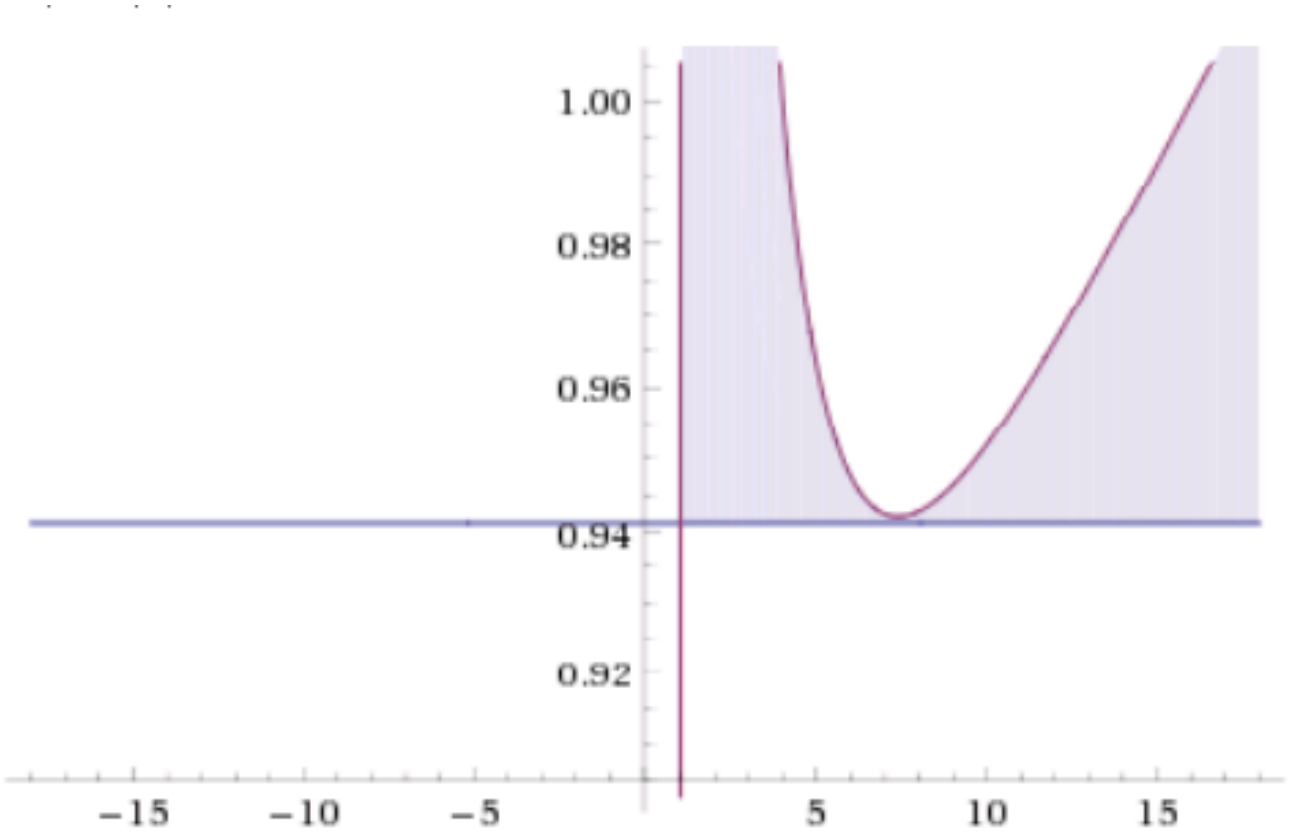
...

Twierdzenie 1

$RAND(8\lambda D, k)$ wyznacza k^{th} element w mniej niż $3log_D n$ fazach z ppb. $\geqslant 1 - 1/n^{\lambda}$

Dowód

...



Rysunek : $16/17 < \sqrt{x}/logx$

Krótko - idea

- szukamy dobrego podziału, ale deterministycznie
- wybieranie elementów dobrze dzielących, z elementów otrzymanych przez dzieci
- wolne
- w każdej fazie, rekurencyjny podział na grupy, dopóki ich wielkości $> O(\sqrt{n})$
- $O(Dlog_D^2 n)$

Krótko - idea

- szukamy dobrego podziału, ale deterministycznie
- wybieranie elementów dobrze dzielących, z elementów otrzymanych przez dzieci
- wolne
- w każdej fazie, rekurencyjny podział na grupy, dopóki ich wielkości $> O(\sqrt{n})$
- $O(Dlog_D^2 n)$

Krótko - idea

- szukamy dobrego podziału, ale deterministycznie
- wybieranie elementów dobrze dzielących, z elementów otrzymanych przez dzieci
- wolne
- w każdej fazie, rekurencyjny podział na grupy, dopóki ich wielkości $> O(\sqrt{n})$
- $O(Dlog_D^2 n)$

Krótko - idea

- szukamy dobrego podziału, ale deterministycznie
- wybieranie elementów dobrze dzielących, z elementów otrzymanych przez dzieci
- wolne
- w każdej fazie, rekurencyjny podział na grupy, dopóki ich wielkości $> O(\sqrt{n})$
- $O(Dlog_D^2 n)$

Krótko - idea

- szukamy dobrego podziału, ale deterministycznie
- wybieranie elementów dobrze dzielących, z elementów otrzymanych przez dzieci
- wolne
- w każdej fazie, rekurencyjny podział na grupy, dopóki ich wielkości $> O(\sqrt{n})$
- $O(Dlog_D^2 n)$

Krótko - idea

- szukamy dobrego podziału, ale deterministycznie
- wybieranie elementów dobrze dzielących, z elementów otrzymanych przez dzieci
- wolne
- w każdej fazie, rekurencyjny podział na grupy, dopóki ich wielkości $> O(\sqrt{n})$
- $O(Dlog_D^2 n)$

Krótko

- redukcja do problemu mediany
- protokół dla dwóch procesorów, każdy z $n/2$ wartościami; można przesłać $\leqslant B$ elementów w jednym komunikacie -> $\Omega(log_B n)$
- dla pokazania dolnej granicy, buduje się specjalny graf, który może zostać przekształcony w powyższy schemat, używając $B = D - 2$

Krótko

- redukcja do problemu mediany
- protokół dla dwóch procesorów, każdy z $n/2$ wartościami; można przesłać $\leqslant B$ elementów w jednym komunikacie -> $\Omega(log_B n)$
- dla pokazania dolnej granicy, buduje się specjalny graf, który może zostać przekształcony w powyższy schemat, używając $B = D - 2$

Krótko

- redukcja do problemu mediany
- protokół dla dwóch procesorów, każdy z $n/2$ wartościami; można przesłać $\leqslant B$ elementów w jednym komunikacie -> $\Omega(log_B n)$
- dla pokazania dolnej granicy, buduje się specjalny graf, który może zostać przekształcony w powyższy schemat, używając $B = D - 2$

Krótko

- redukcja do problemu mediany
- protokół dla dwóch procesorów, każdy z $n/2$ wartościami; można przesłać $\leqslant B$ elementów w jednym komunikacie -> $\Omega(log_B n)$
- dla pokazania dolnej granicy, buduje się specjalny graf, który może zostać przekształcony w powyższy schemat, używając $B = D - 2$

Czego się dowiedzieliśmy

- problem wyboru k^{th} elementu jest ściśle trudniejszy w modelu rozproszonym
- problem wyboru k^{th} elementu w modelu rozproszonym jest ściśle trudniejszy niż funkcje distributive czy algebraic
- istnieje algorytm randomizowany, który osiąga lower bound w.h.p.

Dziękuję za uwagę.