WYKŁAD

4.05.2015

Algorytmy strumieniowe

Nagroda z 2005 roku, artykuł z 1999 roku. — Noga Alon, Yossi Matias, Mario Szegedy

W dzisiejszych czasach jest coraz więcej danych. Zdarza się, że dane te przychodzą w czasie rzeczywistym i nie damy rady ich wszystkich zapisać na dysku. Albo damy, ale to byłoby bardzo dużo, albo nie chcemy. Przykłady to: ruch w sieci, rezultaty eksperymentów naukowych ~~[illegible]~~ ~~[illegible]~~ (działających pewnie cały czas) itd.

Stąd model strumienia i algorytmu strumieniowego. Zakładamy, że przychodzą do nas liczby z ciągu $a_1, a_2, \dots, a_m$, m duże, każde $a_i \in \{1, 2, \dots, n\}$. Zakładamy często, że znamy m, ale czasami nie. Typowo chcielibyśmy policzyć np. liczbę różnych elementów, albo ile ich w ogóle jest, albo jak bardzo różnią się częstości.

Oznaczmy $m_i = |\{j : a_j = i\}|$, czyli ile razy wystąpił element i w ciągu.

Mamy wtedy $m = \sum_{i=1}^{n} m_i$. Ogólniej interesujące są

$$F_k = \sum_{i=1}^{n} m_i^k$$

Np. $F_0 = \sum_{i=1}^{n} m_i^0$ to

liczba różnych elementów. $F_1 = \sum_{i=1}^{n} m_i = m$.

$F_2 = \sum_{i=1}^{n} m_i^2$ odpowiada temu jak

bardzo różnią się częstości różnych elementów,

to ma związek z tzw. surprise index.

Policzyć F_k w pamięci $O(n)$ jest łatwo,

po prostu liczymy wszystkie m_i. Będziemy się starali

liczyć to w mniejszej pamięci, bo n jest duże.

Algorytmy będą randomizowane oraz aproksymacyjne.

Artykuł Alona, Matiasa i Szegedy pokazał niektóre

bardzo ładne wyniki i położył podwaliny pod dziedzinę.

Teraz wiadomo, że:

Tw.

Dla $p \in [0,2]$ wystarczy ~~prostu~~ polylog(n) pamięci

Dla $p > 2$ $n^{1-\frac{2}{p}}$ polylog(n) pamięci wystarczy i jest potrzebne

Nie wszystko pokazali ~~oni~~ AMS, ale pokazali np.

bardzo ładny algorytm dla $p=2$.

Najpierw idea dla prostego algorytmu szacującego

F_0, nie przez AMS.

Powiedzmy, że chcemy zrobić alg. randomizowany, który z prawd. $1-p$ oblicza F_0 z dokładnością do $(1+\alpha)$.

Najpierw uprośćmy problem: dane $T>0$, chcemy algorytm, który z prawd. $1-p$:

- odpowiada TAK jeśli $F_0 > (1+\alpha)T$
- odpowiada NIE jeśli $F_0 < (1-\alpha)T$

Jeśli mamy taki, to możemy zapuścić równolegle algorytmy dla $T = 1, 1+\alpha, (1+\alpha)^2, (1+\alpha)^3, \ldots n$ i z dużym prawd. dostaniemy oszacowanie F_0 z dokładnością do $(1+\alpha)$

Teraz ten algorytm robimy tak:

- losujemy zbiór S t, że $\forall_{i \in \{1,\ldots,n\}} P(i \in S) = \frac{1}{T}$.
- teraz lecimy przez cały strumień i liczymy $\sum_{j=1}^{m} [a_j \in S] =: SUM$

$$Pr = \mathbb{P}[SUM = 0] = \left(1 - \frac{1}{T}\right)^{F_0}$$

$$\text{Dla d.d. } T \quad \left(1-\frac{1}{T}\right)^{F_0} = \left(1 - \frac{1}{T}\right)^{T \cdot \frac{F_0}{T}} \approx e^{-\frac{F_0}{T}}$$

(dla małych precyzyjnie można inaczej rozumieć)

Zatem jeśli $F_0 > (1+\alpha)T$, to

$$Pr \lessapprox e^{-\frac{(1+\alpha)T}{T}} = e^{-(1+\alpha)} < \frac{1}{e} - \frac{\alpha}{3}$$

Jeśli $F_0 < (1-\alpha)T$, to

$$Pr \gtrapprox e^{-\frac{(1-\alpha)T}{T}} = e^{-(1-\alpha)} > \frac{1}{e} + \frac{\alpha}{3}$$

podobnie się da pokazać

Można teraz zwiększyć odstęp między $\frac{1}{e} - \frac{\alpha}{3}$ a $\frac{1}{e} + \frac{\alpha}{3}$ przez robienie tego wiele razy i użycie dekompozycji.

Teraz bardziej konkretnie, pokażemy oszacowanie na F_2 w $O(\log n)$ pamięci.

Tw.

Dla każdego $\lambda > 0$, $\varepsilon > 0$ istnieje algorytm randomizowany, który dla ciągu $a_1, a_2, \dots, a_m$ elementów $\{1, 2, \dots, n\}$ oblicza w jednym przebiegu i używając

$$O\left(\frac{\log\left(\frac{1}{\varepsilon}\right)}{\lambda^2}(\log n + \log m)\right)$$

pamięci liczbę Y t., że $\mathbb{P}(|Y - F_2| > \lambda F_2) \leq \varepsilon$.

D-d

Niech $s_1 = \frac{16}{\lambda^2}$, $s_2 = 2\log\left(\frac{1}{\varepsilon}\right)$.

Idea jest taka.

Stworzymy zmienną losową X, którą będziemy umieli policzyć w jednym przebiegu t., że $\mathbb{E}X = F_2$ oraz $\operatorname{Var} X$ małana. Wtedy X będzie koło F_2.

Jednak aby poprawić dokładność zrobimy tak, że Y to mediana zmiennych $Y_1, \dots, Y_{s_2}$, a zmienna Y_i to średnia zmiennych $X_{i1}, \dots, X_{is_1}$, gdzie każda zmienna X_{ij} ma rozkład taki jak X.

Skupiamy się teraz na X.

Będziemy chcieli policzyć zmienną $Z = \sum_{i=1}^{n} \varepsilon_i m_i$, gdzie $\mathbb{P}(\varepsilon_i = 1) = \mathbb{P}(\varepsilon_i = -1) = \frac{1}{2}$. Idea jest taka, że ε_i są niezależne, ale nam tak naprawdę będzie zależało tylko, by były czwórkami niezależne.

Będziemy $X = Z^2$.

Jak liczymy Z? Przelatujemy po całym strumieniu i jak widzimy a_j, to dodajemy

$Z := Z + \varepsilon_{a_j}$

Pewien problem polega na tym, że w związku z tym musimy wszystkie te ε_i gdzieś pamiętać, a jest ich n (bo potem będziemy chcieli ich może użyć jeszcze raz). Okazuje się, że pomaga nam to, że one mogą być tylko czwórkami niezależne, więc damy radę pamiętać je w $O(\log n)$ pamięci. ~~Może~~ Pewnie pokażemy na ćwiczeniach jak to zrobić.

2-niezal.

Ok, więc $Z = \sum_{i=1}^{n} \varepsilon_i m_i$, $X = Z^2$.

Policzmy $\mathbb{E}X$. $\qquad \mathbb{E}(\varepsilon_i \varepsilon_j) = 0$

$$\mathbb{E}X = \mathbb{E}\left(\left(\sum_{i=1}^{n} \varepsilon_i m_i\right)^2\right) = \mathbb{E}\left(\sum_{i=1}^{n} m_i^2 \varepsilon_i^2 + 2\sum_{1 \le i < j \le n} m_i m_j \varepsilon_i \varepsilon_j\right) =$$

$$= \sum_{i=1}^{n} m_i^2 = F_2.$$

$\mathrm{Var}(X) = \mathbb{E}X^2 - (\mathbb{E}X)^2$, liczymy więc $\mathbb{E}(X^2)$

$$\mathbb{E}X^2 = \mathbb{E}\left(\left(\sum_{i=1}^{n} \varepsilon_i m_i\right)^4\right) = \sum_{i=1}^{n} m_i^4 + 6 \sum_{1 \le i < j \le n} m_i^2 m_j^2$$

$$\text{A więc } \mathrm{Var}(X) = \left(\sum_{i=1}^{n} m_i^4\right) + 6\sum_{1 \le i < j \le n} m_i^2 m_j^2 - \left(\sum_{i=1}^{n} m_i^2\right)^2 = 4\sum_{1 \le i < j \le n} m_i^2 m_j^2$$

Nier. Czebyszewa to $\mathbb{P}(|X - \mathbb{E}X| > t) \le \frac{\mathrm{Var}X}{t^2}$

$\le 2F_2^2$

(chyba nawet F_2^2)

A więc ~~[illegible]~~ $\mathbb{E}(Y_i) = F_2$, $Var(Y_i) \leq \frac{2F_2^2}{s_1}$.

Z Czebyszewa mamy

$$\mathbb{P}[\,|Y_i - F_2| > \lambda F_2] \leq \frac{Var(Y_i)}{\lambda^2 F_2^2} \leq \frac{2F_2^2}{s_1 \lambda^2 F_2^2} = \frac{2F_2^2}{\frac{16}{\lambda^2}\lambda^2 F_2} = \frac{1}{8}$$

Jest nier. Chernoffa, która mówi (jedna z wielu), że

Jeśli $p > \frac{1}{2}$ to pr. sukcesu, to prawd., że w n próbach udziału będzie $\geq \frac{n}{2}$ sukcesów jest

$$\geq 1 - e^{-\frac{1}{2p} n (p - \frac{1}{2})^2}$$

Jak ustawimy $p \geq \frac{7}{8}$ i $s_2 = 16 \log(\frac{1}{\varepsilon})$ (nie umiem z $2 \log(\frac{1}{\varepsilon})$), to wyjdzie, że

$\mathbb{P}(\geq \frac{n}{2}$ sukcesów$) \geq 1 - \varepsilon$.

Jeśli $\geq \frac{n}{2}$ mamy blisko F_2, to mediana też, stąd

$\mathbb{P}(|X - F_2| < \lambda F_2) \geq 1 - \varepsilon$.

c.n.d.

Teraz pokażemy ~~[illegible]~~ dwa dolne ograniczenia, bo są bardzo fajne. Będziemy korzystać ze złożoności komunikacyjnej. Powiedzmy, że ~~[illegible]~~ dana jest funkcja $f: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. Dwie osoby chcą policzyć $f(x,y)$, $x, y \in \{0,1\}^n$, przy czym A ma x, a B ma y. One mogą sobie przesyłać bity, mogą też korzystać z losowości. Złożoność komunikacyjna $C_\varepsilon(f)$ to ~~[illegible]~~ oczekiwana liczba bitów, którą muszą przesłać w najgorszym przypadku (przy użyciu najlepszego protokołu), tak, by się pomylić z prawd. $\leq \varepsilon$.

Rozważmy konkretną funkcję

$DIS_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ t. że

$DIS_n(x,y) = 1$ jeśli ~~[illegible]~~ $x_i \wedge y_i = 1$ dla pewnego indeksu $i \in \{1,\dots,n\}$.

Jest wynik, że dla każdego $\varepsilon < \frac{1}{2}$

$C_\varepsilon(DIS_n) \geqslant \Omega(n)$. Skorzystamy z tego.

Tw.

Dowolny algorytm randomizowany, który dla danego ciągu o co najwyżej $2n$ elementach ze zbioru $\{1,2,\dots,n\}$, oblicza liczbę Y t. że $\mathbb{P}(|Y - F_\infty^*| \geqslant \frac{F_\infty^*}{3}) < \varepsilon$, dla $\varepsilon < \frac{1}{2}$ musi używać $\Omega(n)$ bitów pamięci.

(F_∞^* to $\max_{1 \leq i \leq n} m_i$, ogólnie $F_p^* = \sqrt[p]{F_p}$, F_∞ pasuje do tego, tzn. $F_p^* \to F_\infty^*$ w pewnym sensie).

D-d

Przypuśćmy, że mamy algorytm j.w., który używa s bitów pamięci, pokażemy protokół komunikacyjny, który używa s bitów komunikacji dla $DIS_n(x,y)$. Niech $|x|$ i $|y|$ to odp. długości x i y.

Niech A to ciąg $|x|+|y|$ liczb z $\{1,\dots,n\}$ t. że na początku dajemy elementy j t. że $x_j = 1$, a potem k t. że $y_k = 1$.

Zauważmy, że — $F_\infty^* = 1 \Leftrightarrow DIS(x,y)$

Gracz A zna x, więc realizuje alg. aproksym. na pierwszych $|x|$ elementach ciągu A. Ma teraz stan w s komórkach pamięci. Wysyła to do B, który kontynuuje, bo zna y. ~~[illegible]~~ Jeśli B otrzyma coś $< \frac{4}{3}$ to znaczy, że $F_\infty^* = 1$, więc zwraca $DIS(x,y)$, jeśli otrzyma coś $> \frac{4}{3} = 2 - \frac{1}{3} \cdot 2$, to znaczy, że $F_\infty^* \geqslant 2$, więc zwraca $\neg DIS(x,y)$.

Uwaga: Ta technika działa też dla $m \geq 2n$.
Możemy rozważać ciągi, w których każdy wyraz występuje 0, $\frac{m}{n}$ lub $\frac{2m}{n}$ razy, wtedy wszystko przechodzi.

Pokażemy teraz drugie z dolnych ograniczeń. Mianowicie, że deterministyczne algorytmy radzą sobie dużo gorzej.

Tw.

Dla dowolnego $k \neq 1$ dowolny deterministyczny algorytm, który dla ciągu $\frac{n}{2}$ elementów ze zbioru $\{1, 2, \dots, n\}$ zwraca liczbę Y t. że $|Y - F_k| \leq \frac{1}{10} F_k$ używa $\Omega(n)$ bitów pamięci.

D-d

Niech $\mathcal{G}$ będzie rodziną $t = 2^{\Omega(n)}$ podzbiorów $\{1, 2, \dots, n\}$, każdy o mocy $\frac{n}{4}$ t. że przecięcie dowolnych dwóch ma $\leq \frac{n}{8}$ elementów.
Da się pokazać istnienie takiej rodziny, podobno standardowymi technikami z teorii kodów.
Ustalmy det. alg. aproksymujący F_k dla $k \neq 1$.
Dla dowolnych $G_1, G_2 \in \mathcal{G}$ niech $A(G_1, G_2)$ to ciąg dł. $\frac{n}{2}$, najpierw G_1, potem G_2. Algorytm po wczytaniu pierwszych $\frac{n}{4}$ bitów ma stan pamięci zależny tylko od G_1.
Jeśli pamięć jest $< \log t$, to z zas. szufl. Dirichleta istnieją $G_1, G_2 \in \mathcal{G}$ t. że jest ona taka sama, a więc $A(G_1, G_1) = A(G_2, G_1)$.

Policzmy jednak F_0 i F_k dla $k \geq 2$ dla $A(G_1, G_1)$ i $A(G_2, G_1)$.

	F_0	$F_k, k \geq 2$
$A(G_1, G_1)$	$\frac{n}{4}$	$2^k \frac{n}{4}$
$A(G_2, G_1)$	$\geq \frac{3n}{8}$	$\leq \frac{n}{4} + 2^k \frac{n}{8}$

A więc ~~błąd~~ błąd względny jest $> \frac{1}{10}$ dla przynajmniej jednego z tych przypadków.

A zatem pamięć $\geq \log(t) = \Omega(n)$.

c.n.d.

Teraz kilka prostych ograniczeń dolnych, które można pokazać na ćwiczeniach.

- Pokazać, że algorytm ~~rand~~ randomizowany, ale dokładny $P(Y = F_k) \geq 1 - \varepsilon$ musi użyć $\Omega(n)$ pamięci dla $k \neq 1$

(dowód idzie tak jak dla przybliżenia F_∞^*)

- Pokazać, że alg. $\overset{\text{rand.}}{\vee}$ przybliżający F_0 z dokł. do $\frac{1}{10} F_0$ z prawd. $\frac{3}{4}$ musi użyć przynajmniej $\Omega(\log n)$ bitów

$\underline{\text{D-d}}$

Wiadomo, że $\overbrace{\text{złożoność}}^{\text{rand.}}$ komunikacyjna równości $f(x,y) = 1 \Leftrightarrow x = y$ jest $\Omega(\overset{\log n}{\cancel{\sqrt{n}}})$. Robimy tak jak w dowodzie powyżej, $A(G_1, G_2)$ i $A(G_2, G_1)$ powinny być rozróżnialne.

- Pokaż, że ~~Fa~~ przybliżanie losowe F_1 wymaga ~~wm~~ $\Omega(\log\log m)$ bitów

  (jest min. $\log m$ różnych odpowiedzi, stąd $\log\log m$ bitów na nie)

- Pokaż, że ~~przyb~~ losowe przybliżanie F_2 wymaga $\Omega(\log n + \log\log m)$ bitów

  ($\log n$ jak dla F_0, $\log\log m$ jak dla F_1)

Teraz zobaczymy jeszcze (jeśli starczy czasu, jeśli nie, to może spróbujemy zrobić na ćwiczeniach) jak przybliżyć F_k w pamięci $O(n^{1-\frac{1}{k}}\log(n+m))$.

~~Wniosek~~ Tw.

Dla każdego $k \geq 1$, $\lambda > 0$, $\varepsilon > 0$ istnieje algorytm randomizowany, który dla ciągu $a_1, a_2, \dots, a_m$ elementów $\{1,2,\dots,n\}$ używając jednego przebiegu ~~ora~~ i

$$O\left(\frac{k\log(\frac{1}{\varepsilon})}{\lambda^2} n^{1-\frac{1}{k}}(\log n + \log m)\right)$$

bitów pamięci oblicza Y t.że $\mathbb{P}(|Y - F_k| > \lambda F_k) \leq \varepsilon$.

D-d

Niech $s_1 = \frac{8kn^{1-\frac{1}{k}}}{\lambda^2}$, $s_2 = 2\log(\frac{1}{\varepsilon})$ (a może tak jak poprzednio $16\log(\frac{1}{\varepsilon})$).

Podobnie jak wcześniej algorytm obliczy $Y_1, Y_2, \dots, Y_{s_1}$ i weźmie Y jako medianę.
Podobnie też Y_i to średnia z $X_{i1}, X_{i2}, \dots, Y_{is_2}$, gdzie wszystkie X_{ij} to niezależne zmienne losowe o jednakowym rozkładzie, takim jak X.

Pokażemy teraz jak obliczyć w pamięci $O(\log n + \log m)$ X t. że $\mathbb{E}X = F_k$, $\operatorname{Var} X$ małe.
Robimy tak: losujemy liczbę p ze zbioru $\{1, 2, \dots, m\}$ (jeśli nie znamy m, to też to się da zrobić).
Mamy więc $a_p = l \in \{1, 2, \dots, n\}$.
Niech $r = |\{q : q \geq p, a_q = l\}|$.
Definiujemy

$$X = m \cdot (r^k - (r-1)^k).$$

Wystarczy pamiętać l ($O(\log n)$ bitów) i r ($O(\log m)$ bitów).

$$\begin{aligned}\mathbb{E}X = \frac{1}{m} \cdot m \cdot\ & (1^k + (2^k - 1^k) + \dots + (m_1^k - (m_1 - 1)^k) + \\ & (1^k + (2^k - 1^k) + \dots + (m_2^k - (m_2 - 1)^k) + \\ & \vdots \\ & (1^k + (2^k - 1^k) + \dots + (m_n^k - (m_n - 1)^k) =\end{aligned}$$

$$= \sum_{i=1}^{n} m_i^k = F_k.$$

$$Var(X) = \mathbb{E}(X^2) - (\mathbb{E}X)^2, \text{ policzmy więc } \mathbb{E}(X^2).$$

$$\mathbb{E}X^2 = \frac{m^2}{m}\Big[\big(1^{2k} + (2^k-1^k)^2 + \dots + (m_1^k - (m_1-1)^k)^2\big) +$$
$$+ \big(1^{2k} + \qquad \dots \qquad + (m_2^k - (m_2-1)^k)^2\big) +$$
$$\dots$$
$$+ \big(1^{2k} + \qquad \dots \qquad + (m_n^k - (m_n-1)^k)^2\big)\Big] \leq$$

$$\leq m\Big[\big(k1^{2k-1} + k2^{k-1}(2^k-1^k) + \dots + k\,m_1^{k-1}(m_1^k-(m_1-1)^k) +$$
$$+ \dots\Big] \leq$$

dla $a > b > 0$

$$a^k - b^k$$
$$\shortparallel$$
$$(a-b)(a^{k-1} + a^{k-2}b + \dots + ab^{k-2} + b^{k-1})$$
$$\wedge$$
$$(a-b)\cdot k\,a^k$$

$$\leq m\Big[k\,m_1^{2k-1} + k\,m_2^{2k-1} + \dots + k\,m_n^{2k-1}\Big] =$$

$$= km\cdot F_{2k-1} = k\,F_1\,F_{2k-1}$$

<u>Fakt</u>

$$F_1 \cdot F_{2k-1} \leq n^{1-\frac{1}{k}}(F_k)^2$$

Dowód to 3 linie przekształceń.

Zatem

$$Var(Y_i) = \frac{Var(X)}{s_1} \leq \frac{\mathbb{E}(X^2)}{s_1} \leq \frac{k\,n^{1-\frac{1}{k}}F_k^2}{s_1}.$$

$$\text{Z Czebyszewa } \mathbb{P}\big[|Y_i - F_k| > \lambda F_k\big] \leq \frac{Var(Y_i)}{\lambda^2 F_k^2} \leq$$

$$\leq \frac{k\,n^{1-\frac{1}{k}}F_k^2}{\lambda^2 F_k^2\cdot s_1} \doteq \frac{1}{8}$$

Potem z Chernoffa tak jak poprzednio.

~~[illegible]~~

Jeśli m nie jest znane, to za każdym razem, przy a_k z prawd. $\frac{1}{k}$ wybieramy to, zmieniamy a_0 na a_k i resetujemy r na 1.

c.n.d.