Analiza Algorytmów 2019/2020
(zadania na laboratorium)

Wybór lidera (do 12 III)

Zadanie 1 — Przejrzyj materiały do wykładu i zaimplementuj symulator umożliwiający przetestowanie algorytmu wyboru lidera dla znanej liczby węzłów n (scenariusz drugi) oraz dla znanego ograniczenie górnego u na liczbę węzłów n (scenariusz trzeci). Możesz wykorzystać dowolny język programowania.

Zadanie 2 — Niech zmienna losowa L oznacza liczę slotów od rozpoczęcia algorytmu do czasu wyboru lidera. Wykorzystaj symulator z poprzedniego zadania, aby narysować rozkład empiryczny (histogram) zmiennej losowej L dla obu rozważanych scenariuszy. Dla scenariusza ze znanym ograniczeniem u rozważ trzy przypadki: $n = 2$, $n = u/2$, $n = u$. Uzasadnij wyniki. (10p)

Zadanie 3 — Dla scenariusza ze znaną liczbą węzłów n wykorzystaj symulator do oszacowania wartości $\mathbb{E}[L]$ oraz $\mathbb{V}ar[L]$. Sprawdź, czy wyniki są zgodne z wynikami teoretycznymi. (10p)

Zadanie 4 — Rozważmy scenariusz ze znanym ograniczeniem u. Zgodnie z notacją wprowadzoną w materiałach do wykładu przez $S_{L,n}$ oznaczamy zdarzenie, że w jednej rundzie algorytmu długości $L = \lceil \log_2 u \rceil$ udało się wybrać lidera, jeśli w systemie jest n węzłów. Zaproponuj odpowiednie doświadczenie i za pomocą symulacji potwierdź poprawność Twierdzenia 1 z materiałów do wykładu: $Pr[S_{L,n}] \geq \lambda \approx 0.579$. (10p)

Analiza strumieni danych

Przybliżone zliczanie (kody + raport wysłać na mojego maila do 2 IV)

MinCount($\mathfrak{M}$, h , k)

Inicjalizacja: ustaw k pozycji $M[1], \ldots, M[k]$ tablicy M na 1

for all $x \in \mathfrak{M}$ **do** ▷ $\mathfrak{M}$ to multizbiór
  if $h(x) < M[k] \wedge h(x) \notin M$ **then** ▷ *hasz* $h(x) \in [0, 1)$
    $M[k] \leftarrow h(x)$
    sort(M) ▷ *posortuj* M *rosnąco*
  end if
end for
if $M[k] == 1$ **then**
  Return: $\hat{n} \leftarrow |\{i : M[i] \neq 1\}|$
else
  Return: $\hat{n} \leftarrow (k-1)/M[k]$ ▷ $\hat{n}$ to oszacowanie liczby różnych elementów $\mathfrak{M}$
end if

Zadanie 5 — Przeczytaj notatki do wykładu dotyczące problemu przybliżonego zliczania. Następnie zaimplementuj algorytm MinCount$(k, h, \mathcal{M})$ i przetestuj jego działanie:

a) Rozważ multizbiory $\mathcal{M}_n = (S_n, m)$ takie, że $|S_n| = n$ oraz $n = 1, 2, \ldots, 10^4$.
Czy obecność powtórzeń ma wpływ na wartość estymacji $\hat{n}$ uzyskiwanej w algorytmie?

b) Dla $k = 2, 3, 10, 100, 400$ i multizbiorów z punktu a) narysuj wykres mający na osi poziomej wartości n a na osi pionowej stosunek $\hat{n}/n$.

c) Eksperymentalnie dobierz wartość k tak by w 95% przypadków $|\frac{\hat{n}}{n} - 1| < 10\%$.

(10p)

Zadanie 6 — Dla kilku różnych funkcji haszujących $h : S \to \{0, 1\}^B$ i różnych wartości parametru B przetestuj działanie algorytmu MinCount$(k, h, \mathcal{M})$. Postaraj się znaleźć funkcję haszującą h dla której wyniki algorytmu są istotnie gorsze i wyjaśnij z czego może wynikać utrata dokładności. (10p)

Zadanie 7 — Twoim zadaniem jest porównanie teoretycznych wyników dotyczących koncentracji estymatora $\hat{n}$ wykorzystanego w algorytmie MinCount$(k, h, \mathcal{M})$ uzyskanych przez **a)** nierówność Czebyszewa oraz **b)** nierówność Chernoffa, z wynikami symulacji.
Dla $n = 1, 2, \ldots 10^4$, $k = 400$ i $\alpha = 5\%, 1\%, 0.5\%$ przedstaw na wykresie wartości $\hat{n}/n$ (uzyskane w wyniku eksperymentów) oraz wartości $1 - \delta$ i $1 + \delta$ takie, że

$$Pr\left[1 - \delta < \frac{\hat{n}}{n} < 1 + \delta\right] > 1 - \alpha\,. \quad (10p)$$

Zadanie 8 — Zaimplementuj algorytm HyperLogLog z korektami i przetestuj jego działanie dla różnych wartości parametru m (liczba rejestrów) oraz różnych funkcji haszujących - stwórz wykresy analogiczne do tych z zadania 3. Porównaj dokładność estymacji algorytmów MinCount oraz HyperLogLog, gdy oba mają do dyspozycji taką samą ilość pamięci (możesz założyć, że potrzeba 5 bitów na rejestr w HyperLogLog oraz 32 bity na wartość hasza w MinCount). (20p)

Przybliżone sumowanie (kody + raport wysłać na mojego maila do ~~24 IV~~ 26 IV)

Zadanie 9 — Przeczytaj notatki do wykładu i zaimplementuj opisany tam algorytm przybliżonego sumowania. Wykonaj eksperymenty analogiczne do tych dla przybliżonego zliczania. W szczególności:

a) wykorzystaj metodę odwrotnej dystrybuanty i przetestuj różne funkcje haszujące,

b) sprawdź dokładność algorytmu dla różnych scenariuszy, na przykład kiedy wartości cech $\lambda_1, \lambda_2, \ldots, \lambda_n$ są takie same, losowo wybrane z rozkładu jednostajnego na przedziale (a, b) dla różnych a i b, większość wartości jest podobna ale istnieją wartości odstające,

c) porównaj wyniki eksperymentów z ograniczeniami wynikającymi z nierówności Czebyszewa.

(15p)

Zadanie 10 — Zaproponuj i przetestuj procedurę, która umożliwi w oszacowanie średniej wartości unikalnych elementów

$$\frac{\lambda_1 + \lambda_2 + \ldots + \lambda_n}{n}$$

w ramach jednego przebiegu po multizbiorze $\mathfrak{M}$. (10p)

Blockchain (kody + raport wysłać na mojego maila do 15 V)

Zadanie 11 — Przeczytaj notatki do wykładu. Niech $0 < q < 1/2$ oznacza prawdopodobieństwo wydobycia kolejnego bloku przez adwersarza odpowiadające części mocy obliczeniowej będącej w jego posiadaniu. Niech n oznacza liczbę potwierdzeń (nadbudowanych bloków) potrzebnych by uznać transakcję za potwierdzoną. Niech $P(n, q)$ oznacza prawdopodobieństwo, że adwersarz o mocy q będzie dysponował łańcuchem bloków równym lub dłuższym niż ten budowany przez uczciwych użytkowników w momencie, gdy nadbudowali oni blok zawierający rozważaną transakcję n blokami lub kiedykolwiek później.

a) Porównaj formuły na $P(n, q)$ uzyskane przez Nakamato i Grunspana. W szczególności:
   - ustal $n = 1, 3, 6, 12, 24, 48$ i przedstaw wykresy $P(n, q)$ w zależności od wartości q,
   - ustal dopuszczalne prawd.sukcesu adwersarza $P(n, q) = 0.1\%, 1\%, 10\%$ i narysuj wykresy przedstawiające jak należy dobrać wartość n w zależności od wartości q.

b) Zaimplementuj symulator ataku „double spending", który umożliwi eksperymentalne przybliżenie prawdopodobieństwa zdarzenia $P(n, q)$ w zależności od wartości n i q. Wskazówka: zaprojektuj eksperyment i powtórz go wielokrotnie (Metoda Monte Carlo). W raporcie starannie i dokładnie opisz ideę działania i kod symulatora.

c) Porównaj wyniki symulatora do wyników analitycznych (wykresy). Jeśli pojawią się rozbieżności postaraj się je wyjaśnić.

(30p)

Samostabilizacja (termin: ~~1 VI~~ 5 VI)

Zadanie 12 — Zaimplementuj symulator algorytmu Mutual Exclusion Dijkstry (strona 17). Dla ustalonego n oznaczającego liczbę procesów w pierścieniu, zweryfikuj, że startując z dowolnej konfiguracji początkowej algorytm przejdzie do legalnej konfiguracji. Jeśli z pewnej konfiguracji można przejść do kilku możliwych konfiguracji w zależności od tego, który proces wykona krok jako pierwszy, każde wykonanie powinno zostać zweryfikowane. Jaka jest największa liczba kroków do czasu osiągnięcia legalnej konfiguracji dla ustalonego n? Dla jakich wartości n możesz uzyskać odpowiedź w sensownym czasie? Za zadanie możesz otrzymać $3 \times$ N punktów, gdzie N oznacza największą wartość n, dla której uda Ci się zweryfikować algorytm.

Zadanie 13 — Rozważmy graf $G = (V, E)$. Dwa wierzchołki $v, w \in V$ nazywamy niezależnymi, jeśli $\{v, w\} \notin E$. Podzbiór $S \subseteq V$ wierzchołków nazywamy niezależnym, jeśli wszystkie jego elementy są parami niezależne. Wzorując się na algorytmie Maximal Machting podanym na wykładzie zaprojektuj, zaimplementuj i przetestuj samo-stabilizujący algorytm znajdujący maksymalny zbiór niezależny (ang. Maximal Independent Set) w nieskierowanym grafie spójnym. Podaj przekonywujące uzasadnienie poprawności algorytmu (formalny dowód - zadanie na ćwiczenia). Algorytmy znajdowania maksymalnego zbioru niezależnego mają wiele zastosowań, możesz np. myśleć o problemie przydziału częstotliwości w sieciach bezprzewodowych. (15p)

Funkcje tworzące (termin: 18 VI)

Zadanie 14 — Układając odpowiednie równanie rekurencyjne i wykorzystując funkcje tworzące dla danego n wyznacz liczbę wywołań linii 6 w poniższym algorytmie. Zweryfikuj odpowiedź eksperymentalnie. (15p)

```
1:  f(int n) {
2:  int s = 0;
3:  if ( n == 0 ) then return 1;
4:  else
5:      for int i = 0; i < n; i++  do
6:          s += f(i);
7:      end for
8:      return s;
9:  end if
10: }
```

Zadanie 15 — Algorytm otrzymuje na wejściu tablicę długości $n \geq 0$. Jeśli $n \geq 2$ dla każdego $k \in \{1, 2, 3, \ldots, n\}$ algorytm z prawdopodobieństwem 1/2 wywołuje się rekurencyjnie na pewnej losowej „podtablicy" długości k. Wykorzystując funkcje tworzące wyznacz średnią liczbę wywołań algorytmu dla danego n i przedstaw swoje wyliczenia. Zweryfikuj odpowiedź eksperymentalnie. (15p)