Algorytmy i struktury danych

Laboratorium 3

Termin wysłania (MS Teams): **19 kwietnia 2021 godz. 07:29**

Zadanie 1. [50%]

Zaimplementuj podane na wykładzie algorytmy RANDOMIZEDSELECT oraz SELECT. Program ma przyjmować jeden z dwóch parametrów wejściowych:

- wywołanie `./selection -r` oznacza operowanie na danych losowych długości n,
- wywołanie `./selection -p` oznacza operowanie na losowej permutacji zbioru $\{1, 2, \ldots, n\}$.

Po uruchomieniu, program wczytuje ze standardowego wejścia dwie liczby całkowite: n – długość danych wejściowych oraz $1 \leqslant k \leqslant n$ – numer szukanej statystyki pozycyjnej. Jako wyjście program generuje tablicę danych (zależnie od parametru uruchomienia) i sekwencyjnie uruchamia zaimplementowane algorytmy na wygenerowanych danych.

W czasie wykonywania algorytmów RANDOMIZEDSELECT oraz SELECT na standardowym wyjściu błędów powinien być wypisywany log, tak by można było odtworzyć działanie algorytmu. W szczególności powinien zawierać on tablicę danych, wartość parametru k, kolejno wybierane pivoty, wykonywane porównania i przestawienia oraz podsumowanie zawierające liczbę porównań oraz przestawień elementów. Wynikiem działania algorytmu wyświetlanym na standardowym wyjściu jest tablica z zaznaczoną k-tą statystyką pozycyjną.

Uwaga: w przypadkach, gdy nie jest to niezbędne dla znalezienia k-tej statystyki pozycyjnej, nie powinno następować sortowanie całej tablicy.

Przykładowe wywołanie:

```
./selection -p
7
3
2 1 [3] 5 4 6 7
```

Przeprowadź testy implementowanych algorytmów dla kliku różnych wartości parametru k[1] mające na celu zbadanie liczby wykonywanych porównań oraz przestawień elementów dla obu trybów („losowe dane” i „losowa permutacja”). W tym celu powtarzaj wywołania algorytmów dla tych samych danych wejściowych. Podobnie jak w przypadku Zadania 2. z Listy 2, testy wykonaj dla $n \in \{100, 200, \ldots, 10\,000\}$. Dla każdego n wykonaj m niezależnych powtórzeń, przyjmując np. $m = 100$. Dla wygody statystyki obejmujące liczbę porównań i przestawień mogą być oczywiście zapisywane do plików wyjściowych.

Po przeprowadzeniu eksperymentów wyciągnij wnioski na temat minimalnej i maksymalnej liczby porównań oraz przestawień elementów dla obu algorytmów, policz również średnią i odchylenie standardowe dla zebranych statystyk. Uzyskane wyniki przedstaw przy pomocy odpowiednich wykresów. [2]

Zadanie 2. [25%]

W poznanej na wykładzie i zaimplementowanej w Zadaniu 1. wersji algorytmu SELECT dzielimy tablicę wejściową na $\lceil n/5 \rceil$ grup po 5 elementów każda (ostatnia grupa może mieć mniej elementów). Zbadaj eksperymentalnie, jaki wpływ na złożoność algorytmu (liczbę porównań, liczbę przestawień elementów oraz czas działania) ma liczba grup, na którą dzielona jest tablica w kroku wyszukiwania mediany median.

[1] Np. dla trzech wartości k – „małe” k niezależne od n, np. ustalone $k < 50$, $k = n/2$, „duże” k bliskie n.

[2] Uwaga na marginesie - badając odchylenie standardowe uzyskanych danych eksperymentalnych, warto dodatkowo na jednym z wykresów umieścić wyniki pojedynczych symulacji, tj. po m punktów danych dla każdego n, wraz z krzywą łączącą punkty odpowiadające średniej dla każdego n – wówczas powinno być widać, „jak bardzo” wyniki pojedynczych symulacji skoncentrowane są wokół średniej.

W tym celu wykonaj podobne eksperymenty jak w Zadaniu 1. i wyznacz podobne statystyki dla wariantów algorytmu SELECT, w których w etapie wyznaczania mediany median tablica dzielona jest na $\lceil n/k \rceil$ grup dla $k \in \{3, 5, 7, \ldots 25\}$. Uzyskane wyniki przedstaw przy pomocy odpowiednich wykresów. Wyciągnij wnioski z przeprowadzonych eksperymentów.

Zadanie 3. [25%]

Zaimplementuj rekurencyjny algorytm wyszukiwania binarnego. Program na wejściu otrzymuje posortowaną tablicę długości n oraz wartość v i zwraca 1 w przypadku istnienia elementu v w tablicy oraz 0 w przeciwnym przypadku.

Przetestuj działanie Master Theorem dla zliczonej w trakcie działania algorytmu liczby porównań elementów oraz czasu wykonania dla różnych wartości v (np. dla elementów „blisko początku", „około środka" i „blisko końca" tablicy) oraz dla elementu spoza tablicy.[3] Test powtórz dla rozmiarów tablicy $n \in \{1000,\ 2000,\ \ldots,\ 100\,000\}$. Wykonaj także podobny eksperyment, gdzie szukany element wybierany jest losowo spośród elementów znajdujących się w tablicy (dla każdego n wykonaj odpowiednią liczbę powtórzeń oraz uśrednij otrzymane wyniki). Na podstawie uzyskanych wyników oszacuj czynnik $\mathrm{O}\,(1)$ dla obu tych statystyk (liczba porównań i czas wykonania) w każdym przypadku.

Zadanie 4.* [dodatkowe – 20%]

Wykorzystaj część lub całość algorytmu SELECT w algorytmie QUICKSORT oraz DUALPIVOTQUICKSORT.

Wykonaj testy dla różnych rozmiarów danych (podobne do tych z Zadania 2. z Listy 2). Wyciągnij wnioski z porównania średniego czasu wykonania oraz średniej liczby porównań dla algorytmów z Listy 2 z ich odpowiednikami wykorzystującymi algorytm SELECT.

Następnie zbadaj czas działania i liczę porównań dla „worst-case data" dla bazowej wersji algorytmu QUICKSORT, tj. dla danych, dla których „zwykły" QUICKSORT osiąga pesymistyczną złożoność rzędu $\Theta(n^2)$ (podobnie jak poprzednio, wykonaj testy dla $n \in \{100, 200, \ldots, 10\,000\}$). Wyciągnij wnioski z przeprowadzonych eksperymentów.

W każdym przypadku postaraj się eksperymentalnie wyestymować stałe stojące przy dominującym składniku w wyrażeniu opisującym asymptotyczną liczbę porównań wykonywanych przez badane algorytmy.

[3] W przypadku testów dla elementu spoza tablicy warto zaprojektować eksperymenty tak, żeby nie był to element większy lub mniejszy od wszystkich wartości w tablicy – tak, żeby w kolejnych wywołaniach rekurencyjnych algorytm nie dokonywał „tego samego wyboru". W tym celu można np. przeprowadzić testy dla tablicy liczb parzystych z zakresu od 1 do $2n$, a szukać w niej losowego elementu nieparzystego z tego zakresu.