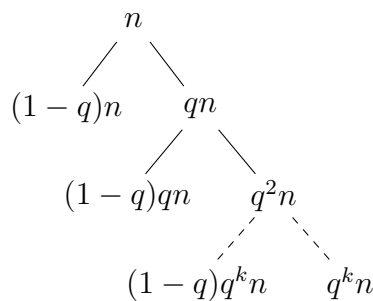


Algorytmy i struktury danych (Lista 5)

Zadanie 1 Udowodnij, że jeśli dla pewnego ustalonego q , takiego że $1/2 < q < 1$, podczas sortowania szybkiego, procedura partition, na każdym poziomie rekurencji podzieli elementy tablicy w stosunku $q : (1 - q)$ to algorytm wykona się w czasie $O(n \log n)$. Wskazówka: udowodnij, że głębokość rekurencji nie przekroczy $-\log n / \log q$ i zaniedbaj błędy zaokrągleń do wartości całkowitych.



$$\begin{aligned}
 q^k n &= 1 \\
 q^k &= \frac{1}{n} \\
 \log q^k &= \log n^{-1} \\
 k \log q &= -\log n \\
 k &= -\frac{\log n}{\log q}
 \end{aligned}$$

Zadanie 2 Ile porównań (zapisz wyniki w notacji O) wykona algorytm quicksort z procedurą partition w wersji Hoare'a, a ile w wersji z procedurę partition w wersji Lomuto dla danych: (a) posortowanych rosnąco, (b) posortowanych malejąco, (c) o identycznych kluczach?

Z procedurą Hoare dla posortowanych rosnąco wykona się $O(n \log n)$

porównań, a z procedurą Lomuto $O(n^2)$. Dla posortowanych malejąco wykona się $O(n^2)$ porównań, a z procedurą Lomuto $O(n \log n)$.

Dla danych o identycznych kluczach dla obu procedur wykona się $O(n^2)$ porównań w wersji

Zadanie 3 Napisz wzór na numer kubełka, do którego należy wrzucić liczbę x w sortowaniu kubełkowym, jeśli kubełków jest n , a elementy tablicy mieszczą się przedziale (a, b) . Numeracja zaczyna się od 0.

$$k = \left\lfloor \frac{n}{b - a} \cdot (x - a) \right\rfloor$$

Zadanie 4 Dla jakich danych sortowanie metodą kubełkową ma złożoność $O(n^2)$?

Dane, które są rozłożone mocno nierównomiernie np. grupują się w jeden kubełek.

Zadanie 5 Jak obliczyć k -tą od końca cyfrę w liczbie x ? Jak obliczyć ilość cyfr liczby x ? Przyjmujemy układ dziesiętny. Jak wyniki zmieniają się w układzie pozycyjnym gdzie różnych cyfr jest m a ich wartości x należą do przedziału $0 \leq x < m$?

$$k_w = \left\lfloor \frac{x}{10^k} \right\rfloor \mod 10$$

$$\text{dla } x = 123456, k = 2, k_w = 4$$

$$\frac{123456}{10^2} = 1234.56 \rightarrow 1234 \mod 10 = 4$$

m różnych cyfr:

$$k_w = \left\lfloor \frac{x}{m^k} \right\rfloor \mod m$$

$$\text{dla } x = 123456, k = 3, m = 6, k_w = 3$$

$$\frac{123456}{6^3} = 123.456 \rightarrow 123 \mod 6 = 3$$

Zadanie 6 Posortuj metodą sortowania pozycyjnego liczby: 101, 345, 103, 333, 432, 132, 543, 651, 791, 532, 987, 910, 643, 641, 12, 342, 498, 987, 965, 322, 121, 431, 350. W pisemnym rozwiązaniu pokaż, jak wygląda zawartość kolejek, za każdym razem, gdy tablica wyjściowa jest pusta i wszystkie liczby znajdują się w kolejkach, oraz jak wygląda tablica wyjściowa, za każdym razem, gdy sortowanie ze względu na kolejną cyfrę jest już zakończone.

0 : (910, 350)
 1 : (101, 651, 791, 641, 121, 431)
 2 : (432, 132, 532, 12, 342, 322)
 3 : (103, 333, 543, 643)
 4 : ()
 5 : (345, 965)
 6 : ()
 7 : (987, 987)
 8 : (498)
 9 : ()

(910, 350, 101, 651, 791, 641, 121, 431, 432, 132, 532, 12, 342, 322, 103, 333, 543, 643, 345, 965, 987, 987, 498)

0 : (101, 103)
 1 : (910)
 2 : (121, 12, 322)
 3 : (431, 432, 132, 532, 333)
 4 : (641, 342, 543, 643, 345)
 5 : (350, 651)
 6 : (965)
 7 : ()
 8 : (987, 987)
 9 : (791, 498)

(101, 103, 910, 121, 12, 322, 431, 432, 132, 532, 333, 641, 342, 543, 643, 345, 350, 651, 965, 987, 987, 791, 498)

0 : (12)
 1 : (101, 103, 121, 132)
 2 : ()
 3 : (322, 333, 342, 345, 350)
 4 : (431, 432, 498)
 5 : (532, 543)
 6 : (641, 643, 651)
 7 : (791)
 8 : ()
 9 : (910, 965, 987, 987)

(12, 101, 103, 121, 132, 322, 333, 342, 345, 350, 431, 432, 498, 532, 543, 641, 643, 651, 791, 910, 965, 987, 987)

Zadanie 7 (2pkt) Które z procedur sortujących:

1. insertionSort - stabilny, zachowuje kolejność bo przestawia elementy od lewej do prawej strony nie zmieniającą tych o tej samej wartości
2. quickSort - niestabilny, [2, 1, 1] 1 z ostatniego indeksu zostanie zamieniona z 2 z 0 indeksu, potem gdy weźmiemy [1,2] to nie będzie już swapa
3. heapSort - niestabilny, [1,1] budowanie kopca nic nie zmieniło, ale na etapie końcowym zamieni nam 1 miejscami i wywoła przesiewanie, które isę przerwie i nic nie zmieni ;')
4. mergeSort - stabilny, ponieważ przy łączeniu dwóch posortowanych tablic zachowuje kolejność elementów o tej samej wartości
5. countingSort - stabilny, ponieważ na końcowym etapie idąc od końca tablicy wejściowej wpisujemy elementy na odpowiednie indeksy przez co jeśli dana liczba była później od innej to tak pozostanie (gdy mają te same wartości)
6. radixSort - stabilny, ponieważ wykorzystuje się kolejki FIFO (first in first out)
7. bucketSort - stabilny, idziemy od końca i do bufora wpisujemy od ostatniego indeksu gdzie elementy z danego kubku powinny być w tablicy wyjściowej

są stabilne? W każdym przypadku uzasadnij stabilność lub znajdź konkretny przykład danych, dla których algorytm nie zachowa się stabilnie.

Zadanie 8 Napisz funkcję `void counting_sort(node* lista, int m)`; sortującą przez zliczanie listę linkowaną liczb całkowitych nieujemnych mniejszych od m . Procedura nie powinna usuwać ani tworzyć nowych węzłów, tylko sprytnie zmieniać pola next wykorzystując tylko $O(m)$ dodatkowej pamięci na wskaźniki.

Zadanie 9 (algorytm Hoare'a) Korzystając funkcji `int partition(int t[], int n)` znanej z algorytmu sortowania szybkiego napisz funkcję `int kty(int t[], int n)`, której wynikiem będzie k -ty co do wielkości element początkowo nieposortowanej tablicy t . Średnia złożoność Twojego algorytmu powinna wynieść $O(n)$.