Algorytmy i struktury danych (Sortowanie i kopce)

Przyjmując, że $t1[] = 1, 2, 3, 4, 5, 6, 7$ oraz $t2[] = 7, 6, 5, 4, 3, 2, 1$ i stosując algorytmy sortujące ściśle wg procedur z pliku sorty2020.cc i wykonaj polecenia:

Zadanie 1 Ile dokładnie porównań (między elementami tablicy) wykona insertionSort(t2) a ile insertionSsort(t1)?

Posortowana: n-1 porównań. Nieposortowana: $\frac{n(n-1)}{2}$ porównań.
Dla t1 wykona 6 porównań. Dla t2 wykona 21 porównań.

Zadanie 2 Ile co najwyżej porównań (między elementami tablic) wykona procedura scalająca merge dwie tablice n-elementowe?

W najgorszym przypadku będzie musiała wykonać $2n - 1$ porównań, gdzie elementy rosnące wystepują naprzemiennie, więc będziemy porównywać każdy element do momentu aż nie trafimy na większy i go nie wpiszemy do tablicy wyjściowej.

Zadanie 3 Jaka jest pesymistyczna złożoność czasowa procedury mergeSort? Odpowiedź uzasadnij.

Dla wersji rekurencyjnej i iteracyjnej: $O(nlogn)$

Twierdzenie o rekurencji uniwersalnej: $T(n) = 2T(\frac{n}{2}) + n$,
gdzie liczba podproblemów to $a = 2$, rozmiar podproblemu to $b = 2$, i mamy n jako liczbę porównań.

$f(n) = \theta(n^{log_2 2}) = \theta(n)$

$T(n) = \theta(nlogn)$

Porównujemy elementy i scalamy je w jedną tablicę. W każdym kroku wykonujemy n porównań. W każdym kroku dzielimy tablicę na dwie części o połowie długości. W sumie wykonujemy $logn$ kroków, scalań. Złożoność czasowa zawsze jest $O(nlogn)$.

Zadanie 4 Ile co najwyżej porównań (między elementami tablicy) wykona procedura partition?

Przy jednym wywołaniu partition wykona się maksymalnie $n + 1$ porównań. To przypadek, gdy nasze odwrócone elementy są najbliżej pivota lub musimy zamienić stronami każdy z elementów. W obu przypadkach musimy każdą wartość po stronie prawej i lewej porównać z pivotem. Mimalnie są zawsz 2 porównania.

Zadanie 5 Jak jest średnia a jaka pesymistyczna złożoność quickSort. Odpowiedź uzasadnij.

Średnia złożoność czasowa: $O(nlogn)$

Pesymistyczna złożoność czasowa: $O(n^2)$

Pesymistyczną złożoność osiągamy, gdy jako pivot zostanie wybrana wartość najmniejsza lub największa w tablicy. Złożoność kwadratowa wynika ze wzoru

$T(n) = n + 1 + T(n-1) -> T(n) = \frac{n(n-1)}{2}$

czyli możemy posłużyć się sumą ciągu arytmetycznego.

$T(n) = O(n^2)$

Zadanie 6 Jaka jest złożoność funkcji buildheap? Przeprowadź dowód - uzasadnij swoją odpowiedź.

Złożoność buildheap wynosi $O(n)$.

Jeśli tablica ma n elementów to $\frac{n}{2}$ elementów jest liśćmi. Do porównań potrzebujemy węzła z dziećmi, więc ich nie mamy z czym porównywać. Pozosłe węzły możemy porównywać z dziećmi, a przy przesiewaniu z dalszymi potomkami. Stąd:

$2(\frac{n}{2} \cdot 0 + \frac{n}{4} \cdot 1 + \frac{n}{8} \cdot 2 + ...)$

Mnożenie przez dwa wynika z sytuacji gdy porównujemy oboje dzieci z rodzicem.

$$2\sum_{i=1}^{h-1} \frac{n}{2^{i+1}} \cdot i$$
$$2(\frac{n}{4} \cdot 1 + \frac{n}{8} \cdot 2 + \frac{n}{16} \cdot 3 + \frac{n}{32} \cdot 4 + \dots)$$
$$\frac{n}{2}(\frac{1}{1} + \frac{2}{2} + \frac{3}{4} + \frac{4}{8} + \dots)$$
$$\frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots = \frac{\frac{1}{1}}{1 - \frac{1}{2}} = 2$$
$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots = 1$$
$$\frac{1}{4} + \frac{1}{8} + \dots = 0.5$$
$$\frac{1}{8} + \dots = 0.25$$
$$\dots$$

$$\frac{n}{2} \cdot \frac{2}{1 - \frac{1}{2}} = \frac{n}{2} \cdot 4 = 2n = O(n)$$

Zadanie 7 Ile dodatkowej pamięci wymaga posortowanie tablicy n-elementowej za pomocą algorytmu: radixsort dodatkowa pamięć O(n + k)

W punktach (e), (f), (g) zakładamy, że ilość kubełków jest m, a liczby do posortowania mają nie więcej niż k cyfr.

Algorytm	Pamięć
`mergesort`	$O(n)$
`quicksort`	$O(n)$
`heapsort`	$O(1)$
`insertionsort`	$O(1)$
`countingsort`	$O(n+m)$
`bucketsort`	$O(n+m)$
`radixsort`	$O(n+k)$

Zadanie 8 Jaka jest średnia a jaka pesymistyczna złożoność czasowa algorytmu:

Algorytm	Średnia	Pesymistyczna
`mergesort`	$O(n \log n)$	$O(n \log n)$
`quicksort`	$O(n \log n)$	$O(n^2)$
`heapsort`	$O(n \log n)$	$O(n \log n)$
`insertionsort`	$O(n^2)$	$O(n^2)$
`countingsort`	$O(n+m)$	$O(n+m)$
`bucketsort`	$O(n+m)$	$O(n^2)$
`radixsort`	$O(nk)$	$O(nk)$

Zadanie 9 Udowodnij, że wysokość (ilość poziomów na których występują węzły) kopca n-elementowego wynosi $\lfloor log_2 n \rfloor + 1$

Maksymalna ilość węzłów w kopcu o wysokości h:

$$n(h) = 2^h - 1$$

$$n(h-1) = 2^{h-1} - 1$$

Minimalna ilość węzłów w kopcu o wysokości h:

$$n(h-1) + 1 = 2^{h-1} - 1 + 1 = 2^{h-1}$$

Ilość węzłów w kopcu o wysokości h:

$$2^{h-1} \leq n < 2^h$$

$$h - 1 \leq \log_2 n < h$$

$$h \leq \log_2 n + 1 < h + 1$$

$$\lfloor \log_2 n \rfloor + 1$$

Zadanie 10 Który element tablicy t jest (a) lewym dzieckiem (b) prawym dzieckim (c) ojcem, elementu t[i] w procedurze heapsort?

$$(a) t[2i+1]$$
$$(b) t[2i+2]$$
$$(c) t[\lfloor \frac{i-1}{2} \rfloor]$$

Zadanie 11 Czy ciąg 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 jest kopcem?

Nie jest to kopiec ze względu na to że 7 jest większa od swojego rodzica, którym jest 6.

Zadanie 12 Zilustruj działanie procedury buildheap dla ciągu 5,3,17,10,84,19,6,22,9. Narysuj na kartce wygląd tablicy/kopca po każdym wywołaniu procedury przesiej.

Zadanie 13 Zasymuluj działanie polifazowego mergesorta dla tablicy 9,22,6,19,14,10,17,3,5. Na każdym etapie sortowania scala się sąsiadujące listy rosnące.

$$9, 22||6, 19||14||10, 17||3, 5$$
$$6, 9, 19, 22||10, 14, 17||3, 5$$
$$6, 9, 10, 14, 17, 19, 22||3, 5$$
$$3, 5, 6, 9, 10, 14, 17, 19, 22$$

Zadanie 14 Zasymuluj działanie mergesort(t2).

$$\begin{aligned} t2[] = & 7, 6, 5, 4, 3, 2, 1 \\ & 7, 6, 5||4, 3, 2||1 \\ & 6, 7||3, 4||1||2, 5 \\ & 3, 4, 6, 7||1||2, 5 \\ & 1||2, 3, 4, 5, 6, 7 \\ & 1, 2, 3, 4, 5, 6, 7 \end{aligned}$$

Zadanie 15 Zasymuluj działanie partition(t2,7).

$$\begin{aligned} t2[] = & 7, 6, 5, 4, 3, 2, 1 \\ x = 4, k = 0, n = 6, swap => & 1, 6, 5, 4, 3, 2, 7 \\ x = 4, k = 1, n = 5, swap => & 1, 2, 5, 4, 3, 6, 7 \\ x = 4, k = 2, n = 4, swap => & 1, 2, 3, 4, 5, 6, 7 \\ return\ 3; \end{aligned}$$

Zadanie 16 Zasymuluj działanie partition(t2,7) w przypadku gdyby piwotem zamiast t[n/2] było t[0]

$$\begin{aligned} t2[] = & 7, 6, 5, 4, 3, 2, 1 \\ x = 7, k = 6, n = 6 \\ return\ 6; \end{aligned}$$

Zadanie 17 Patrz 5.

Zadanie 18 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.

$$\text{Kubełek } k = \lfloor \frac{n}{b-a}(x-a) \rfloor$$

Zadanie 19 Jak obliczyć k-tą od końca cyfrę w liczby x? Jak obliczyć ilość cyfr liczby x? Przyjmujemy układ dziesiętny. Jak wyniki zmienią się w układzie pozycyjnym o 1000 cyfr?

m różnych cyfr:

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

dla $x = 123456, k = 3, m = 1000, k_w = 3$

$$\frac{123456}{1000^3} = 0.123456 \to 0 \mod 1000 = 0$$