Algorytmy i Struktury Danych

Sortowanie i kopce (przygotowanie do kolokwium)

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:

1. Ile dokładnie porównań (między elementami tablicy) wykona `insertion_sort(t2)` a ile `insertion_sort(t1)`?
2. Ile co najwyżej porównań (między elementami tablic) wykona procedura scalająca `merge` dwie tablice n-elementowe?
3. Jaka jest pesymistyczna złożoność czasowa procedury `merge_sort`? Odpowiedź uzasadnij.
4. Ile co najwyżej porównań (między elementami tablicy) wykona procedura `partition`?
5. Jak jest średnia a jaka pesymistyczna złożoność `quick_sort`. Odpowiedź uzasadnij.
6. Jaka jest złożoność funkcji `buildheap`? Przeprowadź dowód – uzasadnij swoją odpowiedź.
7. Ile dodatkowej pamięci wymaga posortowanie tablicy n-elementowej za pomocą algorytmu: (a) `mergesort` (b) `quicksort` (c) `heapsort` (d) `insertionsort` (e) `countingsort` (f) `bucketsort` (g) `radixsort`. W punktach (e), (f), (g) zakładamy, że ilość kubełków jest m, a liczby do posortowania mają nie więcej niż k cyfr.
8. Jaka jest średnia a jaka pesymistyczna złożoność czasowa algorytmu: (a) `mergesort` (b) `quicksort` (c) `heapsort` (d) `insertionsort` (e) `countingsort` (f) `bucketsort` (g) `radixsort`? Zakładamy oznaczenia z poprzedniego zadania.
9. Udowodnij, że wysokość (ilość poziomów na których występują węzły) kopca n-elementowego wynosi $\lfloor \log_2 n \rfloor + 1$.
10. Który element tablicy `t` jest (a) lewym dzieckiem (b) prawym dzieckim (c) ojcem, elementu `t[i]` w procedurze `heapsort`?
11. Czy ciąg `{23, 17,14,6,13,10,1,5,7,12}` jest kopcem?
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`.
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.
14. Zasymuluj działanie `mergesort(t2)`;
15. Zasymuluj działanie `partition(t2,7)`.
16. Zasymuluj działanie `partition(t2,7)` w przypadku gdyby piwotem zamiast `t[n/2]` było `t[0]`.
17. Wykaż, że pesymistyczna złożoność `quicksort` wynosi $O(n^2)$.
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.
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?