Algorytmy i Struktury danych (2023)

Lista zadań 4 (rekurencja uniwersalna, mergesort, heapsort, drzewa)

1. Skorzystaj z metody rekurencji uniwersalnej i podaj dokładne asymptotyczne oszacowania dla następujących rekurencji:
   (a) $T(n) = 2T(n/4) + \sqrt{n}$,
   (b) $T(n) = 3T(n/4) + n$,
   (c) $T(n) = 8T(n/4) + n\sqrt{n}$,
   (d) $T(n) = 2T(\sqrt{n}) + 1$ (potrzebna zamiana zmiennych).

2. Czas działania algorytmu A opisany jest przez rekurencję $T(n) = 7T(n/2) + n^2$. Algorytm konkurencyjny A' ma czas działania $T'(n) = aT'(n/4) + n^2$. Jaka jest największa liczba całkowita a, przy której A' jest asymptotycznie szybszy niż A?

3. Rozważmy warunek regularności $af(n/b) \leqslant cf(n)$ dla pewnej stałej $c \leqslant 1$, który jest częścią przypadku 3 twierdzenia o rekurencji uniwersalnej. Podaj przykład prostej funkcji $f(n)$, które spełnia wszystkie warunki twierdzenia o rekurencji uniwersalnej z wyjątkiem warunku regularności.

4. Zasymuluj działanie polifazowego mergesorta dla tablicy:
   `{9,22,6,19,21,14,10,17,3,5,60,30,29,1,8,7,6,15,12}`.
   W sortowaniu polifazowym na każdym etapie sortowania scala się sąsiadujące podciągi rosnące, to znaczy: w pierwszym przebiegu `{9,22}` z `{6,19,21}`, `{14}` z `{10,17}` itd..

5. a) Czy tablica posortowana malejąco jest kopcem?
   b) Czy ciąg `{23, 17,14,6,13,10,1,5,7,12}` jest kopcem?

6. Zilustruj działanie procedury `buildheap` dla ciągu `{5,3,17,10,84,19,6,22,9,14,3}`. Narysuj na kartce wygląd tablicy i kopca po każdym wywołaniu procedury `przesiej`.

7. Metodą jak na wykładzie, udowodnij, że procedura `build_heap` działa w czasie $O(n)$.

8. Udowodnij, że wysokość kopca n-elementowego wynosi $\lfloor \log_2 n \rfloor + 1$.

9. (2 pkt) Podaj ideę algorytmu, jak przy pomocy struktury kopca, złączyć k posortowanych list jednokierunkowych o łącznej ilości elementów n, w jedną posortowaną listę, za używając nie więcej niż $3n \log_2 k$ porównań.

10. W pliku `spis.txt` umieszczone są w przypadkowej kolejności wszystkie liczby całkowite od `1` do `n` za wyjątkiem jednej (n jest bardzo duże). Jak wyliczyć brakującą liczbę w czasie liniowym nie wykorzystując dodatkowej pamięci plikowej ani RAM za wyjątkiem kilku zmiennych typu `int`? Wskazówka: w C++ działania `+ - *` na liczbach całkowitych obywają się modulo 2^{32}.

11. Niech F_n oznacza ilość różnych kształtów drzew binarnych o n węzłach. Rysując drzewa, łatwo sprawdzić, że $F_0 = 1$, $F_1 = 1$, $F_2 = 2$, $F_3 = 5$, itd. Nie korzystając z internetu:
    (a) Znajdź wzór wyrażający F_n przez $F_0, F_1, F_2, ..., F_{n-1}$ dla $n = 2, 3, 4$ a potem ogólnie.
    (b) Zaprojektuj (na kartce) procedurę, która oblicza kolejne wyrazy ciągu F_n, zapisuje je w tablicy i korzysta z nich przy obliczaniu następnych wyrazów.
    (c) Przeanalizuj ile mnożeń trzeba wykonać, by obliczyć wyrazy od F_1 do F_n . Czy da się ją zapisać w postaci $O(n^k)$ dla pewnego k?
    (d) Jaka byłaby złożoność algorytmu rekurencyjnego, który nie korzysta z wartości zapisanych w tablicy, tylko oblicza je ponownie. Czy da się ją zapisać jako $O(n^k)$?

12. Wykonaj zadanie 11 (a) (b) (c) dla drzew trynarnych (dzieci: left, down, right).