Algorytmy i Struktury danych (2021)

Lista zadań 4 (rekurencja uniwersalna, mergesort, drzewa)

1. Skorzystaj z metody rekurencji uniwersalnej i podaj dokładne asymptotyczne oszacowania dla następujących rekurencji:

   (a) $T(n) = 4T(n/2) + n$,
   (b) $T(n) = 4T(n/2) + n^2$,
   (c) $T(n) = 4T(n/3) + n^3$,

2. Korzystając z twierdzenia o rekurencji uniwersalnej rozwiąż następujące zależności:

   (a) $T(n) = 5T(n/3) + n$,
   (b) $T(n) = 9T(n/3) + n^2$,
   (c) $T(n) = 6T(n/3) + n^2$,
   (d) $T(n) = T(n/2) + 1$,
   (e) $T(n) = 2T(\sqrt{n}) + 1$ (potrzebna zamiana zmiennych).

3. 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?

4. Rozważmy warunek regularności $af(n/n) \leqslant cf(n)$ 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.

5. 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..

6. W tablicy `t[n]` umieszczone są w przypadkowej kolejności wszystkie liczby całkowite od `1` do `n+1` za wyjątkiem jednej. Napisz funkcję `int brakujaca(int t[],int n)` zwracającą brakującą liczbę całkowitą, tak aby ilość potrzebnej pamięci nie zależała od n, a czas wykonania był linowy tzn. $\Theta(n)$.

7. (2pkt) Niech $T_2(n)$ oznacza ilość różnych kształtów drzew binarnych o n węzłach. $T(0) = 1$, $T(1) = 1$, $T(2) = 2$, $T(3) = 5$, ...
   (a) Znajdź wzór rekurencyjny wyrażający $T_2(n)$ przez $\{T_2(i) : 0 \leq i < n\}$.
   (b) Napisz w javascript procedurę rekurencyjną, która używa tego wzoru. Zastosuj `BigNum`.
   (b') Przyspiesz twoją procedurę korzystając z symetrii wzoru rekurencyjnego (a).
   (c) Napisz w javascript procedurę nierekurencyjną, która oblicza po kolei wyrazy ciągu $T_2(n)$ i zapisuje je w tablicy. Przy obliczaniu kolejnych wyrazów, korzysta w poprzednio zapisanych wyników.
   (d) Sprawdź ile wartości ciągu $T_2(n)$ możesz uzyskać procedurą (b), (b'), (c) w czasie 2 minut. Przy każdej wartości $T_2(n)$ wypisuj czas jej obliczenia.

8. (2pkt) Wykonaj punkty (a), (c), (d) zadania 7 dla drzew trynarnych, których węzeł wygląda tak: `struct node3{int k; node3* left; node3* middle; node3* right}`.

9. * Dla ambitnych (3pkt) Napisz procedurę `T(i,n)` obliczającą ilość kształtów drzew i-narnych o n węzłach.