

Spis treści

1	Zagadnienia	2
2	I termin 2010	6
2.1	zadanie 1	6
3	II termin 2010	7
3.1	zadanie 1	7
3.2	zadanie 2	7
3.3	zadanie 3	8
3.4	zadanie 4	8
3.5	zadanie 5	8
3.6	zadanie 6	9
3.7	zadanie 14	9

Rozdział 1

Zagadnienia

[TODO: trzeba podzielić na części egzaminu]

- Algorytmy
 - mnożenie po rosyjsku
 - szybkie potęgowanie
 - szybkie obliczanie liczb Fibonacciego
 - algorytmy sortowania: select i insert
 - maszyna RAM jako model obliczeń
 - notacja asymptotyczna
- Algorytmy zachłanne
 - problem wydawania reszty
 - algorytmy znajdowania minimalnego drzewa rozpinającego
 - * algorytm Prima
 - * algorytm Kruskala
 - * algorytm Boruvki
 - szeregowanie zadań
 - szeregowanie zadań z deadlineami
- Metoda Dziel i Zwyciężaj
 - algorytmy sortowania (mergesort, quick sort)
 - algorytmy mnożenia długich liczb (Karatsuby)
- Metoda Dziel i Zwyciężaj (cd)
 - algorytmy mnożenia długich liczb oparte na podziale na wiele części
 - jednoczesne znajdowanie maksimum i minimum
 - sieć permutacyjna Waksmana-Benesa
 - znajdowanie najbliższej pary punktów na płaszczyźnie
 - zachłanny algorytm aproksymacyjny dla problemu Set Cover
- Programowanie dynamiczne
 - obliczanie współczynnika dwumianowego

- stokrotki
- znajdowanie najkrótszych dróg w grafie między wszystkimi parami wierzchołków
- algorytm Floyd-Warshalla
- najdłuższy wspólny podciąg
- Programowanie dynamiczne (cd)
 - optymalna kolejność mnożenia macierzy
 - przynależność słowa do języka bezkontekstowego
 - drzewa rozpinające drabin
- Dolne granice (cd)
 - gra z przeciwnikiem - jednoczesne znajdowanie minimum i maksimum
 - kopiec
 - * definicja
 - * implementacja tablicowa
 - * realizacja operacji kopcowych
 - * zastosowania: heapsort, kolejka priorytetowa
 - metody dowodzenia dolnych granic
 - * argumentacja teorii informatycznej
 - drzewa decyzyjne - sortowanie; dolna granica w najgorszym i średnim przypadku
 - liniowe drzewa decyzyjne - problem: niepowtarzalność elementów
 - * redukcje
 - nieaproxymowalność problemu komiwojażera
- Kopiec (cd)
 - kopiec minmax
 - Quicksort
 - metody wyboru pivota (deterministyczna, losowa, mediana z małej próbki)
 - analiza oczekiwanego czasu działania przy losowym pivocie
 - usprawnienia Quicksorta
 - * trójpodział
 - * eliminacja rekursji
 - * quicksort w miejscu
- Sortowanie
 - przez zliczanie
 - kubełkowe
 - leksykograficzne ciągów jednakowej długości
- Quicksort
 - prostsza metoda analizy oczekiwanego czasu działania Quicksorta przy losowym wyborze pivota
- Selekcja
 - znajdowanie drugiego co do wielkości elementu

- metoda Hoare’a
- metoda magicznych piątek
- Słowniki
 - drzewa zrównoważone
 - * drzewa AVL (definicja, realizacja operacji słownikowych)
 - * drzewa czerwono-czarne (definicja, realizacja operacji słownikowych)
 - selekcja
 - * zrandomizowany algorytm oparty na próbkowaniu
 - sortowanie ciągów niekoniecznie jednakowej długości z zastosowaniem do sprawdzania izomorfizmu drzew
- Hashowanie
 - przykłady funkcji hashujących
 - kolizje
 - pamiętanie elementów kolidujących (nawlekanie, adresowanie otwarte)
 - struktury samoorganizujące się
 - * idea
 - * heurystyki dla list samoorganizujących się
 - * drzewa rozchylane (splay)
 - * analiza zamortyzowania ciągów operacji splay
 - kopce dwumianowe (wersja leniwa)
 - * analiza zamortyzowana kosztu operacji
 - kopce Fibonacciego
 - * zastosowanie: algorytm Dijkstry
 - * wpływ kaskadowego odcięcia drzew na koszt operacji
 - B-drzewa
 - * definicja
 - * liczba operacji i/o jako miara złożoności
 - * realizacja operacji słownikowych
 - 2-3-drzewa i ich związek z drzewami czerwono czarnymi
 - łączalne kolejki priorytetowe
 - * kopce dwumianowe (wersja gorliwa)
- Hashowanie (cd)
 - usuwanie kolizji w otwartym adresowaniu
 - * metoda liniowa
 - * metoda kwadratowa
 - * podwójne hashowanie
 - oczekiwana liczba prób przy poszukiwaniu elementu
 - uniwersalne rodziny funkcji hashujących
- Słowniki statyczne
 - optymalne drzewa BST

- statyczny słownik Fredman, Komlosa, Szemerédi’ego
- Drzewce
 - definicja
 - istnienie
 - realizacja operacji słownikowych
- Statyczny słownik Fredman, Komlosa, Szemerédi’ego (cd)
 - idea konstrukcji
 - istnienie odpowiednich funkcji hashujących
 - znajdowanie takich funkcji

Rozdział 2

I termin 2010

2.1. zadanie 1

Rozdział 3

II termin 2010

3.1. zadanie 1

Rozwiąż poniższe równanie:

$$T(n) = \begin{cases} c & \text{jeśli } n = 1 \\ 2 \cdot T(n-1) + 1 & \text{jeśli } n > 1 \end{cases}$$

Czy funkcja $T(n)$ jest $O(n^{\log_2 n})$?

Rozwiązanie równania rekurencyjnego:

$$\begin{aligned} T(n) &= 2 \cdot T(n-1) + 1 \\ &= 2 \cdot (2 \cdot T(n-2) + 1) + 1 \\ &= 2^2 \cdot T(n-2) + 2 + 1 \\ &= 2^3 \cdot T(n-3) + 2^2 + 2 + 1 \\ &\dots \\ &= 2^n \cdot T(1) + 2^{n-1} + \dots + 2^2 + 2 + 1 \\ &= 2^n \cdot c + 2^n - 1 \end{aligned}$$

Pokażemy że $T(n) \notin O(n^{\log_2 n})$:

Z powyżej rozwiązanej rekurencji widać że: $T(n) \in \Omega(2^n)$,

$$\exists_{n_0 \in \mathbb{N}} \forall_{n > n_0} n^{\log_2 n} = (2^{\log_2 n})^{\log_2 n} = 2^{(\log_2 n)^2} < 2^n \quad (3.1)$$

Wynika z tego, że $n^{\log_2 n}$ nie jest ograniczeniem górnym.

3.2. zadanie 2

Wyznacz z dokładnością do Θ (przy jednorotnym kryterium) poniższego fragmentu algorytmu:

```
suma ← 0
for i ← 1 to n do
```

```

    k ← 1
    while k ≤ i do
        read(x)
        suma ← suma + x · k
        k ← k + k
    end while
end for

```

$$\sum_{i=0}^n \log_2 i = \log_2 1 + \dots + \log_2 n = \log_2(1 \cdot \dots \cdot n) = \log_2(n!)$$

Algorytm jest $\Theta(\log_2(n!))$

3.3. zadanie 3

Załóżmy, że w definicji drzewa czerwono-czarnego zmienimy warunek mówiący iż dzieci czerwonego ojca są czarne na warunek:
 dzieci czerwonego ojca, którego ojciec też jest czerwony są czarne
 Określ jak zmieni się (z dokładnością) do stałego czynnika maksymalna wysokość tak zdefiniowanych drzew?

Przy oryginalnym założeniu:

$$h \leq 2 \cdot \log(n-1)$$

Po zamianie założeń, minimalna liczba czarnych wierzchołków w każdej ścieżce od korzenia do liścia, zmieniła się z $\frac{h}{2}$ na $\frac{h}{3}$. Warunek na h przyjmuje teraz postać:

$$h \leq 3 \cdot \log(n-1)$$

3.4. zadanie 4

Z ilu drzew może składać się kopiec dwumianowy (wersja eager) zawierający 49 elementów?

Ponieważ jest to kopiec typu eager nie może mieć dwóch drzew dwumianowych tego samego stopnia. Aby przekonać się z ilu drzew dwumianowych się on składa, zamieńmy liczbę jego elementów na liczbę o binarną.

$$49_{10} = 32_{10} + 16_{10} + 1_{10} = 2_{10}^5 + 2_{10}^4 + 2_{10}^0 = 110001_2 \quad (3.2)$$

Widzimy, że kopiec ten składa się z trzech drzew: B_5 , B_4 i B_1 .

3.5. zadanie 5

Dla której z poniżej podanych struktur danych koszt (najgorszego przypadku) wykonania operacji $find(i)$ sprawdzającej czy klucz i jest pamiętany w strukturze jest $O(\log n)$, gdzie n jest rozmiarem struktury?

- (a) drzewo binarnych przeszukiwań,
- (b) drzewo AVL
- (c) kopiec
- (d) kopiec dwumianowy
- (e) kopiec Fibonacciego
- (f) drzewo czerwono-czarne

Dla struktur b, f koszt wykonania operacji $find(i)$ jest $O(\log n)$. Dla pozostałych jest on $O(n)$.

3.6. zadanie 6

Podaj przykłady nietrywialnych zastosowań poniższych algorytmów i struktur danych: kopiec, kopiec Fibonacciego, kopiec dwumianowy, sortowanie leksykograficzne ciągów różnej długości

	Przykład zastosowania
kopiec	algorytm sortowanie przez kopcowanie
kopiec Fibonacciego	algorytm Dijkstry
kopiec dwumianowy	kolejka priorytetowa
sortowanie leksykograficzne ciągów różnej długości	algorytm rozpoznający czy dwa drzewa są izomorficzne

3.7. zadanie 14

W problemie LCS stosowaliśmy redukcję problemu porównując ostatnie litery X i Y . Czy jakieś znaczenie ma fakt, że są to ostatnie litery a nie pierwsze?

Nie jest to istotne.

Weźmy sobie ciągi X, Y i niech algorytm $LCS(X, Y)$, obliczający długość najdłuższego podciągu, wykorzystując redukcję problemu porównując ostatnie litery ciągów. Zaobserwujmy że algorytm $LCS'(X, Y) = LCS(X^R, Y^R)$ również poprawnie oblicza długość najdłuższego podciągu ciągów X, Y , ale można powiedzieć że wykorzystuje on redukcję problemu porównując pierwsze litery X, Y .

Bibliografia

[1] test reference