Lista zadań. Nr 1. 21 marca 2020

Algorytmy i Struktury Danych

IIUWr. II rok informatyki.

1. (**P** 1pkt) Napisz rekurencyjne funkcje, które dla danego drzewa binarnego T obliczają:
   - liczbę wierzchołków w T,
   - maksymalną odległość między wierzchołkami w T.

2. (1pkt) Napisz w pseudokodzie procedury:
   - przywracania porządku
   - usuwania minimum
   - usuwania maksimum

   z kopca minimaksowego. Przyjmij, że elementy tego kopca pamiętane są w jednej tablicy (określ w jakiej kolejności). Użyj pseudokodu na takim samym poziomie szczegółowości, na jakim zostały napisane w Notatce nr 2 odpowiednie procedury dla zwykłego kopca.

3. (1pkt) *Porządkiem topologicznym* wierzchołków acyklicznego digrafu $G = (V, E)$ nazywamy taki liniowy porządek jego wierzchołków, w którym początek każdej krawędzi występuje przed jej końcem. Jeśli wierzchołki z V utożsamimy z początkowymi liczbami naturalnymi to każdy ich porządek liniowy można opisać permutacją liczb $1, 2, , ..., |V|$; w szczególności pozwala to na porównywanie leksykograficzne porządków.

   Ułóż algorytm, który dla danego acyklicznego digrafu znajduje pierwszy leksykograficznie porządek topologiczny.

4. (1pkt) Niech u i v będą dwoma wierzchołkami w grafie nieskierowanym $G = (V, E; c)$, gdzie $c : E \to R_+$ jest funkcją wagową. Mówimy, że droga z $u = u_1, u_2, \ldots, u_{k-1}, u_k = v$ z u do v jest sensowna, jeśli dla każdego $i = 2, \ldots, k$ istnieje droga z u_i do v krótsza od każdej drogi z u_{i-1} do v (przez długość drogi rozumiemy sumę wag jej krawędzi).

   Ułóż algorytm, który dla danego G oraz wierzchołków u i v wyznaczy liczbę sensownych dróg z u do v.

5. (1pkt) Ułóż algorytm, który dla zadanego acyklicznego grafu skierowanego G znajduje długość najdłuższej drogi w G. Następnie zmodyfikuj swój algorytm tak, by wypisywał drogę o największej długości (jeśli jest kilka takich dróg, to Twój algorytm powinien wypisać dowolną z nich).

6. (1.5pkt) Dany jest niemalejący ciąg n liczb całkowitych dodatnich $a_1 \leq a_2 \leq \ldots \leq a_n$. Wolno nam modyfikować ten ciąg za pomocą następującej operacji: wybieramy dwa elementy a_i, a_j spełniające $2a_i \leq a_j$ i wykreślamy je oba z ciągu. Ułóż algorytm obliczający, ile co najwyżej elementów możemy w ten sposób usunąć.

7. (**Z** 2pkt) Skonstruuj algorytm, który wypisze k największych elementów znajdujących się w podanym kopcu binarnym. Załóż, że kopiec jest przechowywany w tablicy, więc możemy w czasie stałym dobrać się do dowolnego elementu, oraz że największy element znajduje się w korzeniu. Elementy można wypisać w dowolnej kolejności, niekoniecznie od największego do k-tego największego. Algorytm powinien działać w czasie $O(k \log \log k)$ lub mniej.

8. (**Z** 2pkt) Rozważmy kopiec binarny przechowujący n elementów, w którego korzeniu znajduje się największy element. Wiemy, że zarówno wstawienie nowego elementu jak i usunięcie największego elementu mogą być wykonane w czasie $O(\log n)$. Skonstruuj strukturę danych, która

umożliwia wstawienie nowego elementu w czasie stałym, oraz wykonuje k-tą operację usunięcia największego elementu w czasie $O(f(n) + \log k)$, gdzie $f(n) = o(\log n)$.

Zadania dodatkowe - do samodzielnego rozwiązywania

1. Co stałoby się z mocą obliczeniową maszyny RAM gdyby instrukcje ADD i MULT zostały usunięte z repertuaru instrukcji? Jak zmieniłby się koszt obliczeńn?

2. Pokaż, że dla każdego programu maszyny RAM istnieje równoważny program maszyny RAM (tj. taki, który dla tych samych danych produkuje te same wyniki) używający nie więcej niż 2^{14} komórek pamięci.

3. Przypomnij sobie notację asymptotyczną dla rzędów funkcji: O, Ω, Θ.

4. Jaka jest najmniejsza wartość n, dla której algorytm o złożoności $100n^2$ działa (na tej samej maszynie) szybciej od algorytmu o złożoności 2^n?

5. Dla każdej funkcji $f(n)$ i czasu t w poniższej tabelce, określ największy rozmiar n danych, dla których algorytm wykona obliczenia w czasie t. Zakładamy, że algorytm rozwiązujący problem potrzebuje $f(n)$ mikrosekund dla danych rozmiaru n.

	1 sekunda	1 minuta	1 godzina	1 dzieńn	1 miesiąc	1 rok	1 wiek
$\log n$							
$\sqrt{n}$							
n							
$n \log n$							
n^2							
n^3							
2^n							
$n!$							

   O ile większe zadania można by rozwiązywać na komputerze 1000 razy szybszym (tj. takim, na którym algorytm potrzebowałby $f(n)$ nanosekund dla danych rozmiaru n)?

6. Skonstruuj program dla maszyny RAM, który dla danej liczby naturalnej n obliczy $n!$.

   Oszacuj złożoność czasową tego programu przy jednorodnym i logarytmicznym kryterium kosztów. Ustal własną miarę "rozmiaru" danych.

7. Napisz w C++, C lub Pascalu funkcję implementującą podany na wykładzie algorytm, który oblicza n-tą liczbę Fibonacciego (modulo stała) w czasie $O(\log n)$.

8. Napisz procedury, które dla danego drzewa binarnych przeszukiwańn T:
   - (0,5pkt) wstawiają zadany klucz do T;
   - (1pkt) usuwają zadany wierzchołek z T;
   - (0,5pkt) dla danego klucza k znajdują następny co do wielkości klucz w drzewie.

9. Napisz funkcję, która dla danej, uporządkowanej rosnąco, tablicy liczbowej T oraz liczby k, obliczy liczbę elementów w T mniejszych od k.

10. Określ z dokładnością do Θ złożoność (przy kryterium jednorodnym) poniższych fragmentów programów:

for $i \leftarrow 1$ **to** n **do**
    $j \leftarrow i$
    while $j < n$ **do**
        $sum \leftarrow P(i,j)$
        $j \leftarrow j + 1$

for $i \leftarrow 1$ **to** n **do**
    $j \leftarrow i$
    while $j < n$ **do**
        $sum \leftarrow P(i,j)$
        $j \leftarrow j + j$

Rozważ dwa przypadki:

- koszt wykonania procedury $P(i,j)$ wynosi $\Theta(1)$
- koszt wykonania procedury $P(i,j)$ wynosi $\Theta(j)$

Krzysztof Loryś