

ALGORYTMY I STRUKTURY DANYCH

IHUWr. II rok informatyki.

1. (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 minimaxowego. 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 $1, 2, \dots, |V|$; 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 $c : E \rightarrow R_+$ jest funkcją wagową. Mówimy, że droga z $u = u_1, u_2, \dots, u_{k-1}, u_k = v$ z u do v jest sensowna, jeśli dla każdego $i = 2, \dots, k$ 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 $a_1 \leq a_2 \leq \dots \leq a_n$. 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. (1,5pkt) Dany jest nieskierowany graf ważony $G = (V, E; c)$ z $c : E \rightarrow R_+$ oraz ciąg v_1, v_2, \dots, v_k różnych wierzchołków z V . Niech D_j ($0 \leq j \leq k$) będzie sumą długości najkrótszych ścieżek między wszystkimi parami wierzchołków pozostającymi w G po usunięciu wierzchołków v_1, v_2, \dots, v_j (wraz z wierzchołkiem usuwamy wszystkie incydentne z nim krawędzie).

Ułóż algorytm obliczający wartości D_0, D_1, \dots, D_k .

8. (1pkt) Ułóż algorytm, który dla danych k uporządkowanych niemalejąco list L_1, \dots, L_k liczb całkowitych znajduje najmniejszą liczbę r , taką że w przedziale $[a, a+r]$ znajduje się co najmniej jedna wartość z każdej z list L_i , dla pewnej liczby a .
- Twój algorytm nie może modyfikować list L_i i powinien być pamięciowo oszczędny (no i oczywiście jak najszybszy).
9. (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 dostać 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.
10. (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 $f(n) = o(\log n)$.