Algorytmy i struktury danych

Zadania z różnych egzaminów, mogą być podobne

Prowadzący: prof. dr hab. Maciej M. Sysło

1. Wypełnij drugą kolumnę w tabeli

Problem	**Podaj liczbę podstawowych operacji (jako funkcję zmiennej *n*) wykonywanych przez najbardziej efektywny algorytm, służący do rozwiązywania problemu po lewej stronie. Wymień, jakie to operacje. Tam, gdzie jest to możliwe, podaj dokładną liczbę operacji.**
Znalezienie binarnej reprezentacji liczby naturalnej *n*.	
Obliczenie dziesiętnej wartości liczby naturalnej *n* danej w systemie pozycyjnym o podstawie 3.	
Scalenie dwóch uporządkowanych ciągów o długościach *k* oraz *l* w jeden ciąg uporządkowany o długości $n = k + l$.	
Uprządkowanie *n* liczb naturalnych z przedziału [1, *m*] metodą przez zliczanie.	
Wyprowadzenie kolejnych dziesiętnych cyfr liczby naturalnej *n* zapisanej w komputerze.	
Znalezienie najmniejszej i największej liczby w ciągu złożonym z *n* liczb naturalnych.	
Podniesienie *x* do potęgi *n*.	
Uprządkowanie stopni wierzchołków grafu o *n* wierzchołkach.	
Zastosowanie algorytmu porządkowania bąbelkowego do ciągu *n* uporządkowanych liczb.	
Zastosowanie porządkowania przez wybór do ciągu *n* uporządkowanych liczb.	
Znalezienie najmniejszej i drugiej najmniejszej liczby w ciągu *n* liczb.	
Obliczenie NWD(*m*, *n*)	
Utworzenie reprezentacji liczby naturalnej *n* przy podstawie *p*.	

2. Danych jest sześć liczb, które należy uporządkować stosując porównania między tymi liczbami.

a) Ile wynosi najmniejsza liczba porównań, jaką musi wykonać jakikolwiek algorytm służący do uporządkowania sześciu liczb za pomocą porównań między elementami, czyli ile wynosi oszacowanie dolne liczby porównań w porządkowaniu sześciu liczb?

b) Ile porównań w najgorszym przypadku wykona algorytm porządkowania przez scalanie zastosowany do sześciu liczb? Przedstaw ten algorytm w tym przypadku.

c) Jeśli liczby porównań w przypadku b) jest większa od liczby porównań w przypadku a), to postaraj się podać algorytm porządkujący sześć liczb, który wykonuje liczbę porównań określoną w puncie a).

3. Przyjmij dla słów porządek alfabetyczny (słownikowy). Zaczynając od pustego drzewa, utwórz kolejne drzewa binarnych poszukiwań wstawiając następujące słowa w podanej kolejności: **if**, **begin**, **mod**, **do**, **abs**, **while**, **goto**, **end**, **for**, **div**, **go**, **to**, **repeat**. Następnie usuń z otrzymanego drzewa najpierw słowo **do**, a później **goto**. Na końcu wstaw do drzewa, które pozostało, najpierw słowo **integer**, a później **real**.

4. Zapisz (w języku programowania lub w pseudo-języku programowania) algorytm o złożoności O(mn), gdzie n jest liczbą wierzchołków, a m – jest liczbą łuków w digrafie D, który służy do badania, czy digraf D jest **silnie spójny**, tzn. każda para jego wierzchołków jest połączona drogą skierowaną. Digraf D jest dany w postaci list sąsiedztwa.

---

3. Danych jest 19 liczb w tablicy:

x[1..19] = [9, 5, 12, 16, 12, 7, 14, 10, 2, 17, 6, 11, 13 18, 1, 4, 8, 3, 19].

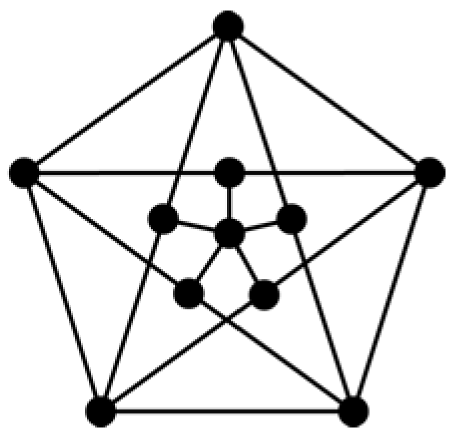
a) Przedstaw kolejne etapy obliczeń, które sprowadzają tablicę x do kopca z elementem **najmniejszym w korzeniu kopca**. Możesz posłużyć się interpretacją kopca w postaci pełnego drzewa binarnego.

b) W otrzymanym w punkcie a) kopcu, element o wartości 3 zostaje zastąpiony przez element o wartości 15. Napraw warunek kopca w tak otrzymanym drzewie.

4. Dane są dwie liczby, dziesiętna liczba naturalna n i dziesiętna liczba naturalna p, gdzie $2 \leq p \leq 10$. Napisz w języku lub pseudojęzyku programowania **rekurencyjny algorytm**, który oblicza sumę cyfr liczby n w reprezentacji przy podstawie p.

5. Graf symetryczny G = (V, E) jest dany w postaci list sąsiadów dla poszczególnych wierzchołków. Napisz w języku lub w pseudojęzyku programowania algorytm, który korzystając z metody przeszukiwania grafu G w głąb sprawdza, czy graf G jest dwudzielny. W przypadku pozytywnej odpowiedzi, wyprowadza podział zbioru wierzchołków V na klasy dwudzielności. Twój algorytm powinien mieć złożoność O(n+m), gdzie n jest liczbą wierzchołków w grafie G, a m jest liczba krawędzi w grafie G.

6. W grafie Mycielskiego, który jest pokazany na rysunku:



pięciu krawędziom wychodzącym z wierzchołka środkowego, przyporządkowujemy wagi 1, 2, 3, 4, 5, krawędziom zewnętrznego cyklu o długości 5 przyporządkowujemy wagi 3, a pozostałym krawędziom wagi 2.

(a) Znajdź najkrótsze drzewo rozpinające w tym grafie.

(b) Znajdź drzewo najkrótszych dróg w tym grafie z jednego z wierzchołków zewnętrznych.

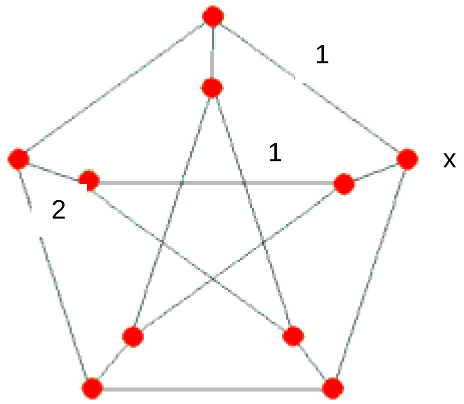
---

1. Podaj, w jaki sposób zostanie obliczona wartość potęgi x^n za pomocą możliwie najmniejszej liczby mnożeń dla wykładnika, który jest dany jako liczba w systemie trójkowym: $n = (1212)_3$. Zapisz ciąg kolejno wykonywanych mnożeń w tym przypadku. Ile ich jest?

2. W komputerze jest zapisana dziesiętna liczba n. Zaprogramuj algorytm, który wyprowadza kolejne cyfry tej liczby w reprezentacji binarnej **począwszy od najbardziej znaczącej**. Użyj rekurencji lub stosu (jawnie).

3. Przyjmij dla słów porządek alfabetyczny (słownikowy). Zaczynając od pustego drzewa, utwórz kolejne drzewa binarnych poszukiwań wstawiając następujące słowa w podanej kolejności: **end**, **do**, **begin**, **while**, **if**, **for**, **go**, **to**, **repeat**. Następnie usuń z otrzymanego drzewa najpierw słowo **if**, a później **to**. Na końcu wstaw do drzewa, które pozostało, najpierw słowo **integer**, a później **real**.

4. W grafie Petersena:



krawędziom zewnętrznego cyklu o długości 5 i krawędziom wewnętrznego cyklu o długości 5 przyporządkowujemy wagi 1, a pozostałym pięciu krawędziom wagi 2.

(a) Znajdź najkrótsze drzewo rozpinające w tym grafie.

(b) Znajdź drzewo najkrótszych dróg w tym grafie z wierzchołka x.

---

1. Czy prawdziwa jest następująca relacja – odpowiedź uzasadnij odpowiednimi obliczeniami:

$$r^{\log_a n} \in \theta(r^{\log_b n}) \text{ dla } a \neq b.$$

2. Dane jest drzewo binarnych poszukiwań. Przyjmujemy, że to drzewo jest reprezentowane w postaci wskaźnikowej, zatem każdy wierzchołek jest rekordem o trzech polach: **info** – słowo w wierzchołku, **Lewe**, **Prawe** – wskaźniki do odpowiednio lewego i prawego poddrzewa. Przy tej reprezentacji drzewo jest dane jako wskaźnik na korzeń. Napisz w pseudokodzie algorytm służący do wypisania na wyjściu wszystkich słów znajdujących się w wierzchołkach tego drzewa (czyli w polach **info**) w kolejności od słowa największego do słowa najmniejszego w porządku leksykograficznym (słownikowym). Twój algorytm nie powinien przekształcać danego drzewa.

3. Jakie słowo zostało zapisane w kodzie Huffmana jako ciąg: **101111111001101111**

Składa się ono z liter: **a**, **g**, **k**, **s**, **u**, których częstości wynoszą:

a	8.7
g	1.4
k	3.1
s	4.6
u	1.9

Uwaga. W kolejnych krokach konstruowania drzewa Huffmana, które posłuży do znalezienia kodu Huffmana liter, węzeł o mniejszej wadze przyjmuj za lewy następnik tworzonego węzła. Przypisywanie kodów: gałęziom wychodzącym z wierzchołka przypisz: lewej – 0, a prawej – 1.

---

1. Uporządkuj następujące funkcje relacją „*o* małe" i uzasadnij każdą z relacji:

$$n^3 \log_2 n\,,\ 2^n \sqrt{n}\,,\ (\log_2 \log_2 n)^2\,,\ (3n+4)^3$$

4. W komputerze jest zapisana dziesiętna liczba *n*.

(a) **Napisz algorytm**, który wyprowadza kolejne cyfry tej liczby począwszy od najbardziej znaczącej.

(b) **Podaj**, w jaki sposób zmodyfikować ten algorytm, aby wyprowadzał cyfry tej liczby zapisanej w systemie pozycyjnym o podstawie *p*.

Wskazówka. Zastosuj rekurencję.

---

1. Podaj sposób obliczania wartości potęgi x^n, w którym jest wykonywanych możliwie najmniej mnożeń. Posłuż się w tym celu schematem Hornera dla liczby *n* zamienionej wcześniej na postać binarną. Wypisz kolejne potęgi występujące podczas obliczania tym sposobem wartości potęgi dla wykładnika $n = 99$. Oszacuj liczbę wykonywanych mnożeń w zależności od *n*.

2. Ciąg liczb [3, 1, 4, 6, 5, 7, 2, 10, 9] uporządkuj metodą przez kopcowanie. W tym celu:

a) najpierw utwórz kopiec z danych liczb – możesz tworzyć kopiec posługując reprezentacją kopca w postaci drzewa ;

b) podaj postacie kolejnych kopców, tworzonych w trakcie sortowania tego ciągu danych.

3. Zaczynając od pustego drzewa utwórz drzewo binarnych poszukiwań wstawiając następujące liczby w podanej kolejności:100, 20, 10, 50, 30, 150, 60, 130, 140, 40. Usuń z tego drzewa liczby 50 i 20. Podaj kolejność pozostałych w drzewie liczb przy przechodzeniu tego drzewa metodą INORDER.

4. Dany jest digraf *G*, reprezentowany za pomocą zbiorów sąsiadów poszczególnych wierzchołków, czyli dla każdego wierzchołka *v* dany jest zbiór wierzchołków *N*(*v*), do których w digrafie istnieje łuk z wierzchołka *v*. Długością drogi w digrafie jest liczba tworzących ją łuków. Zapisz w postaci pseudokodu algorytm, który dla ustalonego wierzchołka *s* w digrafie *G*, w tablicy dist[1:n] umieszcza długości najkrótszych dróg z wierzchołka *s* do wszystkich pozostałych wierzchołków w digrafie *G*; n jest liczbą wierzchołków w digrafie G. Określ złożoność swojego algorytmu w zależności od liczby wierzchołków *n* i liczby łuków *m* w digrafie *G*.

---

1. Uporządkuj pod względem szybkości wzrostu (od najwolniej do najszybciej rosnącej) następujące funkcje:

$$2^{n^2}\,,\ n^{3.01},\ e^{\log_{10} n^3}\,,\ 2^{2n}$$

3. W tablicy d[1:n] został umieszczony ciąg stopni wierzchołków grafu, czyli liczby, których wartości są większe lub równe 0 i mniejsze od n. Podaj algorytm porządkowania tego ciągu, którego złożoność wynosi *O*(n). Wykaż, że złożoność podanego przez Ciebie algorytmu jest rzeczywiście O(*n*).

4. Zapisz w języku lub pseudojęzyku programowania **rekurencyjny algorytm**, który dla liczby naturalne *n* zapisanej w komputerze, oblicza sumę jej dziesiętnych cyfr.

7. Zapisz (w języku programowania lub w pseudo-języku programowania) algorytm o złożoności O(mn), gdzie n jest liczbą wierzchołków, a m – jest liczbą łuków w digrafie D, który służy do badania, czy digraf D jest **silnie spójny**, tzn. każda para jego wierzchołków jest połączona drogą skierowaną. Digraf D jest dany w postaci list sąsiedztwa.

---

1. Posługując się definicją symbolu), sprawdź prawdziwość następujących relacji:

$n^2 \in O(n^3)$; $2^{n+1} \in O(2^n)$; $(n+1)! \in O(n!)$; $\sqrt{n} \in O(\log n)$;

2. Zaczynając od pustego drzewa utwórz drzewo binarnych poszukiwań wstawiając następujące liczby w podanej kolejności:100, 20, 10, 50, 30, 150, 60, 130, 140, 40. Usuń z tego drzewa liczby 50 i 20. Podaj kolejność pozostałych w drzewie liczb przy przechodzeniu tego drzewa metodą INORDER.

---

4. Dany jest digraf G, czyli graf skierowany. **Odległość** wierzchołka u od wierzchołka v w digrafie G definiujemy jako długość najkrótszej drogi z wierzchołka v do wierzchołka u, lub przyjmujemy, że wynosi – 1, jeśli w digrafie G nie ma drogi z wierzchołka v do wierzchołka u. Digraf G jest dany w postaci rodziny zbiorów następników, czyli dla każdego wierzchołka v digrafu G dany jest zbiór $N(v)$ zawierający wierzchołki, które bezpośrednio następują po v w digrafie G. Podaj algorytm, który służy do wyznaczania macierzy zawierającej odległości między każdą parą wierzchołków w digrafie G. Zapisz swój algorytm w postaci pseudokodu. Określ złożoność otrzymanego algorytmu w zależności od liczby wierzchołków n i liczby łuków m w digrafie G.

---

2. Dziesiętna liczba naturalna n jest ***p*-podobna**, gdzie $2 \le p \le 10$, jeśli suma jej cyfr jest równa sumie jej cyfr w reprezentacji przy podstawie p – obie sumy są liczone w systemie dziesiętnym. Na przykład,

21 jest 2-podobna, bo	$(21)_{10} = (10101)_2$	i	$2 + 1 = 1 + 1 + 1 = 3$
23 jest 3-podobna, bo	$(23)_{10} = (212)_3$	i	$2 + 3 = 2 + 1 + 2 = 5$

Zauważ, że każda liczba n jest 10-podobna.

a) Zapisz w wybranej przez siebie notacji (schemat blokowy, pseudo-język programowania, jezyk programowania) algorytm sprawdzający, czy dla danych liczb naturalnych n i p, gdzie $2 \le p \le 10$, n jest liczbą p-podobną.

b) Podaj, ile operacji arytmetycznych (dodawań/odejmowań, mnożeń/dzieleń, branie reszty) w zależności od wartości danych n i p, wykonuje Twój algorytm.

3. Elementy ciągu $a_1, a_2, \ldots, a_n$ należą do zbioru liczb naturalnych $\{1, 2, \ldots, m\}$. Opisz algorytm, który bez porządkowania tego ciągu stwierdza, czy istnieje w element tego zbioru, który występuje w tym ciągu więcej niż $n/2$ razy.

---

1. Sprawdź prawdziwość następujących relacji:

$n^2 \in O(n^3)$; $n^3 \in O(n^{2.99})$; $2^{n+1} \in O(2^n)$; $(n+1)! \in O(n!)$; $\log n \in O(\sqrt{n})$; $\sqrt{n} \in O(\log n)$;