

Egzamin licencjacki/inżynierski

9 września 2024

Informacja dla zdających egzamin na kierunku informatyka

Z sześciu poniższych zestawów zadań (Matematyka I, Matematyka II, Metody programowania, Matematyka dyskretna, Algorytmy i struktury danych, Metody numeryczne) należy wybrać i przedstawić na osobnych kartkach rozwiązania trzech zestawów.

Informacja dla zdających egzamin na kierunku ISIM

Z sześciu poniższych zestawów zadań (Matematyka I, Metody programowania, Matematyka dyskretna, Algorytmy i struktury danych, Metody numeryczne, Języki formalne i złożoność obliczeniowa) należy wybrać i przedstawić na osobnych kartkach rozwiązania trzech zestawów.

Informacja dla wszystkich zdających

Za brakujące (do trzech) zestawy zostanie wystawiona ocena niedostateczna z urzędu. Egzamin uważa się za zaliczony, jeśli student rozwiąże z oceną dostateczną co najmniej 2 zestawy. Wtedy ocena z egzaminu jest średnią arytmetyczną ocen z trzech wybranych zestawów. Na rozwiązanie przeznaczona jest czas $3 \times 40 + 30 = 150$ minut. Po wyjściu z sali egzaminacyjnej w czasie egzaminu nie ma możliwości powrotu do tej sali i kontynuowania pisania egzaminu.

Matematyka I — Logika dla informatyków

Wprowadzenie. Zbiór $L(X)$ wszystkich skończonych list nad danym zbiorem X jest zdefiniowany indukcyjnie w następujący sposób:

- nil jest skończoną listą nad zbiorem X ;
- jeśli x jest elementem zbioru X oraz xs jest skończoną listą nad zbiorem X to $x : xs$ jest skończoną listą nad zbiorem X .

Dla wszystkich zbiorów X operacja konkatencji list $++ : L(X) \times L(X) \rightarrow L(X)$ jest zdefiniowana w następujący sposób. Dla wszystkich elementów $x \in X$ oraz wszystkich list $xs, ys \in L(X)$ przyjmujemy

$$\begin{aligned} nil ++ ys &= ys, \\ (x : xs) ++ ys &= x : (xs ++ ys). \end{aligned}$$

Dla wszystkich zbiorów X definiujemy funkcję $filter : \mathcal{P}(X) \times L(X) \rightarrow L(X)$ w następujący sposób. Dla dowolnego zbioru $S \in \mathcal{P}(X)$, dowolnego elementu $x \in X$ oraz dowolnej listy $xs \in L(X)$ przyjmujemy

$$\begin{aligned} filter(S, nil) &= nil, \\ filter(S, x : xs) &= \begin{cases} x : filter(S, xs), & \text{gdy } x \in S, \\ filter(S, xs), & \text{wpp.} \end{cases} \end{aligned}$$

Właściwe zadanie.

1. Sformułuj zasadę indukcji strukturalnej dla list (w takiej postaci, żeby można było jej użyć w dowodzie w punkcie 2).
2. Korzystając z zasady indukcji sformułowanej w punkcie 1 udowodnij indukcyjnie, że dla dowolnego zbioru X i dowolnych list $xs, ys \in L(X)$ zachodzi równość

$$filter(S, xs ++ ys) = filter(S, xs) ++ filter(S, ys).$$

Uwaga: To jest zadanie z logiki. Podczas jego oceniania zwrócimy szczególną uwagę na poprawność i klarowność rozumowania, w tym na prawidłowe posługiwanie się zmiennymi (np. ich inicjowanie) i kwantyfikatorami, oraz na prawidłowe sformułowanie i użycie założenia indukcyjnego.

Matematyka II — Algebra

Zadanie 1. (4 punkty)

Wykazać, czy wektory $e_3, e_3 - e_2, e_3 - e_1$ są niezależne w przestrzeni \mathbb{R}^3 .

Zadanie 2. (4 punkty)

Jaka jest wartość wyrażenia $3/14$ w grupie (mnożymy) \mathbb{Z}_{17} ?

Zadanie 3. (5 punktów)

$$A = \begin{bmatrix} 5 & 0 & 2 \\ 0 & 5 & 4 \\ 2 & 4 & 4 \end{bmatrix}.$$

Znaleźć wartości własne tej macierzy.

Progi punktowe: 4, 6, 8, 10, 12 punktów.

Metody Programowania

Poniższe zadania należy rozwiązać używając języka Racket lub Plait.

Zadanie 1 Wyjaśnij i zaprezentuj na przykładzie pojęcie nieużytku.

Zadanie 2 Poniższa implementacja funkcji konkatenującej dowolnie głęboko zagnieżdżoną listę do pojedynczej listy generuje nieużytki:

```
(define (concat xs)
  (cond [(null? xs) null]
        [(cons? (car xs)) (append (concat (car xs)) (concat (cdr xs)))]
        [else (cons (car xs) (concat (cdr xs)))]))
```

Napisz implementację `concat-fast`, która nigdy nie generuje nieużytków.

Zadanie 3 Udowodnij przez indukcję, że obydwie definicje (`concat` i `concat-fast`) są równoważne.

Matematyka dyskretna

Dla dowolnego grafu $G = (V, E)$ definiujemy *graf krawędziowy* $L(G)$, którego zbiorem wierzchołków jest E i dwie krawędzie są sąsiednie gdy mają wspólny wierzchołek. Pokaż, że każda klika w $L(G)$ o rozmiarze większym niż 3 składa się z krawędzi grafu G incydentnych do jednego wierzchołka G .

Algorytmy i struktury danych

Za rozwiązanie obydwu zadań z tej części można otrzymać w sumie do 9 punktów. Skala ocen: poniżej 3 punktów — ocena niedostateczna (egzamin niezdany), 3 punkty dają ocenę dostateczną, 4 — dostateczną z plusem, 5 — dobrą, 6 — dobrą z plusem, 7 albo więcej punktów daje ocenę bardzo dobrą.

Zadanie 1: iloczyn ciągu macierzy (4 punkty)

Dany jest ciąg n macierzy liczbowych M_1, M_2, \dots, M_n , przy czym macierz M_i ma wymiary $r_{i-1} \times r_i$ dla $i = 1, 2, \dots, n$. Przyjmujemy, że obliczenie iloczynu dwóch macierzy o rozmiarach odpowiednio $k \times l$ i $l \times m$ kosztuje $k \cdot l \cdot m$ jednostek czasowych. Zadanie polega więc na optymalnym wymnożeniu macierzy $M_1 \times M_2 \times \dots \times M_n$, minimalizując koszt obliczeń (mnożenie macierzy jest łączne, więc można mnożyć macierze w dowolnej kolejności).

Korzystając z metody programowania dynamicznego skonstruuj algorytm, który dla zadanego ciągu n pasujących do siebie wymiarów macierzy wyznaczy optymalny koszt pomnożenia tych macierzy. Opisz ideę rozwiązania a potem zapisz ją w pseudokodzie. Uzasadnij poprawność opisanej metody i oszacuj jej złożoność obliczeniową (pamięciową i czasową).

Uwaga: odpowiedz też na pytanie, jak wyznaczyć optymalną kolejność mnożenia macierzy (a nie tylko wyliczyć optymalny koszt).

Zadanie 2: słownik z operacjami *element* i *number* (5 punktów)

Niech S będzie słownikiem z dodatkowymi operacjami: *element* i *number*. Chcielibyśmy efektywnie wykonywać następujące operacje na tej strukturze danych:

1. $S = \text{new}()$: $S \leftarrow \emptyset$ (utworzenie pustego zbioru);
2. $S.\text{insert}(x)$: $S \cup \{x\}$ (dodanie nowego elementu do zbioru);
3. $S.\text{delete}(x)$: $S \setminus \{x\}$ (usunięcie elementu ze zbioru);
4. $S.\text{search}(x)$: $x \in S$ (sprawdzenie czy element należy do zbioru);
5. $y = S.\text{element}(i)$: wyznaczenie i -tego co do wielkości elementu w zbiorze S (w n -elementowym zbiorze S parametr $i \in \{0, \dots, n-1\}$, przy czym dla $i = 0$ procedura ma wskazać element minimalny a dla $i = n-1$ element maksymalny);
6. $j = S.\text{number}(x)$: wyznaczenie pozycji elementu x względem wielkości (pozycja 0 odnosi się do elementu minimalnego a pozycja $n-1$ do maksymalnego).

Zaprojektuj taką strukturę danych dla zbioru S , która umożliwi wykonywanie wymienionych operacji w czasie logarytmicznym $O(\log n)$, gdzie $n = |S|$ (liczba elementów w S). Możesz wykorzystać jakąś znaną strukturę danych, która efektywnie realizuje operacje słownikowe i zaadoptować ją na potrzeby zadania. Zapisz w pseudokodzie procedurę realizującą operacje *element*(i) i *number*(x) i wyjaśnij jak one działają. Krótko ale precyzyjnie opisz, jak należy zmodyfikować pozostałe standardowe operacje słownikowe. Przeanalizuj złożoność czasową wszystkich operacji na tej strukturze.

Metody numeryczne

Za rozwiązanie zadań można otrzymać łącznie 12 punktów. Otrzymanie 4 pkt. gwarantuje ocenę dostateczną, próg dla dst+ to 5.5 pkt., dla db – 7 pkt., dla db+ 8.5 pkt., a dla bdb – 10 pkt.

1. **4 punkty** Dla jakich wartości x obliczanie wartości wyrażenia $(\sqrt{x^6 + 2024} + x^3)^{-1}$ może wiązać się z **utratą cyfr znaczących** wyniku? Zaproponuj sposób obliczenia wyniku dokładniejszego. Odpowiedź **uzasadnij**.
2. **4 punkty** **Opisz** metodę bisekcji znajdowania przybliżonego miejsca zerowego danej funkcji (pamiętaj o potrzebnych założeniach). Ile kroków według tej metody należy wykonać, żeby wyznaczyć zero α z błędem bezwzględnym mniejszym niż zadana liczba $\varepsilon > 0$? Odpowiedź **uzasadnij**.
3. **4 punkty** Jak powszechnie wiadomo, w języku PWO++ procedura `LegendreZeros(m)` znajduje z dużą dokładnością wszystkie miejsca zerowe m -tego wielomianu Legendre’a. Używając tej procedury, **opracuj i uzasadnij** efektywny **algorytm** wyznaczania takich węzłów $x_0^{(n)}, x_1^{(n)}, \dots, x_n^{(n)}$ oraz współczynników $A_0^{(n)}, A_1^{(n)}, \dots, A_n^{(n)}$, że dla każdego wielomianu w stopnia mniejszego od $2n + 2$ zachodzi

$$\int_{-5}^5 w(x) dx = Q_n(w),$$

$$\text{gdzie } Q_n(f) := \sum_{k=0}^n A_k^{(n)} f(x_k^{(n)}).$$

Języki formalne i złożoność obliczeniowa

Założmy, że mamy wielomianowy algorytm A rozstrzygający, czy wierzchołki danego grafu G można pokolorować trzema kolorami, aby wszystkie pary sąsiadujących wierzchołków były różnych kolorów. Czy z tego wynika, że istnieje algorytm działający w czasie wielomianowym, który znajduje kolorowanie grafu trzema kolorami lub odpowiada, że takie kolorowanie nie istnieje? Porządnie uzasadnij swoją odpowiedź.