Egzamin licencjacki/inżynierski

13 września 2023

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 przeznacza się czas 3x40+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

Kontekst zadania: Na wykładzie z logiki dla informatyków pokazaliśmy, że $\{\wedge, \vee, \neg\}$ jest zupełnym zbiorem spójników. W tym zadaniu pokażemy, że negacja nie jest potrzebna do reprezentacji (niestałych) funkcji monotonicznych. Należy rozwiązać jedną z dwóch wersji zadania, przy czym wersja standardowa będzie oceniana w skali od 2.0 do 5.0 a wersja łatwiejsza w skali od 2.0 do 4.5.

Zadanie (wersja standardowa): Udowodnij, że dla każdej niestałej monotonicznej funkcji boolowskiej istnieje opisująca ją formuła, w której występują wyłącznie spójniki ze zbioru $\{\wedge, \vee\}$.

Zadanie (wersja łatwiejsza): Udowodnij, że dla każdej monotonicznej funkcji boolowskiej istnieje opisująca ją formuła, w której występują wyłącznie spójniki ze zbioru $\{\wedge, \vee, \top, \bot\}$.

Przypomnienie potrzebnych definicji: Naturalny porządek $\sqsubseteq$ na wartościach boolowskich $\mathcal{B} = \{\mathsf{F}, \mathsf{T}\}$, w którym $\mathsf{F} \sqsubseteq \mathsf{T}$, rozszerzamy w standardowy sposób („po osiach") do porządku na krotkach wartości boolowskich wzorem

$$\langle b_1, \ldots, b_n\rangle \sqsubseteq \langle b'_1, \ldots, b'_n\rangle \overset{\text{df}}{\Leftrightarrow} \bigwedge_{i=1}^{n} b_i \sqsubseteq b'_i.$$

Funkcję boolowską $f : \mathcal{B}^n \to \mathcal{B}$ nazywamy *monotoniczną* jeśli dla wszystkich krotek $\langle b_1, \ldots, b_n\rangle$ i $\langle b'_1, \ldots, b'_n\rangle$ zachodzi implikacja

$$\langle b_1, \ldots, b_n\rangle \sqsubseteq \langle b'_1, \ldots, b'_n\rangle \Rightarrow f(b_1, \ldots, b_n) \sqsubseteq f(b'_1, \ldots, b'_n).$$

Dla wszystkich $n \in \mathbb{N}$ mamy dwie *stałe* n-argumentowe funkcje boolowskie (zwracające dla wszystkich argumentów, odpowiednio, F i T). Wszystkie pozostałe funkcje boolowskie nazywamy *niestałymi*.
Formuła φ nad zbiorem zmiennych zdaniowych $\{p_1, \ldots, p_n\}$ *opisuje* n-argumentową funkcję boolowską f jeśli dla wszystkich wartościowań $\sigma : \{p_1, \ldots, p_n\} \to \mathcal{B}$ zachodzi równość

$$\hat{\sigma}(\varphi) = f(\sigma(p_1), \ldots, \sigma(p_n)).$$

Matematyka II — Algebra

Zadanie 1. (5 punktów)
Macierz A ma wartości własne $\lambda_1, \lambda_2, \ldots, \lambda_n$ wektory własne $v_1, \ldots, v_n$. Jakie są wartości własne i wektory własne macierzy A^2?

Zadanie 2. (4 punkty)
Dana jest nieosobliwa macierz $A \in \mathbb{R}^{n\times n}$. Udowodnić, że $(A^T)^{-1} = (A^{-1})^T$.

Zadanie 3. (5 punktów)
V jest przestrzenią liniową na ciałem K. Symbol $\mathbb{O}$ oznacza wektor zerowy. Wykazać, że $\forall\, \alpha \in K \;\; \forall\, v \in V \quad 0 \cdot v = \mathbb{O}$, $\alpha \cdot \mathbb{O} = \mathbb{O}$.

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

Metody Programowania

Poniższe zadania należy rozwiązać używając języka Racket lub Plait.
Rozważamy prosty język funkcyjny **Egz**, w którym *składnia abstrakcyjna* wyrażeń zawiera:

- stałe numeryczne,
- binarną operację dodawania,
- zmienne,
- funkcje jednoargumentowe,
- aplikacje wyrażenia do wyrażenia,

a jego semantyka jest zgodna z semantyką języka **Racket**.

Zadanie 1 Zaproponuj reprezentację składni abstrakcyjnej języka **Egz**. Jeżeli Twoim metajęzykiem jest Racket, to napisz predykat sprawdzający czy jego argument jest poprawnym wyrażeniem.

Zadanie 2 Zaproponuj reprezentację wartości wyrażeń języka **Egz**. Jeżeli Twoja definicja tego wymaga, to wprowadź interfejs środowiska i skorzystaj z niego.

Zadanie 3 Zaimplementuj interfejs środowiska w wybrany przez siebie sposób.

Zadanie 4 Napisz procedurę `eval` stanowiącą interpreter wyrażeń języka **Egz**, tzn. taką, która oblicza wartość wyrażenia z wykorzystaniem środowiska.

Matematyka dyskretna

Niech $\chi(G)$ będzie liczbą chromatyczną grafu G, a $\bar{G}$ dopełnieniem grafu G. Pokaż, że

$$\chi(G) + \chi(\bar{G}) \geq 2\sqrt{n}.$$

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** Funkcję $f(x) = \sin(x/4)$ interpolujemy wielomianem $L_n \in \Pi_n$ w węzłach będących zerami wielomianu Czebyszewa T_{n+1}. **Jak należy dobrać** n, aby mieć pewność, że
$$\max_{x\in[-1,1]} |f(x) - L_n(x)| \leq 10^{-7} \ ?$$
Odpowiedź **uzasadnij**.

2. **4 punkty** Pomiary (t_k, C_k) $(0 \leq k \leq N;\ t_k, C_k > 0)$ pewnej zależnej od czasu wielkości fizycznej C sugerują, że wyraża się ona wzorem
$$C(t) = \frac{1}{2023t^4 + A\sin t + 3}.$$
Stosując aproksymację średniokwadratową, **wyznacz prawdopodobną** wartość stałej A.

3. **4 punkty** Niech dane będą macierze $A, B \in \mathbb{R}^{n\times n}$. Załóżmy, że macierze te są nieosobliwe, i że istnieje rozkład LU macierzy AB. **Opracuj oszczędny** `algorytm` wyznaczania takiej macierzy $X \in \mathbb{R}^{n\times n}$, dla której zachodzi równość $BXA = I_n$, gdzie I_n jest macierzą identycznościową stopnia n. **Wyznacz** jego złożoność czasową i pamięciową.

   Uwaga. W podanym rozwiązaniu **nie możesz jawnie** obliczać odwrotności **żadnych** macierzy, bo – jak wiadomo – nie jest to bezpieczne z numerycznego punktu widzenia.

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: najdłuższy fragment z poprawnym nawiasowaniem (4 punkty)

Dany jest n-elementowy ciąg nawiasów $a_1, a_2, \ldots, a_n$, gdzie $a_i \in \{(,)\}$ dla $i = 1, 2, \ldots, n$. Opracuj efektywny algorytm, który wyznaczy najdłuższy fragment $a_i \ldots a_j$ w tym ciągu z poprawnie rozstawionmi nawiasami — wynikiem działania algorytmu powinna być para ideksów (i, j) identyfikująca wyznaczony fragment albo para $(0, 0)$ gdy takiego fragmentu w ogóle nie ma w zadanym ciągu. Opisz ideę algorytmu a potem zapisz go w pseudokodzie (wraz z niezbędnymi komentarzami). Przeanalizuj złożoność czasową i pamięciową opisanej metody.

Zadanie 2: operacja *increase* i *decrease* w drzewie dwumianowym (5 punktów)

Jak jest zbudowane *drzewo dwumianowe*? Przypomnij definicję. Ile elementów może zawierać takie drzewo oraz jaka jest wysokość drzewa dwumianowego w zależności od ilości elementów? Uzasadnij tą właściwość indukcyjnie.

Co to znaczy, że elementy w drzewie zachowują *porządek kopcowy*? Przypomnij definicję.

Rozważmy dowolne drzewo dwumianowe, w którym zachowany jest porządek kopcowy (w korzeniu znajduje się element maksymalny). Na drzewie takim chcemy wykonywać tylko dwa rodzaje operacji na wskazanym węźle v:

1. *increase(v, d)* — zwiększenie wartości węzła v o wartość $d > 0$;
2. *decrease(v, d)* — zmniejszenie wartości węzła v o wartość $d > 0$.

Jak można efektywnie zaimpelemtować te operacje, aby na końcu zawsze przywrócić porządek kopcowy? Przedstaw ideę realizacji tych operacji, zapisz je pseudokodzie (wraz z niezbędnymi komentarzami) oraz przeanalizuj ich złożoność obliczeniową.

Języki formalne i złożoność obliczeniowa

Przypomnijmy, że automat ze stosem to krotka $(\Sigma, \Gamma, Q, Q_0, \delta, F)$, gdzie

- Q to skończony zbiór stanów, $Q_0 \subseteq Q$ to zbiór stanów początkowych, a F to zbiór stanów akceptujących,
- Σ to alfabet wejściowy a Γ to zbiór symboli stosowych,
- δ to relacja nad $Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\bot\}) \times Q \times \Gamma^*$ o skończonej liczbie krotek.

Dla automatów ze stosem warunkiem akceptacji może być: (a) pusty stos — przebieg na słowie w jest akceptujący, gdy pod koniec przebiegu stos jest pusty, (b) osiągnięcie stanu z F — przebieg na słowie w jest akceptujący gdy ostatni stan przebiegu należy do F, lub (c) kombinacja poprzednich warunków. Wszystkie warunki (a), (b) i (c) są równoważne, t.j. klasa języków rozpoznawanych przez automaty z warunkiem (a) jest taka sama jak klasy języków rozpoznawane przez automaty z warunkiem (b) (odpowiednio (c)) — są to języki bezkontekstowe.

Automat ze stosem jest deterministyczny, jeśli $|Q_0| = 1$ oraz δ jest funkcją $\delta\colon Q \times \Sigma \times (\Gamma \cup \{\bot\}) \to Q \times \Gamma^*$ (bez ϵ-przejść). Czy dla deterministycznych automatów ze stosem warunki akceptacji (a) i (b) są równoważne? Innymi słowy, czy każdy język rozpoznawany przez automat ze stosem akceptujący przy pustym stosie daje się rozpoznawać (być może innym) automatem deterministycznych akceptującym po osiągnięciu stanu akceptującego i vice versa?

Uwaga: rozważamy języki bez słowa pustego, gdyż słowo puste jest zawsze akceptowane przez automat akceptujący pustym stosem.