Egzamin licencjacki/inżynierski — 21 czerwca 2018

Informacja dla zdających egzamin na kierunku informatyka:
Z sześciu poniższych zestawów zadań (Matematyka I, Programowanie, Matematyka dyskretna, Algorytmy i struktury danych, Metody numeryczne, Matematyka II) należy wybrać i przedstawić na osobnych kartkach rozwiązania trzech zestawów.

Informacja dla zdających egzamin na kierunku indywidualne studia informatyczno-matematyczne:
Z sześciu poniższych zestawów zadań (Matematyka I, Programowanie, 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=120 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

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 konkatenacji list $+\!\!+ : L(X) \times L(X) \to L(X)$ jest zdefiniowana w następujący sposób. Dla wszystkich elementów $x \in X$ oraz wszystkich list $xs, ys \in L(X)$ przyjmujemy

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

Dla wszystkich zbiorów X i Y definiujemy funkcję $map : Y^X \times L(X) \to L(Y)$ w następujący sposób. Dla dowolnej funkcji $f : X \to Y$, dowolnego elementu $x \in X$ oraz dowolnej listy $xs \in L(X)$ przyjmujemy

$$\begin{aligned} map(f, \mathsf{nil}) &= \mathsf{nil}, \\ map(f, x : xs) &= f(x) : map(f, xs). \end{aligned}$$

1. Sformułuj zasadę indukcji 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 dowolnych zbiorów X, Y, dowolnej funkcji $f : X \to Y$ oraz dowolnych list $xs, ys \in L(X)$ zachodzi równość

$$map(f, xs +\!\!+ ys) = map(f, xs) +\!\!+ map(f, ys).$$

Uwaga: To jest zadanie z logiki. Przy ocenianiu zwrócimy szczególną uwagę na poprawność i klarowność rozumowania, w szczególności na poprawność użytej zasady indukcji, odpowiednie sformułowanie i użycie wszystkich założeń, odpowiednie użycie kwantyfikatorów i nawiasów itp.

Programowanie

Za tę część egzaminu można otrzymać 20 punktów. Aby otrzymac ocenę dostateczną, należy zdobyć 7 punktów, próg dla dst+ to 9p, dla db – 11p, dla db+ 13p, dla bdb – 15p.

Zadanie 1. Gramatyka G_1 z symbolem startowym S nad alfabetem $\{a, b, c\}$ dana jest za pomocą następującego zbioru produkcji:

$$X \to ab,\ Y \to ac,\ S \to XSY,\ S \to YSX,\ S \to SS,\ S \to \varepsilon$$

Gramatyka G_2 z symbolem startowym S nad alfabetem $\{a, b, c\}$ dana jest za pomocą następującego zbioru produkcji:

$$S \to cSc, S \to ab, S \to aaS,\ S \to bbS,\ S \to \varepsilon$$

Dla gramatyki G przez $L(G)$ rozumieć będziemy język generowany przez G. Dla wyrażenia regularnego r przez $\mathcal{L}(r)$ rozumiemy język opisany przez wyrażenie r.

a) Czy *ababacac* należy do $L(G_1)$? Odpowiedź uzasadnij. **(1)**

b) Czy gramatyka G_2 jest jednoznaczna? Odpowiedź krótko uzasadnij. **(2)**

c) Opisz zbiór $L(G_1) \cap L(G_2)$, odpowiedź uzasadnij. **(3)**

d) Opisz jednym zdaniem, jakie słowa należą do zbioru:

$$A = L(G_1) \cap \mathcal{L}((a + b)^*(a + c)^*))$$

Napisz w języku imperatywnym funkcję, która bierze jako wejście napis i zwraca wartość logiczną, równą True wtedy i tylko wtedy, gdy ten napis należy do zbioru A. Możesz używać języka wybranego z następującej listy: C, C++, Java, C#, Python, Ruby, Go, AWK, Rust. **(4)**

Zadanie 2. (6p) Listę nazwiemy *powtarzalną*, jeżeli składa się z k-krotnego powtórzenia listy o niezerowej długości, dla $k > 1$. Powtarzalna jest na przykład lista `[1,2,3,1,2,3]`, a nie jest `[1,2,3,4,1,2,3,4,1,2,3]`. Uzupełnij poniższy fragment kodu zgodnie z instrukcjami, tak, aby predykat `repeated/1` kończył się sukcesem wtedy i tylko wtedy, gdy jego argument jest powtarzalny.

```
repeated(L) :-
   p1(X,Y, L),
   X = t2,
   L = [_|t3],
   p3(X, L).
```

gdzie `p1` jest nazwą predykatu standardowego, `t2` oraz `t3` to termy, a `p4` to predykat, który powinieneś samodzielnie zdefiniować (wymyślając dla niego bardziej znaczącą nazwę).

Zadanie 3. (4p) Napisz w Haskellu funkcję, która sortuje listę liczb. Funkcja powinna mieć **kwadratową** złożoność. Możesz definiować funkcje pomocnicze, określając w krótkim komentarzu co one robią i podając ich typ.

Matematyka dyskretna

Sześciu mężczyzn i sześć kobiet ma być posadzonych przy okrągłym stole, na zmianę mężczyźni i kobiety. Kto gdzie siedzi nie ma znaczenia, tylko kto jest obok kogo. Ile sposobów usadzenia jest możliwych?

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: selekcja w macierzy kwadratowej uporządkowanej po wierszach i po kolumnach (4 punkty)

Dana jest kwadratowa macierz T o rozmiarach $n \times n$, w której zapisane są liczby rzeczywiste. Liczby w tej macierzy są posortowane po wierszach i po kolumnach, czyli:

$$T_{i,j-1} \leq T_{i,j} \quad \text{dla} \quad i = 0 \ldots n-1\ , \ j = 1 \ldots n-1$$

$$T_{i-1,j} \leq T_{i,j} \quad \text{dla} \quad i = 1 \ldots n-1\ , \ j = 0 \ldots n-1$$

Skonstruuj i opisz efektywny algorytm, który dla zadanej wartości x określi, ile jest elementów mniejszych od x w tablicy T.

- Opisz ideę algorytm, który rozwiązuje ten problem i krótko uzasadnij jego poprawność.
- Zapisz ten algorytm w pseudokodzie.
- Przeanalizuj złożoność obliczeniową (czasową i pamięciową) opisanej metody w zależności od parametru n.

Zadanie 2: zastosowanie zbiorów rozłącznych do wyznaczania minimalnego drzewa rozpinającego w grafie ważonym (5 punktów)

Opisz budowę drzewiastej struktury danych dla *zbiorów rozłącznych*:

- Jakie zadania realizują operacje *union* i *find* w tej strukturze? Na czym polega łączenie według rozmiaru/rangi? Na czym polega kompresja ścieżki podczas wyszukiwania?
- Napisz w pseudokodzie implementację operacji *union* z uwzględnieniem rozmiaru/rangi oraz *find* (może być wersja rekurencyjna) z kompresją ścieżki.
- Jaka jest pesymistyczna złożoność czasowa każdej z tych operacji? Jaka jest złożoność czasowa wykonania ciągu m operacji *union* i *find* na zbiorach rozłącznych z łączeniem według rozmiarów/rang i kompresją ścieżek, które zawierają n różnych elementów?

Następnie napisz jak zastosować opisaną strukturę dla zbiorów rozłącznych do wyznaczenia minimalnego drzewa rozpinającego (MST, ang. minimum spanning tree) w grafie ważonym.

- Opisz krótko algorytm wyznaczania MST z zastosowaniem zbiorów rozłącznych.
- Czy zastosowanie tej struktury ma istotny wpływ na złożoność algorytmu?

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+ 9 pkt., a dla bdb – 11 pkt.

1. $\boxed{\textbf{4 punkty}}$ Wyjaśnij zjawisko utraty cyfr znaczących wyniku. Pokaż na dwóch istotnie różnych przykładach, jak można tego problemu uniknąć.

2. $\boxed{\textbf{4 punkty}}$ Niech dane będą: liczba naturalna n i parami różne liczby rzeczywiste $a_0, a_1, \ldots, a_{n-1}$. Zaproponuj algorytm znajdowania takich liczb $c_0, c_1, \ldots, c_n$, że dla każdego $x \in \mathbb{R}$ zachodzi

$$x^n = c_0 + c_1(x - a_0) + c_2(x - a_0)(x - a_1) + \ldots + c_n(x - a_0)(x - a_1) \cdot \ldots \cdot (x - a_{n-1}).$$

   Podaj jego złożoność obliczeniową i pamięciową.

3. $\boxed{\textbf{4 punkty}}$ Niech dana będzie macierz $A \in \mathbb{R}^{n \times m}$. Przypomnijmy, że *rzędem* macierzy nazywamy maksymalną liczbę jej liniowo niezależnych kolumn. Opracuj algorytm numerycznego wyznaczania rzędu macierzy A. Podaj jego złożoność obliczeniową i pamięciową.

Matematyka II — Algebra

Zadanie 1. (8 punktów)
Niech G będzie grupą a H podgrupą grupy G. Warstwę gH okreslamy jako zbiór

$$gH = \{x \in G \mid \exists h \in H \ x = gh\}.$$

Wykazać, że
a) warstwy g_1H i g_2H są równoliczne,
b) warstwy g_1H i g_2H są albo tożsame albo rozłączne.

Zadanie 2. (5 punktów)
Wektory $v_1, \ldots, v_n$ są liniowo niezależne. Sprawdzić, czy wektory $z_k = \sum_{i=1}^{k} v_i$ są liniowo niezależne.

Progi punktowe: 3, 5, 7, 9, 11 punktów.

Języki formalne i złożoność obliczeniowa

Firma NSA posiada *Dowolnie Szybką Maszynę*$^{\mathrm{TM}}$ (DSM), która jako wejście przyjmuje zwykłą maszynę Turinga M oraz liczbę k, oraz potrafi uruchomić maszynę M (na pustym wejściu) i obliczać jej wartość tak, aby wykonanie każdej instrukcji trwało $1/k$ sekund.

Programy pisane przez pracowników NSA mogą więc zawierać instrukcje postaci $w \leftarrow \mathit{oblicz}(M, k)$, która odpala DSM i po jej zakończeniu (jeśli się skończy) podstawia do w zawartość taśmy M po zakończeniu obliczeń. Wykonanie instrukcji $w \leftarrow \mathit{oblicz}(M, k)$ trwa $T(M)/k$ sekund, gdzie $T(M)$ jest liczbą kroków wykonywaną przez maszynę M (jeśli $T(M) = \infty$, to wykonanie tej instrukcji też trwa nieskończenie długo, to znaczy program się zapętla).

W NSA klasa problemów rozwiązywalnych w czasie wielomianowym oznaczana jest jako $PTIME^{NSA}$ (tzn. jest to klasa takich zbiorów, należenie do których można rozstrzygać przy pomocy programu w czasie wielomianowym).

1. (2 punkt) Czy w NSA można rozwiązywać problemy nierozstrzygalne?
2. (3 punkty) Czy $PTIME^{NSA} = PTIME$, jeśli liczbę k do DSM podaje się zapisaną unarnie? *Wskazówka: Też nie.*

Ocena to liczba zdobytych punktów.