Egzamin licencjacki/inżynierski — 6 lutego 2014

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

Czy istnieje taki zbiór uporządkowany $\langle P, \preceq \rangle$, że

- zbiór P ma moc continuum,
- porządek $\preceq$ jest liniowy,
- w zbiorze P istnieją elementy najmniejszy i największy,
- każdy element z wyjątkiem największego ma następnik, oraz
- każdy element z wyjątkiem najmniejszego ma poprzednik?

Uzasadnij odpowiedź.[1]

Matematyka II — Algebra

Za zadania można otrzymać 13 punktów. Aby otrzymać ocenę dostateczną, należy zdobyć 3 punkty, próg dla dst+ to 5p, dla db – 7p, dla db+ 9p, dla bdb – 11p.

Zadanie 1. (4 punkty)
Dane jest przekształcenie liniowe $T : U \to V$. Wykazać, że *jądro* przekształcenia T jest podprzestrzenią przestrzeni liniowej U, $\ker T < U$.

Zadanie 2. (3 punkty)
Obliczyć multiplikatywną odwrotność liczby 23 modulo 47, tzn. znaleźć x taki, że $23\,x \equiv_{47} 1$.

Zadanie 3. (6 punktów)
Przekształcenie liniowe $T : \mathbf{R}^4 \to \mathbf{R}^4$ ma w pewnej bazie macierz postaci

$$A = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 3 & & 1 & \\ -7 & & & 1 \end{bmatrix}.$$

[1] Dla ułatwienia przypominamy w sposób nieformalny niektóre definicje: moc *continuum* to moc zbioru liczb rzeczywistych; porządek *liniowy* to taki, w którym każde dwa elementy są porównywalne; *następnik* elementu x to taki element y, że $x \prec y$ oraz pomiędzy x i y w porządku $\preceq$ nie ma żadnych innych elementów; *poprzednik* jest dualny do następnika (jest istotnie mniejszy od x i między nim a x nie ma żadnych innych elementów).

i. Uzasadnić, że T jest przekształceniem odwracalnym.
ii. Znaleźć macierz przekształcenia odwrotnego do T.

Programowanie

Za zadanie 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.

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

$$\{A \to a,\ A \to cA,\ A \to Ac,\ A \to cAc,\ B \to b,\ B \to cB,\ B \to Bc,\ B \to cBc,\\ S_1 \to \varepsilon,\ S_2 \to \varepsilon,\ S_1 \to AS_1B,\ S_2 \to BS_2A,\ S \to S_1S_2\} \quad (1)$$

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 *aaccbb* należy do $L(G_1)$? Odpowiedź uzasadnij. **(1)**

b) Czy gramatyka G_1 jest jednoznaczna? Odpowiedź krótko uzasadnij. **(2)**

c) Przedstaw wyrażenie regularne lub gramatykę bezkontekstową generującą zbiór $A_1 = \mathcal{L}(a^*b^*c^*) \cap L(G_1)$ **(2)**

d) Przedstaw wyrażenie regularne lub gramatykę bezkontekstową generującą zbiór $A_1 = \mathcal{L}((ab)^*(ba)^*) \cap L(G_1)$ **(2)**

e) 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 $L(G_1)$. Możesz używać języka wybranego z następującej listy: C, C++, Java, C#, Python, Ruby, PHP, AWK, Pascal. **(3)**

Część 2. Napisz funkcję (w Haskellu) lub predykat (w Prologu) znajdujący największy pierwszy dzielnik danej liczby naturalnej wiekszej od 1. Możesz definiować własne funkcje lub predykaty. Poprzedź definicję opisem algorytmu, który implementujesz, skoncentruj się na czytelności rozwiązania. **(5)**

Część 3. Co robią predykaty:

```
p1(L) :- \+ (append(XX,YY,L), member(X,XX), member(Y,YY), X > Y).
p2(L) :- \+ (member(X,L), member(Y,L), X>Y).
```

Zaproponuj dla obu nazwy, które lepiej oddają ich działanie, a następnie zaimplementuj drugi predykat nie używając negacji (czyli `\+`). **(5)**

Matematyka dyskretna

Jan pożyczył w lombardzie 1000 złotych na 5% miesięcznie. Na koniec każdego miesiąca spłaca 40 złotych. Ile będzie winien po n miesiącach? (Podaj zwarty wzór.)

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: szeregowanie zadań do dwóch procesorów (4 punkty)

Danych jest n zadań o czasach wykonania odpowiednio $t_1, t_2, \ldots t_n$ oraz system komputerowy z 2 procesorami. Zadania te porozdzielane do poszczególnych procesorów i ustawione w kolejkach będą po kolei wykonywane. Jak porozdzielać je do poszczególnych procesorów, aby zminimalizować czas ich przebywania w systemie? Podaj algorytm rozwiązujący ten problem (pseudokod z komentarzami i objaśnieniami), oszacuj jego złożoność obliczeniową (czasową i pamięciową) oraz udowodnij, że znajduje on optymalne rozwiązanie (dowód poprawności jest w tym zadaniu bardzo ważny).

Zadanie 2: dynamiczny zbiór par (5 punktów)

Zaprojektuj strukturę danych umożliwiającą efektywne wykonywanie (najlepiej w czasie logarytmicznym względem bieżącej wielkości zbioru) następujących operacji na początkowo pustym zbiorze S:

- *S.new()* — utworzenie nowego pustego zbioru $S := \emptyset$;
- *S.insert(x,y)* — dodanie pary (x, y) do zbioru $S := S \cup \{(x, y)\}$;
- *S.extract-minx()* — usunięcie z S pary (x, y) o najmniejszej pierwszej składowej x;
- *S.extract-miny()* — usunięcie z S pary (x, y) o najmniejszej drugiej składowej y;
- *S.searchx(x)* — wyszukanie w zbiorze S takiej pary (a, b), że $a = x$;
- *S.searchy(y)* — wyszukanie w zbiorze S takiej pary (a, b), że $b = y$.

Zakładamy, że elementy składowe par ze zbioru S pochodzą ze zbiorów liniowo uporządkowanych. Zakładamy też, że obie składowe każdej pary mogą zajmować dużą przestrzeń pamięciową (nie należy więc powielać żadnej wartości z pary w tej strukturze).

Metody numeryczne

Język programowania `PWO++` ma bogatą bibliotekę funkcji i procedur numerycznych. Wśród nich znajduje się m.in. procedura `Interp_Newton(x,f)` znajdująca dla wektora `x`:= $[x_0, x_1, \ldots, x_n]$ parami różnych liczb rzeczywistych i wektora `f`:= $[f_0, f_1, \ldots, f_n]$ współczynniki b_k $(k = 0, 1, \ldots, n)$ postaci Newtona wielomianu interpolacyjnego $L_n \in \Pi_n$,

$$L_n(x) := b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \ldots + b_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}),$$

spełniającego warunki $L_n(x_i) = f_i$ dla $i = 0, 1, \ldots, n$. Niestety procedura ta ma pewną wadę, mianowicie n **musi być mniejsze** niż 31. W jaki sposób, wykorzystując procedurę `Interp_Newton`, można **szybko i łatwo** wyznaczyć współczynniki postaci Newtona wielomianu $L_{31} \in \Pi_{31}$ spełniającego warunki

$$L_{31}(z_i) = h_i \qquad (i = 0, 1, \ldots, 31;\ z_i \neq z_j \text{ dla } i \neq j;\ h_i \text{ – dane})?$$