Universitatea Al.I.Cuza Iasi **Nume:**
Facultatea de Informatică **Grupa:**

Proiectarea algoritmilor – Test scris 11.04.2014, B

Observaţii:
1. Nu este permisă consultarea bibliografiei.
2. Toate întrebările sunt obligatorii.
3. Fiecare întrebare/item este notată cu un număr de puncte indicat în paranteză. Descrieți conceptele utilizate în răspunsuri.
4. Algorimii vor fi descrişi în limbajul Alk. Se adimit extensii cu sintaxă inspirată din C++ (de exemplu, for, do-while, repeat-until, etc.). Pentru structurile de date utilizate (de exemplu, liste, mulțimi)se va preciza operațiile (fără implementare daca nu se cere explicit) și complexitățile timp și spațiu ale acestora.
5. Nu este permisă utilizarea de foi suplimentare.
6. Timp de răspuns: 1 oră.

1. În contextul căutării cu expresii regulate. Se consderă expresia regula *e* = (a((a + b) c)*)b. (Corectat pe tabla).

a) [0.5] Să se dea definiția expresiilor regulate.

b) [0.5] Să se arate că expresia *e* de mai sus este bine construită (se poate construi arborele sintactic abstract).

c) [0.5] Să se precizeze primele 6 cele mai scurte cuvinte din limbajul desemnat de *e*.

d) [1] Să se construiască automatul asociat lui *e*.

e) [0.5] Să se explice cum se este utilizat automatul pentru a căuta un cuvânt desemnat de e în șirul "aacabcb".

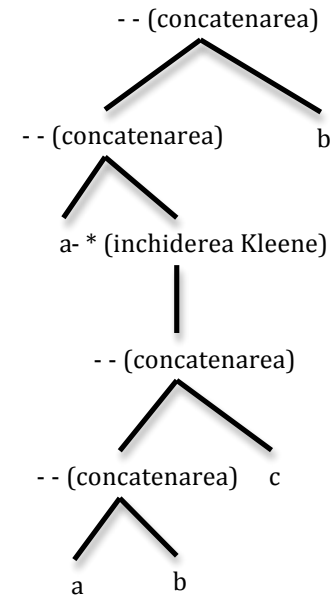
Răspuns.

a)

Mulțimea expresiilor regulate *peste alfabetul A este definită recursiv astfel:*

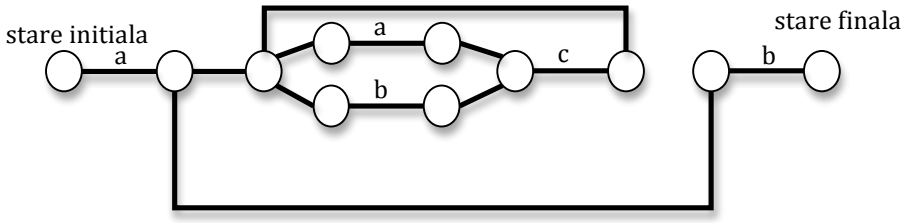
- ε, *empty sunt expresii regulate*
- *orice caracter din A este o expresie regulată;*
- *dacă* e_1, e_2 *sunt expresii regulate, atunci* $e_1 e_2$ *și* $e_1 + e_2$ *sunt expresii regulate;*
- *dacă e este expresie regulată, atunci* (e) *și* e^* *sunt expresii regulate.*

b)



c) L(c) = {c}, L(a+b) = {a, b}, L((a+b)c) = {ac, bc}, L(((a+b)c)*)= {ε, ac, bc, acac, acbc, bcac, bcbc, ...}, L((a((a+b)c)*)b)= {ab, aacb, abcb, aacacb, aacbcb, abcacb, abcbcb, ...}

d)



Trebuie aratat si pasii cum se obtine.

d)

e) Se presupune ca starile sunt numerotate incepand cu 0 de la stanga la dreapta si de sus in jos (din pacate nu au putut fi puse in figura).

1) se pleaca din pozitia i = 0 din subiect si starea initiala (st = 0).

2) primul caracter citit din subiect este a si exista arc etichetat cu a din starea 0; st devine 1, i devine 1.

3) ....

2. În contextul programării dinamice.
a) [0.5] Să se scrie formularea matematică a problemei rucasacului, varianta discretă.
b) [0.5] Să se descrie noțiunea de stare pentru problema rucsacului.
c) [1] Să se descrie cum este utilizat principiul de optim pentru a găsi relația de recurență.
d) [1] Să se explice cum este aplicat algoritmul de programare dinamică pentru instanța n = 4 (numărul de obiecte), M = 10 (capacitatea rucsacului), w = (3, 3, 4, 5) (ponderile) și p = (50, 20, 10, 40). Se va construi tabelul valorilor si se va explica cum este obținută soluția din tabel.

Răspuns.

a)

– funcția obiectiv:

$$\max \sum_{i=1}^{n} x_i \cdot p_i$$

– restricții:

$$\sum_{i=1}^{n} x_i \cdot w_i \leq M$$
$$x_i \in \{0,1\}, i = 1, \ldots, n$$
$$w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \ldots, n$$
$$M \in \mathbb{Z}$$

b)

– funcția obiectiv:

$$\max \sum_{i=1}^{j} x_i \cdot p_i$$

– restricții:

$$\sum_{i=1}^{j} x_i \cdot w_i \leq X$$
$$x_i \in \{0,1\}, i = 1, \ldots, j$$
$$w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \ldots, j$$
$$X \in \mathbb{Z}$$

c)

Dacă $x_j = 0$ (obiectul j nu este pus în rucsac) atunci, conform principiului de optim, $f_j(X)$ este valoarea optimă pentru starea $\text{RUCSAC}(X, j-1)$ și de aici $f_j(X) = f_{j-1}(X)$.

Dacă $x_j = 1$ (obiectul j este pus în rucsac) atunci, din nou conform principiului de optim, $f_j(X)$ este valoarea optimă pentru starea $\text{RUCSAC}(j-1, X-w_j)$ la care se adaugă profitul p_i și de aici $f_j(X) = f_{j-1}(X-w_j) + p_j$.

$$f_j(X) = \begin{cases} -\infty & , \text{dacă } X < 0 \\ 0 & , \text{dacă } j = 0 \text{ şi } X \geq 0 \\ \max\{f_{j-1}(X), f_{j-1}(X-w_j) + p_j\} & , \text{dacă } j > 0 \text{ şi } X \geq 0 \end{cases} \quad (2)$$

d)

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	50	50	50	50	50	50	50	50
2	0	0	0	50	50	50	70	70	70	70	70
3	0	0	0	50	50	50	70	70	70	70	80
4	0	0	0	50	50	50	70	70	90	90	90

X4 = 10, Y4=90
f_4(X4) != f_3(X4), rezulta ca x_3 = 1; X3 = 10-5= 5, Y3 = 90-40=50
f_3(X3) = f_2(X3), rezulta ca x_2 = 0; X2 = X3, Y2 = Y3
f_2(X2) = f_1(X2), rezulta ca x_1 = 0; X1 = X2, Y1 = Y2
f_1(X1) != f_0(X1), rezulta ca x_0 = 1; X0 = 5-3= 2, Y0 = 50-50=0

3. Se consideră problema Acoperirea unei mulțimi (AS):
Intrare:
o mulțime $T = \{t_1, t_2, ..., t_n\}$,
m submulțimi: $S_1, S_2, ..., S_m \subseteq T$ de ponderi (costuri) $w_1, w_2, ..., w_m$.

Iesire: o submulțime $I \subseteq \{1,2,...,m\}$ care minimizează $\sum_{i \in I} w_i$ astfel încât $\bigcup_{i \in I} S_i = T$.
a) [1] Să se arate că AS este în clasa NP. Justificare. Indicatie (pe tabla). Se arata ca varianta ca problema de decizie este in NP.
b) [0.5] Există algoritmi determiniști polinomiali care rezolvă AS? Justificați răspunsul.
c) [1] Să se dea un exemplu de algoritm greedy care calculează o aproximare a soluției optime. Ce se poate spune despre aproximare?
d) [0.5] Precizați complexitatea timp în cazul cel mai nefavorabil a algoritmului greedy.
Răspuns.
a)
Ca problema de decizie: se adauga la intrare un numar intreg K si la iesire raspunsul la intrebarea:

Exista o submulțime $I \subseteq \{1,2,...,m$ astfel încât $\bigcup_{i \in I} Si = T$ si $\sum_{i \in I} wi <= K$?

```
nondetASDec (T, S, w, K) {
 for (i=1; i <= m; ++i)  x[i] = choice(2); // partea de ghicire
 //partea de verificare
 kg = 0;
 for (i=1; i <= m; ++i) kg = kg * x[i];
 if (kg <= K) return succes;
 else return failure;
}
```

b)
Oricare din cele doua variante este problema NP-completa. Deoarece nu se stie daca P = NP, nu se poate da un raspuns exact. Dar se stie ca pana acum nu s-a gasit pentru niciuna din probleme algoritm determinist polinomial care sa o rezolve.
c)
Definim **cost efectiv** al lui S_j ca fiind $w_j / |(S_j - C)|$.

```
asGreedy(T, S, w) {
 I = C = Ø;
 while (T ≠ C) {
   determina Sj cu cel mai mic cost efectiv;
   foreach (ti ∈ Sj-C)
     pret(ti) = wj/|(Sj - C)|;
   I = I ∪ {j};
   C = C ∪ Sj;
 }
}
```

Are ratia de aproximare marginita de $H_n = 1 + 1/2 + \ldots + 1/n$.

d)
- testul $T \neq C$ se executa in O(n), S_j cu cel mai mic cost efectiv in O(m), foreach in O(n), reuniunea se poate executa in O(1)
- bucla while se executa de cel mult n ori (cazul cel mai nefavorabil: fiecare Si include exact un element)
- deosrece m <=n, O(m) = O(n)
- rezulta o complexitate $O(n^2)$.