Universitatea *Al. I. Cuza*, Iași `Numele:`
Facultatea de Informatică `Grupa:`

Proiectarea Algoritmilor - Test Scris (3 iunie 2016), seriile A + E

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 descriși în limbajul Alk (cel utilizat la curs).
5. Nu este permisă utilizarea de foi suplimentare.
6. Timp de răspuns: 1 oră.

---

1. (9p) **Paradigma Greedy**. Se consideră problema rucsacului, varianta continuă (numită mai departe `problema R-C`).
   (a) (3p) Formulați `problema R-C` ca pereche de specificații input-output.

Rezolvare:

Input: `n` - număr de obiecte, `v[0..n-1]` - greutatea obiectelor, `c[0..n-1]` - câştigul asociat fiecărui obiect, `W` - capacitatea rucsacului (toate numere naturale)

Output: cel mai mare număr `y`$\in \mathbb{R}$ astfel încât $\exists p_i \in [0,1] (0 \le i \le n-1)$ şi

i. $y = \sum_i p_i c_i$ (câştigul e `y`) şi
ii. $\sum_i p_i w_i \le W$ (obiectele încap)

(b) (6p) Proiectați un algoritm greedy pentru `problema R-C`:

i. (2p) Descrieți subproblemele rezolvate de algoritmul greedy.

Rezolvare:

`x[n, v[0..n-1], c[0..n-1], W, p[0..n-1]]` = cel mai mare câştig adus de obiectele `i` $\in$ { `0 .. n-1`} de greutate `v[i]`, câştig `c[i]` şi disponibile în proporţie `p[i]`$\in [0,1]$, rucsacul având capacitatea `W`.

ii. (2p) Descrieți criteriul de alegere greedy.

Rezolvare: Se alege **obiectul cu câştig unitar maxim** şi se selectează din el o proporţie cât mai mare (1 dacă obiectul încape integral).

iii. (2p) Scrieți algoritmul greedy pentru `problema R-C`; care este complexitatea-timp în cazul cel mai nefavorabil?

Rezolvare:

```
sortare(n, v, c); // sortează v, c în ordinea descrescătoare a raportului câştig/greutate
câştig = 0;
for (i = 0; i < n; ++i)
{
   if (v[i] ≤ W) {
      p[i] = 1;
      câştig = câştig + c[i];
      W = W - v[i];
   } else {
      p[i] = W / v[i];
      câştig = câştig + p[i] * c[i];
      W = W - p[i] * v[i]; // = 0
   }
}
return câştig;
```

Complexitatea timp este `O(n)`, dată de bucla `for` (plus complexitatea sortării).

2. (9p) **Programare Dinamică**. O *amestecare* a două șiruri de caractere S și T este formată prin intercalarea caracterelor din X și din Y, păstrând caracterele din X și Y în aceeași ordine. De exemplu, $Z =$ **A**B**BBAA**D este o amestecare a șirurilor $X =$ **ABBA** și $Y =$ `BAD` dar $Z =$ `DBAABBA` nu este o amestecare a șirurilor $X =$ `ABBA` și $Y =$ `BAD`, deoarece litera `D` nu apare în ordinea din `BAD`. Proiectați un algoritm care determină daca un șir Z este o amestecare a altor două șiruri X și Y:

   (a) (2p) Formulați problema de mai sus ca pereche de specificații input-output.

Rezolvare:

Input: `Z[0..n-1], X[0..m-1], Y[0..k-1]` - trei șiruri de caractere

Output:
- da, dacă $\exists i_0, \ldots, i_{m-1} \in \{0, \ldots, n-1\}, j_0, \ldots, j_{k-1} \in \{0, \ldots, n-1\}$ astfel încât $i_0 < i_1 < \ldots < i_{m-1}$, $j_0 < j_1 < \ldots < i_{k-1}$, $\{i_0, \ldots, i_{m-1}\} \cap \{j_0, \ldots, j_{k-1}\} = \emptyset$, şi $Z[i_x] = X[x](\forall x = \overline{0, m-1})$ şi $Z[j_y] = Y[y](\forall y = \overline{0, k-1})$.
- nu, altfel

   (b) (7p) Considerați următoarele subprobleme:

$$t[i][j][k] = \begin{cases} 1 & \text{dacă prefixul de lungime } i \text{ a lui } Z \text{ este o amestecare} \\ & \quad \text{a prefixelor de lungime } j \text{ a lui } X \\ & \quad \text{și de lungime } k \text{ a lui } Y \\ 0 & \text{altfel} \end{cases}$$

și răspundeți la următoarele întrebări:

   i. (2p) Cum se pot calcula valorile $t[0][j][k]$ $(0 \leq j \leq len(X),\ 0 \leq k \leq len(Y))$?

Rezolvare:

$$t[0][j][k] = \begin{cases} 1 & \text{dacă } j = k = 0 \text{ (prefixul vid se poate obţine doar prin amestecarea prefixelor vide)} \\ 0 & \text{altfel} \end{cases}$$

   ii. (2p) Care este principiul de optim pentru subproblemele $t[i][j][k]$?

Rezolvare:

Dacă $t[i][j][k] = 1$ şi $Z[i-1] = X[j-1]$ atunci $t[i-1][j-1][k] = 1$; dacă $t[i][j][k] = 1$ şi $Z[i-1] = Y[k-1]$ atunci $t[i-1][k][k-1] = 1$ (ultimul caracter poate "veni" sau din X sau din Y).

   iii. (2p) Cum se poate calcula valoarea $t[i][j][k]$, știind deja valorile $t[i'][j'][k']$ (pentru orice valori $i' < i$, $0 \leq j' \leq len(X)$, $0 \leq k' \leq len(Y)$) și aplicând principiul de optim?

Rezolvare:

$$t[i][j][k] = \begin{cases} 1 & \text{dacă } t[i-1][j-1][k] = 1 \text{ și } Z[i-1] = X[j-1]) \\ 1 & \text{dacă } t[i-1][j][k-1] = 1 \text{ și } Z[i-1] = Y[k-1] \\ 0 & \text{altfel} \end{cases}$$

   iv. (1p) În care dintre subprobleme se găsește răspunsul final al problemei inițiale?

Rezolvare: $t[n][m][k]$.

3. (9p) **Probleme NP-complete**. Se consideră problema 3-colorabilității (dându-se un graf neorientat, să se determine dacă nodurile pot fi colorate folosind maxim 3 culori, astfel încât orice două noduri adiacente să fie colorate diferit).

(a) (1p) Formulați problema 3-colorabilității ca pereche de specificații input-output.

Rezolvare:

Input: un graf neorientat $G = (V, E)$

Output: – da, dacă există $f : V \to \{0, 1, 2\}$ astfel încât $f(v_1) \neq f(v_2)$ pentru orice muchie $\{v_1, v_2\} \in E$.
– nu, altfel

(b) (1p) Definiți clasa `NP`.

Rezolvare: `NP` este clasa problemelor de decizie care pot fi rezolvate în timp polinomial în cazul cel mai nefavorabil de un algoritm nedeterminist.

(c) (2p) Arătați că problema 3-colorabilității face parte din clasa `NP` prin găsirea un algoritm nedeterminist polinomial pentru ea.

Rezolvare:

Considerăm $V = \{x_1, \ldots, x_n\}$; $E = \{(y_1, z_1), \ldots, (y_m, z_m)\}$.

```
for (i = 1; i ≤ n; ++i ) {
   choose col[i] in {0,1,2};
}
for (i = 1; i ≤ m; ++i) {
   if (col[y[i]] == col[z[i]])
     fail;
   }
}
succeed; //complexitate n + m (polinomial)
```

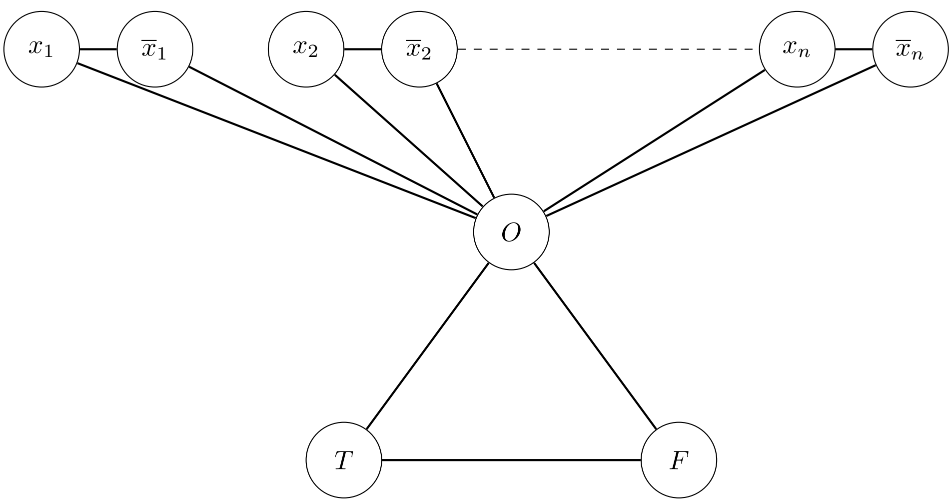
(d) (1p) Definiți noțiunea de problemă `NP`-completă.

Rezolvare:

O problemă este `NP`-completă dacă aparţine lui `NP` şi dacă este `NP`-dificilă.
O problemă `Q` este `NP`-dificilă dacă orice problemă `R∈NP` se reduce la `Q` în timp polinomial.

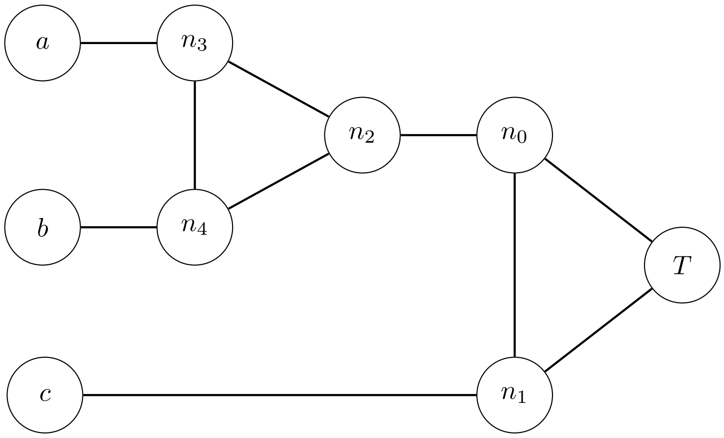
(e) (3p) Știind că problema `3-SAT` este `NP`-completă, arătați că problema 3-colorabilității este `NP`-dificilă. (Indicație: considerați un triunghi format din 3 noduri T, F, O și câte două noduri $x_i, \bar{x}_i$ pentru fiecare variabilă propozițională x_i care apare în formulă, astfel încât nodurile $x_i, \bar{x}_i, O$ să formeze un triunghi. Arătați cum trebuie transpusă fiecare clauză în graf, astfel încât, într-o 3-colorare, literalii "adevărați" să aibă aceeași culoare cu nodul T).

Rezolvare:



(Schiță)
Graful construit este 3-colorabil dacă şi numai dacă formula este satisfiabilă.

Pentru fiecare clauză $a \vee b \vee c$ $(a, b, c \in \{x_1, \bar{x}_1, \ldots x_n, \bar{x}_n)$, adăugăm următoarea construcţie în graf:



Nodurile n_0, n_1, n_2, n_3, n_4 sunt noduri noi. Într-o 3-colorare nu se poate ca a, b şi c să aibă simultan aceeaşi culoare cu nodul F; astfel, cel puţin unul din literalii a, b, c va fi colorat cu aceeaşi culoare ca T.

(f) (1p) Concluzionați că problema 3-colorabilității este NP-completă.

Rezolvare:

Problema 3-colorare este în NP conform (c) şi este NP-dificilă conform (e). Aşadar, problema este NP-completă.