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

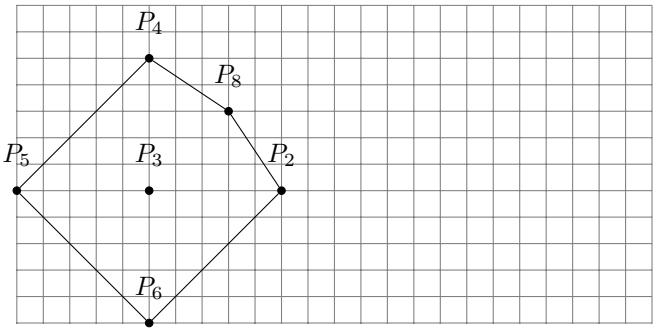
Proiectarea Algoritmilor - Test Scris (23 iunie 2016), toate seriile

Se completează de către profesor	
Oficiu	6
Subiectul 1	
Subiectul 2	
Subiectul 3	
Total	

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. (18p) **Înfășurătoare convexă; Algoritmul lui Jarvis**. Se consideră mulțimea $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ de puncte în plan, unde $P_1 = (8, 8)$, $P_2 = (10, 5)$, $P_3 = (5, 5)$, $P_4 = (5, 10)$, $P_5 = (0, 5)$ și $P_6 = (5, 0)$.
   (a) (3p) Formulați problema înfășurătorii convexe ca pereche *input-output*.

   Rezolvare
   Input: Un vector $P[0..n-1]$ de perechi de numere întregi $(P[i].x, P[i].y)$, reprezentând o mulțime de puncte în plan.
   Output: Varfurile $Q[0..m-1]$ ale celui mai mic poligon convex care conține toate punctele din P în interior (sau pe frontieră).
   (b) (3p) Desenați punctele din mulțimea $\mathcal{P}$ pe grila de mai jos. Care este înfășurătoarea convexă a mulțimii $\mathcal{P}$, dată sub forma unei liste de puncte?



   Rezolvare
   Înfășurătoarea convexă este poligonul ale cărui vârfuri sunt P_6, P_2, P_8, P_4, P_5.
   (c) (3p) Cum este ales, în general, primul punct din înfășurătoarea convex în algoritmul lui Jarvis (cunoscut și sub numele *gift-wrapping method*)? Care este acest punct în exemplul de mai sus?

   Rezolvare
   Este ales punctul Q_1 cu cea mai mică coordonată y (în cazul în care sunt mai multe astfel de puncte, se ia cel mai din dreapta – cu cea mai mare coordonată x). În exemplu, alegem punctul $Q_1 = P_6$.
   (d) (3p) Cum este ales, în general, al doilea punct din înfășurarea convexă în algoritmul lui Jarvis? Care este acest punct în exemplul de mai sus?

   Rezolvare
   Este ales punctul care formează, împreună cu punctul Q_1 ales mai devreme, cea mai mică pantă. În exemplu, alegem punctul $Q_2 = P_2$.
   (e) (6p) Cum sunt alese, în general, celelalte puncte de pe înfășurătoare? Care sunt acestea în exemplul de mai sus?
   Presupunând că ultimele două puncte alese sunt Q_{i-2} și Q_{i-1}, alegem Q_i ca fiind punctul din mulțimea inițială de puncte care maximizează unghiul $\angle Q_{i-2}Q_{i-1}Q_i$. În exemplu, vom alege, pe rând, $Q_3 = P_8$, $Q_4 = P_4$, $Q_5 = P_5$ și la următorul pas ne oprim (deoarece am alege $Q_6 = P_6$, punctul de la care am plecat).

2. (18p) **Căutare peste șiruri; Algoritmul KMP**. Fie $S = ababacb$ un *pattern* și $T = abababacb$ un *text*.
   (a) (3p) Enunțați problema căutării unui șir într-un șir ca pereche de specificații *input-output*.

Rezolvare
`Input`: două șiruri de caractere $S[0..n-1]$ și $T[0..m-1]$ `Output`: cea mai mică poziție $p \in \{0, \ldots, n-1\}$ astfel încât $T[i..i+m-1] = S$ sau -1, dacă nu există o astfel de poziție.
Precizare: se acceptă și varianta de decizie (în care se cere un răspunsul de tipul da/nu) sau varianta în care se cer toate pozițiile pe care apare *pattern*-ul.

   (b) (3p) Ce reprezintă valoarile $f(i)$ ale funcției eșec f asociată unui *pattern* oarecare P ($0 \leq i < len(P)$)?

Rezolvare
$f(i)$ este lungimea celui mai mic prefix propriu al șirului $S[0..i-1]$ care este și sufix (al $S[0..i-1]$).
Precizare: se acceptă orice altă formulare echivalentă și variantele în care se consideră șirul $S[0..i]$ în loc de $S[0..i-1]$.

   (c) (3p) Completați tabelul corespunzător funcției eșec pentru *pattern*-ul $S = ababacb$:

Rezolvare

$S[i]$	a	b	a	b	a	c	b
i	0	1	2	3	4	5	6
$f(i)$	-1	0	0	1	2	3	0

Precizare: se acceptă și variantele în care funcția este shift-ată cu o poziție (deci nu mai apare -1) sau în care valorile funcției sunt shift-ate cu 1 (dar nu variantele inconsistente).

   (d) În algoritmul KMP, se testează dacă *pattern*-ul de mai sus apare la deplasamentul $i = 0$. În cursul execuției algoritmului, s-au testat deja primele $k = 5$ caractere din pattern, care se potrivesc, și tocmai s-a descoperit că al $(k+1)$-lea caracter din pattern, c, nu se potrivește. Situația este exemplificată grafic mai jos:

a	b	a	b	a	b	a	c	b
$=$	$=$	$=$	$=$	$=$	$\neq$			
a	b	a	b	a	c	b		

   i. (6p) Care este următorul deplasament la care algoritmul KMP încearcă potrivirea?

Rezolvare

a	b	a	b	a	b	a	c	b
		$=$	$=$	$=$	?			
		a	b	a	b	a	c	b

Deplasamentul crește cu 2 (devine $i = 2$), deoarece funcția eșec are valoarea $f(5) = 3$ și deci pattern-ul trebuie deplasat cu $5 - 3$ poziții (astfel încât prefixul de lungime 3 să se suprapună peste sufixul de lungime 3 care s-a potrivit).

   ii. (3p) Care este umătoarea comparație de caractere făcută de algoritm? (indicați pozițiile acestor caractere în text și respectiv în pattern)

Rezolvare
Știm deja (datorită valorii funcției eșec $f(5) = 3$) că trei caractere se potrivesc. Deci se compară al 4-lea caracter al pattern-ului ($S[3] = b$) cu caracterul $T[5] = b$.

3. (18p) **Probleme NP-complete**. Fie următoarele probleme:
Vertex Cover
Instance: un graf neorientat $G = (V, E)$, un număr natural K.
Question: există o submulțime $V' \subseteq V$ de noduri astfel încât orice muchie să aibă *cel puțin un capăt în* V' și $|V'| \leq K$?
Independent Set
Instance: un graf neorientat $G = (V, E)$, un număr natural L.
Question: există o submulțime $V' \subseteq V$ de noduri de cardinal $|V'| \geq L$ astfel încât orice muchie să aibă *cel mult un capăt în* V'?

(a) (3p) Definiți clasa NP.

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

(b) (6p) Arătați că problema **Independent Set** face parte din clasa NP prin găsirea un algoritm nedeterminist polinomial pentru ea.

```
IS(V, E, L) // V - lista de noduri, E - lista de muchii
{
  count = 0;
  // ghiceste o multime independenta de noduri
  for (i = 0; i < len(V); ++i) {
    choose x[i] in { 0, 1 };
    if (x[i] == 1) {
      ++count; // calculeaza cate noduri sunt in multime
    }
  }
  if (count < L) {
    fail; // multimea ghicita are prea putine noduri
  }
  for (i = 0; i < len(E); ++i) {
    if (x[E[i].fst] == 1 && x[E[i].snd] == 1) {
      fail; // am gasit o muchie in care amandoua nodurile
            // sunt in multimea ghicita; deci multimea nu este independenta
    }
  }
  succes; // am ghicit o multime de noduri independente de dimensiune >= L
}
```

Algoritmul este polinomial în cazul cel mai nefavorabil deoarece conține doar două bucle `for`, cu marginele superioare date de numărul de noduri și respectiv de muchii.
Precizare: Se acordă 4 puncte pentru algoritm și 2 puncte pentru justificarea complexității.

(c) (3p) Definiți noțiunea de problemă NP-dificilă.
O problemă P este NP-dificilă dacă orice problemă $Q \in$ NP se reduce la P în timp polinomial.

(d) (3p) Știind că problema Vertex Cover este NP-completă, arătați că problema Independent Set este NP-dificilă, găsind o relație de reducere potrivită.
Fie $G = (V, E)$ un graf neorientat. Se poate observa foarte ușor că V' este un *independent set* dacă și numai dacă $V \setminus V'$ este un *vertex cover*. Așadar, $G = (V, E)$ conține un *vertex cover* V' de dimensiune $\leq K$ dacă și numai dacă conține un *vertex cover* $V \setminus V'$ de dimensiune $\geq L = |V| - K$.
Folosind observația de mai sus, reducem problema Vertex Cover la problema Independent Set:

```
vertex-cover(V, E, K)
{
  return independent-set(V, E, |V| - K);
} // reducere Karp în O(1)
```

Cum Vertex Cover este NP-completă, orice problemă Q din NP se reduce în timp polinomail la Vertex Cover. În plus, tocmai am arătat că Vertex Cover se reduce în timp polinomial la Independent Set. Deci orice problema Q din NP se reduce în timp polinomial la Independent Set. Deci Independent Set este NP-dificilă.

(e) (3p) Concluzionați că problema Independent Set este NP-completă.
La punctul (b) am arătat că IS este în NP și la punctul (d) că este NP-dificilă. Deci, prin definiție, IS este NP-completă.