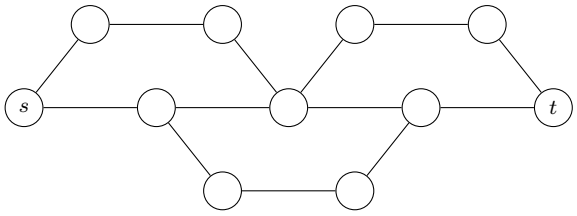
Setul de probleme 2

soluţiile se primesc
miercuri 2 decembrie între orele 10 şi 12, la cabinetul C-402

22 noiembrie 2015

Problema 1. Fie s şi t două vârfuri neadiacente în graful G. Notăm cu $p_l(s,t;G)$ numărul maxim de drumuri intern disjuncte (ca vârfuri) de la s la t în graful G, de lungime cel mult l ($l \in \{1,\ldots,|G|\}$). De asemenea, notăm cu $k_l(s,t;G)$ cardinalul minim al unei mulţimi de vârfuri, diferite de s şi t, prin îndepărtarea cărora din graf nu mai există drumuri de la s la t de lungime cel mult l.
a) Demonstraţi că are loc inegalitatea $p_l(s,t;G) \leq k_l(s,t,G) \quad (*)$.
b) Demonstraţi, utilizând graful de mai jos, că în relaţia $(*)$ inegalitatea poate fi strictă.



c) Specificaţi valori ale lui l din mulţimea $\{2,\ldots,|G|\}$ pentru care $(*)$ are loc cu egalitate pentru orice graf G şi orice două vârfuri s şi t.

(1+2+1= 4 puncte)

Problema 2. Fie $G=(V,E)$ un graf $K_{1,3}$-free şi conex.
a) Demonstraţi că dacă M este un cuplaj de cardinal maxim în G atunci există cel mult un vârf expus relativ la cuplajul M.
b) Se execută o parcurgere `dfs` a lui G dintr-un vârf oarecare. Demonstraţi că arborele (parţial) `dfs` T construit este binar (orice vârf are cel mult doi descendenţi).
c) Din arborele T se construieşte un arbore parţial T' punând pentru fiecare vârf v (cu excepţia rădăcinii), $parent(v) \leftarrow w$, unde w este primul vecin (în G) al lui v întâlnit pe drumul din T de la rădăcina lui T la v. Demonstraţi că descendenţii imediaţi (*children*) în T' ai oricărui vârf diferit de rădăcina lui T' formează o clică în G.
d) Descrieţi un algoritm care în timpul $O(|V|+|E|)$ construieşte un cuplaj de cardinal $\lfloor \frac{|V|}{2} \rfloor$ în graful G.

(2+1+1+2= 6 puncte)

Problema 3. Fie U o mulţime de puncte din $\mathbf{R}^3$ şi $d : U \times U \to \mathbf{R}_{\geq 0}$ distanţa euclidiană. Pentru orice partiţie a lui U cu k clase, $(S_1, \ldots, S_k)$, definim *calitatea* ei ca fiind cea mai mică distanţă dintre două puncte din clase diferite. Algoritmul de mai jos determină partiţia lui U cu k clase, de *calitate* maximă.

Algorithm Kruskal-clustering
Input: $U = \{P_1, \ldots, P_n\}$, $2 \leq k < n$
Output: a partition of U with k classes, of maximum inter-classes distance
1: **for** $i, j \in \{1, \ldots, n\}, i < j$ **do** compute $d(P_i, P_j)$
2: sort the $n(n-1)/2$ elements $e = (P_i, P_j, d(P_i, P_j)$ increasing by key $d(P_i, P_j)$
3: **for** $i = 1, n$ **do** `Make-set`(P_i)
4: $added := 0$; $index := 0$
5: **while** $added \leq n - k$ **do**
6: $index := index + 1$; $e_{index} := (P, Q, d)$
7: **if** `Find`$(P) \neq$ `Find`(Q) **then**
8: `Union`(P, Q)
9: $added := added + 1$
10: return the k sets obtained

În linia 3 se construieşte partiţia lui U cu n clase, $(\{P_1\}, \{P_2\}, \ldots, \{P_n\})$, iniţializând o structura de date pentru utilizarea procedurilor *union-find*.
a) Justificaţi corectitudinea algoritmului *Kruskal-clustering*.
b) Stabiliţi complexitatea timp a algoritmului *Kruskal-clustering*.

(**2+2= 4 puncte**)

Precizări

1. Este încurajată asocierea în echipe formate din 2 studenţi care să realizeze în comun tema.
2. Depistarea unor soluţii copiate între echipe diferite conduce la anularea punctajelor tuturor acestor echipe.
3. Nu e nevoie să se rescrie enunţul problemelor. Nu uitaţi să treceţi numele şi grupele din care fac parte membrii echipei la începutul lucrarii.
4. Este încurajată redactarea latex a soluţiilor.
5. Nu se primesc soluţii prin e-mail.