Tema 2 - Algoritmica Grafurilor

Marin Mălina A6

23.11.2020

Exercitiul 1

a)

Fie g_S gradul fiecarui nod din bipartitia S si g_T gradul fiecarui nod din bipartitia T.
G este bipartit $\Rightarrow$ suma gradelor nodurilor din bipartitia S = suma gradelor nodurilor din bipartitia T = E.
Rezulta ca $|S| * g_S = |T| * g_T \Rightarrow |S| = |T| * g_T / g_S$.
Consideram $|S| \leq |T| \Rightarrow |T| * \frac{g_T}{g_S} \leq |T| \Rightarrow \frac{g_T}{g_S} \leq 1 \Rightarrow g_S \geq g_T$.(3)
Pentru a avea cuplaj, evident $|S| > 0$ deci $g_T > 0$ si $|T| > 0$.
Utilizam Teorema lui Hall: G are cuplaj care satureaza toate nodurile din S ddaca $|N_G(A) \geq |A|, \forall A \subseteq S$.
Fie m_i numarul de muchii incidente cu noduri din $N_G(A)$. Rezulta in cazul nostru ca $m_i = |N_G(A)| * g_T$. (1)
Dar $m_i \geq |A| * g_S$. (2)
Din (1) si (2) rezulta ca $|A| * g_S \leq |N_G(A)| * g_T \Rightarrow |A| \leq \frac{|N_G(A)| * g_T}{g_S}$. Dar din (3) avem ca $\frac{g_T}{g_S} \leq 1$. Rezulta ca $\frac{|N_G(A)| * g_T}{g_S} \leq |N_G(A)| \Rightarrow |A| \leq |N_G(A)|$. Rezulta ca intr-un graf σ-bipartit exista cuplaj care satureaza toate nodurile din bipartitia S (la modul general, bipartitia cu numar de noduri mai mic sau egal cu numarul de noduri din cealalta bipartitie).

b)

Cartile dintr-un pachet de carti sunt sortate in urmatoarele categorii:
-13 carti de inima
-13 de romb
-13 trefla
-13 de frunza
Printre cele sase carti alese la intamplare, cu siguranta vor exista doua carti cu acelasi simbol (doua sau mai multe), pentru ca numarul total de simboluri este de doar 4.
Alegem drept carte ascunsa cartea al carei simbol se repeta si cu valoare maxima dintre cele care au acel simbol.
Prima carte asezata de la stanga la dreapta va fi una din cartile ramase care au acel simbol. Aceasta prima carte va da simbolul cartii ascunse.
Restul de 4 carti vor fi asezate in functie de valoarea lor. Mai intai stabilim ordinea crescatoare
$Minim < Mediu < Mare < Maxim$
Daca avem carti cu aceeasi valoare, consideram: $inima < romb < trefla < frunza$.
Putem spune ca am ordonat cartile astfel: A de inima = 1, A de romb = 2, A de trefla = 3, A de frunza = 4, 2 de inima = 5 etc.
Sunt asadar 4!=24 posibilitati de a le aseza, fiecare mod de asezare corespunzand unui numar, adica diferenta dintre cartea ascunsa si prima carte din stanga. Acest offset poate lua valori din 1,2, ..., 12, deci am avea nevoie de doar 12 posibilitati.
Pentru ca avem de fapt 24 posibilitati de a aseza cele 4 carti si trebuie sa le acoperim pe toate, vom proceda astfel: fiecare offset va corespunde la 2 ordonari diferite. (*)
Fiecare tip de asezare va corespunde unui offset astfel:
Minim Mediu Mare Maxim = 1
Minim Mediu Maxim Mare = 1
Minim Mare Mediu Maxim = 2
Minim Mare Maxim Mediu = 2

Minim Maxim Mediu Mare = 3
Minim Maxim Mare Mediu = 3
Mediu Minim Mare Maxim = 4
Mediu Minim Maxim Mare = 4
Mediu Mare Minim Maxim = 5
Mediu Mare Maxim Minim = 5
Mediu Maxim Minim Mare = 6
Mediu Maxim Mare Minim = 6 etc
La acest offset adunam valoarea primei carti din stanga si obtinem valoarea cartii ascunse, simbolul il stim automat pentru ca e cel al primei carti, deci am gasit cartea ascunsa.
Un astfel de sistem va functiona intotdeauna, putem face o analogie cu punctul a al problemei astfel: consideram graful bipartit G, bipartitia S reprezinta toate offset-urile posibile ce conduc la gasirea valorii cartii ascunse(1-12), bipartitia T reprezinta toate modalitatile posibile de a ordona cele 4 carti (in total 24).Evident,$|S| \leq |T|$. Gradele nodurilor din S = 2 (conform (*)), gradele nodurilor din T = 1, rezulta ca exista un cuplaj care satureaza toate nodurile din S(adica mereu vom putea ghici cartea, fiind acoperite toate offset-urile posibile).

Exercitiul 2

a)

Din punctul de vedere al serverelor, reteaua e conexa, iar initial clientii pot comunica cu toate serverele. Intre serverele care sunt la distanta 2 se creeaza o legatura directa, rezulta ca se creeaza un graf complet K3 intre oricare 3 servere care initial erau la distanta 2. Fie s1,s2,s3 o grupare de astfel de servere.
Daca unul din aceste 3 servere pica, intre celelalte doua se va pastra un fir de legatura directa, iar cel putin unul din acestea doua e conectat la randul lui la cel putin un alt server(reteaua ramane conexa).
Orice client e conectat la minim 2 servere. Daca clientul era conectat la serverul ce a picat, el mai e conectat la inca un server, deci nu va fi afectat. Daca nu era conectat la serverul ce a picat, tot nu va fi afectat.

b)

Se va forma o retea alcatuita doar din 3-clici, orice server va fi conectat la minim 3 alte servere. Daca eliminam 2 legaturi directe ale aceluiasi server cu 2 servere, el va ramane conectat la un minim alt server, care e conectat la randul sau la alte servere, deci reteaua ramane conexa. Daca eliminam 2 legaturi intre servere diferite, acestea vor ramane conectate la cate minim 2 alte servere, deci reteaua ramane conexa.

Exercitiul 3

a)

Presupunem prin reducere la absurd ca 2 b-subgrafuri A,B ale lui G ar avea mai mult de un nod in comun. Atunci ar trebui ca $A \cup B$ sa nu fie 2-conex. Pentru ca A, B sunt b-subgrafuri, nu vom putea deconecta $A \cup B$ stergand un nod care este fie doar in A, fie doar in B. Dar nu vom putea deconecta nici daca stergem unul din nodurile comune, pentru ca mereu vom ramane cu minim un alt nod in comun, si acela va pastra conexiunea intre A si B.

Fie A,B doua b-subgrafuri si x nodul comun.
A 2-conex rezulta A-x ramane conex.
B 2-conex rezulta B-x ramane conex.

$A \cup B$ nu este 2-conex ($A \cup B$ subgraf indus al lui G care contine pe A strict, nu este 2-conex)

$A \cup B - x \Rightarrow$ 2 componente conexe $\Rightarrow$ G se deconecteaza $\Rightarrow$ x punct articulatie.

Mai sus am aratat ca doua sau mai multe b-subgrafuri ale lui G (A,B) pot avea un singur nod in comun (x), acel nod fiind punct de articulatie.

Presupunem prin reducere la absurd ca exista un punct de articulatie care nu este la intersectia a doua b-subgrafuri. Dar asta ar insemna ca G sa nu fie deconectat prin eliminarea acelui nod, pentru ca G este conex si singurele astfel de noduri sunt cele care au gradul = 1.

b)

Situatia in care s-ar forma cicluri ar fi aceea in care doua sau mai multe b-subgrafuri ar avea mai mult de un nod in comun(de exemplu A nod in comun cu B, B nod in comun cu C, C nod in comun cu A). Dar acest lucru nu este posibil(La punctul a) am aratat ca doua sau mai multe b-subgrafuri ale lui G (A,B) pot avea un singur nod in comun (x), acel nod fiind punct de articulatie).

In plus, de la a) avem ca orice punct de articulatie este in intersectia a cel putin 2 b-subgrafuri, rezulta ca nu vom avea puncte de articulatie u care sa nu fie adiacente cu b-subgrafuri H $\Rightarrow$ G este conex

c)

Algoritmul dat este similar cu DFS, insa este modificat.
Daca x este radacina si are 2 copii, orice nod din subarborele unui copil este conectat la orice nod din subarborele celuilalt copil prin nodul radacina. Daca eliminam radacina, cei 2 subarbori se deconecteaza. Analog pentru mai mult de 2 copii(vom avea numar de componente conexe egale cu numarul de copii), rezulta x punct de articulatie.
Daca x nu este radacina, este obligatoriu in primul rand ca x sa nu fie frunza (stergerea frunzei nu deconecteaza G).
In algoritm, order[x] semnifica ordinea(momentul de timp) in care este vizitat nodul x(similar DFS), iar high[x] are rolul de a detecta(daca este cazul) la care stramos se poate ajunge din x printr o muchie de intoarcere.
$order(x) \leq high(v)$ semnifica faptul ca v sau orice copil al lui v nu este conectat cu un stramos al lui x printr-o muchie de intoarcere.
Daca copilul/copiii lui v nu pot ajunge decat prin intermediul lui pe un nivel superior în arborele DFS, atunci eliminarea nodului v duce la deconectarea subarborelui copiilor, deci v e punct de articulatie.

Exercitiul 4

a)

Presupunem ca suntem in momentul in care avem p arbori si ca suntem in cadrul for-ului.
Presupunem ca gasim muchia $e_i = uv$, cu : $uv \in E, u \in V(T_i), v \notin V(Ti) = Tj$, de cost minim.
Dar la un moment dat va trebui sa cautam si o muchie $e_j = st$ unde $st \in E, s \in V(Tj), t \notin V(Tj)$, de cost minim. Dar aceasta muchie a fost gasita deja anterior, este chiar muchia uv. Deci $t \in Ti$ si $e_i = e_j$, (uv=st).
Deci pentru fiecare arbore la care gasim muchie de cost minim care sa respecte conditiile, exista un al doilea arbore la care nu mai atasam nicio muchie. Deci numarul de arbori afectati de adaugarea muchiilor este cel mult p/2.
Rezulta ca dupa fiecare iteratie while numarul de componente conexe se injumatateste cel putin.

b)

Presupunem ca suntem in momentul in care avem p arbori.
Fie T1...Tp componente conexe ale lui T.
Presupunem prin reducere la absurd ca dupa o iteratie while, T are ciclu intr-o componenta conexa Tk. Asta ar insemna ca, in cadrul while - ului, am ales o muchie care are ca extremitati: un nod din Tk si un alt nod, tot din Tk - contradictie cu conditia de alegere a fiecarei muchii uv :$u \in V(Ti), v \notin V(Ti)$.
Rezulta ca dupa fiecare iteratie while T este aciclic.

c)

Algoritmul este similar cu cel de determinare a unui MST. (Prim/kruskal)
Diferenta este doar in modul/ordinea de alegere a muchiilor:
La alg Kruskal se aleg muchii de cost minim netinand cont de noduri, dar avand grija sa nu se formeze niciun ciclu.
La alg Prim se aleg muchii de cost minim dintre cele care au ca extremitati: unul din nodurile deja vizitate si un nod nevizitat.
La acest algoritm se aleg mai intai muchiile de cost minim care pleaca de la fiecare nod astfel incat sa fie vizitate toate nodurile (doar daca este cazul adaugam muchie - am aratat la punctul a)), urmand apoi sa adaugam acele muchii de cost minim care leaga o componenta conexa de alta, pana cand avem 1 componenta conexa finala,adica MST-ul asociat grafului dat.
In concluzie T este un MST al lui G.

d)

La punctul a) am aratat ca dupa fiecare iteratie while, nr de componente conexe ale lui T se injumatateste cel putin. Conditia din while este ca T sa fie neconex.
Initial T are n componente conexe. (nodurile lui G). Dupa o iteratie while, T va avea maxim $\frac{n}{2}$ componente conexe. Dupa 2 iteratii while, T va avea maxim $\frac{n}{2^2}$ comp conexe..Dupa n iteratii while, T va avea maxim $\frac{n}{2^n}$ comp conexe. Executia while-ului se opreste cand T devine conex in intregime.
In alte cuvinte, problema de dimensiune n este divizata in subprobleme de dimensiune n/2 pana cand ajungem la probleme de dimensiune 1 (ne aflam in caz de arbore binar, inaltimea arborelui binar este in cazul nostru $log(n)$).
Deci while-ul se va executa de $log(n)$ ori.
Complexitatea unei iteratii while depinde de numarul de muchii - se cauta muchia de cost minim printre cele care leaga o componenta conexa de alta de un numar constant de ori, numarul total de muchii este m.
Consideram implementarea cu liste de adiacenta (nodurile au 2 campuri, avem si cost pe muchii) $\Rightarrow$ O(1) pentru a adauga o muchie, O(m) pentru a determina muchiile care respecta conditiile (initial avem n componente conexe, se va trece prin toate listele de adiacenta si in plus G e conex, deci are minim n-1 muchii $\Rightarrow$ O(m) $\Rightarrow$ Complexitate finala O(m).
In concluzie complexitatea totala este O($m * log(n)$)).