Tema grafuri 3 bill gates

??

??

1. a) Deoarece M nu este cuplaj perfect, exista macar un nod care nu este saturat de acesta. Vom incerca sa gasim un drum de crestere in T, pornind din acest nod (fie el x). Daca se gaseste un drum de crestere, fie el $P = xv_1v_2...v_n$ (v_n este nod expus), atunci putem aplica operatia $M' \leftarrow M\Delta P$ pentru a obtine un nou cuplaj care va contine muchiile $xv_1, v_2v_3, ..., v_{n-1}v_n$ si va avea $|M'| = k + 1$. Daca M' nu expune deja niciun nod pendant, vom sterge o muchie din el astfel incat sa indeplineasca aceasta conditie; altfel, stergem o muchie la intamplare. Astfel, $|M'| = k$ si am demonstrat ca exista cuplajul cautat. Daca nu exista niciun drum de crestere, inseamna ca orice drum alternat din T care porneste din x se va incheia intr-un nod satisfacut de M. T fiind conex, exista un drum alternat din x pana intr-un nod terminal oarecare y, fie el $P = xv_1v_2...v_ny$ (acest drum are un numar par de muchii, altfel ar fi drum de crestere). Aplicand operatia $M' \leftarrow M\Delta P$ vom obtine un nou cuplaj care va contine muchiile $xv_1, v_2v_3, ..., v_{n-1}v_n$, cu $|M'| = k$. M' va expune nodul pendant y, deci am demonstrat ca exista cuplajul cautat.

b) Vom demonstra a 2-a afirmatie folosind prima:
Daca T are un cuplaj perfect, orice cuplaj de cardinal maxim este perfect. Astfel, orice cuplaj de cardinal maxim va satisface toate nodurile, inclusiv pe cele pendante.

Vom demonstra prima afirmatie folosind-o pe a 2-a:
Daca toate cuplajele de cardinal maxim satureaza toate nodurile pendante, atunci presupunem prin absurd ca exista un astfel de cuplaj care nu este perfect. Conform demonstratiei de la subpunctul **a**, exista si un alt cuplaj de acelasi cardinal (maxim) care expune un nod pendant. Dar, asta este in contradictie cu a 2-a afirmatie, deci este imposibil. Astfel, am aratat ca T are un cuplaj perfect.

2. Conform constructiei sale, reteaua prezentata (exceptand nodul destinatie) este un graf aciclic fara muchii inapoi. Astfel, algoritmul prezentat profita de aceste particularitati si este foarte asemanator cu algoritmul lui Ford & Fulkerson, nodurile de scanat fiind alese intr-o maniera DFS.

Variabila α reprezinta fluxul maxim care ar putea intra intr-un nod (fiind limitat de capacitatile arcelor precedente), iar variabila $flowOut$ reprezinta fluxul care va iesi dintr-un nod. La final, $flowOut$ va reprezenta fluxul care iese din s, adica fluxul maxim din R.

Tabloul x este un flux deoarece respecta proprietatile unuia:

- $0 \leq x[ij] \leq c[ij], \forall (i, j) \in V \times V$
  Algoritmul nu lucreaza cu nicio valoare negativa, deci niciun flux nu poate fi negativ. De asemenea, deoarece α este restrictionat la $min\{\alpha, c[uv]\}$ din apelul anterior, el nu va depasi niciodata $c[uv]$. Datorita **if**-ului din **for**, $flowOut$ nu va depasi niciodata valoarea lui α, deci si $flowOut \leq c[uv]$.

- $\sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0, \forall i \in V \setminus \{s, t\}$
  Un apel al functiei va returna valoarea fluxului care iese dintr-un nod. Astfel, asignarea $x[uv] \leftarrow$ Flow(v, min($\{c[uv], \alpha\}$); va asigura ca fluxul care intra in nodul v este egal cu cel care iese din el.

x este un flux maxim deoarece nu exista niciun drum de crestere in R. Intotdeauna algoritmul va incerca sa asigneze valoarea maxima permisa unui flux, fiind restrictionat doar de capacitatile arcelor. α din apelul initial al functiei este suma tuturor capacitatilor, adica este destul de mare cat sa permita asignarea unui flux care sa satisfaca toate arcele (daca exista unul), deci asignarile nu vor fi restrictionate de α din apelul initial, ci doar de capacitatile arcelor intalnite.

Complexitatea algoritmului este dominata de complexitatea parcurgerii DFS. Restul operatiilor din functie se executa in timp constant, deci complexitatea totala a algoritmului este cea a parcurgerii DFS, adica $O(|V| + |E|)$. $V \setminus t$ fiind un arbore, complexitatea este chiar de $O(|V|)$.

3. a) Calculam $|S|$ si $|T|$:
S este format din X si $n-r$ noduri din W' (care, fiind adaugate, sunt diferite de cele din W, deci si din X). $|S| = |X| + (n-r) = r + n - r = n$.
T este format din Y si restul de r noduri din W'. $|Y|$ poate fi calculat ca $|W| - |X| = n - r$. Deci, $|T| = n - r + r = n$.
In concluzie, $|S| = |T| = n$.

Notam $|\{xy : x \in X, y \in Y, xy \in F\}| = x$.
$|\{xy \in E : x \in S, y \in T\}| = (|\{xy : x \in X, y \in Y\}| - x) + |\{xt : x \in X, t \in T \setminus Y\}| + |\{sy : s \in S \setminus X, y \in Y\}| + |\{st : s \in S \setminus X, t \in T \setminus Y\}| = [r(n-r) - s] + r(n-r) + r^2 + (n-r)^2 = n^2 - x$. Cum $x \geq k \Rightarrow n^2 - x \leq n^2 - k \Rightarrow n^2 - x \leq p$.

b) Presupunem (prin reducere la absurd) ca $X = \emptyset$. Atunci, $X = W \cap S = \emptyset \Rightarrow Y = W \cap T = W$. Din moment ce $|X| = |Y| = n$ si $|S| = |T| = \frac{|V|}{2} = n$ si $X \subseteq S$, $Y \subseteq T$, putem concluziona ca $S = W'$ si $T = W$. Atunci, numarul de muchii dintre S si T va fi $|S| \cdot |T| = n^2$. Dar, conform enuntului numarul acestor muchii trebuie sa fie $\leq p = n^2 - k$ si $k \in N^*$, deci am ajuns la contradictie. Se poate demonstra analog ca si $Y \neq \emptyset$.

Din nou, notam $|\{xy : x \in X, y \in Y, xy \in F\}| = x$.
$|\{xy \in E : x \in S, y \in T\}| = |\{xy : x \in S, y \in T\}| - x = n^2 - x$.
Dar numarul acestor muchii trebuie sa fie $\leq p = n^2 - k$, deci $n^2 - x \leq n^2 - k \Rightarrow \geq k$ (q.e.d.).

c) In primul rand, demonstram ca **MBC** este **NP**:
Verificarea unei solutii se poate face in timp polinomial ($O(m)$), deoarece aceasta consta in numararea muchiilor dintre S si T.

In al doilea rand, demonstram ca **simple-MAX-CUT** (care este **NP**-completa) se poate reduce in timp polinomial la **MBC**:
Fie $H = (W, F)$ o instanta a **simple-MAX-CUT**. Aplicam constructia descrisa in enunt pentru a obtine o instanta de **MBC** - costul timp al acestei constructii este de $O(n)$ (adaugarea a n noduri) + $O(n^2)$ (adaugarea muchiilor intre W si W'), deci in total $O(n^2)$. Apoi, conform **b)**, daca rezolvam **MBC** pe aceasta instanta cu $p = n^2 - 1$, inseamna ca am rezolvat si **simple-MAX-CUT**.

Deci, **MBC** este **NP**-completa.

4. a) Vom construi o retea de transport $R = (G, s, t, c)$ astfel. Exceptand nodurile s si t, aceasta va fi impartita in 2 clase X si Y, $|X| = p$ fiind grupele vechi si $|Y| = r$ fiind grupele noi. Vor exista arcele $sx_i, \forall i \in \{1, ..., p\}$ cu $c(sx_i) = n_i$ de la s la toate nodurile din X (care vor reprezenta numarul initial de studenti din fiecare grupa) si arcele $y_j t, \forall j \in \{1, ..., r\}$ cu $c(y_j t) = m_j$ (care vor reprezenta numarul final de studenti din grupele noi). De asemenea, vor exista si $p \cdot r$ arce $x_i y_j, \forall (i, j) \in (X \times Y)$ cu $c(x_i y_j) = c$ care vor reprezenta cati studenti se vor transfera de la o grupa la alta.

b) Un flux maxim in reteaua prezentata va reprezenta o solutie, dar doar daca acest flux va avea $x[sx_i] = c[sx_i], \forall i \in \{1, ..., p\}$ (altfel, inseamna ca nu s-a gasit o repartizare pentru toti studentii, ci doar pentru o parte din acestia). $x[x_i y_j]$ va reprezenta numarul de studenti care se vor transfera din vechea grupa i in noua grupa j.

c) Rezolvarea acestei probleme se reduce la gasirea unui flux maxim in reteaua prezentata. Pentru asta, vom discuta 3 algoritmi:

- Ford & Fulkerson: Acest algoritm are complexitatea $O(nmU)$ (U fiind majorant al tuturor capacitatilor). Aceasta poate fi problematica daca c este mare, deci se recomanda utilizarea acestuia doar pentru retele cu capacitati scazute.

- Edmonds & Karp: Aceasta optimizare a algoritmului lui Ford & Fulkerson modifica complexitatea la $O(m^2 n)$. Aceasta este mai avantajoasa doar in cazurile in care capacitatile sunt mari (mai mari decat m).

- Ahuja & Orlin: Pentru a aplica acest algoritm, trebuie sa aplicam un preflux $x[sx_i] = n_i, \forall i \in \{1, ..., p\}$ retelei noastre. Algoritmul are complexitatea $O(nm + n^2 \log U)$, care daca U este mic, este mai eficient decat Ford & Fulkerson, dar chiar si pentru valori mai mari complexitatea ar putea fi acceptabila deoarece se calculeaza cu $\log U$, nu U.