Analiza Algoritmilor
Tema 4 - Implementarea unei Reduceri Polinomiale

Termen de predare: **14.01.2018** (100% punctaj)
17.01.2018 (70% punctaj)

Ultima actualizare: *09.12.2017*

1 Introducere

O *reducere Turing* $A \leq_T B$, unde A și B sunt probleme, ne spune că B este *cel puțin la fel de grea* ca A, sau, cu alte cuvinte, dacă putem rezolva B atunci putem rezolva și A folosind o transformare *calculabiă* T.

O *reducere polinomială*[1] $A \leq_p B$, adaugă o constrângere la definiția de mai sus: transformarea T a unei instanțe a problemei A într-o instanță a problemei B poate fi calculată în timp polinomial ($\exists c \in \mathbb{N}, T = O(n^c)$).

Ne propunem ilustrarea unei astfel de reduceri polinomiale prin implementarea transformării T, adică proiectarea și implementarea unui algoritm care transformă instanța in_A a problemei A, într-o instanță $in_B = T(in_A)$ a problemei B, a.î. $A(in_A) = B(in_B)$, pentru orice in_A.

Notă: Ultima egalitate din paragraful de mai sus este ceea ce face ca transformarea să fie **corectă**.

2 k-Colorability & SAT

O *k-colorare*[2] într-un graf neorientat $G = (V, E)$ este o funcție $K : V- > \{1, 2, ..., k\}$ astfel încât $K(u) \neq K(v)$ pentru oricare (u, v) din E.

Informal, o k-colorare este o modalitate de a colora vârfurile unui graf, folosind cel mult k culori, astfel încât nu există două vârfuri adiacente de aceeași culoare.

Problema de decizie **k-Colorability** se enunță astfel:

Există o k-colorare pentru graful G?

Reamintim problema de decizie **SAT**[3]:

Dându-se o expresie booleană φ, există o interpretare I astfel încât $I \models \varphi$?

3 Cerință

Se cere implementarea reducerii $k - Colorability \leq_p SAT$ într-unul din limbajele de programare: *C, Java*.

Programul va primi ca input un graf neorientat și va trebui să returneze expresia booleană rezultată ca urmare a aplicării unei tehnici de reducere **corectă**.

Alături de codul sursă, va fi necesară includerea unui *Makefile* cu următoarele target-uri:

- **build**: compilează codul sursă
- **run**: rulează programul
- **clean**: șterge toate fișierele generate de target-urile anterioare, cu excepția celui de output.

Notă: *make build*, *make run*, *make clean* vor trebui să fie comenzi valide din root-ul arhivei trimise.

De asemenea, este necesara existenta unui fisier README in care sa fie explicata pe scurt transformarea si sa se precizeze complexitatea acesteia pentru a demonstra ca este polinomiala.

Notă: Obtinerea punctajului acordat de checker este conditionata de existenta fisierului README.

3.1 Input

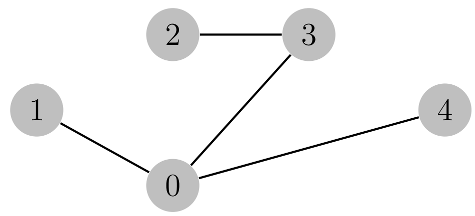
Fișierul de intrare va fi **test.in**.

Pe prima linie se vor afla 3 numere, n, m, k, reprezentând numărul de noduri din graf, numărul de muchii ale grafului, respectiv numarul de culori disponibile.

Pe fiecare din următoarele m linii se va afla câte o pereche de forma (u, v), $0 \leq u, v \leq n - 1$, cu semnificația *există muchie între nodul u și nodul v*.

Exemplu

5 4 3
1 0
0 4
0 3
3 2



3.2 Output

Fișierul de ieșire va fi **test.out**.

Outputul constă într-o singură linie pe care se va afla o expresie booleană. Ca nume de variabile se vor folosi **xk**, $k = 0..N-1$, unde N este numărul total de variabile necesare.

Pentru disjuncție se va folosi caracterul **V**, pentru conjuncție $\wedge$ (shift - 6), iar pentru negație $\sim$ (tilda). Spațiile vor fi ignorate.

Notă: Deoarece nu definim precedența celor 3 operatori, se vor folosi paranteze rotunde oriunde există ambiguități. Spre exemplu, expresia **x1**$\wedge$**x2**$\vee$**x3** nu este validă. În schimb, **(x1**$\wedge$**x2)**$\vee$**x3**, **x1**$\wedge$**(x2**$\vee$**x3)**, **x1**$\vee$**x2**$\vee$**x3**$\vee$**x4** și **x1**$\vee\sim$**x2** sunt expresii valide.

4 Punctaj

Tema valorează **0.5 puncte** din nota finală. Testarea va fi automată.

Ficare zi de intarziere va aduce cu sine o penalizare cu 10 procente din valoarea totala a temei. Dupa 3 zile, nu se vor mai aplica penalizari, punctajul acordat va fi 0.

5 Restricții

Datorită modului de testare a temei (verificarea echivalenței formulelor booleene utilizând ROBDD-uri) este necesar să se definească o convenție de numire a variabilelor folosite.

Astfel, considerăm variabilele implicate ca fiind $x_i,\ i = 0, 1, ..., kn - 1,$ unde n = numărul de noduri, k = numărul de culori, cu semnificația că $x_{ik+j} = 1$ atunci când nodul i are culoarea j, $i = 0...n - 1, j = 0...k - 1$.

Un nod poate avea o singură culoare, iar aceasta trebuie să fie diferită de a tuturor vecinilor săi.

6 Referințe

[1] Polynomial-time reduction
https://en.wikipedia.org/wiki/Polynomial-time_reduction

[2] k-Colorability
https://ro.wikipedia.org/wiki/Colorarea_grafurilor#Colorarea_nodurilor

[3] Boolean satisfiability problem
https://en.wikipedia.org/wiki/Boolean_satisfiability_problem