ANALIZA ALGORITMILOR - TEMA 2
- Octavian's Saga -

(As this Saga enters and after your draw step, add a lore counter. Sacrifice after III.)

Responsabili:
Gabriel Păvăloiu, Alexandru Buzea, Radu Nichita

Deadline soft: **19.01.2023**
Deadline hard: **19.01.2023**

CUPRINS

1 INTRODUCERE

1.1 Obiectiv

- Asocierea și modelarea unor aplicații practice cu probleme NP-grele.
- Implementarea reducerilor polinomiale la problema SAT.
- Aproximarea rezultatului unei probleme NP-grele într-un timp rezonabil.

1.2 Definiții

oracol = este o entitate teoretica capabilă de a rezolva o problemă. Poate fi privit ca o "cutie magica" (black box), care poate produce o soluție pentru orice instanță dată a unei probleme.
CNF = forma conjunctiv normală.
SAT = problema satisfiabilității unei formule logice în format CNF. Este o problemă NP - Completă, conform Teoremei lui Cook.

1.3 Exemplu de reducere la problema SAT

1.3.1 Problema acoperirii cu Varfuri (Vertex Cover)

Se dă un graf neorientat G (cu V noduri și E muchii) și un număr k. Există pentru graful dat o acoperire de k noduri? (O acoperire de k noduri este o submulțime de k noduri ale lui V, cu proprietatea că pentru oricare muchie (u,v) din E, cel puțin unul dintre nodurile u și v aparține submulțimii)

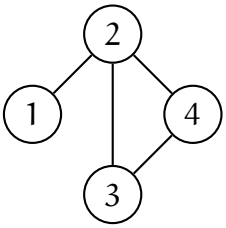
1.3.2 Reducere Vertex Cover $\leqslant_p$ CNF-SAT

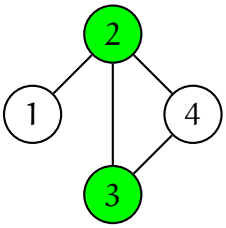
Fie n - numărul de noduri al grafului, construim o expresie booleană cu proprietatea că aceasta este satisfiabilă dacă și numai dacă G are o acoperire de dimensiune k.
Pentru $1 \leqslant i \leqslant n$ și $1 \leqslant r \leqslant k$ se introduc variabilele $y_{i,r}$ cu semnificația: $y_{i,r}$ este adevărată dacă nodul i este al r-lea din acoperire.
Se introduc următoarele clauze:

- $(y_{1,r} \vee y_{2,r} \vee ... \vee y_{n,r})$, pentru fiecare $r, 1 \leqslant r \leqslant k$ - **aceste clauze forțează că există măcar un nod pentru fiecare element al acoperirii**).
- $(\neg y_{i,r} \vee \neg y_{i,s})$ pentru $1 \leqslant i \leqslant n$, $1 \leqslant r, s \leqslant k$ și $r \neq s$ - **aceste clauze forțează că un nod se află cel mult o dată în acoperire (nu este ales pentru mai multi indecsi ai acoperirii**).
- $(y_{u,1} \vee y_{u,2} \vee ... \vee y_{u,k} \vee y_{v,1} \vee y_{v,2} \vee ... \vee y_{v,k})$, pentru fiecare muchie (u,v) din G - **aceste clauze forțează ca oricare muchie să aibă cel puțin un capăt în acoperire**).

Având în vedere construcția de mai sus, expresia este satifiabilă dacă și numai dacă G conține o acoperirie de dimensiune k.

1.3.3 *Exemplu*





k = 2

Pentru graful dat, o posibilă acoperire este mulțimea formată din nodurile 2 și 3. Expresia booleana echivalenta a problemei este:

$E(y_{1,1}, y_{1,2}, y_{2,1}, y_{2,2}, y_{3,1}, y_{3,2}, y_{4,1}, y_{4,2}) =$

$(y_{1,1} \vee y_{2,1} \vee y_{3,1} \vee y_{4,1}) \wedge$	*primul element posibil al acoperirii*
$(y_{1,2} \vee y_{2,2} \vee y_{3,2} \vee y_{4,2}) \wedge$	*al doilea element posibil al acoperirii*
$(\neg y_{1,1} \vee \neg y_{1,2}) \wedge$	*nodul 1 poate fi considerat cel mult o data in acoperire*
$(\neg y_{2,1} \vee \neg y_{2,2}) \wedge$	*nodul 2 poate fi considerat cel mult o data in acoperire*
$(\neg y_{3,1} \vee \neg y_{3,2}) \wedge$	*nodul 3 poate fi considerat cel mult o data in acoperire*
$(\neg y_{4,1} \vee \neg y_{4,2}) \wedge$	*nodul 4 poate fi considerat cel mult o data in acoperire*
$(y_{1,1} \vee y_{1,2} \vee y_{2,1} \vee y_{2,2}) \wedge$	*muchia (1,2) are cel putin un capat in acoperire*
$(y_{2,1} \vee y_{2,2} \vee y_{3,1} \vee y_{3,2}) \wedge$	*muchia (2,3) are cel putin un capat in acoperire*
$(y_{2,1} \vee y_{2,2} \vee y_{4,1} \vee y_{4,2}) \wedge$	*muchia (2,4) are cel putin un capat in acoperire*
$(y_{3,1} \vee y_{3,2} \vee y_{4,1} \vee y_{4,2})$	*muchia (3,4) are cel putin un capat in acoperire*

Aceasta este satisfiabilă, iar o soluție validă este: $y_{2,1}$ = true, $y_{3,2}$ = true, iar restul variabilelor false.

2 PROBLEMA DISCUTATĂ

2.1 Backstory

Săturat de tractorit la GwentStone, Octavian s-a hotărât să se apuce serios de singurul joc de cărți care conteaza: "Magic: The Gathering"™, cel mai complex joc și **primul** collectible card game. Am menționat că e și Turing Complete? Mai multe detalii aici.

Fiind un colecționar la suflet, Octavian nu este interesat de dubluri și formate în care acestea sunt permise. Om de cultură, el joacă doar Singleton și Commander.

2.2 Enunț

Octavian și-a făcut o listă de cărți pe care vrea neapărat să le aibă în colecție, atât ca să le pună in ramă, cât și să isi construiasca niste deck-uri de temut. El are deja niște cărți, care s-ar putea (sau nu) să facă parte din lista dorită.

Eroul nostru poate cumpăra niște 'pachețele' de cărți de pe net, al căror conținut îl cunoaște. Aceste pachețele au un număr variabil de cărți și pot conține și exemplare de care Octavian nu este interesat.

Octavian vrea să afle care este numărul minim de pachețele pe care trebuie să le achiziționeze pentru a-și întregi colecția și care sunt acestea.

Protagonistul nostru a trecut deja materia, așa că vă roagă pe voi sa îi rezolvați problema prin 2 moduri: reducere și aproximare.

2.2.1 Date de intrare

Datele se vor citi de la **stdin**

N M P
a_1
a_2
...
a_N
b_1
b_2
...
b_M
x_1
$s_{1,1}$
$s_{1,2}$
...
s_{1,x_1}
...

```
x_P
s_{P,1}
s_{P,2}
...
s_{P,x_P}
```

Pe prima linie se află 3 numere întregi **N**, **M** și **P**, reprezentând numărul cărților deținute, numărul cărților din colecția dorită și respectiv numărul de pachețele care pot fi achiziționate.

Pe următoarele **N** linii se află numele cărților deja deținute. Pe următoarele **M** linii se află numele cărților din colecția dorită. Urmează **P** secvențe de următorul tip: x_i numărul de carți din pachet, urmat de x_i linii cu numele cărților din pachetul respectiv.

ATENȚIE! Numele cărților conțin spații. Fiecare nume de carte ocupa un rând întreg.

2.2.2 Date de ieșire

Datele se vor afișa la stdout.

```
K
i_1 i_2 ... i_K
```

Pe prima linie se va afișa numărul minim de pachețele achiziționate pentru a întregi colecția. Pe următoarea linie se vor afișa indicii acestor pachețele.

Este garantat că există o soluție.

2.2.3 Exemplu

stdin	stdout	explicatie
1 5 3 Black Lotus Black Lotus Mox Sapphire Ancestral Recall Timetwister Time Walk 3 Mox Sapphire Ancestral Recall Invoke Prejudice 3 Ancestral Recall Timetwister Time Walk 3 Ancestral Recall Timetwister Black Lotus	2 1 2	Octavian are un Black Lotus (WOW!) și vrea să colecționeze cărțile albastre din Power 9. Pentru asta, alege pachețelele 1 și 2. 1 conține și o carte de care nu are nevoie, care va fi ignorata. La final, el are o dublură de Ancestral Recall, dar nu se supără.

3 CHAPTER I: TRIAL OF THE ORACLE

3.1 Enunț

Octavian a găsit un asa-zis "oracol" pe un site dubios. Cei de pe site pretind că acesta ar putea rezolva problema SAT într-un timp rezonabil. Avand experiențe neplăcute cu oracole, Octavian nu este foarte convins, dar, în cazul in care acesta functionează, i-ar prinde bine in rezolvarea dilemei sale.

Înainte să împartășească lista sa prețioasă de cărti, el vrea să testeze temeinic codul găsit pentru a nu avea surprize. Prima voastră sarcină este să puneți la încercare acest oracol, implementând reducerea polinomială de la problema Set Cover la SAT.

Fiind dată o mulțime de forma $\{x_1, x_2, ..., x_n\}$, un număr m de submulțimi nevide ale acesteia și un număr k de submulțimi care trebuie alese, identificați dacă există o alegere a celor k submulțimi astfel încât reuniunea lor să fie chiar mulțimea inițială, reducând instanța de Set Cover astfel primită la o instanță a problemei SAT.

3.1.1 Date de intrare

Datele se vor citi de la stdin.

```
N M K
x1 a1,1 a1,2 ... a1,x1
x2 a2,1 a2,2 ... a2,x2
...
xM aM,1 aM,2 ... aM,xM
```

Pe prima linie se află 3 numere întregi **N**, **M** si **K**, reprezentând cardinalul universului (dimensiunea mulțimii inițiale), numărul total de seturi și respectiv numărul de seturi alese.

Pe următoarele **M** linii se află secvente de urmatorul tip: x_i cardinalul setului curent, urmat de x_i numere întregi, elementele setului.

3.1.2 Date de ieșire

Datele se vor afișa la stdout.
Dacă problema nu are o soluție validă, se va afișa doar *False*

```
False
```

Altfel, dacă problema are soluție, se va afișa următorul output-ul în următorul format.

```
True
K
i1 i2 ... iK
```

Pe prima linie se află o valoare booleană (True sau False), corespunzătoare tipului de expresie oferită în fișierul de input (satisfiabilă/nesatisfiabilă). Pe a doua linie, se va afișa un număr întreg **K**, reprezentand numarul de seturi alese.

Pe ultima linie se află **K** indici ai seturilor alese.

3.1.3 Exemplu

stdin	stdout
4 3 2 2 1 2 3 2 3 4 2 2 3	2 1 2

3.2 Oracolul (SAT)

Nu bate, nu troncăne și presupunem că nu minte.

3.2.1 Date de intrare

Toată interacțiunea cu Oracolul se va face prin fișierele **sat.cnf** și **sat.sol**.

```
p cnf <nr_variables V> <nr_clauses F>
<var_1> <var_2> ...  <var_n1> 0
<var_1> <var_2> ...  <var_n2> 0
...
<var_1> <var_2> ...  <var_nF> 0
```

Prima linie a fișierului **"sat.cnf"** va conține stringul *"p cnf"* urmat de numerele **V** și **F**.

Următoarele **F** linii conțin variabilele pentru fiecare clauză din formula finală. Variabilele sunt despărțite de un spațiu, iar la final veți adăuga un 0, ce va reprezenta terminarea clauzei.

3.2.2 Date de iesire

Răspunsul pe care îl veți primi de la Oracol va conține doar alegerea variabilelor. Va trebui să faceți și o interpretare a acestui răspuns pe baza reducerii pe care ați ales-o.

```
True
V
var_1 var_2 ... var_V
```

Pe prima linie a fișierului **"sat.sol"** se va afla răspunsul Oracolului:

- **True**, dacă formula voastră are soluție.
- **False**, dacă formula voastră nu are soluție.

În cazul în care formula are soluție, pe a doua linie va fi numărul **V**, reprezentat de numărul variabilelor formulei.

Pe ultima linie vor fi **V** numere reprezentând numele variabilelor din formula voastră. Aceste numere vor fi pozitive sau negative, cele pozitive reprezentând atribuirea valorii **True** acestui literal, iar cele negative reprezentând atribuirea valorii **False** literalului respectiv.

3.2.3 Restricții și precizări

- Variabilele nu pot conține 0, deci va trebui să începeți numirea variabilelor de la 1.

3.2.4 Exemplu

sat.cnf	sat.sol
p cnf 6 9 1 2 3 0 4 5 6 0 -1 -4 0 -2 -5 0 -3 -6 0 1 4 0 1 2 3 4 5 6 0 2 3 5 6 0 2 5 0	True 6 1 -2 -3 -4 5 -6

Observație: Interpretarea soluției oracolului depinde de rezolvarea voastră!

4 CHAPTER II: RISE OF THE FALSE PROPHET

Din moment ce Oracolul pare să meargă, Octavian îi încredințează lista sa de cărți și vă roagă să îl folosiți pentru a afla ce pachețele trebuie să cumpere.

Datele de intrare / ieșire sunt exact cele din descrierea problemei.

5 CHAPTER III: REDEMPTION BY APPROXIMATION

Din nefericire, așa-zisul Oracol a dat leak listei de cărți, așa că Octavian va avea un dezavantaj la campionatul de CEDH la care plănuia să participe. Dar eroul nostru nu se dă niciodată bătut! El este dispus să cumpere alte pachețele, chiar și mai multe decât e strict necesar, atât timp cât acest număr este suficient de mic și obține rezultatul într-un timp scurt (se grăbește la concurs).

Vă roagă să îi scrieți un program care să îi indice un număr cât mai mic de pachețele, dar nu neapărat minim.

Cumva, Octavian a pus mâna pe catalogul de AA și este dispus să vă strecoare un bonus dacă acest program are o marjă de eroare suficient de mică.

Datele de intrare / ieșire sunt exact cele din descrierea problemei.

6 CLARIFICĂRI PENTRU FOLOSIREA ORACOLULUI

Să presupunem că în urma reducerii pe care ați făcut-o pentru o problemă aveți o formulă de tipul:
$(x \vee y \vee \neg z \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg y \vee w) \wedge (x \vee \neg y \vee w) \wedge (\neg y \wedge \neg z)$ Pentru a codifica această formulă într-un fișier de tipul DIMACS care să poată fi folosit de Oracol (SAT solver), va trebui să oferim fiecărei variabile câte un număr pozitiv de la 1 până la 4 (numărul total de variabile). Să spunem că alegem $x \to 1$, $y \to 2$, $z \to 3$, $w \to 4$. Atunci fișierul .cnf corespunzător acestei formule va fi

```
p cnf 4 5
1 2 -3 -4 0
-1 3 0
-2 4 0
1 2 4 0
-2 -3 0
```

Prima linie conține antetul obligatoriu *p cnf* urmat de numărul de variabile si numărul de clauze. Următoarele linii vor conține codificările pentru fiecare clauză. Acestea sunt formate dintr-un șir de numere semnificând variabilele din clauză, terminându-se cu cifra 0. Fiecare din numerele pentru clauze pot fi negative sau pozitive, un număr negativ reprezentând că variabila corespunzătoare numărului este negată în clauză, iar unul pozitiv că variabila nu este negată.

Fișierul pe care oracolul o să îl creeze va avea forma

```
True
4
1 -2 3 4
```

Acest fișier arată că formula poate fi satisfăcută și după oferă o alegere a variabilelor din formulă. Pentru cazul nostru, se observa ca alegând y ca *False*, și restul ca *True*, formula este satisfăcută.

7 PUNCTARE

- Punctajul temei este de 100 puncte, distribuit astfel:
  - **Reducere SET -> SAT**: 40p
  - **Rezolvarea prin reducere**: 20p
  - **Rezolvarea prin aproximare**: 30p
  - Comentarii și README: 10p

- Punctajul pe README și comentarii este condiționat de obținerea a unui punctaj strict pozitiv pe cel puțin un test.

- Se poate obține un bonus de maxim 20p pentru trecerea unor teste private ale problemei de aproximare, disponibile doar pe vmchecker. În total se pot obține 120 de puncte (**NU** se trunchiază).

- Vor exista mai multe teste pentru fiecare caz în parte: problema initiala, reducerea SET->SAT. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste.

- În fișierul README va trebui să descrieți soluția pe care ați ales-o pentru fiecare problemă, să precizați complexitatea pentru fiecare și alte lucruri pe care le considerați utile de menționat.

- Pentru a primi punctajul complet pe rezolvarea prin reducere, este nevoie să implementați atât cererea spre oracol, cât și descifrarea răspunsului, deoarece checker-ul verifică doar validitatea răspunsului final, permițându-vă astfel să aveți libertate deplină în alegerea reducerilor pe care doriți să le folosiți.

7.1 Checker

- Atât checkerul cât și scheletul sunt disponibile pe Github.

- Arhiva se va trimite **OBLIGATORIU** pe vmchecker, unde tema se va testa atât folosind setul de teste public, cât și unul privat (pentru bonus).

- Pentru citirea in Java se recomanda folosirea **BufferedReader**.

8 FORMAT ARHIVĂ

• Temele vor fi testate automat pe vmchecker. Acesta suportă temele rezolvate în C/C++ sau Java.
• Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_Tema2.zip** (ex: 399CX_BanigreiOctavian_Tema2.zip).

Arhiva trebuie să conină în directorul **RĂDĂCINĂ** doar următoarele:

- Codul sursă al programului vostru
- Un fișier **Makefile** care să conțină regulile **build** și **clean**.
  - Regula **build** va compila codul vostru și va genera următoarele executabile:
    * **trial** pentru reducerea SET-SAT
    * **rise** pentru problema de mtg cu reducere
    * **redemption** pentru problema de aproximare
  - Regula **clean** va șterge **toate** fișierele generate la build (executabile, binare intermediare etc).
- Un fișier **README** care să conțină prezentarea implementării alese de voi. **NU** copiați bucăți din enunț.

• **ATENȚIE!** Arhiva temei NU va conține: fișiere binare, fișiere de intrare/ieșire folosite de checker, checkerul, orice alt fișier care nu este cerut mai sus.
• **ATENȚIE!** Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).

• **ATENȚIE!** Orice nerespectare a restricțiilor duce la un punctaj **NUL** pe temă (după regulile de mai sus).

8.1 Alte precizări

- Tema poate fi submisă de oricâte ori fără depunctări până la deadline.
- O temă care **NU** compilează pe vmchecker **NU** va fi punctată.
- O temă care compilează, dar care **NU** trece niciun test pe vmchecker, NU va fi punctată.
- Punctajul pe teste este cel acordat de **check** rulat pe **vmchecker**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).

- Ultima temă submisă pe vmchecker poate fi rulată de către responsabili de mai multe ori în vederea verificării faptului că nu aveți buguri în sursă. Vă recomandăm să verificați **local** tema de mai multe ori pentru a verifica că punctajul este mereu același, apoi să încărcați tema.