Analiza Algoritmilor
Tema 2 - K–$CLIQUE \leq_p SAT$

Termen de predare: **15.Jan.2023, 23:50**

Responsabili temă: Mihai-Valentin DUMITRU

Data publicării: *19.Dec.2022*
Ultima actualizare: *19.Dec.2022*

Pentru cea de-a doua temă, va trebui să implementați programatic o reducere de la problema $K–CLIQUE$ la SAT.

1 $K–CLIQUE$

Într-un graf $G = (V, E)$[†], o "clică" este o colecție de noduri, în care fiecare nod are muchie către orice alt nod din clică. În alte cuvinte, este un *"subgraf complet"* al lui G.

O *"k-clică"* este o clică cu k noduri.

Problema de decizie $K–CLIQUE$ primește ca input un graf G și un număr k; dacă graful conține o clică de dimensiune k, răspunsul este $TRUE$; dacă graful nu conține o clică de dimensiune k, răspunsul este $FALSE$.

2 SAT

O formulă logică ϕ, ce constă în variabile booleene și operații logice între acestea, este *satisfiabilă* dacă există o interpretare a variabilelor (o asociere de `TRUE/FALSE` pentru fiecare variabilă) astfel încât formula să fie adevărată.

În contextul temei, o să lucrăm exclusiv cu formule în *forma normală conjunctivă*[‡].

O formulă în formă normală conjunctivă constă într-o *conjuncție* de una sau mai multe *clauze*. O clauză constă într-o disjuncție de unul sau mai mulți *literali*. Un literal este fie o variabilă, fie negarea unei variabile.

Exemplu de formulă CNF:

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (x_2 \vee x_3 \vee \overline{x_4})$$

3 $K–CLIQUE \leq_p SAT$

O posibilă reducere polinomială de la $K–CLIQUE$ la SAT este următoarea:

- Vom avea $k \cdot |V|$ variabile de forma x_{ij}, ce indică faptul că al i-elea nod din clică este nodul v. Deci vom vrea să punem condiții astfel încât să ne asigurăm că, pentru fiecare i de la 1 la k, acest nod există, este unic, și are muchie către toate celelalte.

- Pentru a ne asigura că există un "al i-elea nod" în clică, vom adăuga k clauze de forma:

  $$\bigvee_{v \in V} x_{iv}$$

  pentru $1 \leq i \leq k$.

- Pentru ca toate elementele clicii să fie unice, vom adăuga, pentru fiecare i, j $(1 \leq i < j \leq k)$ și $v \in V$, câte o clauză de forma:

  $$\overline{x_{iv}} \vee \overline{x_{jv}}$$

- Vrem ca oricare două noduri din clică să aibă muchie între ele. Adică vrem ca, pentru orice două noduri între care nu există muchie, cel puțin unul dintre ele să nu facă parte din clică. Deci, pentru fiecare i, j $(1 \leq i < j \leq k)$ și pentru fiecare $u, v \in V$, astfel încât $(u, v) \notin E$, vom adăuga câte o clauză de forma:

  $$\overline{x_{iv}} \vee \overline{x_{ju}}$$

[†] V este mulțimea de noduri, E este mulțimea de muchii.

[‡] În alte surse, este posibil să găsiți această problemă sub denumirea $CNFSAT$.

4 Cerință

Va trebui să implementați programatic o reducere de la K–$CLIQUE$ la SAT. Reducerea implementată poate fi cea descrisă în secțiunea precedentă, sau orice altă reducere corectă. Dacă implementați o altă reducere:

- dacă ați citit despre reducere dintr-o altă sursă, va trebui să o menționați în README.
- dacă este o reducere proprie, va trebui să schițați, în README, demonstrația corectitudinii

Programul vostru va fi invocat cu doi parametrii din linia de comandă:

1. numele unui fișier de intrare, care conține descrierea grafului G și numărul k
2. numele unui fișier de output, în care va trebui să scrieți formula rezultată din transformare

În continuare, prezentăm formatele de input și output.

5 Format input

Pe prima linie a inputului, se găsesc două numere: $|V|$, numărul de noduri din graf, urmat de k. Nodurile grafului sunt notate cu numere de la 1 la $|V|$.

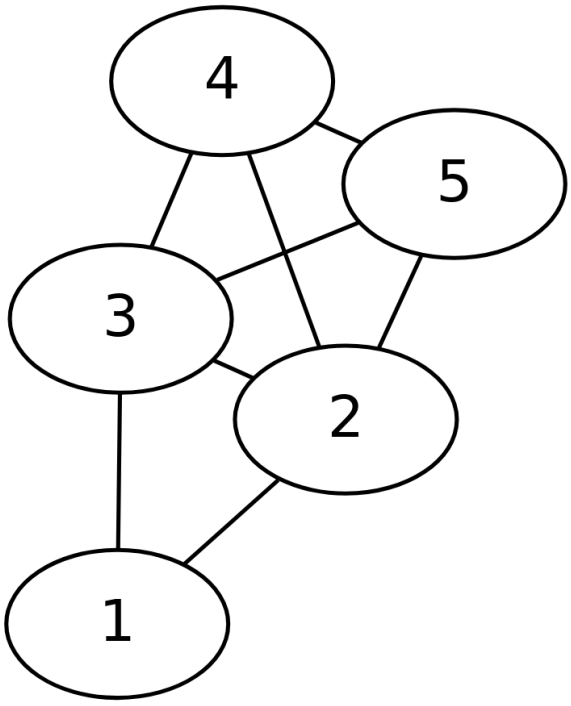
Grafurile vizate în cadrul temei vor avea **maxim 100 de noduri.**

În continuare, urmează $|V|-1$ linii ce reprezintă matricea de adiacență a grafului astfel: pe fiecare linie i se găsesc cel mult $|V|-1$ noduri, reprezentând vecinii **cu indice mai mare** ai nodului i. Astfel, dacă există o muchie între nodul 1 și nodul 2, lista de vecini ai nodului 1 va conține "2", dar lista de vecini a nodului 2 nu va conține "1". *Atenție însă la faptul că graful este neorientat!*.

Exemplu:

```
5 4
2 3
3 4 5
4 5
5
```

Reprezintă graful:



Pentru care se pune întrebarea dacă există o clică de dimensiune 4.

Atenție 1. Sunteți liberi să alegeți orice fel de reprezentare internă a grafului (e.g. matrice de adiacență).

6 Format output

Pentru a reprezenta o instanță de SAT, vom folosi formatul DIMACS, folosit în competiții de SAT-solving.

Prima linie va începe cu cuvintele cheie "p cnf", urmate de numărul de variabile n și de numărul de clauze m, separate de câte un spațiu. Variabilele poartă nume de la 1 la n.

Următoarele m linii reprezintă fiecare câte o clauză. Linia i conține până la n numere întregi corespunzând literalilor care compun clauza i, separate de câte un spațiu. Dacă un literal apare în forma negată, numărul respectiv va fi precedat de un minus "-".

Fiecare linie se termină cu un 0 (separat printr-un spațiu de ultimul literal).

Exemplu:

Formula: $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3)$

Va fi reprezentată astfel:

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

7 Checker și teste automate

Pentru a facilita dezvoltarea temei, vă punem la dispoziție un checker automat și o suită de teste. Checkerul va invoca programul scris de voi pe fiecare test în parte. Rezultatul reducerii voastre este apoi trimis ca input către un SAT solver.

SAT solverul folosit la temă este `Kissat_MAB-HyWalk`, câștigătorul ediției din 2022 a "The International SAT Competition"[§] – o competiție anuală, afiliată cu una dintre conferințele de top din domeniu.[††]

Solverul se găsește în directorul cu același nume; în subfolderul `bin/`, există un executabil precompilat, `kissat`, care va fi invocat de checker. În caz că executabilul nu funcționează pe mașina voastră, puteți să-l recompilați. Din directorul `Kissat_MAB-HyWalk`, rulați:

```
./configure
make
cp build/kissat bin/kissat
```

Există 30 de teste, fiecare test valorând 3 puncte. Pentru fiecare test există o limită de trei secunde de rulare; limita se referă strict la realizarea reducerii polinomiale, nu și la timpul petrecut de SAT solver pentru a determina satisfiabilitatea formulei generate.

În folderul `in/` se găsesc cele 30 de teste, cu un nume de forma `dataX.in`, unde `X` e numărul testului. Pentru fiecare test, există un fișier corespunzător în directorul `ref/`, cu un nume de forma `dataX.ref`, ce conține un singur "1", dacă graful conține o clică de dimensiunea căutată, respectiv un singur "0", dacă graful nu conține o clică de dimensiunea căutată.

Pentru a rula checkerul pe toate testele, este suficient să rulați:

```
./checker.sh
```

Puteți să rulați selectiv pe unul sau mai multe teste (e.g. pe testele 2, 7, 15), menționându-le ca argumente din linia de comandă:

```
./checker.sh 2 7 15
```

[§] https://satcompetition.github.io/2022/results.html
[††] http://satisfiability.org/SAT22/

8 Punctaj

Tema valorează 10 puncte din totalul de 60 de puncte de parcurs. Tema va fi punctată de la 0 la 100, după următoarele criterii.

- 90 de puncte pentru testele automate.
- 10 puncte vor fi acordate în urma unei examinări manuale ce are ca scop evaluarea clarității implementării și a prezentării acesteia în README. Tema nu este axată pe bune practici de dezvoltare software; deci deși vă recomandăm cu căldură să urmăriți un coding-style consecvent și restrictiv, nu se vor scădea puncte pentru aspecte particulare (e.g. linii prea lungi, funcții prea lungi etc.) Codul trebuie să fie însă inteligibil, în caz contrar se vor putea scădea puncte după de caz.

Atenție 2. Scopul testelor este să ajute cu dezvoltarea cerinței și cu punctarea temei.
Voi trebuie însă să rezolvați *cerința*, indiferent de teste. În cazul în care implementarea voastră nu respectă cerința, punctajul oferit de checker poate fi anulat total sau parțial în urma examinării manuale.

9 Trimitere temă

Va trebui să trimiteți următoarele:

- tot codul sursă necesar pentru implementarea temei
- un fișier `Makefile`
- un fișier `README`

Codul sursă poate consta în oricâte fișiere, grupate în orice fel de ierarhie de directoare.

Fișierul `Makefile` va trebui să conțină o regulă de `build`; `make build` trebuie să producă un executabil numit `main`.

În fișierul `README` ar trebui să descrieți, high-level, implementarea voastră. În caz că ați folosit o altă reducere decât cea menționată aici în enunț, va trebui să precizați asta în README.

Acestea trebuiesc puse într-o arhivă `.zip` cu numele:

IDLDAP_Grupa_Tema2.zip

Unde IDLDAP este ID-ul cu care vă logați pe moodle.

De exemplu:

mihai.dumitru2201_321CB_Tema2.zip

Fișierele `Makefile` și `README` trebuie să fie în rădăcina arhivei (nu într-un director în arhivă); `make build` va fi rulat după dezarhivare.

Tema va trebui încărcată pe moodle la această adresă.

9.1 Deadline

Tema trebuie trimisă până la data de **15 ianuarie 2023, ora 23:50**.

10 Forum

Pentru orice întrebări sau neclarități legate de conținutul temei, folosiți forumul temei. Evitați punerea de întrebări pe canale private (e.g. Teams, mail).

Nu postați pe forum fișierul cu implementarea voastră, sau fragmente din acesta. Nu postați pe forum fișierul cu lista de configurații obținute pe un input, sau fragmente din acesta.

11 Plagiat

Rezolvarea temei este individuală.

Copierea totală sau parțială a temei de la un alt student va rezulta în aplicare sancțiunilor menționate în regulament[‡‡] pentru *toți studenții implicați*, fiind irelevant cine de la cine a luat.

Desigur, aveți voie să discutați între voi idei generale de soluții. Dar nu vă uitați pe *implementarea* unui coleg; nu preluați exemple concrete de tranziții, nu cereți ajutor cu depanarea concretă a implementării voastre (i.e. nu prezentați cuiva lista cu configurații pe un anumit input).

În situațiile în care nu sunteți siguri ce constituie un act de plagiat, întrebați pe forumul temei.

[‡‡] https://ocw.cs.pub.ro/ppcarte/doku.php?id=aa:intro:rules