Analiza Algoritmilor

Tema 4 - Reducere k-Vertex-Cover $\leq_P$ SAT

Responsabili: Pirоșca Radu-Andrei, Dorobanțu Florin-Claudiu, Popovici Dan Matei

Termen de predare: 17.01.2021, ora 23:59

Obiective

Obiectivele temei sunt identificarea unei reduceri k-Vertex-Cover $\leq_P$ SAT și implementarea acesteia în limbajul Python.

Motivație

Astfel de reduceri sunt foarte utile în a rezolva probleme care nu fost studiate îndeajuns de mult sau la care nu s-a găsit încă un algoritm optim. De exemplu, dacă avem o problemă X din NP, știind că problema SAT a fost studiată în detaliu și există multe biblioteci care implementează SAT−Solvers, putem reduce problema noastră X la SAT, iar apoi să rezolvăm problema SAT rezultată folosind un SAT−Solver. Astfel vom obține un rezultat pentru problema SAT, care va fi și rezultatul pentru problema X (k-Vertex-Cover).

Problema k-Vertex-Cover

O acoperire a unui graf neorientat este o submulțime de noduri din graf astfel încât, pentru orice muchie din graf, cel puțin unul dintre capetele acesteia se află în submulțime. Altfel spus, este o selecție de noduri care acoperă toate muchiile grafului.

Problema k-Vertex-Cover întreabă dacă, fiind date un graf neorientat G și un număr natural k, există o acoperire de dimensiune k în graful G.

Putem descrie problema în mod explicit în felul următor:

Fie graful neorientat $G = (V, E)$, unde $V = \{v_1, v_2, \ldots, v_n\}$ reprezintă mulțimea nodurilor din graf, iar $E = \{e_1, e_2, \ldots, e_m\}$ reprezintă mulțimea muchiilor din graf, unde $e_i = (u, v)$.

k-Vertex-Cover(G, k) = 1, dacă există o acoperire $V' \subseteq V$ de dimensiune k astfel încât pentru orice muchie $(u, v) \in E$, avem $u \in V'$ sau $v \in V'$.

Problema satisfiabilității booleene (SAT)

SAT întreabă dacă o formulă booleană în **formă normal conjunctivă** este satisfiabilă (dacă există o interpretare care satisface formula). Altfel spus, întreabă dacă există o alegere de valori (True / False) pentru variabilele unei formule booleene astfel încât formula să se evalueze la True (să fie adevărată).

Exemple:

$formula_1 = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_3)$

Această formulă este satisfiabilă, deoarece pentru următoarele atribuiri de valori variabilelor din formulă, valoarea formulei este True: x_1 = False, x_2 = True, x_3 = True. Deci, rezultatul problemei SAT este DA, pentru că există o interpretare care satisface formula.

$formula_2 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$

Această formulă nu este satisfiabilă, deoarece pentru orice interpretare, valoarea formulei este False. Deci, rezultatul problemei SAT este NU, pentru că nu există nicio interpretare care să satisfacă formula.

Cerință

În cadrul acestei teme, ne propunem să identificăm o transformare polinomială **T**, prin care să reducem problema k-Vertex-Cover la problema SAT și să o implementăm în limbajul Python:

k-Vertex-Cover $\leq_p$ SAT

Programul va primi ca input o instanță a problemei k-Vertex-Cover (un graf neorientat **G** și un număr natural **k**) și va trebui să întoarcă o instanță a problemei SAT (o formulă booleană în **formă normal conjunctivă**), cu proprietatea că **G** conține o acoperire de dimensiune **k** dacă și numai dacă formula e satisfiabilă.

Input

Inputul va fi primit în felul următor:

- Pe prima linie se află un singur număr natural **K**, ce reprezintă dimensiunea acoperirii căutate în graful **G**
- Pe a doua linie se află un singur număr natural **N**, ce reprezintă numărul de noduri din graful **G** (nodurile vor fi reprezentate prin numerele naturale de la **1** la **N**)

- Pe a treia linie se află un singur număr natural **M**, ce reprezintă numărul de muchii din graful **G**
- Pe următoarele **M** linii se află câte 2 numere naturale cu spațiu între ele, ce reprezintă nodurile care formează o muchie în graful **G**

Exemplu:

```
2
5
7
4 2
4 1
2 3
4 5
2 1
3 4
3 5
```

Pentru acest exemplu: **K = 2, N = 5, M = 7, G = (V, E), V = {1, 2, 3, 4, 5}, E = {(4, 2), (4, 1), (2, 3), (4, 5), (2, 1), (3, 4), (3, 5)}**.

Inputul programului se citește de la stdin și se garantează a fi întotdeauna valid.

Output

Outputul va trebui să fie o formulă booleană în **formă normal conjunctivă**, reprezentată printr-un șir de caractere, în care:

- Variabilele vor fi codificate ca numere naturale consecutive, începând cu 1 (de exemplu, dacă aveți 5 variabile, acestea vor fi numite 1, 2, 3, 4, 5)
- Negația va fi codificată folosind caracterul ‘~’
- Disjuncția va fi codificată folosind caracterul ‘V’ (majuscula V)
- Conjuncția va fi codificată folosind caracterul ‘^’
- Clauzele vor fi încadrate între paranteze rotunde

Un exemplu de formulă rezultat pentru exemplul anterior este următoarea:

```
(7V3V8V4)^(7V1V8V2)^(3V5V4V6)^(7V9V8V10)^(3V1V4V2)^(5V7V6V8)^(5V9V6V10)^(1V3V5V7V9)^(~1V~3)^(~1V~5)^(~1V~7)^(~1V~9)^(~3V~5)^(~3V~7)^(~3V~9)^(~5V~7)^(~5V~9)^(~7V~9)^(2V4V6V8V10)^(~2V~4)^(~2V~6)^(~2V~8)^(~2V~10)^(~4V~6)^(~4V~8)^(~4V~10)^(~6V~8)^(~6V~10)^(~8V~10)^(~1V~2)^(~3V~4)^(~5V~6)^(~7V~8)^(~9V~10)
```

În funcție de transformarea identificată și de implementarea aleasă, pot exista mai multe șiruri de caractere, ce reprezintă formule corecte. Pentru exemplul precedent, programul va trebui să întoarcă o formulă echivalentă cu cea anterioară, deci nu neapărat pe aceeași.

Outputul programului (formula booleană rezultat) va trebui scris la stdout.

Trimiterea temei

Fișierul sursă Python se va încărca pe platforma hackerrank, unde se va realiza și testarea automată a temei: https://www.hackerrank.com/aa-homework-4.

Pe platforma moodle se va încărca un fișier README în format pdf, în care să explicați transformarea identificată și să arătați că aceasta este polinomială.

Punctaj

Punctajul maxim ce poate fi obținut este 100, acesta fiind împărțit astfel:

- 20p – README
- 80p – Trecerea celor 20 de teste (4p pe test)