

Reducerea polinomială k-Clique \leq_p SAT

I.

1. Demonstratia reducerii polinomiale:* Reducerea polinomială k-Clique \leq_p SAT:Se dem că: $\begin{cases} k\text{-Clique} \leq_p \text{SAT} \\ T \text{ polinomială} \end{cases}$ Consider:
 $\begin{cases} V = \text{multimea de noduri} \\ E = \text{multimea de laturi} \end{cases}$ 

$k\text{-Clique}(G(V, E), k) = 1 \Leftrightarrow \text{SAT}(f) = 1. \simeq$
 $G(V, E)$ este o k-clique $\Leftrightarrow f$ este satisfiabilă \simeq
 $G(V, E)$ este o k-clique, $\exists C \subseteq V$ o clică cu $|C| = k$ a.i.
 $\forall u, v \in C, \exists (u, v) \in E \Leftrightarrow f$ este satisfiabilă.

$T = \begin{cases} \text{Construiește } (C \wedge \Delta \wedge E) \text{ unde: } k = \text{cardinalul clicăi.} \\
 \begin{aligned}
 &\text{- Pt fiecare } i, 1 \leq i \leq k, \exists \text{ cel puțin un nodul } u \in V \\
 &\text{astfel încât nodul } u \text{ este al } i\text{-lea element din clică.} \\
 &C = \bigwedge_{i=1}^k \left(\bigvee_{u \in V} x_i^u \right) \\
 &\text{- Pt fiecare } i, 1 \leq i \leq k, \text{nici o pereche de două laturi} \\
 &\text{nu pot fi simultan a } i\text{-lea latură din clică.} \\
 &\Delta = \bigwedge_{i=1}^k \bigwedge_{\substack{u, v \in V \\ u \neq v}} (\neg x_i^u \vee \neg x_i^v) \\
 &\text{- Pt fiecare } (u, v) \notin E, \text{fie nodul } u \text{ nu aparține clicăi (nu} \\
 &\text{poate fi al } i\text{-lea element din clică), fie nodul } v \\
 &\text{nu aparține clicăi (nu poate fi al } j\text{-lea element)} \\
 &E = \bigwedge_{(u, v) \notin E} \bigwedge_{1 \leq i, j \leq k} (\neg x_i^u \vee \neg x_j^v)
 \end{aligned}
 \end{cases}$

Demonstrația implicăției:

I. „ \Rightarrow ”: $G(V, E)$ este o k -clique $\Rightarrow \exists C \subset V$ cu $|C| = k$, o clică a G
 $\forall u, v \in C, \exists (u, v) \in E \Rightarrow \exists k$ noduri notate x_i^u ale clicăi
 având litera lui corespunzător: $x_i^u = 1$. (x_i^u reprezintă al i -lea
 nod din clică, $1 \leq i \leq k$.)

• Pt formula booleană: ①
 În fiecare dintre clauzele componente ale formulei C , \exists
 cel puțin un literal care va fi alocat valoră
 true. (Iclauză: cel puțin unul dintre toate nodurile grafului
 este nodul 1 din clică; Iclauză: „ ” este
 nodul 2 din clică etc) $\Rightarrow C$ satisfiabilă.

• Pt formula D : ②
 Spune că sau x_i^u este al i -lea nod sau v este al i -lea
 nod, și nu pot fi simultane.
 Din moment ce am asignat true unui singur literal
 corespunzător lui i , iar orice alt $x_i^v = 0$ (dacă v nu e
 nodul i din clică) \Rightarrow situația $\neg 1 \vee \neg 1 = 0 \vee 0$ este imposibilă
 $\Rightarrow D$ satisfiabilă.

• Pt formula E : ③
 Situația $\neg 1 \vee \neg 1 = 0 \vee 0$ este imposibilă pt că ar însemna
 ca două noduri care aparțin clicăi să nu aibă același
 litera $(u, v) \in E$; \forall alte situații de $1 \Rightarrow E$ satisfiabilă.

①, ②, ③ $\Rightarrow T(G(V, E), k)$ produce o formulă satisfiabilă
 \Rightarrow inputul pt SAT este satisfiabil \approx satisfiabil.

II. " \leq " : $SAT(P) = 1 \Rightarrow$ P este o formulă booleană satisfiabilă \Rightarrow În fiecare clausă a lui P , \exists un literal adevărat. \Rightarrow :

$\left\{ \begin{array}{l} c = \text{true}, \text{ pt că } \exists \text{ literal în clausă } x_i^u = \text{true.} \\ \text{reprezentând } k \text{ rezolvări ale clizei.} \\ P = \text{true}, \text{ în fiecare clausă } \exists \text{ doar un literal adevărat!} \\ \Rightarrow \text{ adică 2 literaluri nu sunt ambele adevărate} \\ E = \text{true } \nexists \exists x_i^u = 1 \wedge x_i^u = 0 \text{ unde } i \notin E: \text{ deci } u \text{ nu} \\ \text{poate fi al i-lea el din cliză.} \end{array} \right.$

2. Dem T polinomială : discutata în cod.

-> Complexitate rezultată: $O(n^2) + O(n^3) + O(n^3) = O(n^3)$, polinomială

II.

Compararea timpilor:

```

RUNNING BACKTRACKING category1
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
RUNNING BACKTRACKING category2
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
RUNNING BACKTRACKING category3
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
[TEST6] - PASSED
[TEST7] - PASSED
[TEST8] - PASSED
[TEST9] - PASSED
[TEST10] - PASSED
TOTAL: 21/21
BACKTRACKING TOTAL TIME: 0.736s

```

```

RUNNING REDUCTION category1
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
RUNNING REDUCTION category2
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
RUNNING REDUCTION category3
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
[TEST6] - PASSED
[TEST7] - PASSED
[TEST8] - PASSED
[TEST9] - PASSED
[TEST10] - PASSED
TOTAL: 21/21
REDUCTION TOTAL TIME: 37.662s

```

```

laura@DESKTOP-400QF11: ~/mhc/e/facultate
RUNNING BACKTRACKING category1
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
TOTAL: 4/4
BACKTRACKING TOTAL TIME: 0.143s

RUNNING REDUCTION category1
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
TOTAL: 4/4
REDUCTION TOTAL TIME: 1.572s

REDUCTION / BACKTRACKING: 10.993

```

```

RUNNING BACKTRACKING category2
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
TOTAL: 6/6
BACKTRACKING TOTAL TIME: 0.257s

RUNNING REDUCTION category2
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
TOTAL: 6/6
REDUCTION TOTAL TIME: 33.054s

REDUCTION / BACKTRACKING: 128.614

```

```

RUNNING BACKTRACKING category3
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
[TEST6] - PASSED
[TEST7] - PASSED
[TEST8] - PASSED
[TEST9] - PASSED
[TEST10] - PASSED
TOTAL: 11/11
BACKTRACKING TOTAL TIME: 0.399s

RUNNING REDUCTION category3
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
[TEST6] - PASSED
[TEST7] - PASSED
[TEST8] - PASSED
[TEST9] - PASSED
[TEST10] - PASSED
TOTAL: 11/11
REDUCTION TOTAL TIME: 3.216s

REDUCTION / BACKTRACKING: 8.060

```

Optimizari:

Pt optimizarea algoritmului de reducere am incercat sa renunt, pe cat posibil, la o parte din functiile din biblioteca string.h.

*in loc de strlen, incrementez intr-un contor strlen_C de fiecare data cand adaug un caracter in secventa

*pt concatenare la sfarsitul secventei, strcat-ul ar fi parcurs de fiecare data stringul rezultat de la pozitia 0 pt a gasi finalul, asa ca folosesc un pointer char *point_last_poz care indica ultima pozitie unde se poate adauga un caracter nou, iar acesta va fi incrementat de fiecare data cand se adauga un nou caracter.

*tot pentru evitarea strlen-ului in momentul in care vreau sa concatenez un numar format din mai multe cifre la string trebuie sa stiu cu cat voi incrementa pointerul(cate pozitii trebuie sa sar), iar pentru asta am folosit functia update_nr_of_positions care mai intai verifica daca s-a trecut un prag(10, 100, 1000 etc) si apoi incrementeaza nr_of_positions de sarit.

Timpii de rulare:

Comparand timpii de rulare obtinuti pe fiecare categorie se observa ca pt categoria 2 timpul este semnificativ mai mare. Acest lucru se datoreaza faptului ca inputurile pt categoria 2 genereaza grafuri foarte mari (de ex nr de noduri: 145, 135 etc) si implicit si formula booleana SAT rezultata este una foarte complexa si lunga. Rezolvarea unei astfel de formule de catre un sat

solver o sa dureze mult pentru ca vor rezulta n^k simboluri carora li se vor atribui true/false in toate combinatiile posibile.