Bonyolultságelmélet gyakorlat – 03
Visszavezetések

Recap: eldöntési probléma

Egy probléma **eldöntési probléma**, ha definíciójában az output **egybites** (igen/nem, 1/0, Accept/Reject)

Néhány eldöntési probléma:

SAT

- **Input:** egy F formula CNF alakban

  Recap: CNF

  ítéletkalkulus-beli konjunktív normálforma, pl. $(p \vee q) \wedge \neg p \wedge \neg q$

- **Output:** kielégíthető-e F? (a megadott példa pl. egy „nem” példány, minden értékadásra hamis lesz)

ez nagyon fontos

DNF-tautológia

- **Input:** egy F formula DNF alakban

  Recap: DNF

  diszjunktív normálforma, pl. $(p \wedge q) \vee \neg p \vee \neg q$

- **Output:** tautológia-e F? (a megadott példa pl. egy „igen” példány, minden értékadásra igaz lesz, azaz tautológia)

Recap: az input elkódolása számokkal

A fenti problémákban **nem számsorozatok** az inputok, ez nem baj: azt kell vegyük az input n méretének, ahány biten „értelmesen” el tudjuk tárolni, pl:

- ha egy N-csúcsú **gráf** az input: $n = O(N^2)$, pl. a szomszédsági mátrixot használva
- ha egy N hosszú **string**: $n = O(N)$, feltéve, hogy az ábécénk fix darab, előre rögzített betűt használ
- ha egyetlen N **szám**: $n = O(\log N)$ (ezt már láttuk korábban)
- ha egy **formula**: mintha string lenne

Eldöntési problémák egymáshoz képesti nehézségét **visszavezetésekkel** definiáljuk: kb. „a B probléma legalább olyan nehéz, mint az A, ha egy B-t megoldó algoritmus egy gyors $A \to B$ inputkonverzió után meg tudja oldani A-t is”, azaz:

Recap: Visszavezetés

ez nagyon fontos

Az A eldöntési probléma (hatékonyan) visszavezethető a B eldöntési problémára, jelben $A \leq B$, ha van olyan f inputkonverziós függvény, ami

- A inputjaiból B inputjait készíti,
- **polinomidőben** (ettől „hatékony”)

  azaz pl. ha A inputja n-bites, akkor f (mondjuk) $O(n^2)$ lépésben készíti el ebből B-nek az $f(x)$ inputját, vagy $O(n^3)$ stb, valami polinom időkorláttal,
- és **választartó** módon: minden x inputjára A-nak $A(x) = B(f(x))$

  azaz f az A „igen” példányaiból B-nek is „igen” példányait készíti, „nem”-ből pedig „nem”-et

Ha A-ról B-re f egy visszavezetés, és B-re van (hatékony) solverünk, akkor A-ra is van:

```
1 bool solveA(x) {
2   return solveB(f(x))
3 }
```

Recap: komplementer

Az A eldöntési probléma **komplementere**, jelben $\overline{A}$, az a probléma, melynek ugyanazok az inputjai, mint A-nak, és minden egyes inputon épp a másik választ definiálja, mint A.

Például $\overline{\text{SAT}}$ inputjai ugyanúgy CNF-ek, mint ahogy SAT-nak is, de pont hogy a kielégíthetetlenekre kell „igen”-t és a kielégíthetőkre „nem”-et mondjunk.

1. feladat.

Adjunk meg egy hatékony visszavezetést SAT **komplementeréről** DNF-TAUTOLÓGIÁra!

PÁROSÍTÁS

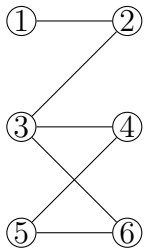
- **Input:** egy G gráf
- **Output:** Van-e G-ben teljes párosítás?

  Recap: teljes párosítás

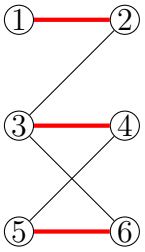
  azaz: tudunk-e kiválasztani élei közül néhányat úgy, hogy minden csúcsra **pontosan egy** kiválasztott él illeszkedjen?

Irányítatlan gráfokat általában G-vel, irányítottakat $\vec{G}$-vel fogunk jelölni.

Például a következő gráf a PÁROSÍTÁS problémának egy **igen** példánya...



...mert a **piros** élek benne egy teljes párosítást alkotnak:



2. feladat.

Adjunk meg egy hatékony visszavezetést PÁROSÍTÁSról SATra!

2-SZÍNEZÉS

- **Input:** egy G gráf
- **Output:** ki lehet-e színezni G **csúcsait** két színnel úgy, hogy szomszédos csúcsok mindig különböző színt kapjanak?

ez kicsit fontos

3. feladat.

Adjunk meg egy hatékony visszavezetést 2-SZÍNEZÉSről SATra!

4. feladat.

Adjunk meg egy hatékony visszavezetést SATról DNF-TAUTOLÓGIÁra!

1. feladat megoldása.

Kell adjunk egy olyan f, gyorsan kiszámítható függvényt (gyors algoritmust), ami

- egy F CNF-ből ($\overline{\text{SAT}}$ inputjából) egy $f(F)$ DNF-et (DNF-TAUTOLÓGIA inputját) készít
- választartó módon, azaz:
  - kielégíthetetlen F-ból tautológia $f(F)$-et (a $\overline{\text{SAT}}$ „igen” példányaiból a DNF-TAUTOLÓGIA „igen” példányait)
  - kielégíthető F-ből pedig **nem** tautológia $f(F)$-et („nem” példányból „nem” példányt)

Itt eszünkbe juthat logikáról két-három dolog:

- kielégíthetetlen formula negáltja tautológia lesz,
- kielégíthető formula negáltja nem lesz tautológia,
- és egy negált CNF-en ha alkalmazzuk a deMorgan-azonosságokat, akkor **gyorsan** kapunk egy, a negált CNF-fel ekvivalens DNF-et.

Tehát f: az input CNF minden egyes literáljának vegyük a komplementerét, és cseréljük meg a $\vee$ és $\wedge$ jelek szerepét is. (ez az „alkalmazzuk a deMorgan-szabályokat a CNF negáltján” rész)

Pl: $(p \vee q) \wedge \neg p \wedge \neg q \ \mapsto \ (\neg p \wedge \neg q) \vee p \vee q$, erre tényleg igaz, hogy

- gyors algoritmus: $O(n)$, azaz lineáris időben végre tudjuk hajtani, egyszer kell hozzá csak balról jobbra végigolvasni a formulát és egyből tehetjük is outputra a készített DNF elemeit,
- választartó, hiszen a generált DNF az input CNF negáltjával lesz ekvivalens, így kielégíthetetlenből tényleg tautológiát készít, kielégíthetőből pedig nem tautológiát.

2. feladat megoldása.

Amikor a SAT problémára vezetünk vissza valami A problémát (most a PÁROSÍTÁSt), érdemes a következő sémában gondolkodni:

1. keressünk valamiféle „tanúsítvány”t az A probléma inputjaihoz, ami
   - az adott inputra megmutatja, hogy a kérdésre a válasz miért „igen”,
   - maga a „tanúsítvány” mérete kicsi,
   - ha valaki ad nekünk egy állítólagos „tanút”, azt gyorsan tudjuk ellenőrizni,
   - az „igen” példányokra legyen tanú (nem baj, ha több is van, de legyen legalább egy), a „nem” példányokra ne legyen tanú

Ebben az esetben pl. egy „lehetséges tanú” az éleknek egy részhalmaza, ami akkor lesz tényleg tanú, ha minden csúcsra pontosan egy eleme illeszkedik.

2. ezután keresünk egy „logikai kódolást”, amiben logikai változók értékeivel kódolhatunk el egy-egy lehetséges tanút

Most az éleknek egy részhalmaza egy lehetséges tanúsítvány, amit pl. el lehet úgy kódolni, hogy minden egyes e élhez bevezetünk egy x_e logikai változót, úgy, hogy ha x_e-t 1-re állítjuk, az jelentse azt, hogy e benne van a kiválasztott élek halmazában, ha pedig 0-ra, az azt, hogy nincs.

3. végül, megpróbálunk készíteni egy (rövid, gyorsan felírható) formulát az így bevezetett változókon úgy, hogy ha a változók egy értékadása (ami egy „lehetséges tanú”) tényleg egy tanút kódol el, akkor a formula értéke igaz legyen, ha pedig nem tanút, akkor hamis legyen.

Most pl. azt kéne felírnunk, hogy minden csúcsra pontosan egy olyan „változó” (él) illeszkedik, aminek 1 az értéke. Ezt legegyszerűbben logikában úgy tudjuk felírni, hogy felírjuk h minden csúcsra legalább egy 1 értékű változó illeszkedik (ehhez csak össze kell vagyoljuk minden csúcsra a rá illeszkedő élek változóit, ezeket összeéselni, eddig jó, mert ez egy CNF), és hogy minden csúcsra legfeljebb egy 1 értékű változó illeszkedik (ezt pl. úgy, hogy ha u is és v is illeszkedik egy csúcsra és $u \neq v$, akkor felvesszük a $\neg(x_u \wedge x_v)$ tagot, azaz a $(\neg x_u \vee \neg x_v)$ klózt).

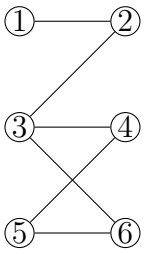
Tehát a visszavezetés összefoglalva: ha $G = (V, E)$ az input gráf, akkor

- minden $e \in E$ élhez felveszünk egy x_e logikai változót,
- ciklusban végigmegyünk G csúcsain, és ha a v csúcsra az $e_1, \ldots, e_k$ élek illeszkednek, akkor felvesszük az $(x_{e_1} \vee x_{e_2} \vee \ldots \vee x_{e_k})$ klózt („legalább egy igaz közülük”),
- ezen kívül, ha v-re e_1 és e_2 is illeszkedő, különböző élek, akkor a $(\neg x_{e_1} \vee \neg x_{e_2})$ klózt is („legfeljebb egy igaz közülük”).

Ez az inputkonverzió választartónak választartó, még a futásidején kell elgondolkodni: de mivel ciklusban végigmegyünk a csúcsokon (ez $O(n)$-szer fut le), és minden csúcson belül egyszer végigmegyünk a szomszédain (ez a ciklusmagban egy $O(n)$ időigény), majd egyszer pl. két egymásba ágyazott ciklussal végigmegyünk a különböző szomszédain (ez pedig $O(n^2)$), így a teljes időigény $O(n) \times (O(n) + O(n^2)) = O(n^3)$ lesz, ami polinom, tehát hatékonynak számít.

Egy példa erre:

Ha az input gráfunk az alábbi...



...akkor a kapott formulánk:

$$
\begin{aligned}
& (x_{12}) \\
\wedge\ & (x_{12} \vee x_{23}) && \wedge\ (\neg x_{12} \vee \neg x_{23}) \\
\wedge\ & (x_{23} \vee x_{34} \vee x_{36}) && \wedge\ (\neg x_{23} \vee \neg x_{34}) \wedge (\neg x_{34} \vee \neg x_{36}) \wedge (\neg x_{23} \vee \neg x_{36}) \\
\wedge\ & (x_{34} \vee x_{45}) && \wedge\ (\neg x_{34} \vee \neg x_{45}) \\
\wedge\ & (x_{45} \vee x_{56}) && \wedge\ (\neg x_{45} \vee \neg x_{56}) \\
\wedge\ & (x_{36} \vee x_{56}) && \wedge\ (\neg x_{36} \vee \neg x_{56})
\end{aligned}
$$

---

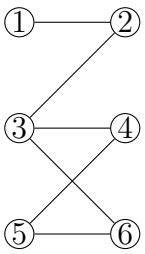
3. feladat megoldása.

Az előző feladat módszerét alkalmazva:

- itt egy **lehetséges tanú** egy színezés: minden csúcs (pl) a kék és a piros színek egyikét kapja,
- ami akkor **tanú**, ha minden élnek a két vége különböző színű lett,
- amit el lehet úgy *kódolni*, hogy minden v **csúcs** kapjon egy x_v logikai változót,
- a tanú ellenőrzése pedig: minden $e = (u, v)$ élre felírjuk, hogy x_u és x_v különböző, amit az $x_u \leftrightarrow \neg x_v$ formulával tehetünk meg, ami CNF-ben $(x_u \vee x_v) \wedge (\neg x_u \vee \neg x_v)$ lesz.

Egy példa erre:

Ha az input gráfunk az alábbi...



...akkor a kapott formulánk:

$$
\begin{aligned}
& (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \\
\wedge\ & (x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \\
\wedge\ & (x_3 \vee x_4) \wedge (\neg x_3 \vee \neg x_4) \\
\wedge\ & (x_3 \vee x_6) \wedge (\neg x_3 \vee \neg x_6) \\
\wedge\ & (x_4 \vee x_5) \wedge (\neg x_4 \vee \neg x_5) \\
\wedge\ & (x_5 \vee x_6) \wedge (\neg x_5 \vee \neg x_6)
\end{aligned}
$$

4. feladat megoldása.

Ha ez nem sikerült, ne búsuljunk, eddig senkinek nem sikerült a világon adni egy ilyet, sem pedig bizonyítani, hogy nincs eközött a két probléma között visszavezetés.