Oplossingen oefeningen Theoretische Informatica II
26 april 2001

1.a. Uit Definition 7.15 herinneren we ons dat een algoritme V een verifier is voor een taal A als voor elke mogelijke input w geldt dat:

$$w \in A \quad \Longleftrightarrow \quad \exists c : V \text{ aanvaardt } \langle w, c \rangle$$

In ons geval is:

- A de taal CLIQUE gedefiniëerd onderaan pagina 245;
- V het algoritme beschreven in de proof van Theorem 7.20;
- elke mogelijk input w van de vorm $\langle G, k \rangle$.

We gaan eerst $\Rightarrow$ na. Gegeven $\langle G, k \rangle \in$ CLIQUE. Dit wil zeggen dat G een k-clique heeft; noem deze k-clique Q, dit is dus een verzameling van k knopen uit G zodat elk koppeltje van knopen uit Q verbonden is in G. We moeten aantonen dat er een c bestaat zodat $\langle \langle G, k \rangle, c \rangle$ zal aanvaard worden door V. De gezochte c is gewoon een opsomming van de knopen in Q. Inderdaad:

- Test 1 van V zal dan lukken;
- Test 2 zal ook lukken, Q is immers een clique!

We gaan nu $\Leftarrow$ na. Gegeven is dus dat er een c bestaat zodat $\langle \langle G, k \rangle, c \rangle$ aanvaard wordt door V. Dus moet al zeker test 1 van V lukken. Dus moet c een verzameling van k knopen uit G zijn. Ook test 2 moet lukken. Dus moet elk koppeltje knopen uit c verbonden zijn in G. We besluiten dat c een k-clique voorstelt. G heeft dus een k-clique, en dus $\langle G, k \rangle \in$ CLIQUE wat we moeten aantonen.

Tenslotte, is V polynomiaal? Daarvoor moet V lopen in tijd polynomiaal in de lengte van $\langle G, k \rangle$ alleen, zie Definition 7.15. Noteer deze lengte als ℓ. Een eerste opmerking is dat k ten hoogste gelijk is aan n, het aantal knopen van G. We bekijken nu de tests die V moet uitvoeren:

1. Gewoon door c lopen, c moet een lijstje van knoopnummers zijn gescheiden door komma's. Elke nummer moet een nummer zijn tussen 1 en n. Het aantal nummers mag niet meer zijn dan k. Dit kan allemaal efficiënt gecheckt worden.
2. Voor elk koppeltje opzoeken of de overeenkomstige edge in G aanwezig is. Er zijn $k \times k \leq n^2$ koppeltjes. Dit is dus opnieuw polynomiaal in n.

Besluit: we zijn polynomiaal in n. Omdat $n \leq \ell$ (ℓ is immers de totale lengte van de opsomming van alle n knopen, plus nog eens alle edges, plus nog eens de beschrijving van k) zijn we dus zeker polynomiaal in ℓ.

1.b Zij V de verifier uit de proof van Theorem 7.21. We tonen aan:

$$\langle S, t\rangle \in \text{SUBSET-SUM} \iff \exists c : V \text{ aanvaardt } \langle\langle S, t\rangle, c\rangle$$

We gaan eerst $\Rightarrow$ na. Gegeven is $\langle S, t\rangle \in$ SUBSET-SUM. Dit wil zeggen dat er een $Y \subseteq S$ is zodat $\sum_{y\in Y} y = t$. We moeten een c vinden zodat $\langle\langle S, t\rangle, c\rangle$ wordt aanvaard door V. We kunnen hiervoor gewoon een opsomming nemen van de elementen uit Y. Inderdaad, testen 1 en 2 van V lukken dan duidelijk.

We gaan nu $\Leftarrow$ na. Gegeven is dat er een c bestaat zodat $\langle\langle S, t\rangle, c\rangle$ wordt aanvaard door V. Wegens test 1 moet dus c een verzameling Y van getallen zijn zodat $\sum_{y\in Y} y = t$. Wegens test 2 moet $Y \subseteq S$. We besluiten dat $\langle S, t\rangle \in$ SUBSET-SUM zoals gevraagd.

Tenslotte, loopt V polynomiaal in de lengte van $\langle S, t\rangle$? Hiervoor moeten we testen 1 en 2 wel gelijktijdig uitvoeren! We lezen telkens een getalletje uit c, als dit getalletje te lang is (langer dan t) stoppen we al. We kijken direct of dit getalletje voorkomt in S, indien neen stoppen we, indien ja kruisen we het getal in S uit. Als het getalletje vroeger reeds uitgekruist was stoppen we ook (c moet immers een verzameling zijn en mag dus geen herhalingen bevatten). We houden verder een som bij van de reeds geziene getallen. Van zodra deze som te groot wordt (groter dan t) stoppen we. Enkel als we aan het einde zijn gekomen van c en tegelijkertijd is de som gelijk aan t aanvaarden we.

2. Op input $\langle\langle G, s, t\rangle, c\rangle$:

1. Test of c een lijst knoopnummers is uit G van lengte hoogstens n, het aantal knopen in G.
2. Test of c begint met s en eindigt met t.
3. Test of c geen herhalingen bevat.
4. Test of elke knoop uit G voorkomt in c.
5. Test of tussen opeenvolgende knopen in c een edge is in G.
6. Als alle testen lukken, aanvaard; anders reject.

3.a. Op input $\langle G, a, b, k\rangle$:

test of $k \leq n$ met n aantal knopen van G; indien niet, reject;
markeer a;
Laatstgemarkeerd := $\{a\}$;
Nieuwgemarkeerd := $\varnothing$;
doe k keer het volgende:
  voor elke knoop x in Laatstgemarkeerd doe:
    voor elke knoop y met een pijl $x \to y$ in G doe:
      als y nog niet gemarkeerd, markeer y en stop in Nieuwgemarkeerd;
  Laatstgemarkeerd := Nieuwgemarkeerd; Nieuwgemarkeerd := $\varnothing$
als b gemarkeerd is accept, anders reject.

Dit algoritme doet $k \leq n$ iteraties. Elke iteratie vergt $O(n^2)$ tijd. Totaal $O(n^3)$.

3.b. Op input $\langle\langle G, a, b, k\rangle, c\rangle$:

1. Test of c een lijst is van knoopnummers uit G van lengte hoogstens n, het aantal knopen in G.
2. Test of c begint met a en eindigt met b.
3. Test of tussen opeenvolgende knopen in c een edge is in G.
4. Test of c minstens bestaat uit k knopen.
5. Als alle testen lukken, aanvaard; anders reject.

4. Op input G:

found := false;
for each node x in G do
  for each node y in G met $y \neq x$ do
    for each node z in G met $z \neq x$ en $z \neq y$ do
      if er zijn edges $\{x, y\}$, $\{y, z\}$ en $\{z, x\}$ in G
      then found := true;
if found then accept else reject.

Totaal aantal iteraties is n^3, opzoeken van de drie edges $O(n)$, dus $O(n^4)$.

5. Unie. Op input $\langle w, c\rangle$:

1. Roep V_A op met input $\langle w, c\rangle$. Als deze aanvaardt, aanvaarden wij ook.
2. Roep V_B op met input $\langle w, c\rangle$. Als deze aanvaardt, aanvaarden wij ook.
3. Anders rejecten we.

Deze verifier is duidelijk correct, en hij is ook polynomiaal omdat V_A en V_B beiden polynomiaal zijn.

5. Concatenatie. Op input $\langle w, c\rangle$:

1. Zij n de lengte van w. We schrijven w als $x_1 x_2 \ldots x_n$.
2. Test of c van de vorm is i, c_A, c_B met i een getal tussen 0 en n, en c_A en c_B willekeurige strings.
3. Roep V_A op met input $\langle x_1 \ldots x_i, c_A\rangle$;
4. Roep V_B op met input $\langle x_{i+1} \ldots x_n, c_B\rangle$;
5. Als beiden aanvaarden, aanvaarden we ook; anders rejecten we.

Deze verifier is duidelijk correct: het certificaat bestaat uit een plaats waar w kan gesplitst worden, tesamen met certificaten voor lidmaatschap van het linkerstuk in A en het rechterstuk in B.

Hij is ook polynomiaal omdat V_A en V_B polynomiaal zijn.

Addertje onder het gras: Hij moet polynomiaal lopen in n. De strings c_A en c_B kunnen echter willekeurig lang zijn. Strikt genomen is onze verifier dus *niet* polynomiaal. We kunnen hem echter wel polynomiaal maken als volgt. We weten dat V_A polynomiaal is, b.v., met een complexiteit $O(n^4)$. Ook V_B is polynomiaal, b.v., complexiteit $O(n^7)$. Het heeft dus geen zin aan V_A een certificaat aan te bieden dat langer is dan n^4. Analoog heeft het geen zin aan V_B een certificaat aan te bieden dat langer is dan n^7. We voegen daarom een test toe helemaal in het begin:

- Als c langer is dan $\log n + n^4 + n^7 + 2$ (de $\log n$ is voor de i, de $+2$ is voor de twee kommas), dan reject.

6. Op input $\langle w, c\rangle$:

1. Zij n de lengte van w.
2. Test of c van de volgende vorm is:

   $$c_0, i_1, c_1, i_2, c_2, \ldots, i_m, c_m$$

   waarbij:

   - m een getal is tussen 0 en n;
   - elke i_j een getal is zodat $0 \leq i_1 < i_2 < \cdots < i_m \leq n$;
   - elke c_j een willekeurige string is.

   Dit certificaat bestaat dus uit m posities waar we de string w doorkappen, zodat we $m+1$ stukken krijgen, tesamen met $m+1$ certificaten (c_0 tot c_m) voor lidmaatschap van deze stukken in A.
3. Als $m = 0$ roep gewoon V_A op met input $\langle w, c\rangle$.
4. Stel nu $m > 0$.

   - Roep V_A op met input $\langle w_0, c_0\rangle$. Hier is w_0 de substring die gaat van het begin van w tot vlak voor positie i_1.
   - Roep V_A op met input $\langle w_m, c_m\rangle$. Hier is w_m de substring van w die gaat vanaf positie i_m tot het einde.
   - Verder, voor $j = 1, \ldots, m-1$, roep V_A op met input $\langle w_j, c_j\rangle$. Hier is w_j de substring van w die gaat vanaf positie i_j tot vlak voor positie i_{j+1}.

   Als al deze aanroepen van V_A aanvaarden, aanvaarden we ook; anders rejecten we.