# АЛГОРИТАМ И АЛГОРИТАМСКА РЕШЛИВОСТ

**Дино Сејдиновиќ,**
Отсек за математика, ПМФ, Сараево

## Поимот алгоритам

Првата дефиниција за алгоритам со која се среќаваме е: *Алгоритам е секоја јасна и недвосмислена низа чекори којашто води до решавање на проблемот или до одговор дека проблемот нема решение.* Но, оваа дефиниција е прилично непрецизна. Потребно е, пред сé, дефиницијата да се прошири со: *Алгоритам е секоја јасна и недвосмислена* ***конечна*** *низа чекори којашто води до решавање на проблемот или до одговор дека проблемот нема решение.* Кога би имале постапка којашто "води" до решавање на проблемот по бесконечно многу чекори, тогаш прашање е дали можеме воопшто да кажеме дека таа постапка навистина води до решение. Самиот збор ***алгоритам*** потекнува од неправилен изговор на последниот збор од името на персискиот математичар од деветтиот век чие полно име е Abu Ja'far Mohamed ibn Musa al-Khowarizm**.** Тој во своето капитално дело од 825 година *Kitab al-jabr wa'l-muqabala* (да забележиме дека од името на ова дело потекнува зборот *алгебра*) прв формулирал јасни и прецизни постапки – алгоритми – за извршување на четири основни операции за сметање со помош на молив и хартија.

Со наведената дефиниција за алгоритам се јавува следниот проблем: за одредени постапки не е познато дали претставуваат алгоритми или не, т.е. дали по конечен број чекори водат до решение на проблемот. Класичен пример е функцијата на Collatz или Ulam. Неа, независно еден од друг, ја формулирале **Lothar Collatz** и **Stanislaw Ulam**. Таа гласи:

$$C(n)=\begin{cases}1, & n=1\\ 1+C(\frac{n}{2}), & \text{ако } n \text{ е парен број}\\ 1+C(3n+1), & \text{ако } n \text{ е непарен број, } n\neq 1\end{cases}$$

Пресметувањето на вредностите на оваа функција може да се преведе во една јасна и недвосмислена низа чекори, па според наведената дефиниција за алгоритам, таа е кандидат за алгоритам:

1. Внеси некој природен број;
2. Ако бројот е парен, тогаш подели го со 2, а во спротивно помножи го со 3 и на резултатот додај 1;
3. Ако новодобиениот број е различен од 1, оди на чекор 2;
4. Крај.

Ако за некоја фиксирана вредност на бројот $n$ сакаме да ја пресметаме вредноста на **Collatz**-овата функција, тогаш добиваме рекурзивна низа од меѓу резултати кои ќе терминираат (т.е. ќе запрат по одреден број повикувања) доколку како член на низата се појави единица, за чија вредност функцијата е дефинирана. Дали ова навистина се случува за секој природен број $n$, т.е. дали добиената низа терминира за секој природен број $n$? Тоа би значело дека **Collatz**-овата функција е добро дефинирана, односно дека претставува алгоритам.

На пример, нека $n = 7$. Тогаш добиваме рекурзивна низа од повикувања на **Collatz**-овата функција со следните вредности на аргументите: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. Значи, низата добиена за $n = 7$ терминира, што значи дека **Collatz**-овата функција е дефинирана за $n = 7$ е дефинирана и нејзината вредност за овој број изнесува 18. Да забележиме дека важи следното: низата рекурзии терминира ако и само ако се изврши рекурзивно повикување, при што вредноста на аргументот е степен на 2.

Проблемот на **Collatz**-овата функција е отворен и за неговото решение е распишана голема награда, макар што е проверено дека низата терминира за сите природни броеви до 40-цифрени вредности (во некои случаи и со астрономски број на рекурзивни повикувања) и контрапример не е најден. Од наведеното го извлекуваме следното прашање: **Collatz**-овата функција може лесно да се преведе во механичка постапка, но дали таа постапка можеме да ја сметаме за алгоритам? Или поинаку: како да дознаеме дека некоја јасна и недвосмислена низа чекори навистина го решава проблемот, т.е. претставува алгоритам?

Вториот проблем со наведената дефиниција на алгоритам претставува оној момент при решавањето на проблемот којшто обично го нарекуваме инспирација или интуиција (анг. *isight*). Јасно е дека синтагмата "јасна и недвосмислена низа чекори" не е прецизно

дефинирана. За многу проблеми решавањето бара извесна досетливост и инспирација и постапките со чија помош се доаѓа до решенијата никако не се очигледни или механизирани. Дали за таквите решавања постои "јасна и недвосмислена низа чекори" и дали таквите постапки на решавање можеме да ги сметаме за алгоритми? Ова нé доведува до следната модификација на дефиницијата за алгоритам: *Алгоритам е низа чекори кои се изведуваат механички, т.е. автоматски без посегање по каква било инспирација, интуиција или интелигенција и кои за конечно време доведуваат до решение на проблемот или до одговор дека проблемот нема решение.* Значи, зборуваме за низа чекори коишто може да ги изведува машина.

Тоа нé доведува до клучниот проблем во дефиницијата на алгоритам. Што всушност значи "низа чекори којашто се изведува механички"? За каква машина овде станува збор? Дискусиите на оваа тема се бескрајни и сé уште не е дојдено до општо прифатлив одговор. Формирани се разни апстрактни модели таканаречени универзални машини: **Alan Turing** ја вовел тјуринговата машина, **Emil Leon Post** – постовата машина, а **Joachim Lambek** – бесконечниот абакус. Сето ова се обиди да се моделираат машини кои се способни да ги изведуваат сите операции што човекот може да ги изведува "механички" – без интуиција или интелигенција. Најпозната меѓу нив е секако тјуринговата машина. Таа е составена од подвижни магнетни ленти со неограничена должина (и со самото тоа е практично неизводлива, па има само теоретско значење), на која може да се запшат симболи од некоја азбука – било да се тоа само симболите 0 или 1, цифрите на декадниот броен систем или буквите на абецедата – како и главата којашто може да го прочита симболот на лентата и евентуално да го замени. Лентата се поместува налево или надесно и се зема дека машината може да се наоѓа во конечно многу различни состојби од кои зависи акцијата на машината откако на одредено место ќе се прочита одреден симбол. Иако на прв поглед делува како сосема чудна алатка со огараничени можности, покажано е дека секој проблем којшто може да се реши со механички алатки, може да се реши и со тјурингова машина. На овој начин доаѓаме до формалната дефиниција на алгоритам: *Алгоритам е низа чекори коишто можат да се извршат на тјурингова машина и кој за конечно време води кон решение на проблемот или до одговор дека проблемот нема решение.* Разгледувани се разни модификации на тјуринговата машина,

но покажано е дека секоја модификација е еквивалентна со почетниот модел, односно дека тјуринговата машина може да ја имитира секоја своја модификација. Од друга страна, сите три наведени модели: тјуринговата и постовата машина, како и бесконечниот абакус, иако немаат скоро никакви сличности, можат меѓусебно да се симулираат. Тоа значи дека наведената дефиниција важи и доколку тјуринговата машина ја замениме со постовата машина или, пак, со бесконечниот абакус.

Во теориската компјутерска наука, славната и оспорувана Church –Turing-овата теза тврди дека која било постапка која е изведлива, изведлива е и на тјурингова машина. Со други зборови, оние проблеми за кои ни теориски не постои алгоритам не може да ги реши ни човек, што значи дека човекот секогаш постапува според алгоритми, макар што тие можат да бидат и енормно сложени, па не можат ни лесно да се искажат!

Многу значајно прашање коешто навлегува во сферата на вештачката интелигенција е: *дали е можно да се механизираат постапките на човечкиот мозок?* Јасно, ако важи Church–Turing-овата теза, одговорот е позитивен. Постојат две струи на научници кои имаат спротивставени мислења по ова прашање:

– Силна вештачка интелигенција (анг. Strong Artificial Intelligence – *Strong AI*):

Постои механизиран алгоритам кој одговара на постапките на човечкиот мозок, но прашање е дали тој некогаш ќе може да се синтетизира. Ова мислење денес е доминантно.

– Слаба вештачка интелигенција (анг. *Week AI*): Човечкото мислење не е можно да се алгоритмизира, бидејќи се наоѓа надвор од сите модели. Имено, мозокот користи закони на физиката коишто сé уште не познати – сите познати (и применети) закони можат да се алгоритмизираат. Ова мислење го застапува **Roger Penrose** во книгата *Emperor's New Mind* [1]. Тој сугерира и дека мозокот работи според квантната теорија на гравитацијата, но оваа теорија е спекулативна.

## Алгоритамска решливост

Откако се запознавме со поимот алгоритам, природно е да се постави прашањето дали за секој проблем постои алгоритам што води

кон негово решавање (или до одговор дека не постои решение). За жал, одговорот е одречен. Заради тоа, велиме дека проблемот е алгоритамски решлив ако е решлив на еден од споменатите модели (на пример на тјурингова машина). Проблемот е алгоритамски нерешлив ако не е можно да се реши на тјурингова машина. Треба да напоменеме дека алгоритамската нерешливост значи дека е **докажано** дека алгоритамско решение не постои, а не дека тоа просто не е познато или дека не е најдено.

Постојат многу проблеми за кои е покажано дека се алгоритамски нерешливи. На пример, покажано е дека е невозможно да се напрви програма којашто ќе испитува дали дадена програма ќе заврши за конечно време, т.е. дали програмата терминира. Овој проблем се вика уште и *Halting* проблем.

Да го разгледаме значењето на овој проблем: да забележиме дека секоја теорема во аритметиката (па и во многу други, ако не и во сите области на математиката) може да се напише во предикатска форма[1]: $(\forall x \in X)\, P(x)$ или $(\exists x \in X)\, P(x)$ [2], при што $X$ е некое множество природни броеви или подредена $k$-торка природни броеви: $X \subseteq \mathbb{N}^k$ и $k \in \mathbb{N}$. На пример, големата теорема на Ферма[3] би можела да се искаже во следната форма:

$$(\forall (x,y,z,n) \in \mathbb{N}^4)\ \ (n \geq 3 \Rightarrow x^n + y^n \neq z^n)\,.$$

Со мало размислување може да се дојде до револуционерен заклучок: решливоста на Halting проблемот би овозможила автоматско

---

[1] Запис со помош на *предикати* (уште се викаат и *исказни функции*). Една декларативна реченица (искажана со зборови или симболи) во којашто фигурира една или повеќе променливи се вика *исказна функција* ако и само ако при секоја замена на променливите со конкретни објекти се добива конкретен исказ. На пример, од исказната функција $1 \in X$ се добива вистинит исказ ако $X$ се замени со конкретно множество кое го содржи 1 како елемент. Во горниот текст исказната функција е означена со $P(x)$. (Заб. прев.)

[2] Симболот $\forall$ се чита "за секој" и се вика *знак за универзален квантификатор*, а симболот $\exists$ се чита "постои" или "за некој" и се вика *знак за егзистенцијален квантификатор*. Повеќе во врска со искази и исказни функции може да се прочита кај Ѓ. Чупона, *Алгебарски структури и релани броеви*, СМН, 1976, стр. 3 – 7 . (Заб. прев.)

[3] Големата теорема на Ферма гласи: Равенката $x^n + y^n = z^n$ нема решение за $x, y, z \in \mathbb{N}$, ако $n \geq 3$. (Заб. прев.)

докажување (или побивање) на сите теореми од аритметиката! На пример, за еднодимензионалниот случај, односно случајот кога $X = \mathbb{N}$, би било доволно да се испита дали следната програма терминира или не (користена е синтакса од програмскиот јазик $C++$, што не ја намалува општоста на разгледувањето):

```
int x = 1
while (P(x)) x++ ;
```

Доколку наведента програма не терминира, точна е теоремата $(\forall x \in X)\, P(x)$, а во спротивно, не е. Примерот лесно може да се обопшти за повеќе димензионални случаи, како и за случаи во кои наместо квантификаторот $\forall$ се јавува квантификаторот $\exists$ (доволно е наместо предикатот $P$ да се разгледува неговата негација $\neg P$). Значи, кога Halting проблемот би бил решлив, сите математички теореми би можеле да бидат докажани или побиени! Но, не е сé така убаво, зашто се покажало дека Halting проблемот е нершлив. Тоа се покажува едноставно како варијација на познатиот Раселов парадокс[4]: *Ако берберот во едно село ги бричи сите луѓе што не се бричат сами, тогаш кој го бричи берберот?* Двете можни претпоставки – таа дека тој се бричи сам, или дека го бричи некој друг – доведуваат до апсурд. Каква врска има Halting проблемот со селскиот бербер? Да претпоставиме дека имаме функција `zapri_se` којашто испитува дали некоја програма запира за конечно време. Значи функцијата `zapri_se` би можела да изгледа како (повторно ја користиме синтаксата на програмскиот јазик $C++$, што не ја намалува општоста бидејќи истата идеја може да се искаже без да се потпираме на ниту еден програмски јазик):

```
Bool zapri_se (string ime){
...
}
```

Од друга страна, имаме програма дадена со:

```
int main() {
if(zapri_se(<<abc.cpp>>)) while(true);
   return 0;
   }
```

[4] Читателот може да прочита повеќе во врска со овие парадокси од следната литература: А. Самарџиски, Н. Целакоски, *Збирка задачи по алгебра, Множества*, трето издание на УКИМ, Скопје, стр. 106 – 112 и Ј. Маркоска, *Парадокси*, Сигма бр.73, стр. 46-48. (Заб. прев.)

Јасно е дека оваа програма терминира ако и само ако програмата снимена под името <<abc.cpp>> не терминира. Парадоксот се јавува токму во моментот кога оваа програма ја снимаме под името <<abc.cpp>>, т.е. ако му ја дадеме улогата на селскот бербер. Ова нé води до заклуок дека функција како што е функцијата zapri_se не може да постои. Секако, од погоре наведеното не следува дека и проблемот на автоматско докажување на теоремите е алгоритамски нерешлив. Дилемите околу ова завршиле кога се покажа дека Halting проблемот може да се напише во вид на предикати, односно со теоремите на аритметиката, па кога сите теореми би биле решливи, тогаш би бил решлив и Halting проблемот, што веќе констатиравме дека е невозможно.

Halting проблемот е пандан на *Геделовата теорема за противречност*, којашто покажува дека секоја и малку покомплицирана математичка теорија е непотполна или противречна. Halting проблемот и Геделовата теорема за противречност се сведливи еден на друг.

Со оглед на тоа дека Halting проблемот е доста општ, не изненадува што е нерешлив. Многу повеќе изненадува тоа што многу побанални проблеми се алгоритамски нерешливи. Таков е примерот со Thue проблемот на зборови или уште познат како *проблем на зборови за полугрупи*. Да претпоставиме дека е зададено конечно множество дозволени трансформации коишто овозможуваат замена на една група букви со друга. Овој проблем се состои во следното: за дадени два збора да се утврди дали постои низа трансформации која едниот збор ќе го претвара во другиот. На пример, нека е зададено множеството правила:

$$\text{EAT} \leftrightarrow \text{AT},\ \ \text{ATE} \leftrightarrow \text{A},\ \ \text{LATER} \leftrightarrow \text{LOW},\ \ \text{PAN} \leftrightarrow \text{PILLOW},\ \ \text{CARP} \leftrightarrow \text{ME}$$

Да се запрашаме дали е можно да се трансформира зборот KATERPILLAR во зборот MAN. Ако одговорот е потврден, решението секогаш може да се најде со груба сила (анг. *brute force*). Конкретно, за овој пример одговорот е потврден, а бараната низа трансформации е следната:

$$\underline{\text{CATE}}\text{RPILLAR} \rightarrow \text{CARPILL}\underline{\text{A}}\text{R} \rightarrow \text{CARPIL}\underline{\text{LATER}} \rightarrow \text{CAR}\underline{\text{PILLOW}} \rightarrow$$
$$\underline{\text{CARP}}\text{AN} \rightarrow \text{ME}\underline{\text{A}}\text{N} \rightarrow \text{M}\underline{\text{EATE}}\text{N} \rightarrow \text{M}\underline{\text{ATE}}\text{N} \rightarrow \text{MAN}$$

Но, доколку одговорот е одречен, тогаш тој одговор не можеме да го добиеме со груба сила, зашто бројот на трансформации што можеме да ги примениме не е ограничен. За давање одречен одговор потребна е интелигенција. На пример, зборот CARPET не може да се трансформира во зборот MEAN со наведеното множество трансформации. Имено, секоја од трансформациите не го менува збирот на појавувања на буквите A, W и M во новодобиениот збор, а за наведените зборови овие броеви не се еднакви – CARPET ја има само буквата A, додека MEAN има две букви од од оваа група: M и A. Но, ова е решение на само еден специјален проблем и за некое друго множество правила најверојатно нема да постои никакво аналогно решение. Рековме дека Thue проблемот е алгоритамски генерално нерешлив, што значи дека не постои универзален алгоритам што би работел за *произволно множество правила*. Едно такво множество правила, за кое е алгоритамски невозможно да се утврди евентуална неможност за трансформација од еден збор во друг, дале G.S. Tseitin и Dana Scot 1955 година. Тоа гласи:

$$\text{AH} \leftrightarrow \text{HA}, \quad \text{OH} \leftrightarrow \text{HO}, \quad \text{AT} \leftrightarrow \text{TA}, \quad \text{OT} \leftrightarrow \text{TO}, \quad \text{TAI} \leftrightarrow \text{IT},$$
$$\text{HOI} \leftrightarrow \text{IH}, \quad \text{THAT} \leftrightarrow \text{ITHT}$$

Листата на досега познати алгоритамски нерешливи проблеми е депримирачки голема. Овде ќе ги наведеме некои од најпознатите и најважните алгоритамски нерешливи проблеми. Примерите покажуваат дека алгоритамски нерешливите проблеми можат да бидат од најразлична природа:

- Halting проблем;
- Проблемот на зборови на Thue;
- Десеттиот Хилбертов проблем (*проблем на решливост на Диофантови равенки*): Нека $P(x_1, x_2, ..., x_n)$ е даден полином со целобројни коефициенти. Испитај дали овој полином има целобројни нули. **Yuri Matiyasevich** решавајќи еден поопшт проблем докажал дека Десеттиот Хилбертов проблем генерално е алгоритамски нерешлив. За случаите кога $n = 1$ и $n = 2$ проблемот е решлив, додека за секој $n \geq 4$

нерешливоста е докажана. Случајот кога $n = 3$ и натаму останува отворен;

● Ако $f$ е некоја произволна реална функција од една променлива, генерирана од композицијата на операциите собирање, одземање, множење, делење и функцијата $\sin x$, тогаш алгоритамски нерешливи проблеми се: да се одреди дали оваа функција има реални нули и дали нејзиниот интеграл $\int_{-\infty}^{+\infty} f(x)dx$ конвергира;

● Да се утврди дали два програми се еквивалентни, т.е. дали за исти влезни податоци секогаш даваат исти излезни податоци;

● Да се утврди дали некоја програма е или содржи вирус во некој свој дел, при што под вирус ќе ја подразбираме програмата за способност да прави свои копии или да подига некој малициозен код;

● Tiling проблем – да се утврди дали е можно рамнината да се поплочи со фигури од однапред зададено можество фигури (слично на популарната игра *Tetris*). Penrose своите тврдења ги заснова токму на овој проблем. Имено, човечкиот мозок сепак интуитивно наоѓа методи за решавање на овој проблем (во конкретни случаи, не и општо);

● Да се утврди дали една конечно генерирана група е Абелова;

● *Постов проблем на кореспонденција* – Ако имаме низа картички на кои од двете страни се испишани некои букви, при што различни видови картички се конечно многу, а од секој вид можеме да имаме произволен број примероци, тогаш да се одреди дали е можно картичките да наредат во таков редослед за од двете страни да биде напишан истиот збор;

● Да се најде веројатноста $\Omega$, позната како Chaitin-ова константа или бројот на мудрости (анг. *Number of Wisdom*) за произволно генериран машински код да се заврши во конечно време на конкретен процесор или Тјуринговамашина. Да запреме за момент и да констатираме дека таквиот број е добро дефиниран и дека навистина постои! Она што е посебно интересно е дека кога би се знаел овој број, би можел да биде решен Halting проблемот и дека би било доволно да се знаат првите 10000 бита на бројот $\Omega$ за да се реши скоро секој математички проблем! Логичарот **Gregory Chaitin** во една прилика бил запрашан кое прашање би го поставил доколку би можел да направи еден телефонски

разговор со Господ. Chaitin, како што претпоставувате, би побарал Господ да му ги издиктира првите 10000 цифри на бројот $\Omega$ (во декаден броен систем тоа се нешто повеќе од 3000 цифри).

Со текот на времето се јавила претпоставката дека можеби постои сметачка постапка којашто го проширува поимот механизирана постапка, односно постапка којашто ја надвишува моќта на тјуринговата машина. Предложени се разни такви модели, наречени *хиперсметачи*, ама за ниту еден од предложените модели не се гледа можност за вистинска физичка реализација. Лесно е да се покаже дека дури ни евентуален изум на некој хиперсметач не овозможува алгоритамско решавање на сите проблеми. Имено, може да се покаже дека *множеството од сите алгоритми е пребројливо*, независно од тоа за која дефиниција на алгоритам се одлучиме (и за кој модел на сметачка машина). Ќе дадеме неформален вербален доказ на ова тврдење. Имено, ако алгоритам е конечно множество чекори, тогаш тој може да се изрази преку конечна низа реченици (инструкции) во некоја конечна азбука, што значи преку некоја конечна низа симболи. Нека, на пример, азбуката има $N$ симболи. Тогаш симболите од азбуката можат да се протолкуваат како цифри во броен систем со основа $N$. Тогаш, самиот алгоритам го разбираме како природен број запишан во броен систем со основа $N$. Различни алгоритми се запишани како различни низи од симболи, па им одговараат различни природни броеви. На овој начин конструиравме едно инјективно пресликување од множеството на сите алгоритми во множеството на сите природни броеви, а тоа значи дека множеството од сите алгоритми е најмногу пребројливо. Со оглед на тоа дека во еден алгоритам можеме да додаваме инструкции, т.е. алгоритмот можеме да го прошируваме, следува дека множеството алгоритми е бесконечно.

Од друга страна, *множеството од сите проблими не е пребројливо*, дури ни ако се ограничиме на проблеми коишто се однесуваат само на природните броеви! Ова произлегува од фактот дека постојат непребројливо многу пресликувања од $\mathbb{N}$ во $\mathbb{N}$, што се покажува едноставно со добро познатиот Канторов дијагонален принцип. Навистина, ако сме ги подредиле во низа сите функции од $\mathbb{N}$ во $\mathbb{N}$: $f_1(x), f_2(x), ..., f_k(x), ...$ тогаш функцијата $g : \mathbb{N} \to \mathbb{N}$ за која $g(n) \neq f_n(n)$ за секој $n \in \mathbb{N}$ не припаѓа на оваа низа. Ова нé води до

заклучокот дека постојат функции од $\mathbb{N}$ во $\mathbb{N}$ кои не може да се пресметаат, т.е. не постојат алгоритми за нивно пресметување, а со тоа ни сите проблеми коишто се однесуваат на природните броеви не можат да бидат алгоритамски решливи! Постојат многу зачајни конкретни примери на функции од $\mathbb{N}$ во $\mathbb{N}$ што не може да се пресметаат, а ние овде ќе се ограничиме на следните два примера.

*Пример* 1: Функцијата $BB(n)$ позната под името *busy beaver* функција, се дефинира за конкретен процесор или, уште почесто, за тјурингова машина. Нека е дадена програма која зазема $n$ мемориски локации. Со $BB(n)$ ќе го означиме најголемиот број мемориски локации коишто дадената програма може да ги "згреши", т.е. да ја смени нивната содржина. Оваа функција е добро дефинирана, но се покажало дека не може да се пресмета. Од претпоставката дека $BB$ е функција што може да се пресмета, се добива парадоксална ситација слична на онаа како кај Halting прогблемот.

*Пример* 2: Функцијата на Колмогоров $K(n)$ дефинирана како минимален број битови на кои програма од $n$ инструкции може да се компресира. Дека оваа функција не може да се пресмета, се добива со варијајација на Barry-евиот парадокс, кој е еден од најсилните парадокси во математиката и кој предизвикал барем исто толку полемики колку што предизвикал и Раселовиот или познатиот Зенонов парадокс. Ќе наведеме една од можните формулации на Barry-евиот парадокс. Можеме да тврдиме дека постојат доволно сложени природни броеви такви што за нивното опишување (еднозначно укажување на тој и тој број) требаат повеќе од 30 зборови од некој однапред фиксиран речник. Речникот е конечен и множеството од сите можни комбинации од 30 или помалку зборови од тој речник е конечен. Значи, такви броеви несомнено постојат. Тие се едно подмножество од множеството на природните броеви $\mathbb{N}$. Бидејќи множеството $\mathbb{N}$ е добро подредено[5], следува дека ова множество има најмал елемент. Тоа е: *најмалиот*

[5] Нека $M$ е подредено множество и $A$ е непразно подмножество од $M$. За елементот $a \in A$ велиме дека е *најмал* во $A$ ако и само ако за секој $x \in A$, $a \leq x$. За едно множество $M$ велиме дека е *добро подредено* ако и само ако секое непразно подмножество од $M$ има најмал елемент. За релации, подредувања и подредени множества може да се прочита кај Ѓ. Чупона, *Алгебарски структури и релани броеви*, СМН, 1976, стр. 19 – 31 (Заб. прев.)

*природен број којшшо не може да се оūише со ūомалку од 30 збора*. Но, токму ова еднозначно го опишува со помалку од 30 збора! Аналогијата меѓу компресирањето реченици од програмата и опишувањето броеви е евидентна.

Литература:

[1] **Roger Penrose**: *The Emperor's New Mind – Concerning Computers, Minds, and The Laws of Physics*, Vintage Books, London, 1990

[2] **Željko Jurič**, *Informatika za 1,2 i 3 razred gimanzije*, Sarajevo publishing, Sarajevo, 2003

[3] **Željko Jurič, Dino Sejdinovič**: *Matematički algoritmi*, PMF Sarajevo (rdna skripta)

[4] **S.F. Barker**: *Philosophy of Mathematics*, Prentice Hall, Englewood Cliffs, 1964

Превод од српско-хрватски: Ц.J. В.