Kapitola 1: Prerekvizity

Riešenie nasledujúcich úloh je dobrovoľné. Pokrývajú témy, u ktorých by ma potešilo, aby ste im rozumeli skôr, než začneme s našim predmetom.

Úlohy zjavné

Toto sú úlohy, na ktoré by ste mali pozrieť a vedieť, aká je odpoveď, resp. ako sa k nej dopracovať.

$\alpha 1$. Majú množiny $\mathbb{N}$ a $\mathbb{N} \times \mathbb{N}$ rovnakú mohutnosť?

$\alpha 2$. Aká je mohutnosť množiny $\{a, b\}^*$ (teda množiny všetkých konečných reťazcov z písmen a a b)?

$\alpha 3$. Ku každému nedeterministickému zásobníkovému automatu sa dá mechanicky zostrojiť ekvivalentný nedeterministický Turingov stroj.

$\alpha 4$. Predchádzajúce tvrdenie platí aj ak druhý výskyt slova „nedeterministický“ zmeníme na „deterministický“.

Úlohy trochu náročnejšie

U týchto úloh môže byť potrebná trocha zamyslenia, prípadne môže byť treba vytiahnuť a použiť pero a papier. Ale nemalo by to zase príliš bolieť.

$\beta 1$. Nech χ a φ sú čiastočné funkcie jednej premennej na $\mathbb{N}$, t. j., ich definičný obor je podmnožinou $\mathbb{N}$. Pritom nech $\chi \circ \varphi$ je totálna funkcia, t. j., funkcia definovaná na celom $\mathbb{N}$.

Musí byť χ totálna? Musí byť φ totálna?

(Ak je funkcia f nedefinovaná pre vstup x, značíme to $f(x) = \perp$. Pre konzistenciu sa dohodneme, že pre ľubovoľnú čiastočnú funkciu f platí $f(\perp) = \perp$. Potom $f \circ g$ je funkcia definovaná predpisom $\forall x : (f \circ g)(x) = f(g(x))$.)

$\beta 2$. Dokážte alebo vyvráťte: pre každý ε-free nedeterministický konečný automat $A = (K, \Sigma, \delta, q_0, F)$ je jazyk

$$\{w \mid \text{ak } (q_0, w) \vdash_A^* (q, \varepsilon) \text{ tak } q \in F\}$$

regulárny. (Slovne: ide o jazyk slov, na ktorých sú všetky výpočty, ktoré dočítajú vstup, akceptačné.)

$\beta 3$. Je rozhodnuteľné, resp. aspoň čiastočne rozhodnuteľné, či pre daný nedeterministický konečný automat A platí $L(A) = L(A)^R$?

$\beta 4$. Máme *nedeterministický* Turingov stroj s jednosmerne nekonečnou páskou, ktorý nevie hýbať hlavou klasicky. Namiesto toho vie robiť dva druhy pohybov: krok o políčko doprava a skok doľava na začiatok pásky. Popíšte, ako na takomto NTS simulovať klasický.

$\beta 5$. Nech $\Sigma = \{0, 1\}$. Definujme reláciu $<$ na Σ^* nasledovne: $u < v$ platí, ak buď $|u| < |v|$ alebo ($|u| = |v|$ a existuje w také, že u má prefix $w0$ a v má prefix $w1$).

Dokážte alebo vyvráťte: jazyk L je rekurzívny práve vtedy, ak existuje Turingov stroj, ktorý enumeruje slová L usporiadané podľa $<$.

Úlohy ešte náročnejšie

U týchto úloh už nie je potrebné, aby ste ich vedeli vyriešiť – stačí, keď rozumiete zadaniu. Ich riešenie môže byť aj bolestivé, ale o to lepší je pocit, keď si s niektorou z nich poradíte `:-)`.

γ1. Je nasledujúci jazyk regulárny?

$$L = \{x_1y_1x_2y_2\ldots x_ny_n \mid n > 0 \ \wedge \ 3 \cdot (x_1x_2\ldots x_n)_2 = (y_1y_2\ldots y_n)_2\}$$

(Slovne: keď sa na $x_1x_2\ldots x_n$ a $y_1y_2\ldots y_n$ dívame ako na binárne zápisy čísel X a Y, tak $3X = Y$. Čísla X aj Y môžu začínať niekoľkými nulami.)

γ2. Čo sa stane, keď v úlohe β4 zmeníme slovo „nedeterministický“ na „deterministický“? Bude takýto DTS vôbec ešte mať rovnakú výpočtovú silu ako klasický?

γ3. Je rozhodnuteľné, resp. aspoň čiastočne rozhodnuteľné, či sú všetky výpočty daného nedeterministického Turingovho stroja A na slove w konečné?

γ4. Toto je zovšeobecnenie úlohy β5. Dokážte alebo vyvráťte: Pre každý total order $<$ na Σ^* platí, že jazyk L je rekurzívny práve vtedy, ak existuje Turingov stroj, ktorý enumeruje slová L usporiadané podľa $<$.

γ5. Uvažujme deterministický Turingov stroj s jednosmerne nekonečnou páskou (na ľavom konci so zarážkou \$), ktorý môže písať blanky, ale nemôže písať nič iné, lebo má prázdnu pracovnú abecedu. Je pre takýto TS rozhodnuteľný problém zastavenia?

Kapitola 2: Automatové modely

1. Ku danému deterministickému Turingovmu stroju zostrojte ekvivalentný Minského registrový stroj. Rozmyslite si podrobne nasledovné detaily:

   - Turingov stroj na vstupe dostáva slovo, nie číslo. Ako by ste riešili vstup pre Minského stroj? (Pozor na konečnosť počtu použitých registrov!)
   - Ako definovať, či Minského stroj dané slovo akceptuje?
   - Počas simulácie Turingovho stroja si potrebujeme pamätať nielen obsah jeho pásky, ale aj jeho aktuálny stav. Ako sa toto dá spraviť?

2. Uvažujme triviálny deterministický Turingov stroj, ktorý akceptuje slová, v ktorých je počet a deliteľný tromi.

   K nemu zostrojíme ekvivalentný automat s dvoma zásobníkmi.

   K tomu ekvivalentný automat s tromi počítadlami (prvý zásobník kódovaný do čísla, druhý zásobník kódovaný do čísla, pomocné počítadlo).

   No a k tomu zostrojíme ekvivalentný automat s dvomi počítadlami (pôvodné tri počítadlá kódované do jedného, pomocné počítadlo).

   Tento výsledný stroj spustíme na vstupe *aababab*. Odhadnite, aká najväčšia hodnota sa počas výpočtu zjaví v niektorom z počítadiel.

3. Program pre registrový stroj vieme ľahko znázorniť pomocou vývojového diagramu. Vysvetlite ako.

4. Napíšte program pre registrový stroj, ktorý bude počítať funkciu $f(n) = \lfloor \sqrt{n} \rfloor$.

5. Napíšte program pre registrový stroj, ktorý bude počítať funkciu $g(n) = 2^{2^n}$.

6. Napíšte program pre registrový stroj, ktorý bude počítať funkciu $h(n) = \lceil \log_2 n \rceil$ (pričom pre $n = 0$ vracia 0).

7. Uveďte high-level popis toho, ako by ste zostrojili registrový stroj, ktorý pre vstup n vráti na výstupe $(n + 1)$. prvočíslo.

8. Zostrojte deterministický dvojsmerný konečný automat s jedným počítadlom, ktorý bude rozpoznávať jazyk všetkých palindrómov nad abecedou $\{a, b\}$.

Kapitola 3: Gramatikové modely

1. V skriptách v odseku o univerzalite Markovovych algoritmov úmyselne ignorujem blanky; na prednáške som tento detail okrajovo spomenul. (Turingovmu stroju sa môže stať, že pri pohybe hlavou vybehne na prázdne políčko. Tieto prípady treba pri konštrukcii nejak doplniť.)

   Detailne vysvetlite, ako doplniť konštrukciu o tieto prípady. (Hint: využite, že pravidlo s vyšším číslom sa použije len vtedy, ak sa nič s nižším číslom použiť nedá.)

2. Zostrojte Markovov algoritmus, ktorý sa na slove $w \in \{a, b\}^*$ zastaví práve vtedy, keď $w = w^R$.

3. Nájdite explicitnú konštrukciu, ktorá k ľubovoľnému Markovovmu algoritmu vyrobí ekvivalentnú frázovú gramatiku.

4. Uvažujme nasledovný tag systém: $(1, \{a, b, c, d, e, f, g\}, \{a \to \varepsilon, b \to feebac, c \to \varepsilon, d \to afgafgafgaf, e \to babadge, f \to caga, g \to \varepsilon\})$.

   Nájdite jazyk všetkých vstupov, ktoré tento tag systém akceptuje (t.j. vstupov, pre ktoré prepisovanie po konečnom počte krokov skončí).

Kapitola 4: Exotické modely

1. Označme $WT(f(n))$ triedu jazykov, pre ktoré existuje Wang tile set, ktorý tento jazyk rozpoznáva s priestorovou zložitosťou $g(n) = O(f(n))$.

   Nájdite jazyk z $WT(\log_2 n) - WT(1)$.

2. Dokážte alebo vyvráťte: trieda bezkontextových jazykov $\mathscr{L}_{CF}$ je podmnožinou $WT(n)$.

3. Navrhnite sadu Wang tiles ktorá bude rozpoznávať jazyk $\{ww \mid w \in \{a, b\}^*\}$.

4. Jeden z pôvodných Wangovych cieľov bolo navrhnúť algoritmus, ktorý by pre danú sadu dlaždíc zistil, či sa nimi dá vydláždiť celá rovina. Toto sa mu aj (r. 1961) podarilo – ale len za predpokladu, ktorý nevedel dokázať: že každé takéto dláždenie musí byť periodické.

   Pomerne nepríjemné prekvapenie priniesol (r. 1965) Robert Berger, ktorý našiel konečnú sadu dlaždíc (20 426 typov), ktorou sa síce dá nekonečná rovina vydláždiť, ale žiadne dláždenie nie je periodické. V súčasnosti je známa sada s touto vlastnosťou, ktorá má len 13 typov dlaždíc. (Jej detailná analýza: `home.gwu.edu/~robinson/Marseille.pdf`)

   Oproti Bergerovi máte nadhľad – poznáte už výsledky z ďalších > 40 rokov. Navrhnite vlastnú sadu dlaždíc s touto vlastnosťou.

   (Alebo zľahčená verzia: o jednom type dlaždíc môžete navyše povedať, že aspoň jedna dlaždica tohto typu musí byť použitá. Dostanete teda zadarmo „štartovaciu“ dlaždicu.)

5. Napíšte program pre interpreter `SUBLEQ`u, ktorý bude robiť násobenie: do `memory[3]` uloží súčin hodnôt, ktoré sú na začiatku v `memory[1]` a `memory[2]`. (Môžete pre jednoduchosť predpokladať, že násobené hodnoty sú kladné.)

6. Máme inštrukciu `SUBEQ`, ktorá robí to isté ako `SUBLEQ`, až na to, že skok sa vykoná len vtedy, ak výsledkom odčítania bola presne nula.

   Napíšte vo svojom obľúbenom programovacom jazyku (alebo aspoň slovne popíšte) program, ktorý na vstupe dostane postupnosť čísel predstavujúcu program pre náš interpreter `SUBLEQ`u a na výstupe vyrobí ekvivalentný program pre interpreter `SUBEQ`u.

7. h4x0r challenge: Napíšte program pre interpreter `SUBLEQ`u, ktorý bude testovať prvočíselnosť hodnoty, ktorá je na začiatku v `memory[1]`.

Kapitola 5: Primitívna rekurzia

V týchto cvičeniach niektoré sú a niektoré nie sú označené hviezdičkou ($\star$). Tie, ktoré sú označené, sú ťažšie. Tým ostatným treba určite rozumieť.

1. Z definície dokážte primitívnu rekurzívnosť nasledujúcich zaujímavých funkcií:
   - $p(x) = \begin{cases} 0 & \leftarrow x = 0 \\ x - 1 & \leftarrow \text{inak} \end{cases}$
   - $sub(x, y) = \begin{cases} x - y & \leftarrow x \geq y \\ 0 & \leftarrow \text{inak} \end{cases}$
   - $hop(x) = 47x + 74$
   - $sgn(x) = \begin{cases} 0 & \leftarrow x = 0 \\ 1 & \leftarrow \text{inak} \end{cases}$
   - $\overline{sgn}(x) = \begin{cases} 1 & \leftarrow x = 0 \\ 0 & \leftarrow \text{inak} \end{cases}$
   - $diff(x, y) = |x - y|$.
   - $max(x, y)$
   - $median(x, y, z)$
   - $rovnasa(x, y) = \begin{cases} 1 & \leftarrow x = y \\ 0 & \leftarrow \text{inak} \end{cases}$
   - $fact(x) = x!$

2. Dokážte: Každý polynóm $p(x)$, ktorého koeficienty sú prirodzené čísla, je primitívne rekurzívny.

3. Dokážte alebo vyvráťte: K ľubovoľnému polynómu $p(x)$ s celočíselnými (potenciálne aj zápornými!) koeficientami existuje primitívne rekurzívna funkcia f_p taká, že $\forall n \in \mathbb{N} : f_p(n) = \max(0, p(n))$.

4. Dokážte primitívnu rekurzívnosť logickej spojky xor. Teda dokážte, že ak p a q sú primitívne rekurzívne predikáty s rovnakou aritou, tak je primitívne rekurzívny aj predikát r, ktorý vracia 1 práve pre tie vstupy, pre ktoré jeden z p a q vracia 1 a druhý 0.

5. Dokážte, že k ľubovoľnej primitívne rekurzívnej funkcii f existuje primitívne rekurzívna funkcia g taká, že $\lim_{n\to\infty} f(n)/g(n) = 0$.

6. Dokážte: Pre ľubovoľnú primitívne rekurzívnu funkciu f je funkcia $g(x) = \sum_{i<x} f(i)$ primitívne rekurzívna. (Funkciu g voláme prefixovým súčtom funkcie f.)

   ($\star$) Dokážte aj všeobecnejšie tvrdenie: pre ľubovoľné primitívne rekurzívne funkcie lo, hi a f je funkcia

   $$g(\overline{x}) = \sum_{lo(\overline{x}) \leq i \leq hi(\overline{x})} f(i, \overline{x})$$

   primitívne rekurzívna. (Značenie $\overline{x}$ je skráteným zápisom pre $x_1, \ldots, x_k$.)

7. Pomocou všetkých doteraz známych výsledkov dokážte, že sú primitívne rekurzívne nasledovné funkcie:

   - $mod(x,y) = \begin{cases} x \bmod y & \leftarrow y > 0 \\ 0 & \leftarrow \text{inak} \end{cases}$
   - $divides(x,y) = \begin{cases} 1 & \leftarrow y > 0 \ \wedge \ y \text{ delí } x \\ 0 & \leftarrow \text{inak} \end{cases}$
   - $div(x,y) = \begin{cases} \lfloor x/y \rfloor & \leftarrow y > 0 \\ 0 & \leftarrow \text{inak} \end{cases}$

8. ($\star$) Dokážte vetu o *ohraničenej minimalizácii*:
   Pre ľubovoľné primitívne rekurzívne funkcie $f(y,\overline{x})$ a $g(\overline{x})$ je funkcia

   $$h(\overline{x}) = \begin{cases} \min\{i \mid i < g(\overline{x}) \ \wedge \ f(i,\overline{x}) > 0\} & \leftarrow \text{ak také } i \text{ existuje} \\ g(\overline{x}) & \leftarrow \text{inak} \end{cases}$$

   primitívne rekurzívna.
   Slovne, $h(\overline{x})$ si môžeme predstaviť tak, že postupne počíta $f(0,\overline{x})$, $f(1,\overline{x})$, ..., až kým buď prvýkrát nedostane nenulovú hodnotu, alebo nedosiahne vopred určenú hranicu $g(\overline{x})$.

9. Pomocou vety o ohraničenej minimalizácii vieme triviálne ukázať primitívnu rekurzívnosť niektorých funkcií, pre ktoré je to priamo z definície nepríjemné. Dokážte takto primitívnu rekurzívnosť nasledujúcich funkcií:

   - $ceil(x,y) = \begin{cases} \lceil x/y \rceil & \leftarrow y > 0 \\ 0 & \leftarrow \text{inak} \end{cases}$
   - $sqrt(x) = \lceil \sqrt{x} \rceil$
   - ($\star$) $isprime(x) = \begin{cases} 1 & \leftarrow x \text{ je prvočíslo} \\ 0 & \leftarrow \text{inak} \end{cases}$

   (Hint: *ceil* je primitívne rekurzívna, lebo ju môžeme definovať zhruba nasledovne: $ceil(x,y)$ je najmenšie také z, pre ktoré $yz \geq x$, pričom z stačí hľadať v rozsahu od 0 po x. Rozmyslite si, ako by tento argument vyzeral formálne. Pri testovaní prvočíselnosti budete pravdepodobne potrebovať niekoľko pomocných funkcií.)

10. Dokážte, že n-té prvočíslo (číslované od 0) je menšie alebo rovné ako 2^{2^n}.

11. Pomocou predikátu *isprime* a výsledku predchádzajúceho cvičenia dokážte, že funkcia, ktorá pre vstup n vráti $(n+1)$. prvočíslo, je primitívne rekurzívna.

12. Uvažujme konštantnú funkciu $t(x) = 1000$. Táto funkcia je primitívne rekurzívna, teda k nej existuje „recept“: postupnosť funkcií $f_1, f_2, \ldots, f_k = t$ taká, že každá f_i je nulárna nula, successor, niektorá projekcia, alebo vzniká z niektorých predchádzajúcich f_j operáciou kompozície alebo operáciou primitívnej rekurzie.
    Dokážte alebo vyvráťte: v každom recepte pre t je $k \geq 1000$.

13. Dokážte, že $f(x) =$ (cifra rádu 10^{-x} v desatinnom rozvoji čísla $\sqrt{2}$) je primitívne rekurzívna funkcia.
    (Hint: Jedna možnosť je dokázať primitívnu rekurzívnosť $f(x) = \lfloor x\sqrt{2} \rfloor$.)

14. Nájdite vzorec v uzavretom tvare pre funkciu $l(x)$ zo skrípt.

15. Dokážte, že pre kódovanie konečných postupností z prednášky (do mocnín prvočísel) je funkcia $reverz$ primitívne rekurzívna. (Funkcia $reverz$ má nasledovnú vlastnosť: pre ľubovoľnú postupnosť $a_1, \ldots, a_n$ platí, že ak K je kód $a_1, \ldots, a_n$ a $\overline{K}$ je kód $a_n, \ldots, a_1$, tak $reverz(K) = \overline{K}$.)

16. Postupnosti môžeme do čísel kódovať aj ináč. Ukážeme si postup, ktorý je bežne používaný vo funkcionálnych programovacích jazykoch: reprezentovať zoznam ako usporiadanú dvojicu (prvý prvok, zvyšok zoznamu).

    Nech s je successor a c párovacia funkcia z prednášky. Ich kompozíciou dostaneme párovaciu funkciu $\overline{c}$, ktorá nikdy nevráti 0. Teraz môžeme kód postupnosti definovať rekurzívne podľa jej dĺžky: prázdna postupnosť má kód 0, postupnosť $a_1, \ldots, a_n$ má kód $\overline{c}(a_1, k)$, kde k je kód $a_2, \ldots, a_n$.

    Upravte dôkaz vety o primitívne rekurzívnej časovej zložitosti tak, aby použil takéto kódovanie.

Kapitola 6: Všeobecná rekurzia

1. Nech A je Ackermannova funkcia dvoch premenných. Uvažujme funkcie B a C definované nasledujúcim predpisom:

$$B(x,y,z) = \begin{cases} 1 & \leftarrow A(x,y) < z \\ 0 & \leftarrow A(x,y) \geq z \end{cases}$$

$$C(x,y,z) = \begin{cases} 1 & \leftarrow A(x,y) > z \\ 0 & \leftarrow A(x,y) \leq z \end{cases}$$

   Rozhodnite a dokážte, či sú B a C primitívne rekurzívne.

2. Ľubovoľnými prostriedkami dokážte primitívnu rekurzívnosť 2D primitívnej rekurzie.

   T. j., dokážte, že pre ľubovoľné primitívne rekurzívne g_1, g_2 a h je primitívne rekurzívna aj nasledovne definovaná f:

$$\begin{aligned} f(0,y) &= g_1(y) \\ f(x+1,0) &= g_2(x) \\ f(x+1,y+1) &= h(\, f(x,y+1),\, f(x+1,y),\, x,\, y) \end{aligned}$$

3. Videli sme, že vyhodnocovanie Ackermannovej funkcie je konečné preto, že pri každom rekurzívnom volaní sa buď zmenší prvý parameter, alebo prvý ostane nezmenený a druhý sa zmenší.

   Nájdite rekurzívnu funkciu B dvoch premenných, ktorá nebude mať túto vlastnosť, a napriek tomu bude pre každé x, y vyhodnocovanie $B(x,y)$ konečné.

   (Jemne ťažšia verzia: Nájdite rekurzívnu funkciu C, ktorá spĺňa to čo B a navyše platí: nech si povieme ľubovoľne veľké p, q, vždy budú existovať x, y také, že na vyhodnotenie $C(x,y)$ potrebujeme nejakú hodnotu $C(x',y')$, kde $x' > x+p$ a $y' > y+q$.)

4. Pre unárnu funkciu f definujme $H(f) = \{f(x) \mid x \in \mathbb{N}\}$.

   Dokážte alebo vyvráťte: ku každej primitívne rekurzívnej unárnej funkcii f existuje primitívne rekurzívna unárna funkcia g taká, že $H(g) = H(f)$ a navyše g nadobúda každú hodnotu z $H(g)$ nekonečne veľa krát.

5. Dokážte alebo vyvráťte: Nech f je prostá unárna primitívne rekurzívna funkcia. Potom existuje primitívne rekurzívna funkcia g taká, že $\forall n : g(f(n)) = n$.

6. Uvažujme nejaké konkrétne číslovanie $\varphi_0, \varphi_1, \ldots$ všetkých unárnych primitívne rekurzívnych funkcií. Funkcia U nech je univerzálna funkcia pre ne. (Tá z prednášky, t.j. $\forall n, x : U(n,x) = \varphi_n(x)$.)

   Hovoríme, že binárna funkcia V sa podobá na univerzálnu, ak $\forall n, x : |U(n,x) - V(n,x)| \leq 10$.

   Dokážte alebo vyvráťte: Neexistuje primitívne rekurzívna funkcia, ktorá sa podobá na univerzálnu.

7. Definujte univerzálnu funkciu pre všetky čiastočné rekurzívne funkcie. Zaujíma nás, či je táto funkcia čiastočne rekurzívna.

   Dá sa použiť ten istý kontrapríklad ako pre univerzálnu primitívne rekurzívnu funkciu? Ak nie, kde to zlyhá?

8. Dá sa definovať efektívne číslovanie (a teda univerzálna funkcia) pre všetky (totálne) rekurzívne funkcie?

9. Uvažujme nasledujúcu definíciu *alternatívnej minimalizácie*:

   Alternatívnou minimalizáciou (možno čiastočnej) funkcie $f(y, \overline{x})$ vzniká funkcia $MIN[f] = g$ taká, že

$$g(\overline{x}) = \begin{cases} \min\{i \mid f(i, \overline{x}) > 0\} & \leftarrow \text{ak také } i \text{ existuje} \\ \bot & \leftarrow \text{inak} \end{cases}$$

   Na rozdiel od klasickej minimalizácie teda nepožadujeme, aby $\forall j < i : f(j, \overline{x}) = 0$, v tomto prípade niektoré z týchto hodnôt môžu byť aj nedefinované.

   Dokážte alebo vyvráťte: Trieda čiastočne rekurzívnych funkcií je uzavretá na alternatívnu minimalizáciu.

Riešenie cvičenia 5

V tomto cvičení bolo našou úlohou dokázať alebo vyvrátiť nasledovné tvrdenie: „Nech f je prostá unárna primitívne rekurzívna funkcia. Potom existuje primitívne rekurzívna funkcia g taká, že $\forall n : g(f(n)) = n$.“

Toto riešenie je úmyselne dlhé. Netreba sa ho ale zľaknúť. Samotné riešenie by sa dalo napísať na pár riadkov, tu je však navyše aj jeden možný myšlienkový proces, ktorý k nemu vedie.

Zamyslime sa, ako k takejto úlohe pristupovať. Na úvod môžeme skúsiť dokázať, že tvrdenie platí. Keďže f vôbec nepoznáme, asi bude tento dôkaz vyzerať spôsobom „zoberiem program, ktorý počíta f, a pomocou neho zostrojíme nový program, ktorý bude počítať g“. No ale bez nejakej hlbšej analýzy toho, čo a ako f počíta, máme v princípe jedinú možnosť, ako naprogramovať g:

```
g(y):
    x = 0
    while f(x) != y: x += 1
    return x
```

Tento postup má ale hneď dva problémy. Prvým je použitie while-cyklu, ktoré nám môže pokaziť primitívnu rekurzívnosť. A druhým je to, že podľa zadania f nemusí byť bijektívna. Môže teda existovať y také, že $\forall x : f(x) \neq y$. No a čo spraví náš program pre g, keď dostane takéto y na vstupe? Bude bežať do nekonečna. To veru nechceme. Vyzerá to teda, že k dôkazu tvrdenia priamočiara cesta nevedie. Možno budeme úspešnejší pri snahe vyvrátiť ho.

Čo potrebujeme na vyvrátenie tvrdenia? Nájsť funkciu f, ktorá sa počíta „ľahko“ (teda napr. vieme jej časovú zložitosť vyjadriť nejakou prim. rek. funkciou), ale každá inverzná funkcia k nej sa počíta „ťažko“ (teda nie je primitívne rekurzívna).

Funkcií, ktoré nie sú primitívne rekurzívne, nepoznáme zatiaľ veľa, v princípe len dve – Ackermannovu a univerzálnu pre primitívne rekurzívne funkcie. Asi sa teda oplatí najskôr skúšať variácie na tieto známe témy. Univerzálna primitívne rekurzívna funkcia sa správa divoko, navyše má ďaleko do prostej funkcie. Nádejnejšie vyzerá Ackermannova funkcia. Intuitívne: jej inverzná funkcia by mohla byť primitívne rekurzívna, lebo počítame z veľkej hodnoty malú. Uvidíme.

Tak teda prvý pokus. Môžeme ako g chcieť mať priamo unárnu Ackermannovu funkciu a? (Teda funkciu, pre ktorú $\forall n : a(n) = A(n, n)$. Špeciálne teda $a(0) = A(0, 0) = 1$, $a(1) = A(1, 1) = 3$, $a(2) = A(2, 2) = 7$, $a(3) = A(3, 3) = 61$ a $a(4)$ už je nepredstaviteľne obrovské.)

Nemôžeme. Totiž také f, ku ktorému by bola a inverznou funkciou, by muselo $a(0)$ zobraziť na 0, $a(1)$ na 1, $a(2)$ na 2, atď. A tým sme si už minuli všetky prirodzené čísla a neostalo nám žiadne, ktoré by sme mohli vrátiť ako $f(0)$ či ako $f(47)$.

To je ale len syntaktický problém, ktorý vieme ľahko vyriešiť vhodným pomasírovaním oboch funkcií. Zvolíme teda funkciu φ definovanú nasledovným predpisom: ak $\exists n : x = a(n)$, tak $\varphi(x) = 2n$, inak $\varphi(x) = 2x + 1$.

Takto upravená funkcia teda robí nasledovné: ak na vstupe dostaneš nejaký výstup Ackermannovej funkcie, vráť párne číslo (ktoré je dvojnásobkom príslušného vstupu pre ňu), inak vráť nepárne číslo. Naša funkcia φ je teda nejakou vhodne upravenou inverznou Ackermannovou funkciou.

Uvedomte si ešte, že korektnosť definície φ aj jej prostosť obe vyplývajú z toho, že funkcia a je ostro rastúca, a teda prostá.

Začíname už možno mať pocit, že toto by mohol byť ten protipríklad, ktorý hľadáme. Potrebujeme teda dokázať dve tvrdenia:
– že je naša funkcia φ primitívne rekurzívna,
– že žiadna funkcia ψ, pre ktorú platí $\forall n : \psi(\varphi(n)) = n$, primitívne rekurzívna nie je.

Začneme tým druhým. Predpokladajme, že máme takúto funkciu ψ, ktorá je primitívne rekurzívna. Ako pomocou ψ určiť hodnotu $x = a(z) = A(z, z)$ pre konkrétne z? Vieme, že pre toto konkrétne x vráti funkcia φ hodnotu $2z$. A teda ψ musí pre vstup $2z$ vrátiť hodnotu x.

Inými slovami, unárnu Ackermannovu funkciu a vieme vypočítať pomocou ψ nasledovným predpisom: $\forall z : a(z) = \psi(2z)$. A to je spor, lebo potom by aj a musela byť primitívne rekurzívna.

Zostáva zdôvodniť, že je φ primitívne rekurzívna. Na to budeme potrebovať nasledovné ľahko dokázateľné tvrdenie: $\forall m, n : A(m, n) > m \wedge A(m, n) > n$. (Dôkaz indukciou.)

My hľadáme pre dané x konkrétne n také, že $A(n, n) = x$. To môžeme spraviť tak, že nájdeme úplne všetky m, n také, že $A(m, n) \leq x$ (a vypočítame aj ich hodnoty). A na to zjavne stačí uvažovať len $m, n < x$.

Celý výpočet môžeme teda spraviť nasledovným primitívne rekurzívnym programom:

```
sprav pole A[0..x-1][0..x-1]
pre m od 0 po x-1:
    pre n od 0 po x-1:
        ak m==0:
            A[m][n] = n+1
        inak:
            ak n==0:
                A[m][0] = A[m-1][1]
            inak:
                tmp = A[m][n-1]
                if tmp >= x:
                    A[m][n] = x+1
                inak:
                    A[m][n] = A[m-1][tmp]
```

(Hodnotu $x + 1$ použijeme pre všetky políčka, ktorých skutočná hodnota je väčšia ako x. Polia sme síce v našom jazyku z prednášky zatiaľ nepovolili, ale ľahko si domyslíte, že konečne veľké pole vieme primitívne rekurzívne zakódovať do jedného čísla.)

Program pre funkciu φ teda zostrojí vyššie popísaným spôsobom pole A a následne prezrie všetky políčka na jeho uhlopriečke. Ak nájde i také, že $A[i][i] = x$, vráti $2i$, inak vráti $2x + 1$.

Kapitola 7: Vypočítateľné reálne čísla

1. Dokážte, že intuitívna a moderná definícia vypočítateľných čísel zo skrípt sú ekvivalentné.

2. Nezáporné reálne číslo x voláme $\mathbb{Q}$-vyčísliteľné, ak existuje program počítajúci funkciu $f_x : \mathbb{N} \to \mathbb{N}^2$ s nasledujúcou vlastnosťou:

$$\forall n \geq 0 : \quad b_n \neq 0 \ \wedge \ \left| x - \frac{a_n}{b_n} \right| < \frac{1}{10^n}, \qquad \text{kde } (a_n, b_n) = f_x(n)$$

   Aký je vzťah medzi množinou vyčísliteľných a množinou $\mathbb{Q}$-vyčísliteľných čísel?

3. Nezáporné reálne číslo x voláme konvergentne vyčísliteľné, ak existuje program počítajúci funkciu $f_x : \mathbb{N} \to \mathbb{N}^2$ s nasledujúcou vlastnosťou:

   Nech $f_x(n) = (a_n, b_n)$. Potom $\forall n : b_n > 0$ a postupnosť $\{a_n/b_n\}_{n=0}^{\infty}$ konverguje k x.

   Aký je vzťah medzi množinou vyčísliteľných a množinou konvergentne vyčísliteľných čísel?

4. Poriadne dokážte, že ľubovoľné nezáporné reálne algebraické číslo je vyčísliteľné.

5. Dokážte alebo vyvráťte: tvoria vyčísliteľné čísla pole?

Kapitola 8: Busy Beaver a iné nerekurzívne funkcie

1. Dokážte alebo vyvráťte: Busy Beaver funkcia B rastie rýchlejšie ako hociktorá rekurzívna funkcia.

2. Uvažujme analógiu Busy Beavera pre registrové stroje: Aké najväčšie číslo vie v niektorom registri vyrobiť program s n inštrukciami v okamihu, keď zastane? Dokážte, že príslušná busy-beaver-like funkcia B_M nie je vypočítateľná.

3. Nájdite čo najviac presných hodnôt funkcie B_M.

4. Uvažujme Turingove stroje z prednášky o Busy Beaver funkcii (obojsmerne nekonečná páska, pracovná abeceda obsahuje len 0 a 1). Nech $R(A) = 0$ ak stroj A na prázdnom vstupe nezastaví, inak nech $R(A)$ je počet políčok, o koľko je na konci výpočtu čítacia hlava napravo od políčka, kde začínala. (Pre niektoré A môže byť $R(A)$ aj záporné.) Nech $R(n)$ je maximum z $R(A)$ cez všetky n-stavové A.

   Teda $R(n)$ hovorí, že každý n-stavový TS, ktorý na prázdnom vstupe zastane, zastane s hlavou nanajvýš $R(n)$ políčok napravo od miesta, kde začínal.

   Je funkcia $R(n)$ rekurzívna?

Kapitola 9: Množiny a redukcie

1. Pre ľubovoľnú (aj čiastočnú) funkciu $f : \mathbb{N} \to \mathbb{N}$ nazveme grafom f množinu $Graf_f = \{ (x, y) \mid x \in \mathbb{N} \wedge y = f(x)\}$.

   Dokážte alebo vyvráťte každé z nasledujúcich tvrdení:

   a) Ak je funkcia f primitívne rekurzívna, tak jej graf je primitívne rekurzívny.

   b) Ak je funkcia f totálna a jej graf je primitívne rekurzívny, tak je f primitívne rekurzívna.

   c) Ak je funkcia f rekurzívna, tak jej graf je rekurzívny.

   d) Ak je funkcia f totálna a jej graf je rekurzívny, tak je f rekurzívna.

   e) Ak je funkcia f totálna a jej graf je rekurzívne vyčísliteľný, tak je f rekurzívna.

2. Dokážte alebo vyvráťte: Pre každú čiastočne rekurzívnu funkciu je jej definičný obor rekurzívne vyčísliteľná množina.

3. Nech $A \subseteq \mathbb{N}$, pričom $A \neq \emptyset$ a $A \neq \mathbb{N}$. Dokážte alebo vyvráťte: A je rekurzívna práve vtedy, keď $A \leq_m \{47\}$.

4. Dokážte, že ak pre rekurzívne vyčísliteľnú množinu platí $A \leq_m \overline{A}$, tak je rekurzívna.

5. Nájdite množinu B, pre ktorú platí $B \leq_m \overline{B}$, ale B nie je rekurzívna.

6. Dokážte, že pre každé $k > 1$ obsahuje nami definovaná postupnosť funkcií $\varphi_n^{(k)}$ všetky k-árne čiastočne rekurzívne funkcie.

7. Dokážte, že ak má čiastočne rekurzívna funkcia rekurzívny definičný obor, tak existuje aspoň jedno jej zúplnenie, ktoré je rekurzívne.

8. Prečo nedostaneme spor, keď v dôkaze, že U nemá rekurzívne zúplnenie zadefinujeme funkciu f pomocou U namiesto U'?

9. Dokážte m-úplnosť množiny $UNIV$.

10. Dokážte m-úplnosť množiny $\{n \mid 47 < |W_n|\}$, kde W_n je efektívne číslovanie rekurzívne vyčísliteľných množín.

11. Nech φ_n je naše číslovanie unárnych čiastočne rekurzívnych funkcií.

    Nech $MONOTONE = \{n \mid \varphi_n$ je totálna a $\forall x : \varphi_n(x) \leq \varphi_n(x + 1)\}$.

    Dokážte, že $\overline{HALT} \leq_m MONOTONE$.

12. Nech A je jednoduchá množina. Uvažujme množinu $B = \{c(a, n) \mid a \in A \wedge n \in \mathbb{N}\}$, kde c je naša párovacia funkcia.

O každom z nasledujúcich tvrdení rozhodnite, či platí vždy, nikdy, alebo niekedy:
– B je rekurzívne vyčísliteľná.
– B je rekurzívna.
– B je kreatívna (t. j. m-úplná).
– B je jednoduchá.

13. Dokážte, že ak má čiastočne rekurzívna funkcia rekurzívny definičný obor, tak existuje aspoň jedno jej zúplnenie, ktoré je rekurzívne.

14. Každé políčko štvorcovej siete v prvom kvadrante vyplníme čiernou alebo bielou. Pre každý riadok a stĺpec teraz môžeme definovať množinu čiernych a množinu bielych políčok v ňom.

    Napr. $WR_{47} = \{y \mid (47, y) \text{ je biele}\}$.

    Existuje také vyfarbenie, pri ktorom žiadna z uvedených množín nie je rekurzívna?

    Existuje také vyfarbenie, pri ktorom žiadna z uvedených množín nielenže nie je rekurzívna, ale dokonca neobsahuje žiadnu nekonečnú rekurzívnu podmnožinu?

    Existuje také vyfarbenie, pri ktorom žiadna z uvedených množín nielenže nie je rekurzívna a neobsahuje žiadnu nekonečnú rekurzívnu podmnožinu, ale navyše je každá z nich rekurzívne vyčísliteľná?

15. Dokážte alebo vyvráťte: Množina A je rekurzívna práve vtedy, ak existuje rekurzívna funkcia f spĺňajúca nasledujúce podmienky:

    - $\forall a \in A: \ \exists n \in \mathbb{N}: \ f(n) = a$
    - $\forall n \in \mathbb{N}: \ f(n) \in A$
    - $\forall n \in \mathbb{N}: \ f(n) < f(n+1)$

16. Dokážte alebo vyvráťte: Existuje množina prirodzených čísel A taká, že ani A, ani $\mathbb{N} - A$ neobsahuje nekonečnú aritmetickú postupnosť.

17. Dokážte: Existuje nekonečná rekurzívne vyčísliteľná množina, ktorá nemá žiadnu nekonečnú primitívne rekurzívnu podmnožinu.

18. Dokážte alebo vyvráťte: Každá nekonečná rekurzívne vyčísliteľná množina má nekonečnú rekurzívnu podmnožinu.

19. Prečo v predchádzajúcej úlohe nezafunguje konštrukcia, ktorá fungovala v o jedno skoršej úlohe?

20. Množinu voláme *imúnna*, ak je nekonečná, ale nemá žiadnu nekonečnú rekurzívne vyčísliteľnú podmnožinu. Zjavne komplementom každej jednoduchej množiny je nejaká imúnna množina.

    Nájdite imúnnu množinu, ktorej komplement nie je jednoduchá množina.

21. Existuje imúnna množina, ktorej komplement je tiež imúnna množina?

Kapitola 10: Logika a Gödelove vety

1. Zostrojte v našej predikátovej logike prvého rádu reťazec, ktorý bude zodpovedať predikátu „x je zložené číslo“.

2. Zostrojte v našej predikátovej logike prvého rádu reťazec, ktorý bude zodpovedať výroku „existuje nekonečne veľa prvočísel“.

3. Nájdite konkrétny formálny systém, v ktorom je množina dokázateľných tvrdení rekurzívna.

4. Množina axióm je *minimálna*, ak sa žiadna z nich nedá dokázať z ostatných.

   Uvažujme rozhodovací problém, ktorého inštanciou je dvojica (A, D), kde A je konečná množina axióm a $D(x, y, z)$ je ternárny rekurzívny predikát „reťazec y je dôkazom tvrdenia x z množiny axióm zakódovanej do z“. Otázkou, ktorú treba zodpovedať, je, či je daná množina A minimálna pre daný predikát D.

   Dokážte, že tento rozhodovací problém nie je rozhodnuteľný, ale že jeho komplement (zistiť, že A minimálna nie je) je čiastočne rozhodnuteľný.

Kapitola 11: Vety o rekurzii

1. Vyberte si ľubovoľný programovací jazyk a napíšte v ňom program, ktorý vypíše samého seba.

   Potom ho upravte, napríklad tak, aby vypísal samého seba, ale zmenil pritom všetky samohlásky na a.

2. Vyberte si ľubovoľný programovací jazyk a napíšte v ňom dva rôzne programy X a Y také, že na prázdnom vstupe X vypíše Y a Y vypíše zase X.

   (Viete nájsť súvis s vetami o rekurzii? Dalo by sa ukázať, že aj táto úloha ide riešiť v ľubovoľnom rozumnom programovacom jazyku?)

3. Ako treba zvoliť transformáciu f v Rogersovej verzii druhej vety o rekurzii, aby sme ukázali, že existuje program, ktorý vypíše samého seba?

4. Môže v druhej vete o rekurzii niekedy existovať viac ako jeden pevný bod? Teda môžu napr. v Rogersovej verzii k jednému f existovať dva rôzne programy n_1 a n_2, ktoré počítajú dve rôzne čiastočne rekurzívne funkcie, ale platí že n_1 počíta rovnakú funkciu ako $f(n_1)$ a n_2 zase rovnakú ako $f(n_2)$?

5. Prečítajte si o `http://www.ioccc.org/2012/endoh2/endoh2.c` (popis tu: `http://www.ioccc.org/2012/endoh2/hint.html`) a potom sa zamyslite, ako takéto niečo (funkcionalitu, nie nutne formátovanie) naprogramovať.