1. Uvažujme abecedu Σ, t.ž., symbol $R \notin \Sigma$, a následující kódování deterministického konečného automatu do Turingova stroje: pro $A = (Q, \Sigma, \delta, q_0, F)$ sestrojíme TS $M_{sim}(A) = (Q \cup \{q_0^M, q_f^M\}, \Sigma, \Sigma \cup \{\Delta\}, \delta_M, q_0^M, q_f^M)$, kde $Q \cap \{q_0^M, q_f^M\} = \emptyset$ a δ_M je definována následovně:

- $\delta_M(q_0^M, \Delta) = (q_0, R)$
- $\forall f \in F : \delta_M(f, \Delta) = (q_f^M, R)$
- $\delta_M(q, a) = (p, R) \Leftrightarrow \delta(q, a) = p$

Množina kódů turingových strojů vzniklých transfomaci DKA $KA = \{\langle M \rangle \mid M = M_{sim}(A)$ pro nějaký DKA $A\}$ je rozhodnutelná, protože lze jednoduše ověřit tvar přechodové funkce.

Rozhodněte a dokažte, zda následující jazyky jsou (resp. nejsou) rekurzivní (resp. rekursivně vyčíslitelné):

- $L_1 = \{\langle M \rangle \# \langle w \rangle \mid w \notin L(M) \wedge \langle M \rangle \in KA\}$
- $L_2 = \{\langle M \rangle \# \langle w \rangle \mid w \notin L(M) \wedge \langle M \rangle \notin KA\}$

20 bodů

a) $L_1 = \{<M>\#<w> \mid w \notin L(M) \wedge <M> \in KA\}$

- L_1 je rozhodnutelný problém, tedy $L_1 \in \mathcal{L}_{REC}$
- Pokud $L_1 \in \mathcal{L}_{REC}$, poté existuje TS T takový, že $L(T) = L_1$

obecný popis konstrukce TS T:

1, TS T zkontroluje, že má na svém vstupu řetězec $<M>\#<w>$ v korektním tvaru. Pokud ne, zamítá TS T svůj vstup.

2, TS T ověří, že $<M>$ je validní kód TS, který vznikl transformací z DKA (ověření tvaru přechodové funkce). Pokud se nejedná o validní kód, TS T zamítne svůj vstup.

3, TS T spustí simulaci stroje M nad řetězcem w. Simulace nad řetězcem vždy zastaví.

4, Pokud TS M odmítne vstup w, akceptuje TS T svůj vstup.

5, Pokud TS M akceptuje vstup w, odmítne TS T svůj vstup.

$$\boxed{L_1 \in \mathcal{L}_{REC}}$$

b) $L_2 = \{\langle M\rangle \# \langle w\rangle \mid w \notin L(M) \wedge \langle M\rangle \notin KA\}$

- L_2 je nerozhodnutelný problém, tedy $L_2 \notin \mathcal{L}_{RE}$
- jako důkaz použijeme redukci co-HP $\leq L_2$
- funkce $\sigma(\langle M\rangle \# \langle w\rangle) \to (\langle M'\rangle \# \langle w'\rangle)$:
  1) Pokud $\langle M\rangle \# \langle w\rangle$ není korektní kód, pak M' je kód TS, který v 1 kroce okamžitě přijme svůj vstup ($L(M') = \Sigma^*$), a $\langle w'\rangle$ je libovolné slovo nad Σ^*
  2) Vrať $\langle M'\rangle \# \langle w'\rangle$, kde $\langle w'\rangle$ je kód libovolného slova (např. ε) a $\langle M'\rangle$ je kód TS M':

kód TS M' pro vstup w''

1) Simuluj M na w
2) akceptuj w''

Ověření:

1) $\langle M\rangle \# \langle w\rangle \in$ co-HP $\Rightarrow M$ nad w cyklí $\Rightarrow M'$ cyklí nad libovolným w'' $\Rightarrow L(M') = \emptyset \wedge \langle M'\rangle \notin KA$ (jelikož v sobě simuluje M nad w, což DKA nedokáže) $\Rightarrow \langle M'\rangle \# \langle w'\rangle \in L_2$

2) $\langle M\rangle \# \langle w\rangle \notin$ co-HP $\Rightarrow M$ nad w zastaví $\Rightarrow M'$ akceptuje libovolné w'' $\Rightarrow L(M') = \Sigma^* \Rightarrow \langle M'\rangle \# \langle w'\rangle \notin L_2$

$$\boxed{\text{co-HP} \leq L_2 \wedge \text{co-HP} \notin \mathcal{L}_{RE} \Rightarrow L_2 \notin \mathcal{L}_{RE}}$$

2. Jan a Eliška si vymysleli novou hru. Mají barevné křídy o b ($b \geq 2$) barvách. Na chodník si nakreslili křídou kolečka a některá z nich propojili čárami. Teď by rádi do každého kolečka namalovali x ($x \geq 2$) barevných značek (barvy značek v jednom kolečku se mohou opakovat) tak, aby kolečka propojená čárou nebyla označená stejně. Pořadí značek v kolečku nehraje roli. Otázka zní, jestli je takového označení možné.

Formálně definujme hru Jana a Elišky jako n-tici $H = (K, C, b, x)$, kde

- K je konečná množina koleček,
- $C \subseteq \{\{a, b\} \mid a, b \in K\}$ je množina čar,
- $b \geq 2$ je počet barvev,
- $x \geq 2$ požadovaný počet značek.

Hra H má řešení, pokud existuje zobrazení $O : K \times \langle 1, b\rangle \rightarrow \mathbb{N}$ takové, že

- $\forall a \in K : \sum_{i=1}^{b} O(a, i) = x$ (počet značek v jednom kolečku je roven x)
- $\forall \{a, b\} \in C\ \exists i : O(a, i) \neq O(b, i)$ (označení dvojice míst spojených čárou se liší)

Dokažte, že problém existence řešení pro hru H je NP-úplný.

(Pozn: Pomůže Vám NP-úplnost některého z problémů uvedených zde:
`https://en.wikipedia.org/wiki/NP-completeness#NP-complete_problems`
v odstavci „*NP-complete problems*“.) 15 bodů

1) Nejprve je vhodné ověřit, jestli zadaný problém náleží NP.

NTS nejdříve ověří, že každé kolečko obsahuje přesně X značek a značky ze všech koleček jsou namalovány maximálně b barvami. Poté NTS uhádne správné označení a ověří, zda je splnitelné. Proto zadaný herní problém náleží NP.

2) Redukce NP-úplného problému na H

- Jako NP-úplný problém v tomto případě zvolíme SUDOKU
- klasické sudoku má herní mřížku 9x9

$$\text{Sudoku} \overset{?}{\leq_p} H$$

Pokud dokážeme tuto redukci, dokážeme i, že Hra je NP-úplný problém.

Postup redukce pro Sudoku 9x9

- definujeme instanci $H = (K, C, b, x)$, kde
  - K: Každé kolečko odpovídá 1 políčku v Sudoku
  - b: Počet barev pevně nastavíme na hodnotu 2
  - x: Zvolíme rovno hodnotě 8 pro Sudoku 9x9
  - C: Čáru vytvoříme mezi kolečky, pokud jsou:
    - ve stejném řádku
    - ve stejném sloupci
    - ve stejném bloku

    $\Big\}$ Pravidla sudoku
- každé kolečko musí obsahovat jednu z dvojic $\{(0,8);(1,7);(2,6);\dots;(7,1);(8,0)\}$, aby byl součet čísel ve dvojici roven 8. Pokud tedy například číslo v sudoku je 1, přiřadíme mu dvojici $(0;8)$. Pro číslo 2 je to dále $(1;7)$, atd. až po 9 odpovídá $(8;0)$ (na pořadí čísel ve dvojici záleží, tj $(x,y) \neq (y,x)$, kde x a y značí počet značek jedné z barev podle b)

Korektnost redukce:

- pokud existuje řešení sudoku, přiřadíme uzlům vhodné dvojice čísel tak, aby:
  - → součet každé dvojice odpovídal číslu 8
  - → spojená kolečka čárami měla různou dvojici značek, protože každá dvojice odpovídá číslům 1 až 9 v sudoku

$$\boxed{\text{Hra} \in NP \wedge \text{Sudoku} \leq_p \text{Hra} \Rightarrow \text{Hra} \in NP\text{-complete}}$$

3. Uvažujeme funkci get_next, která má na vstupu řetězec nad abecedou $\Sigma = \{A, ..., Z\}$ a jeho délku l. Funkce vrácí následující řetězec vzhledem k lexikografickému uspořádání. Funkce $next\_char$ vrací následující znak latinské abecedy. Analyzujte a zdůvodněte amortizovanou časovou složitost libovolné posloupnosti n operací $str := get\_next(str, l)$. Na začátku je zafixována konstanta $l > 0$ a počáteční hodnota $str = A^l$ (řetězec obsahující l symbolů A).

Předpokládejme uniformní cenové kriterium, kde každý řádek má cenu 1 (vyjímkou je řádek 7 s cenou 0).

```
1  Function get_next(char [] str, int l)
2      fin := false;
3      while ¬(fin) ∧ l > 0 do
4          l := l − 1;
5          if str[l] = Z then
6              str[l] := A;
7          else
8              next_char(str[l]);
9              fin := true;
10     return str;
```

15 bodů

- Abeceda Σ má 26 znaků $(|\Sigma| = 26)$

- 26. krok způsobí přepis opět na A, proto je potřeba během každé operace našetřit na tento přepis

- přepis nastane i pro 26^2 krok, 26^3 krok ...., 26^l krok a i na toto je potřeba našetřit

Případ*	Cena	Kredity
iter=1	$2+5+1 = 8$	$8 + \frac{4}{26} + \frac{4}{26^2} + \dots + \frac{4}{26^l} = 8 + \frac{4}{25} = \frac{204}{25}$
iter=2	$2+4+5+1 = 12$	$8 + \frac{4}{25} = \frac{204}{25}$
$\vdots$	$\vdots$	$\vdots$
iter=l	$2+(l-1)\cdot 4+4+1$	$7 + \frac{4}{25} = \frac{179}{25}$

*kolik iterací cyklu se provede při zavolání funkce get_next.

- speciální případ nastává, pokud $l=1$. Poté probíhá vždy 1 iterace s cenou 8 nebo 7 podle aktuálního znaku.

Složitost posloupnosti n operací je poté

$$\boxed{n \cdot \max\left(\frac{204}{25}, \frac{179}{25}\right) = n \cdot \frac{204}{25} \in O(n)}$$

4. Uvažujme funkci *find_suffix*, která má na vstupu pole čísel $array$ o velikosti $size$ (chybné vstupy neuvažujte) a která se snaží nalést v rámci pole suffix takový, že součet čísel v tomto suffixu je roven hodnotě $final$.

a) • Analyzujte časovou složitost funkce *find_suffix* v nejlepším případě

b) • Analyzujte časovou složitost funkce *find_suffix* v nejhorším případě.

c) • Navrhněte funkci *find_opt*, která bude dávat stejný výsledek jako funkce *find_suffix*, ale bude mít lepší asymptotickou složitost v nejhorším případě.

d) • Analyzujte časovou složitost funkce *find_opt* v nejhorším případě.

Předpokládejme uniformní cenové kriterium, kde každý řádek má cenu 1.

```
Function find_suffix(int * array, int size, int final)
    int i, j;
    i := 0;
    while i < size do
        j := i;
        int tmp := 0;
        while j < size do
            tmp := tmp + array[j];
            j := j + 1;
        if tmp = final then
            return ANO
        i := i + 1;
    return NE
```

„Zavedeme $n = size$"

15 bodů

a) „Suma hodnot celého pole je rovna hodnotě final" ($\sum_{i=1}^{size} array[i] = final$), poté se provede 1 iterace vnějšího cyklu a dále „doběhne" celý vnitřní cyklus.

Složitost vnitřního cyklu je $O(3n+1) \subseteq O(n)$

Celková složitost je $\boxed{O(2+1+1\cdot(2+3n+1+2)) \subseteq O(n)}$

b) „Hledaný suffix neexistuje", poté se provede „size" iterací vnějšího cyklu a pro každou iteraci dále proběhne vnitřní cyklus

Pro každou iteraci vnějšího cyklu se provede $2+(size-i)\cdot 3+1+2$ kroků, proto lze celkovou složitost vyjádřit jako

$$\boxed{O(2+n\cdot 5+3\cdot[n+(n-1)+(n-2)+\dots+(n-n+1)]+1) \subseteq O(n^2)}$$

c)

```
Function find_opt(int * array, int size, int final)
    int i := size - 1;
    int tmp = 0;
    while i >= 0 do
        tmp := tmp + array[i];
        if tmp = final then
            return ANO;
        i := i - 1;
    return NE;
```

d) V nejhorším případě je složitost funkce

$$\boxed{O(2 + n \cdot 3 + 1 + 1) \subseteq O(n)}$$

5. Mějme teorii T se signaturou $\langle\{Kral_{/0}, Dama_{/0}, Vez_{/0}, Kun_{/0}, Pesec_{/0}\}, \{ohrozuje_{/2}, =_{/2}\}\rangle$ ($=$ je standardní rovnost) se speciálními axiomy

$$\forall x(x = Kral \lor x = Dama \lor x = Vez \lor x = Kun \lor x = Pesec)$$
$$\forall x\, \neg ohrozuje(x, Kral)$$
$$\forall x, y(ohrozuje(x, y) \land ohrozuje(y, x) \Rightarrow x \neq Kun)$$
$$\forall x(ohrozuje(Pesec, x) \Rightarrow ohrozuje(x, Pesec) \lor x = Kun)$$

i) Rozhodněte a stručně zdůvodněte, zda T je: a) bezesporná, b) úplná a c) rozhodnutelná (tj. množina důsledků T je rozhodnutelná).

15 bodů

a) Teorie T je bezesporná právě tehdy, když má model.
- pokud se nám tedy podaří nalézt model teorie T, dokážeme, že je teorie bezesporná

$D_{M_a} = \{Kral, Dama, Vez, Kun, Pesec\}$

$M_a(ohrozuje) = \{(Vez, Dama), (Dama, Vez), (Pesec, Dama)\}$

- model M_a je validní model teorie T, proto teorie

$\boxed{\text{T je bezesporná}}$

b) Pokud má teorie T jediný model, poté je úplná
- pokud tedy nalezneme 2. model, dokážeme, že teorie T není úplná

$D_{M_b} = \{Kral, Dama, Vez, Kun, Pesec\}$

$M_b(ohrozuje) = \{(Pesec, Kun), (Kun, Dama)\}$

- model M_b je opět validní model, proto můžeme říct, že teorie má více modelů, a tedy

$\boxed{\text{T není úplná}}$

c) Aby byla teorie rozhodnutelná, musí být efektivní, bezesporná a úplná. Jelikož teorie T není úplná, nemůže být ani rozhodnutelná.

Pokud je teorie T efektivní, může být alespoň částečně rozhodnutelná.

- Teorie obsahuje konečný počet axiomů
- Teorie obsahuje konečný počet prvků

Vzhledem k omezenému počtu axiomů a prvků domény můžeme efektivně ověřit korektnost modelu, a tedy

$\boxed{T \text{ je částečně rozhodnutelná}}$