

22. Turingov stroj

video prezentácia

[turing machine](#)

Alan Turing

[Alan Turing](#) ([tiež](#)) (1912-1954) sa považuje za zakladateľa modernej informatiky - rok 2012 sa na celom svete oslavoval ako [Turingov rok](#).

[Turingov stroj](#) je veľmi zjednodušený model počítača, vďaka ktorému sa informatika dokáže zaoberať zložitou problémami - Turing ho publikoval v roku 1936 (keď mal 24 rokov).

Základná idea takéhoto stroja je nasledovná:

- pracuje s nekonečnou páskou - postupnosť políčok, na každom môže byť nejaký symbol (my si to zjednodušíme obyčajnými znakmi, t.j. jednoznakovými reťazcami) - táto páska je nekonečná oboma smermi
- nad páskou sa pohybuje čítacia/zapisovacia hlava, ktorá vie prečítať symbol na páske a prepísať ho iným symbolom
- samotný stroj (riadiaca jednotka) sa v každom momente nachádza v jednom zo svojich stavov: na základe momentálneho stavu a prečítaného symbolu na páske sa riadiaca jednotka rozhoduje, čo bude robiť
- na začiatku sa na políčkach pásky nachádzajú znaky nejakého vstupného reťazca (postupnosť symbolov), stroj sa nachádza v počiatočnom stave (stavy pomenujeme ľubovoľnými reťazcami znakov, napríklad 's1' alebo 'end'), celý zvyšok nekonečnej pásky obsahuje prázdne symboly (my sa dohodneme, že to budeme označovať symbolom '_')
- činnosť stroja (program) sa definuje špeciálnou dvojrozmernou tabuľkou, tzv. **pravidlami**
- každé pravidlo je takáto päťica: (stav1, znak1, znak2, smer, stav2) a popisuje:
 - keď je stroj v stave stav1 a na páske (pod čítacou hlavou) je práve symbol znak1, tak ho stroj prepíše týmto novým symbolom znak2, posunie sa na páske podľa zadaného smeru smer o (-1, 0, 1) políčko a zmení svoj stav na stav2
 - napríklad pravidlo (s0, a, b, >, s1) označuje: v stave 's0' so symbolom 'a' na páske ho zmení na 'b', posunie sa o jedno políčko vpravo a zmení svoj stav na 's1'
 - pre lepšiu čitateľnosť budeme smery na posúvanie hlavy označovať pomocou znakov '<' (vľavo), '>' (vpravo), '=' (bez posunu)
 - takéto päťice sa niekedy označujú aj v tomto tvare: (stav1, znak1) -> (znak2, smer, stav2), t.j. pre danú dvojicu stav a znak, sa zmení znak
- program má väčšinou viac stavov, z ktorých niektoré sú špeciálne, tzv. koncové a majú takýto význam:
 - keď riadiaca jednotka prejde do koncového stavu, výpočet stroja sa zastaví a stroj oznámi, že **akceptoval** vstupné slovo, vtedy sa môžeme pozrieť na obsah pásky a táto informácia môže byť výsledkom výpočtu
- stroj sa zastaví aj v prípade, že neexistuje pravidlo, pomocou ktorého by sa dalo pokračovať (stroj skončil v nekonečnom stave), vtedy
 - hovoríme, že stroj oznámil, že **zamietol** (neakceptoval) vstupné slovo
- zrejme sa takýto stroj môže niekedy aj zacykliť a neskončí nikdy (pre informatikov je aj toto veľmi dôležitá informácia)

Na internete sa dá nájsť veľa rôznych zábavných stránok, ktoré sa venujú Turingovmu stroju, napríklad

- [Alan Turing's 100th Birthday](#)

- [LEGO Turing Machine](#)
- [Turing Machine-IA Lego Mindstorms](#)
- [A Turing Machine - Overview](#)
- [Turing machine running binary counter algorithm](#)

Zostavme náš prvý program pre Turingov stroj:

```
(0, a) -> (A, >, 0)
(0, _) -> (_, =, end)
```

Vidíme, že pre počiatočný stav (zrejme je to stav s menom 0) sú tu dve pravidlá: buď je na páske symbol 'a' alebo symbol '_', ktorý označuje prázdne políčko. Ak teda čítacia hlava má pod sebou na páske symbol 'a', tak ho prepíše na symbol 'A' a posunie hlavu na políčko vpravo. Riadiaca jednotka pritom stále ostáva v stave 0. Vďaka tomuto pravidlu sa postupne všetky písmená 'a' nahradia 'A'. Ak sa pritom narazí na iný symbol, prvé pravidlo sa už nebude dať použiť a stroj sa pozrie, či nemá pre tento prípad iné pravidlo. Ak je na páske už len prázdny symbol, stroj sa zastaví a oznámi radostnú správu, že vstupné slovo **akceptoval** a vyrobil z neho nový reťazec. Ak ale bude na páske za postupnosťou 'a' aj nejaké iné písmeno, stroj nenájde pravidlo, ktoré by mohol použiť a preto sa zastaví. Oznámi pritom správu, že takéto vstupné slovo **zamietol**.

Priebeh výpočtu pre vstupné slovo 'aaa' by sme si mohli znázorniť, napríklad takto:

```
aaa
^ 0
Aaa
^ 0
AAa
^ 0
AAA_
^ 0
AAA_
^ end
True
```

True na konci výpisu znamená, že stroj sa úspešne zastavil v koncovom stave, teda stroj **akceptoval** vstupné slovo.

Všimnite si, že v našej vizualizácii sa na páske automaticky objavujú prázdne symboly, keďže páska je nekonečná a okrem vstupného slova obsahuje práve tieto znaky.

Ak by sme zadali, napríklad slovo 'aba', tak by výpočet prebiehal takto:

```
aba
^ 0
Aba
^ 0
False
```

False tu znamená, že stroj sa zastavil v inom ako koncovom stave, teda zamietol vstup.

Môžeme povedať, že náš prvý program pre Turingov stroj akceptuje len také vstupné slová, ktoré sú zložené len zo znakov 'a'. Okrem toho sa na páske znaky 'a' zmenia na 'A'.

Návrh simulátora

Aby sme mohli s takýmto strojom lepšie experimentovať a mať možnosť si uvedomiť, ako sa pomocou neho riešenia úlohy, naprogramujeme si veľmi jednoduchý simulátor. Začneme tým, že navrhujeme triedu **Paska**, ktorá bude popisovať pásku Turingovho stroja a jej metódy:

```

class Paska:
    def __init__(self, obsah=''):
        self.paska = list(obsah or '_')
        self.poz = 0

    def symbol(self):
        return self.paska[self.poz]

    def zmen_symbol(self, znak):
        self.paska[self.poz] = znak

    def __str__(self):
        return ''.join(self.paska) + '\n' + ' '*self.poz + '^'

    def vpravo(self):
        self.poz += 1
        if self.poz == len(self.paska):
            self.paska.append('_')

    def vlavo(self):
        if self.poz > 0:
            self.poz -= 1
        else:
            self.paska.insert(0, '_')

    def text(self):
        return ''.join(self.paska).strip('_')

```

Atribúty sú asi zrejme:

- **paska** - zoznam (typu `list`), ktorého prvky sú jednoznakové reťazce - tieto reprezentujú políčka pásky
 - pásku Turingovho stroja by sme mohli namiesto zoznamu reprezentovať aj pomocou znakového reťazca, zvolili sme zoznam, nakoľko v ňom sa jednoduchšie mení hodnota jedného prvku
 - všimnite si zápis `self.paska = list(obsah or '_')` a konkrétne použitie operácie `or`, v tomto prípade má takýto význam: ak je premenná `obsah` prázdna (Python ju interpretuje ako `False`), výsledkom operácie je druhý operand, teda `'_'`, inak je výsledkom samotný reťazec `obsah`
- **poz** - pozícia čítacej/zapisovacej hlavy, pomocou tejto pozície budeme indexovať zoznam `paska`
 - metódami `vpravo()` a `vlavo()` sa páska automaticky nafukuje, ak by hlava prišla mimo momentálne prvky zoznamu

Túto triedu využije trieda `Turing`, v ktorej sa okrem pásky bude uchovávať aj samotný program, teda množina prepisovacích pravidiel:

```

class Paska:
    def __init__(self, obsah=''):
        self.paska = list(obsah or '_')
        self.poz = 0

    def symbol(self):
        return self.paska[self.poz]

    def zmen_symbol(self, znak):
        self.paska[self.poz] = znak

    def __str__(self):
        return ''.join(self.paska) + '\n' + ' '*self.poz + '^'

    def vpravo(self):
        self.poz += 1
        if self.poz == len(self.paska):
            self.paska.append('_')

```

```

def vlavo(self):
    if self.poz > 0:
        self.poz -= 1
    else:
        self.paska.insert(0, '_')

def text(self):
    return ''.join(self.paska).strip('_')

symbol = property(symbol, zmen_symbol)
text = property(text)

class Turing:
    def __init__(self, program, obsah='', start=None, koniec={'end', 'stop'}):
        self.program = {}
        self.stav = start
        for riadok in program.split('\n'):
            riadok = riadok.split()
            if len(riadok) == 5:
                stav1, znak1, znak2, smer, stav2 = riadok
                self.program[stav1, znak1] = znak2, smer, stav2
                if self.stav is None:
                    self.stav = stav1
        self.paska = Paska(obsah)
        self.koniec = koniec

    def __str__(self):
        return str(self.paska) + ' ' + self.stav

    def krok(self):
        stav1, znak1 = self.stav, self.paska.symbol
        try:
            znak2, smer, stav2 = self.program[stav1, znak1]
        except KeyError:
            return False
        self.paska.symbol = znak2
        self.stav = stav2
        if smer == '>':
            self.paska.vpravo()
        elif smer == '<':
            self.paska.vlavo()
        return True

    def rob(self, vypis=True):
        if vypis:
            print(self)
        while self.stav not in self.koniec:
            if not self.krok():
                return False
            if vypis:
                print(self)
        return True

```

Atribúty tejto triedy:

- `prog` je tabuľka pravidiel - tieto pravidlá tu môžu byť uvedené v ľubovoľnom poradí a riadiaca jednotka si vyhľadá to správne
 - každé pravidlo sa skladá z piatich prvkov: v akom sme stave, aký je symbol na páske, na aký symbol sa prepíše, ako sa posunie hlava (buď '<' alebo '>') a do akého stavu sa prejde
 - pravidlá budeme ukladať do slovníka - asociatívneho poľa (typ `dict`) tak, že kľúčom bude dvojica (**stav, znak**) a hodnotou pre tento kľúč bude trojica (**nový_znak, smer_posunu, nový_stav**)

- `paska` bude „nekonečná“ postupnosť symbolov, využili sme tu našu triedu `Paska`
- `stav` označuje, momentálne meno stavu (môže byť, napríklad `'s1'` alebo `'0'`), v ktorom sa nachádza Turingov stroj
- `koniec` označuje, množinu koncových stavov

Metódy

- metóda `__init__()` inicializuje atribúty, má tieto parametre:
 - `program` zoznam pravidiel: zadáva sa ako jeden dlhý reťazec, v ktorom sú pravidlá oddelené znakom `'\n'`, každé pravidlo sa skladá z 5 prvkov, pravidlo, ktoré neobsahuje práve 5 prvkov, sa ignoruje (môžeme použiť, napríklad ako komentár)
 - `obsah` počiatočný obsah pásky, hlava sa nastaví na prvý znak tohto reťazca
 - `start` počiatočný stav, v ktorom štartuje Turingov stroj, ak ho ne zadáme, ako počiatočný sa vyberie stav z prvého pravidla
 - `koniec` je množina koncových stavov, predpokladáme, že najčastejšie to bude `{'end', 'stop'}`
- metóda `krok()` - Turingov stroj vykoná jeden krok programu; vráti `False`, keď sa krok nepodari vykonať (teda pre daný `stav` a `znak` na páske neexistuje pravidlo v slovníku `prog`), inak vráti `True`
- metóda `rob()` - riadiaca jednotka bude postupne vykonávať kroky programu, kým sa stroj nezastaví
 - metóda vráti `True`, ak program akceptoval symboly na páske, inak vráti `False`

Do metódy `rob()` sme pridali kontrolný výpis, ktorý môžeme vypnúť parametrom `vypis`.

Turingov stroj otestujeme jediným pravidlom:

```
t = Turing('0 a A > 0', 'aaa')
print(t.rob())
```

Dostávame tento výpis:

```
aaa
^ 0
Aaa
^ 0
AAa
^ 0
AAA_
^ 0
False
```

Zrejme to nemohlo dopadnúť inak ako `False`, lebo náš program neobsahuje pravidlo, ktorého výsledkom by bol koncový stav.

Doplňme druhé pravidlo:

```
t = Turing('0 a A > 0\n0 _ _ = end', 'aaa')
print(t.rob())
```

```
aaa
^ 0
Aaa
^ 0
AAa
^ 0
AAA_
^ 0
AAA_
^ end
True
```

Ďalší turingov program bude akceptovať vstup len vtedy, keď obsahuje jediné slovo `'ahoj'`:

```
print(Turing('''
1 a a > 2
2 h h > 3
3 o o > 4
4 j j > 5
5 _ _ = stop
''', 'ahoj').rob())
```

Všimnite si, že stavy tohto programu sú **1, 2, 3, 4, 5** a **stop**, pričom **1** je počiatočný stav a **stop** je koncový stav.

Program zadaný vstup **'ahoj'** akceptuje, ale napríklad **'hello'** nie. Doplňme program tak, aby akceptoval obe slová **'ahoj'** aj **'hello'** (na páske bude buď **'ahoj'** alebo **'hello'**):

```
prog = '''
1 a a > 2
1 h h > 6
2 h h > 3
3 o o > 4
4 j j > 5
5 _ _ = stop
6 e e > 7
7 l l > 8
8 l l > 9
9 o o > 5
'''

t = Turing(prog, 'hello')
print(t.rob())
```

Podobný tomuto je program, ktorý akceptuje vstup len vtedy, keď sa skladá z ľubovoľného počtu dvojíc znakov **'ok'**:

```
prog = '''
1 o o > 2
1 _ _ = end
2 k k > 1
'''

t = Turing(prog, 'ok' * 100)
print(t.rob())
```

Zostavme ešte takýto program:

- predpokladáme, že na páske je postupnosť **0** a **1**, ktorá reprezentuje nejaké dvojkové číslo, napríklad **'1011'** označuje desiatkové číslo **11**
- čítacia hlava je na začiatku nastavená na prvej číslici
- program pripočíta k tomuto dvojkovému číslu **1**, t.j. v prípade **'1011'** bude výsledok na páske **'1100'** teda číslo **12**
- program bude postupovať takto:
 - najprv nájde koniec vstupného reťazca, teda prázdny znak **'_'** za poslednou cifrou (toto sa zabezpečuje v stave **'s1'**)
 - potom prechádza od konca a všetky **'1'** nahrádza **'0'**
 - keď pritom príde na **'0'**, túto nahradí **'1'** a skončí
 - ak sa v čísle nachádzajú iba **'1'** a žiadna **'0'**, tak namiesto prázdneho znaku **'_'** pred číslom dá **'1'** a skončí

Program pre turingov stroj:

```
(s1, 0) -> (0, >, s1)
(s1, 1) -> (1, >, s1)
(s1, _) -> (_, <, s2)
```

```
(s2, 1) -> (0, <, s2)
(s2, 0) -> (1, =, end)
(s2, _) -> (1, =, end)
```

Otestujeme naším programom:

```
prog = '''
s1 0 0 > s1
s1 1 1 > s1
s1 _ _ < s2

s2 1 0 < s2
s2 0 1 = end
s2 _ 1 = end
'''

t = Turing(prog, '1011')
print(t.rob())
```

po spustení:

```
1011
^ s1
1011
^ s1
1011
^ s1
1011
^ s1
1011_
^ s1
1011_
^ s2
1010_
^ s2
1000_
^ s2
1100_
^ end
True
```

Závěrečná ukážka demonštruje zložitejší Turingov stroj: zistí, či je vstup zložený len z písmen 'a' a 'b' a či je to palindrom:

```
palindrom = '''
s1 a _ > sa
s1 b _ > sb
s1 _ _ = end
sa a a > sa
sa b b > sa
sa _ _ < saa
saa a _ < s2
saa _ _ < end
sb a a > sb
sb b b > sb
sb _ _ < sbb
sbb b _ < s2
sbb _ _ < end
s2 a a < s2
s2 b b < s2
s2 _ _ > s1
'''

t = Turing(palindrom, 'aba')
```

```
print(t.rob())
```

```
aba
^ s1
_ba
^ sa
_ba
^ sa
_ba_
^ saa
_b_
^ s2
_b_
^ s2
_b_
^ s1
____
^ sb
____
^ sbb
____
^ end
True
```

Cvičenia

L.I.S.T.

- riešenia **aspoň 10 úloh** odovzdaj na úlohový server <https://list.fmph.uniba.sk/>
- pozri si **Riešenie úloh 22. cvičenia**

1. Pomocou programu `Turing` z prednášky otestuj tieto pravidlá:

```
2. (stav, x) -> (y, >, stav)
3. (stav, _) -> (z, =, end)
4. prog = '''
5.
6.
7. '''
8. t = Turing(prog, ...)
9. print(t.rob())
```

Zvoľ takú vstupnú pásku, aby Turingov stroj akceptoval vstup (metóda `rob()` vráti `True`). Otestuj s rôzne veľkými páskami.

2. Zisti, aké reťazce bude akceptovať nasledovný Turingov stroj:

```
3. start q q > jedna
4. jedna q q > dva
5. dva q q > start
6. jedna _ _ = stop
```

Nájdi aspoň 3 rôzne dlhé reťazce aspoň dĺžky 8, ktoré budú akceptované.

3. Napíš pravidlá (program) pre Turingov stroj, ktorý bude akceptovať len takú postupnosť symbolov na páske, ktorá obsahuje len písmená {a, b, c}. Otestuj s rôznymi vstupnými reťazcami, napríklad:

```
4. for retazec in 'abcb', 'cccca', 'cba'*10, ...:
5.     t = Turing(''
6.         ...program...
7.         '', retazec)
8.     print(retazec, t.rob(False))
```

4. Na vstupnej páske je postupnosť znakov 'x'. Napíš program, ktorý akceptuje tento vstup len, ak obsahuje nepárny počet týchto znakov. Napríklad:

```
5. >>> prog = ''
6.     ...program...
7.     ...
8. >>> print(Turing(prog, 'xxxxx').rob(False))
9. True
10. >>> print(Turing(prog, 'xx' * 5).rob(False))
11. False
```

5. Program z úlohy (4) oprav tak, aby akceptoval len reťazce zložené z písmen {x, y}, v ktorých je nepárny počet znakov 'x' (na počte 'y' by nemalo záležať).

```
6. >>> prog = ''
7.     ...program...
8.     ...
9. >>> print(Turing(prog, 'xxxxx').rob(False))
10. True
11. >>> print(Turing(prog, 'yyy').rob(False))
12. False
13. >>> print(Turing(prog, 'yx' * 5).rob(False))
14. True
```

6. Na vstupnej páske je reťazec zložený len zo znakov 'e'. Napíš program, ktorý akceptuje len reťazce z písmen 'e', pričom na ich koniec pripíše jeden zo znakov {0, 1, 2} a to podľa zvyšku po delení 3 počtu 'e'. Napríklad:

```
7. for pocet in 10, 9, 8, 7:
8.     t = Turing(''
9.         ...program...
10.         '', 'e' * pocet)
11.     print(t.rob(False), t.paska.text)
```

by mal vypísať:

```
True eeeeeeeee1
True eeeeeeeee0
True eeeeeeee2
True eeeeeee1
```

7. Otestuj Turingov stroj z prednášky, ktorý pripočítaval 1 k dvojkovému zápisu pre nejaké väčšie číslo, napríklad takto:

```

8. cislo = 555
9. retazec = f'{cislo:b}'
10. t = Turing(...) #zavolaj Turingov stroj s daným reťazcom
11. t.rob(False)
12. vysledok = t.paska.text
13. print(cislo, retazec, vysledok, int(vysledok, 2))

```

Uvedom si, že v premennej `retazec` je dvojkový zápis daného čísla `cislo` a v premennej `vysledok` by mal byť dvojkový zápis tohto čísla zväčšeného o 1.

Skontroluj tento program aj pre iné čísla, napríklad pre `2**20-1`

8. Ručne bez počítača preved' číslo `1000` do dvojkovej sústavy. Svoje riešenie skontroluj, napríklad pomocou:

```

9. >>> int('...dvojkový zápis...', 2)

```

9. Ručne zisti, aké číslo má dvojkový zápis `10101010`. Ručne preved' toto číslo aj do 16-ovej sústavy. Môžeš to skontrolovať, napríklad takto:

```

10. >>> int('10101010', 2)
11. >>> int('...šestnástkový zápis...', 16)

```

10. Navrhni Turingov stroj, ktorý zo vstupného reťazca `'matfyz'` vytvorí na páske slovo `'python'`. Pravidlá by mali mať jediný stav (okrem koncového). Otestuj, napríklad:

```

11. >>> t = Turing(prog, 'matfyz')
12. >>> t.rob(False)
13. True
14. >>> t.paska.text
15. 'python'

```

11. Uprav pravidlá Turingovho stroja zo (7) úlohy tak, aby na páske vzniklo dvojkové číslo o 1 menšie. Napríklad:

```

12. >>> t = Turing(prog, '1000')
13. >>> t.rob(False)
14. True
15. >>> t.paska.text
16. '111'

```

12. Navrhni Turingov stroj, ktorý číslo v dvojkovej sústave vynásobí dvoma a pripočíta 1. Napríklad:

```

13. >>> cislo = 27
14. >>> t = Turing(prog, f'{cislo:b}')
15. >>> t.rob(False)
16. True
17. >>> int(t.paska.text, 2)
18. 55

```

13. Na páske je číslo zapísané v desiatkovej sústave. Navrhni Turingov stroj, ktorý toto číslo zvýši o 1. Napríklad:

```

14. >>> cislo = 1299
15. >>> t = Turing(prog, str(cislo))
16. >>> t.rob(False)
17.      True
18. >>> t.paska.text
19.      '1300'

```

14. Navrhni Turingov stroj, ktorý predpokladá, že na páske je postupnosť znakov '1'. Po skončení práce stroja (metóda `rob()` vráti `True`) bude na páske celá táto postupnosť jednotiek skopírovaná za seba aj s prázdny symbolom, napríklad:

```

15. >>> t = Turing(prog, '1111')
16. >>> t.rob(False)
17.      True
18. >>> t.paska.text
19.      '1111_1111'

```

15. Navrhni Turingov stroj, ktorý predpokladá, že na páske je postupnosť znakov 'x'. Po skončení práce stroja (metóda `rob()` vráti `True`) bude na páske celá táto postupnosť iksov skrátená na polovicu, napríklad:

```

16. >>> t = Turing(prog, 'xxxx')
17. >>> t.rob(False)
18.      True
19. >>> t.paska.text
20.      'xx'
21. >>> t = Turing(prog, 'x' * 15)
22. >>> t.rob(False)
23.      True
24. >>> t.paska.text
25.      'xxxxxxx'

```

16. Čísla od 0 do 255 sú uložené v jednom bajte (8 bitov). Čísla do 65535 sú uložené v dvoch bajtoch (16 bitov). Napíš funkciu `bajty(cislo)`, ktorá z daného čísla vráti postupnosť bajtov. Zrejme posledný bajt vypočítaš ako `cislo % 256` a zvyšné bajty sú potom `cislo // 256` - toto budeš opakovať, kým nebude `cislo` nula. Napríklad:

```

17. >>> bajty(100)
18.      [100]
19. >>> bajty(1000)
20.      [3, 232]
21. >>> bajty(10000)
22.      [39, 16]
23. >>> bajty(2 ** 16 - 1)
24.      [255, 255]
25. >>> bajty(3 ** 33)
26.      [19, 191, 239, 166, 90, 187, 131]

```

17. Každý bajt (číslo od 0 do 255) sa dá zapísať jedným dvojčiferným číslom v 16-ovej sústave. Napíš funkciu `do_hex(post)`, ktorá postupnosť bajtov zo (16) úlohy vypíše v 16-ovej sústave. Výsledok porovnaj so šestnástkovým výpisom pôvodného čísla. Napríklad:

```

18. >>> do_hex(bajty(1000))
19.      03 e8
20. >>> f'{1000:04x}'

```

```
21. '03e8'
22. >>> do_hex(bajty(3 ** 33))
23. 13 bf ef a6 5a bb 83
24. >>> f'{3**33:x}'
25. '13bfefa65abb83'
```

12. Týždenný projekt

L.I.S.T.

- riešenie odovzdaj na úlohový server <https://list.fmph.uniba.sk/>

Zapíš metódy triedy `Turing` s týmito metódami:

```
class Turing:
    def __init__(self, program, obsah=''):
        self.program = {}
        ...

    def restart(self, stav=None, obsah=None, n=None):
        # od noveho stavu (ak nie je None), s novou páskou (ak nie je None) a zavola rob()
        return False, 0

    def rob(self, n=None):
        return False, 0

    def text(self):
        return ''
```

Tento Turingov stroj by sa mal správať veľmi podobne, ako ten z prednášky, líšiť sa bude len v niekoľkých detailoch:

- inicializácia `__init__()` má dva parametre `program` a počiatočný stav pásky, pričom `program` sa zadáva inak ako bolo v prednáške (uvádzame nižšie)
- metódy `restart()` a `rob()` majú posledný parameter `n`, ktorý, ak je zadáný, určuje maximálny počet vykonávaných pravidiel, ak by sa mal tento počet presiahnuť, výpočet končí tak, ako keby neakceptoval vstup (Turingov stroj vtedy vykoná maximálne len `n` pravidiel, ale `n+1`-pravidlo už nie)
- obe tieto metódy vrátia dvojicu: `True/False` a počet vykonaných pravidiel
- metóda `restart()` môže mať tieto ďalšie 2 parametre, ktorým môžeme nastaviť počiatočný stav, resp. zmeniť obsah pásky (pri zmene obsahu sa pozícia na páske nastaví na 0), táto metóda okrem nastavovania stavu a pásky zavolá metódu `rob()`
- množina koncových stavov je `{'end', 'stop'}`
- metóda `text()` vráti momentálny stav pásky, ktorý je očistený od úvodných a záverečných prázdnych znakov `'_'` - my ju budeme používať ako atribút - property
- v triede `Turing` môžeš dodefinovať ďalšie metódy aj atribúty

Formát zadávaného programu pre Turingov stroj

Doteraz sme definovali Turingov stroj ako zoznam pravidiel, pričom každé z nich bolo päticou:

```
stav1 znak1 znak2 smer stav2
```

Napríklad:

```
s1 0 0 > s1
```

```

s1 1 1 > s1
s1 _ _ < s2

s2 1 0 < s2
s2 0 1 = end
s2 _ 1 = end

```

Tento istý Turingov stroj môžeme definovať aj takouto tabuľkou:

	s1	s2
0	0 > s1	1=end
1	1 > s1	0 < s2
_	_ < s2	1=end

V tejto tabuľke sú v prvom riadku vymenované všetky stavy (okrem koncových), v prvom stĺpci sú všetky sledované symboly, pre ktoré existuje pravidlo. Každé vnútorné políčko tabuľky reprezentuje jedno pravidlo: **stav1** z príslušného stĺpca a **znak1** z príslušného riadka, samotné políčko obsahuje trojicu **znak2 smer stav2**. Ak pre nejakú dvojicu **stav1**, **znak1** neexistuje pravidlo, tak na príslušnom mieste tabuľky je namiesto trojice reťazcov jeden znak '.'.

Niekedy sa takáto tabuľka môže trochu sprehľadniť tým, že sa môžu vynechať časti **znak2**, resp. **stav2**, ak sa zhodujú so **znak1**, resp. **stav1**. Predchádzajúca tabuľka by mohla vyzeráť aj takto a popisovala by ten istý Turingov stroj:

	s1	s2
0	>	1=end
1	>	0<
_	<s2	1=end

Všimni si, že sme tu vynechali medzery v trojiciach vo vnútri tabuľky. Takúto tabuľku budeme do triedy **Turing** zadávať pri inicializácii, napríklad takto:

```

prog = '''
    s1    s2
0      >    1=end
1      >    0<
_      <s2   1=end
...
t = Turing(prog, '1011')

```

Uvedom si, že ak má Turingov stroj **n** stavov (okrem koncových), tak prvý riadok súboru obsahuje **n** reťazcov - názvov stavov (prvý z nich bude štartový), ktoré sú navzájom oddelené aspoň jednou medzerou. Každý ďalší riadok (podľa počtu rôznych rozlišovaných znakov) obsahuje presne **n+1** reťazcov navzájom oddelených aspoň jednou medzerou.

Ďalší príklad ukazuje tabuľku aj s políčkami bez pravidiel:

	s1	s2	s3	s4	s5
a	>s2
h	.	>s3	.	.	.

	s1	s2	s3	s4	s5
o	.	.	>s4	.	.
j	.	.	.	>s5	.
—	=end

Obmedzenia

- svoje riešenie odovzdaj v súbore `riesenie.py`, pričom sa v ňom bude nachádzať len jedna definícia triedy `Turing`
- atribút `program` v triede `Turing` bude obsahovať slovník (asociatívne pole) s pravidlami Turingovho stroja (vo formáte z prednášky)
- prvé tri riadky tohto súboru budú obsahovať:

```
# 12. zadanie: turing
# autor: Janko Hraško
# datum: 20.12.2021
```

- zrejme ako autora uvedieš svoje meno
- program by nemal počas testovania testovačom nič vypisovať (žiadne testovacie `print()`)

Testovanie

Keď budeš spúšťať svoje riešenie na svojom počítači, môžeš do súboru `riesenie.py` pridať testovacie riadky, ktoré však testovač nebude vidieť, napríklad takto:

```
if __name__ == '__main__':
    prog = '''
        s1    s2
    0    >    1=end
    1    >    0<
    —    <s2   1=end
    '''
    t = Turing(prog, '1011')
    print(t.program)
    print(t.rob())
    print('vysledok =', t.text)
    print(t.restart('s1', '10102010'))
```

Tento test by mal vypísať:

```
{('s2', '_'): ('1', '=', 'end'), ('s1', '_'): ('_', '<', 's2'), ('s2', '1'): ('0', '<', 's2'),
('s1', '0'): ('0', '>', 's1'), ('s1', '1'): ('1', '>', 's1'), ('s2', '0'): ('1', '=', 'end')}
(True, 8)
vysledok = 1100
(False, 4)
```

Projekt `riesenie.py` odovzdávaj na úlohový server <https://list.fmph.uniba.sk/> najneskôr **7. januára 2022** do 23:00. Môžeš zaň získať **5 bodov**.