Lehký příklad: Sekvence (6 bodů)

Napište program `sequence.py`, který v zadané matici celých čísel najde nejdelší souvislou část řádku, nebo sloupce skládající se z nezáporných sudých čísel.

Vstup jméno souboru je první argument příkazové řádky. Soubor obsahuje vstupní matici zadanou celými čísly, oddělenými jednou mezerou.

Výstup Písmeno 'r' nebo 's' a tři celá čísla oddělená mezerou nebo klíčové slovo `NEEXISTUJE`, pokud v matici není žádná nezáporná sudá posloupnost. Písmeno 'r' určuje, že nejdelší posloupnost je součástí řádku, písmeno 's' určuje, že nejdelší posloupnost je součástí sloupce. První dvě čísla označují souřadnici začátku posloupnosti v matici v pořadí `řádek sloupec`, třetí pak její délku.

Pozn:

- Posloupnost nabývá délky 1 a více
- Jako počátek definujeme prvek sekvence s nejmenším indexem řádku (pro sloupcové sekvence), nebo sloupce (pro řádkové sekvence)
- Číslování řádků a sloupců začíná od 0
- Nejdelší posloupnost je vždy jednoznačně určena.
- Vstupní matice je vždy zadaná správně, tj. obsahuje pouze celá čísla oddělená jednou mezerou a všechny řádky mají stejný počet prvků
- Vstupní matice má vždy velikost alespoň 1×1

Bodování

Popis části	**Počet testů**	**Timeout**	**Max. bodů**	**Bodování**
Čtvercové matice (posl. existuje)	10	2 s/test	**2**	2b pokud všechny správně
Obecné matice (posl. existuje)	10	2 s/test	**2**	2b pokud všechny správně
Obecné matice včetně těch bez posloupnosti	10	2 s/test	**2**	2b pokud všechny správně

Při řešení můžete použít libovolné funkce jazyka Python, včetně standardních knihoven dostupných v systému Brute.

Testovací data

Příklad I Volání `python3 sequence.py mat1.txt`
Obsah souboru `mat1.txt`

```
1 2 -2  4
3 0  5 -2
1 1 -1 -3
```

Výstup

```
s 0 1 2
```

Komentář Matice obsahuje dvě řádkové jednoprvkové sekvence nezáporných sudých čísel a jednu jednoprvkovou a jednu dvouprvkovou sloupcovou sekvenci. Nejdelší z nich je ta dvouprvková sloupcová, která začíná číslicí 2 na prvním řádku (index 0) a druhém sloupci (index 1).

Příklad II Volání `python3 sequence.py mat2.txt`
Obsah souboru `mat2.txt`

```
-1 -2 -1 0
 2  0  4 0
-1 -1 -3 1
```

Výstup

```
r 1 0 4
```

Komentář Matice kromě jednoprvkových sekvencí obsahuje dvouprvkovou sloupcovou sekvenci a 4-prvkovou řádkovou sekvenci, která je nejdelší – od pozice `1 0`.

Příklad III Volání `python3 sequence.py mat3.txt`
Obsah souboru `mat3.txt`

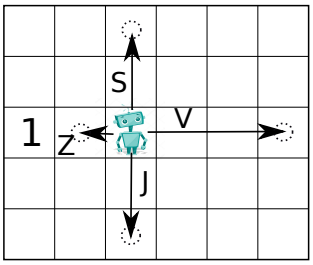
```
-1 -2 -2 -3
-2 -6 -3 -9
-5 -5 -6 -4
-8 -9 -1 -3
```

Výstup

```
NEEXISTUJE
```

Komentář Matice obsahuje pouze záporná čísla, tj. neexistuje žádná nezáporná sekvence.

Těžký příklad: Robot v kluzišti (14 bodů)



Napište program `maze.py`, který najde posloupnost akcí pro robota v kluzišti. Pokud se vykoná jedna akce, robot se vydá zadaným směrem, zastaví se pouze buď o překážku nebo o konec bludiště (okraj matice).

Vstup Jméno souboru je první argument příkazové řádky. Tento soubor obsahuje definici bludiště pomocí celočíselné matice $m \times n$ s prvky:

- `0` je prádné políčko,
- `1` je překážka,
- `2` je startovní pozice - prázdné políčko
- `4` je cílová pozice - prázdné políčko

Výstup Program vypíše na standardní výstup

- Nejkratší posloupnost pohybů, které určují cestu ze startovní pozice do cílové pozice. Posloupnost sestává z písmen `S, J, V, Z` podle světových stran a každý prvek určuje jednu akci k dosažení cílové pozice:
  - `S` – pohyb po sloupci do řádku s indexem 0, nebo před první překážku
  - `J` – pohyb po sloupci do řádku s indexem $m - 1$, nebo před první překážku
  - `Z` – pohyb po řádku do sloupce s indexem 0, nebo před první překážku
  - `V` – pohyb po řádku do sloupce s indexem $n - 1$, nebo před první překážku
- `NEEXISTUJE`, pokud taková sekvence akcí neexistuje

Při pohybu se robot nemůže zastavit uprostřed kluziště, může se zastavit pouze o okraj, nebo o překážku - v tom případě skončí pohyb na volném políčku před překážkou ve směru jeho pohybu.

- Správnost vstupů není potřeba testovat. Matice obsahuje pouze čísla 0,1,2,4 oddělené mezerou a má rozměry minimálně 2×2. Všechny řádky mají stejný počet prvků.
- Vypište jednu z možných nejkratších cest.
- Výslednou posloupnost vypisujte bez mezer mezi jednotlivými prvky.

Bodování

Popis části	Počet testů	Timeout	Max. bodů	Hodnocení
Úlohy s řešením do 3 akcí	8	4 s/test	**4**	0.5b za každý správný
Úlohy s řešením do 15 akcí	12	5 s/test	**6**	6b pokud všechny správně
Úlohy i bez řešení	10	10 s/test	**4**	4b pokud všechny správně

Při řešení můžete použít libovolné funkce jazyka Python pro zpracování řetězců, včetně knihoven.

Testovací data

Příklad I Při volání `python3 maze.py maze1.txt` a obsahu soboru `maze1.txt`

```
2 0 4 0
0 0 0 0
0 1 0 0
```

Je nejkratší řešení:

```
VJZS
```

Vizualizace řešení je následující. Po každém tahu (v závorce) se naše aktuální pozice (vyznačená jako `R`) posune určeným směrem. Políčka projetá posledním tahem jsou označena tečkou (`.`)

```
R 0 4 0  (V)  . . . R   (J)  2 0 4 .  (Z)   2 0 4 0  (S)  2 0 R 0
0 0 0 0       0 0 0 0        0 0 0 .        0 0 0 0       0 0 . 0
0 1 0 0       0 1 0 0        0 1 0 R        0 1 R .       0 1 . 0
```

Příklad II Při volání `python3 maze.py maze2.txt` a obsahu soboru `maze2.txt`

```
0 0 0 0
2 0 4 1
0 0 0 0
```

Je očekávaný výstup:

```
V
```

Cílovou pozici lze dosáhnout jedním tahem.

Příklad III Při volání `python3 maze.py maze3.txt` a obsahu soboru `maze3.txt`

```
2 0 0 0 0
0 1 1 1 0
0 1 4 0 0
0 0 0 1 0
```

Je očekávaný výstup:

```
JVS
```

Příklad IV Při volání `python3 maze.py maze4.txt` a obsahu soboru `maze4.txt`

```
0 0 0 1
2 0 4 0
0 0 0 0
```

Je očekávaný výstup:

```
NEEXISTUJE
```

Na cílovém místě není jak zastavit.