ALGORITMI 2024/2025
DOMAČA NALOGA 1

Časovna zahtevnost

1. naloga (1 točka)

Določi časovno zahtevnost algoritma:

```
funkcija t1(n: int) -> int {
    s: int = 0
    for i: int = 0; i < n; i++ {
        s += i + 1
    }
    vrni s
}
```

2. naloga (1 točka)

Določi časovno zahtevnost algoritma:

```
funkcija t2(n: int) -> int {
    s: int = 0
    for i: int = 0; i < n; i++ {
        for j: int = 1; j < i; j *= 2 {
            s += t1(i)  // t1 je funkcija iz prejšnje naloge
        }
    }
    vrni s
}
```

3. naloga (2 točki)

Določi časovno zahtevnost algoritma:

```
funkcija cantor(n: int, a: float, d: float) {
    if n < 1 {
        return d - a
    }
    b: float = 2 * a / 3 + d / 3
    c: float = a / 3 + 2 * d / 3

    vrni cantor(n / 2, a, b) + cantor(n / 2, c, d)
}
```

Rodovne funkcije

4. naloga (4 točke)

Andrej bi rad vedel, koliko je načinov, da z evrskimi kovanci sestavimo znesek 13 centov. Želeno število kombinacij izračunaj s pomočjo rodovnih funkcij.

Petra pa zanima na koliko načinov lahko sestavimo isti znesek samo s kovanci za 1 in dva centa. Tudi njemu pomagaj izračunati želeno število kombinacij s pomočjo rodovnih funkcij.

5. naloga (3 točke)

Zapiši rodovno funkcijo za trikotniška števila (https://en.wikipedia.org/wiki/Triangular_number). S pomočjo dobljenega rezultata poišči stoto trikotniško število.

6. naloga (4 točke)

S pomočjo rodovnih funkcij pokaži, da lahko vsako pozitivno celo število zapišemo na natanko en način kot vsoto različnih potenc števila 2.

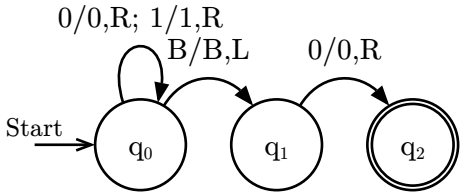
Turingovi stroji

Turingov stroj zapišemo kot sedmerico lastnosti:

- Q je končna množica stanj
- Γ je končna abeceda traku
- $b \in \Gamma$ je prazen simbol in edini, ki se lahko neskončnokrat pojavi na traku
- $\Sigma \subseteq \Gamma \setminus \{b\}$ je abeceda vhodnih simbolov
- $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ je prehodna funkcija
- q_0 je začetno stanje
- $F \subseteq Q$ so končna stanja

Vsak turingov stroj sme uporabiti le en trak. Turingov stroj je lahko deterministične ali nedeterminističen. Na začetku izvajanja programa je glava turingovega stroja vedno nad najbolj levim znakom vhodne besede.

Pri vsakem turingovem stroju v nekaj povedih zapiši, kako deluje (kakšna je ideja algoritma). Prehodna funkcija turingovega stroja naj bo narisana v grafični obliki. Primer turingovega stroja, ki preveri, če je podano binarno število sodo:



Vsak prehod je oblike $\mathbf{a/b,\ c}$, kjer a predstavlja znak pod bralno-pisalno glavo ob branju, b znak po pisanju pod bralno-pisalno glavo, c pa premik glave v levo oziroma desno smer. Stanje q_2 je dvojno obkroženo, kar predstavlja končno stanje (binarno število je sodo). Če prehod ni definiran, pomeni, da vhodna beseda ni sprejeta (binarno število je liho).

7. naloga (1 točka)

Zapiši turingov stroj, ki preveri, če je vhodna beseda palindrom. Abeceda vhodnih simbolov naj bo $\Sigma = \{A, C, G, T\}$. Primer vhodne besede `ACCGTTGCCA`:

...	B	B	A	C	C	G	T	T	G	C	C	A	B	B	...
			↑												
			q_0												

8. naloga (2 točki)

Zapiši Turingov stroj, ki za dani niz oklepajev in zaklepajev preveri, če so pravilno gezdeni (t.j. vsak oklepaj se zaključi z zaklepajem in vsak zaklepaj zaključi oklepaj). Abeceda vhodnih simbolov naj bo $\Sigma = \{(,)\}$. Primer vhodne besede `(((()))())()`:

...	B	B	(	(	(	)	(	)	)	)	(	)	)	(	)	B	B	...
			↑															
			q_0															

9. naloga (3 točke)

Zapiši turingov stroj, ki preveri, če je podan vhod oblike ww, kjer je $w \in (0, 1)^+$ (neprazen niz ničel in enic). Torej nizi, ki so konkatenacija dveh enakih binarnih podnizov. Abeceda vhodnih simbolov naj bo $\Sigma = \{0, 1\}$. Primer vhodne besede `0100110101001101`:

...	B	B	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	B	B	...
			↑																		
			q_0																		

Drevesa

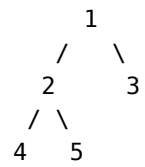
10. naloga (4 točk)

Podan je razred `Node`, ki predstavlja binarno drevo (v `Pythonu` in `Javi`):

```python
# zig_zag.py:
class BinaryTree:
    def __init__(value):
        self.value = value
        self.left = None
        self.right = None

class ZigZag:
    def __init__(root):
        self.tree = root

    def show():
        # TODO
```

```java
// Node.java:
public class Node {
    int value;
    Node left, right;

    Node(int x) {
        value = x;
        left = null;
        right = null;
    }
}

// ZigZag.java:
class ZigZag {
    Node tree;

    public ZigZag(Node root) {
        tree = root;
    }

    public void show() {
      // TODO
    }
}
```

Podan je tudi razred `ZigZag`. V njem implementiraj metodo `show`, ki izpiše cik-cak sprehod nad podanim drevesom (vrednosti vozlišč, ki so ločene z `", "`). Cik-cak sprehod je sprehod po drevesu, kjer se po nivojih izmenično sprehajamo levo in desno. Za dano drevo,



bi bil izpis za cik-cak sprehod sledeč:

```
1, 3, 2, 4, 5
```

Oddaj `zig_zag.py` oziroma `ZigZag.java`.