Cvičení z Algoritmizace a Programování 1

10. cvičení

Algoritmizace

Rekurze

(Klasická) rekurze

- Řešení skládám z řešení podproblémů
- Rekurzivní volání používám k vyřešení podproblémů

- Ukončovací podmínka: problém je malý (triviální)
  - už nejde rozdělit
  - umím ho vyřešit jinak

- Příklady úloh
  - n-té Fibonacciho číslo, faktoriál
  - výška binárního stromu
  - symetričnost bin. stromu
  - vyhodnocování aritmetických výrazů

Rekurzivní generování („najděte všechna řešení“)

- Postupně tvořím 1 řešení
- Rekurzivní volání používám k prozkoumání všech možností (větvení)

- Ukončovací podmínka: řešení je hotové

- Příklady úloh
  - generování (všech) variací
  - zaplacení sumy mincemi

```python
def fibonacci(n):
    if n <= 1:
        return 0
    else:
        n_1 = fibonacci(n-1)
        n_2 = fibonacci(n-2)
        return n_1 + n_2
```

```python
def height(node):
    if node is None:
        return -1 # trik
    else:
        h_left_subtree = height(node.left)
        h_right_subtree = height(node.right)
        return 1 + max(h_left_subtree, h_right_subtree)
```

```python
def variace_s_opakovanim(n, k):
    """Vytiskne seznam vsech variaci k-te tridy z {1...n}"""


    def _variace(i):
        """nageneruj [i:k]"""
        for j in range(1, n+1):
            v[i] = j
            if i < k-1:
                _variace(i+1) # dogeneruj [i+1:k]
            else:
                print(v)


    v = [0] * k
    _variace(0)
```

Řešení DÚ Zaplacení sumy mincemi

Krabičky

- http://atest.geometry.cz/
- Přihlašte se jako host
- Vyřešte úlohy:
  1. „Zkuste si“ > „Ovládání (algoritmizace)“
  2. „Těžší“ > „První (nv)“
  - Tlačítko „Odeslat zobrazené řešení“ provede vyhodnocení

- Jak (programem) najít řešení?
- Jak (programem) najít všechny hodnoty, které lze vytvořit ze zadaných krabiček (čísel + operátorů)?

Domácí úloha - Krabičky

- https://recodex.mff.cuni.cz/app/assignment/256244e7-e844-4a91-897b-2cd4445c4603
- Do 5. 1. 2023
  - … ale s ohledem na zápočet doporučuji odevzdat dříve

Programování

Slovník – `dict`, `{}`

- Dvojice klíč-hodnota
- Rychlé operace (průměrně O(1))
  - Vložení `slovnik[klic] = hodnota`
  - Získání hodnoty `ulozena_hodnota = slovnik[klic]`
  - Je klíč ve slovníku? `klic in slovnik`
- Klíč musí být hashovatelný
  - ANO: číslo, string, tuple
  - NE: list, dict, set
  - vlastní třída: každá instance je unikátní
    - Lze změnit (metody `__hash__` a `__eq__`)
- Položky nejsou uspořádané!

Práce se soubory

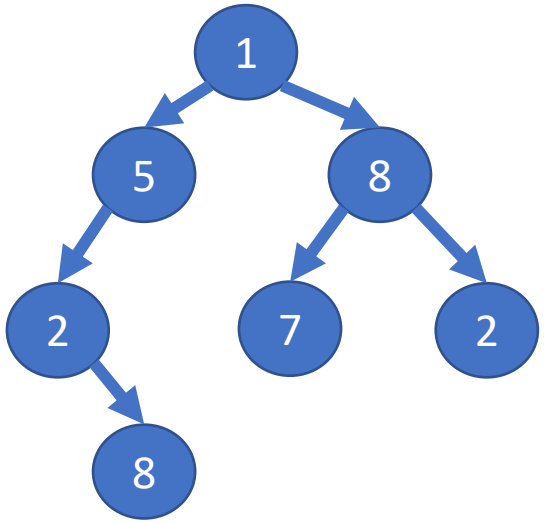
- Viz Files.py

Načítání stromu ze souboru – nestihlo se

- Soubor na webu (strom.txt)
- Formát:
  - Každý vrchol má unikátní identifikátor (typu string) a hodnotu (int)
  - 1. řádek: identifikátor kořene
  - Další řádky:
    - 1 řádek = 1 vrchol stromu, v libovolném pořadí!
    - Formát řádku:
      - `id -> hodnota, id_levého_dítěte, id_pravého_dítěte`
      - id dětí mohou být prázdná (= vrchol dané dítě nemá)
        - Např. `n1 -> 5, ,n2` – vrchol „n1“ má hodnotu 5, levé dítě nemá, jeho pravé dítě je vrchol „n2“

Binární stromy – nestihlo se

- Maximum
  - Rekurzivně
  - Nerekurzivně
  - „Nejmělčí“ (nejbližší kořeni)



Domácí úkol – Jeden proti stu

- https://recodex.mff.cuni.cz/app/assignment/bc3feaff-f9f5-4673-928c-3ff21465fadb