EADŠ - cvičenie 7 - opakovanie

3. novembra 2022

Zložitosti - prehľad

$$f(n) \in O(g(n)) \iff \exists n_0, c; \forall n > n_0 : f(n) \leq c \cdot g(n)$$

Trieda	Intuitívne	Limita
$f(n) \in O(g(n))$	$f(n) \leq g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$
$f(n) \in \Omega(g(n))$	$f(n) \geq g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$
$f(n) \in \Theta(g(n))$	$f(n) \approx g(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0, \infty$

Určovanie zložitosti - prehľad

- zanedbávame konštanty:
  $O(c \cdot f(n)) = O(f(n))$
  (napr. $O(3n) = O(n)$)
- berieme najväčšiu funkciu
  $O(f(n) + g(n)) = O(\max(f(n), g(n)))$
  (napr. $O(n^5 + n^2 + n) = O(n^5)$)
- poradie zložitostí:
  $\forall a > 1 : O(1) \in O(\log^a n) \in O(n^a) \in O(a^n) \in O(n!) \in O(n^n)$
  Intuitívne:
  $\forall a > 1 : 1 \leq \log^a n \leq n^a \leq a^n \leq n! \leq n^n$

Asymptotická analýza - príklad

```
arr = [...]
l, r = 0, n

while r >= l:
    mid1 = l + (r-l) // 3
    mid2 = r - (r-l) // 3

    if key < arr[mid1]:
        r = mid1 - 1
    elif key > arr[mid2]:
        l = mid2 + 1
    else:
        l = mid1 + 1
        r = mid2 - 1
```

Asymptotická analýza - príklad

$f(n)$	$g(n)$	O	Θ	Ω
$1,0001^n$	n^{200}	✓/✗	✓/✗	✓/✗
$2^{log_2(n)}$	n^2	✓/✗	✓/✗	✓/✗
$9\,651\,600\,000$	$log_2\ \sqrt[200]{n}$	✓/✗	✓/✗	✓/✗

PQ a halda - prehľad

Priority queue

- čierna škatuľka (abstraktný dátový typ) s operáciami:
  - vyber minimum
  - vlož prvok x

Halda

- konkrétna implementácia dátovej štruktúry *priority queue*
- úplný binárny strom
- minimum v koreni
- rodič má menší kľúč (menšiu prioritu) ako potomok
- vieme vkladať prvky a vybrať minimum

Halda - príklad

Nakreslite, ako sa postupne bude vyvíjať minimová halda:

1. vlož prvky `1, 7, 2, 5, 4, 0, 8`
2. 3 krát vyber minimum
3. vlož prvky `6, 1, 3`

Triedenie - prehľad

- zoraďujeme prvky (od najmenšieho po najväčšie)
- ako o jednom z mála problémov v informatike vieme dokázať, dolný odhad zložitosti
- často sa využíva pri prístupoch ako je "zametanie"
- Algoritmy:
  - pomalé - $O(n^2)$ (min sort, insertion sort, ...)
  - rýchle - $O(n \log n)$ (quicksort, merge sort, heapsort, ...)
  - špeciálne - $O(n)$ (counting sort radix sort, bucket sort, ...)

Slovník (`dict`) - opakovanie

Abstraktná dátová štruktúra s operáciami:

- `insert(key, value)`
- `search(key)`
- `delete(key)`

Rôzne implementácie:

- neutriedené pole
- utriedené pole
- linked list
- hash tabuľka
- binárny vyhľadávací strom

Hashovanie - prehľad

1. Chceme objekty uložiť do poľa tak, aby vyhľadávanie bolo rýchle.
2. h(x) je funkcia, ktorá priradí každému objektu prirodzené číslo $0, ..., m-1 =$ **kľúč**.
3. h(x) musí byť deterministická
4. Kolízia : ak h(x) = h(y) ale x $\neq$ y

Druhy:

- hashovanie s uzavretou adresáciou (kýbliky / spájané zoznamy)
- hashovanie s otvorenou adresáciou (skúšame iné indexy)
  - lineárne
  - kvadratické
  - dvojité

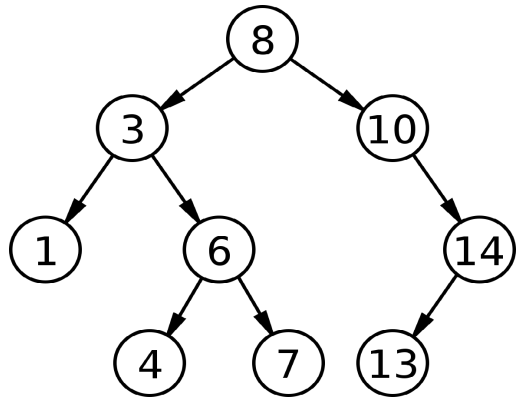
Hashovanie - príklad

Majme funkciu $h(x) = (3x + 7) mod\ 11$ a hodnoty $1, 7, 11, 0, 8, 12$. Hashovacia tabuľka má veľkosť 11.

- napíšte ako bude vyzerať naša hashovacia tabuľka, ak použijeme uzavreté adresovanie.
- napíšte ako bude vyzerať naša hashovacia tabuľka, ak použijeme otvorené adresovanie a pri kolízii na i-tom indexe skúsime index i+1.

BST- prehľad

- stromová dátová štruktúra
- nie nutne *úplný* binárny strom
- ľavý syn je *menší* ako rodič, pravý syn je *väčší* ako rodič



Dobré zložitosti iba ak je vyvážený. Algoritmy na vyvažovanie:

- AVL
- scapegoat
- red-black tree
- treap
- ...

Stromy a pre/in/post order

Spôsoby prechádzania stromu.

- `preorder` - najprv koreň, potom ľavý podstrom, potom pravý podstrom
- `inorder` - najprv ľavý podstrom, potom koreň, potom pravý podstrom
- `postorder` - najprv ľavý podstrom, potom pravý podstrom, potom koreň

1. preorder

```
def preorder(v):
  print(v.value)
  preorder(v.left)
  preorder(v.right)
```

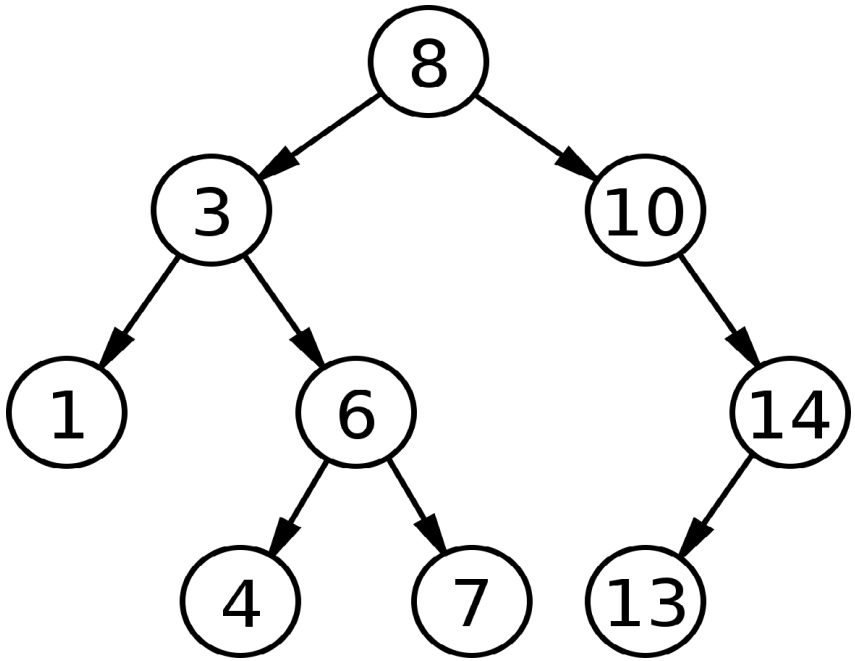
2. inorder

```
def inorder(v):
  preorder(v.left)
  print(v.value)
  preorder(v.right)
```

3. postorder

```
def postorder(v):
  preorder(v.left)
  preorder(v.right)
  print(v.value)
```

BST, pre/in/post order - príklad



Vypíšte poradie, v akom navštívime vrcholy, ak ich budeme prechádzať v pre/in/post order poradí.

Trie - príklad

Nakreslite komprimovanú trie pre slová
dog, dot, pump, fair, first

Vyhľadávanie v texte - prehľad

- naivne - v čase $O(nm)$ - skúšanie každého písmena ako potenciálny začiatok výskytu
- Rabin-Karp - očakávane v čase $O(n+m)$ - pomocou rolling hash, potom porovnanie nie je v $O(m)$ ale v $O(1)$, celé podslovo porovnávame len ak sa hash zhoduje
- KMP - v čase $O(n+m)$ - predpočítanie automatu a prechádzanie hlavného textu. Posúvame (a pamätáme si) iba "aktívny stav v automate".

Rabin-Karp - príklad

Ukážte, že ak zvolíme nasledujúce funkcie ako hash funkcie pre vytváranie rolling hashu, tak ich vieme nasledujúci hash vytvoriť v čase O(1).

- $\Sigma = \{a, b\}$ a $h(X) =$ počet výskytov znaku a
- $\Sigma = \{0, 1, 2, \ldots 9\}$ a
  $h(X) = x_0 - x_1 + x_2 - x_3 + \cdots + x_{m-2} - x_{m-1}$

Konštrukcia KMP automatu - príklad

Vytvorte KMP automat pre slovo EELEELE.

Suffix tree - príklad

Dostaneme string S. Chceme zistiť, aký je v ňom najdlhší opakujúci sa podreťazec.