

Binárne vyhľadávanie

Vyhľadávanie v utriedenom poli možno, oproti lineárnemu vyhľadávaniu, zefektívniť (zrýchliť). Keďže pole je utriedené, ak porovnáme hľadanú hodnotu s hodnotou v strede poľa, hneď môžeme vylúčiť polovicu prvkov poľa z prehľadávania (pole sme rozdelili na dve časti, preto názov binárne). Môžu nastať tri prípady. Po prvé, hodnota v strede poľa sa rovná hľadanej hodnote, môžeme skončiť. Po druhé, ak je hodnota v strede poľa väčšia ako hľadaná hodnota, hľadaná hodnota, ak je vôbec v poli, sa musí vyskytovať v dolnej polovici poľa. Po tretie, ak je hodnota v strede poľa menšia ako hľadaná hodnota, hľadaná hodnota, ak je vôbec v poli, sa musí vyskytovať v hornej polovici poľa a následne stačí tú prehľadať. Opísanú myšlienku delenia poľa môžeme uplatniť aj na dolnú alebo hornú polovicu poľa, následne štvrtinu, osminu,... Delenie úsekov končí nájdením hľadanej hodnoty alebo zistením, že úsek na prehľadávanie „má nulovú veľkosť“.

B.1 Vytvorte a použite funkciu, ktorá efektívne zistí, či sa zadaná hodnota nachádza v utriedenom poli.

Funkciu nazveme `vratBinVyskyt()` a bude vracieť jednu z hodnôt `False` alebo `True`.

```
import pole, vratcislo

def vratBinVyskyt(pole, hladat):
    dh = 0 # dolná hranica skúmanej oblasti
    hh = len(pole)-1 # horná hranica skúmanej oblasti
    while dh <= hh:
        stred = (dh+hh)//2
        if pole[stred] == hladat:
            return True
        if pole[stred]<hladat:
            dh = stred + 1
        else:
            hh = stred - 1
    return False
```

Použitie:

```
poleInt = pole.vytvorIntNahodne(0,100)
poleInt.sort()
print(poleInt)
# testovací cyklus
while True:
    hladat = vratcislo.cele("Hľadať hodnotu: ")
    if vratBinVyskyt(poleInt, hladat):
        print("{} sa v poli nachádza.".format(hladat))
    else:
        print("{} sa v poli nenachádza.".format(hladat))
```

B.2 Vytvorte a použite funkciu na efektívne zistenie miesta výskytu zadanej hodnoty v utriedenom poli.

Funkciu nazveme `vratBinMiesto()`. Pokiaľ sa hľadaná hodnota v poli vyskytuje, nemáme problém s návratovou hodnotou, je ňou zrejme hodnota premennej `stred`; ak sa hľadaná hodnota v poli nevyskytuje, nech sa vyvezie hodnota `-1`. Algoritmus je prakticky totožný s algoritmom z úlohy B.1.

```
def vratBinMiesto(pole, hladat):
    dh = 0
    hh = len(pole)-1
    while dh <= hh:
        stred = (dh+hh)//2
        if pole[stred] == hladat:
            return stred
```

```

if pole[stred]<hladat:
    dh = stred + 1
else:
    hh = stred - 1
return -1

```

Použitie:

```

poleInt = pole.vytvorIntNahodne(0,100)
poleInt.sort()
print(poleInt)
# testovací cyklus
while True:
    hladat = vratcislo.cele("Hľadať hodnotu: ")
    index = vratBinMiesto(poleInt, hladat)
    if index >= 0:
        print("{} sa v poli nachádza na indexe {}".format(hladat, index))
    else:
        print("{} sa v poli nenachádza (index: {})".format(hladat, index))

```

B.3 Vytvorte a použite funkciu na efektívne zistenie, či sa v utriedenom poli vyskytuje zadaná hodnota, a ak nie, nech ju vloží do poľa na také miesto, aby pole zostalo utriedené.

Funkciu nazveme `binVlozit()` a bude na úrovni príkazu - buď zmení (doplní) parametrické pole alebo nezmení.

Algoritmus vo funkcii `binVlozit()` sa skladá z dvoch častí. V prvej časti funkcia zisťuje, či sa `hladat` nachádza v poli (algoritmus totožný s predchádzajúcou funkciou). Hodnoty premennej `stred` sa čoraz viac blížia k miestu, kde by sa mala v utriedenom poli nachádzať hodnota `hladat`. Vypočíta stred medzi indexami `dh` a `hh` a ak hodnota `pole[stred]` nie je hľadanou hodnotou, vie rozhodnúť, keďže pole je utriedené, či má pokračovať v hľadaní v časti s hodnotami `pole[dh]` až `pole[stred-1]` alebo `pole[stred+1]` až `pole[hh]`. Približovanie končí nájdením hľadanej hodnoty alebo ak už nie je čo prehľadávať, t.j. `dh > hh`.

Po skončení prvej časti algoritmu si treba uvedomiť, že ak sa hľadaná hodnota v poli našla, v premennej `stred` je jej pozícia v poli a ak nie, čo je pre nás dôležitejšie, premenná `stred` „ukazuje“ priamo na miesto alebo tesne pred miesto, kde má byť hľadaná hodnota vložená (keďže v prvej časti sa k tomuto miestu hodnoty premennej `stred` blížia). Napríklad pri každom poli, kde hodnota `hladat` je najväčšia, sa `stred` zastaví na poslednom prvku poľa, t.j. pred miestom, kde treba vložiť hľadanú hodnotu. Ak teda `stred` ukazuje pred miesto, kde má byť `hladat` vložená, treba `stred` posunúť o jednu pozíciu doprava (zabezpečí príkaz `if`). Potom už len stačí vložiť hodnotu `hladat` do poľa na index `stred` a to príkazom `zoznam.insert(kam_vložit, čo_vložit)`, ktorý zároveň zabezpečí posunutie všetkých ostatných prvkov poľa od vloženého o jednu pozíciu doprava.

```

def binVlozit(pole, hladat):
# prvá časť
    dh = 0
    hh = len(pole) - 1
    while dh <= hh:
        stred = (dh + hh)//2
        if pole[stred] == hladat:
            break
        if pole[stred] < hladat:
            dh = stred + 1
        else:
            hh = stred - 1
# druhá časť
    if dh > hh:
        if pole[stred] < hladat:
            stred += 1
        pole.insert(stred, hladat)

```

```
# druhá časť - riešenie študentky Natálie Holkovej (je zrejmé, že aj dh a hh sa približujú k hľadanému miestu)
if dh > hh:
    pole.insert(dh, hladat)
```

Použitie:

```
poleInt = pole.vytvorIntNahodne(0,100)
poleInt.sort()
print(poleInt)
#testovací cyklus
while True:
    hladat = vratcislo.cele("Hľadať hodnotu: ")
    binVlozit(poleInt, hladat)
    pole.vypis(poleInt)
```

Úlohy

- Funkciu binVlozit() zadania B.3 upravte tak, aby vložila do poľa aj v poli sa vyskytujúcu hodnotu, a to tak, aby pole zostalo utriedené.
- Funkcie binárneho vyhľadávania otestujte aj pre polia (zoznamy) obsahujúce reťazce, napríklad mená, a aj pre polia obsahujúce reálne čísla. Nezabudnite, že testovať môžete len na utriedených zoznamoch!

Poznámka:

Python má modul bisect, ktorý obsahuje funkcie na prácu s utriedenou skupinou dát. Nebudeme sa ním podrobnejšie zaoberať (môžete pozrieť na internete), uvádzame len „našu“ funkciu binVlozit() zrealizovanú cez funkciu insert_left() modulu bisect. Príkaz bisect.insert_left(pole, hladat) by mohol byť nahradený aj príkazom pole.insert(bisect.bisect_left(pole, hladat, 0, len(pole)), hladat), v ktorom sme uviedli všetky parametre (dolnú a hornú hranicu prehľadávanej oblasti, t.j. 0 a len(pole), by sa mohli vynechať).

```
import bisect
```

```
def binVlozitFu(pole, hladat):
    if hladat not in pole:
        bisect.insort_left(pole, hladat)
        #pole.insert(bisect.bisect_left(pole, hladat, 0, len(pole)), hladat)
```

Poznámka

Porovnanie efektívnosti vyhľadávania v neutriedenom a utriedenom poli nájdete v študijnom texte Správnosť a výpočtová zložitosť algoritmu a programu.