Eksperimentalno ovrednotenje zahtevnosti

Napišite program v programskem jeziku Java za empirično primerjavo dveh algoritmov za iskanje elementa v urejeni tabeli:

- navadnega zaporednega iskanja in
- dvojiškega iskanja.

Oba algoritma poženite večkrat za različne velikosti tabele. Pri tem izmerite in izračunajte povprečni čas iskanja. Izpišite čase za oba algoritma. Predlagamo, da izziv rešujete postopoma po naslednjih točkah.

a) Generiranje testnih primerov

Za generiranje testnih primerov napišite metodo, ki vam za podani *n* vrne (urejeno) tabelo celih števil z vrednostmi od 1 do n. Npr.

- `int[] generateTable(int n)`

b) Implementacija obeh algoritmov iskanja elementa

Napišite oba algoritma za iskanje elementa. Npr.

- `int findLinear(int[] a, int v)`
- `int findBinary(int[] a, int l, int r, int v)`

Pri tem je *a* tabela elementov, *v* iskana vrednost, *l* leva meja v tabeli in *r* desna meja v tabeli, vrnemo pa mesto (indeks), kjer se element *v* nahaja. Kakšno vrednost imata *l* in *r* ob prvem klicu *findBinary(...)*?

c) Izvedba enega testa za tabelo dolžine *n*

Napišite metodi (za vsak način iskanja svojo metodo), ki izmerita povprečni čas iskanja v tabeli dolžine *n*. Npr.

- `long timeLinear(int n)`
- `long timeBinary(int n)`

Vsaka izmed metod naj izvede naslednje:

- Ustvari tabelo dolžine *n* z metodo, ki ste jo implementirali predhodno.
- Začne meriti čas.
- Nato 1000-krat ponovi naslednje
  - Ustvari naključno število med 1 in *n*.
  - Poišče število v tabeli.
- Ustavi merjenje časa.
- Izračuna povprečje.

Čas izvajanja merite na sledeči način:

```
long startTime = System.nanoTime();
// iskanje elementa
long executionTime = System.nanoTime() - startTime;
```

d) Eksperimentalno ovrednotenje algoritmov

Na učilnico oddajte **izvorno kodo** (javanska datoteka in ne zip ipd.) te naloge.

Za vrednosti $n \in [10^3, \ldots, 10^5]$ s korakom 10^3 tabelirajte povprečni čas izvajanja. Izpišite tabelo s tremi stolpci:

- prvi stolpec naj vsebuje *n*,
- drugi povprečni čas izvajanja navadnega iskanja,
- tretji pa povprečni čas dvojiškega iskanja.

Primer izpisa:

```
   n     |    linearno  |   dvojisko   |
---------+--------------+--------------
    1000 |          662 |           41
    2000 |         1444 |           46
    3000 |         2135 |           50
    4000 |         2706 |           47
    5000 |         3433 |           52
    6000 |         3751 |           51
    ...            ...             ...
```

e) Razmislite

- Zakaj so časi pri vas drugačni kot v zgornji tabeli?
- Kateri algoritem je hitrejši?
- Kdaj bi lahko bil počasnejši algoritem hitrejši?
- Kako je čas odvisen od velikosti naloge (linearno, kvadratno, ...)?
- Je časovna odvisnost dvojiškega iskanja bližje linearni ali konstantni?

f) Grafični prikaz - neobvezno, a zabavno in poučno

Narišite graf zgoraj izmerjenih vrednosti časa izvajanja v odvisnosti od dolžine tabele. Za izris priporočamo uporabo knjižnice StdLib in razreda StdDraw. Za barvno izvedbo potrebujete dodati 6 vrstic v zgornjo rešitev.

V skrajni sili lahko uporabite tudi druge možnosti:

- Uporabite lahko program za izdelavo razpredelnic (OpenOffice, Excel, ...).
- Izris neposredno iz Jave z uporabo enostavne knjižnice jMathPlot. Na povezavi najdete enostaven uvod v uporabo te knjižnice.
- itd.