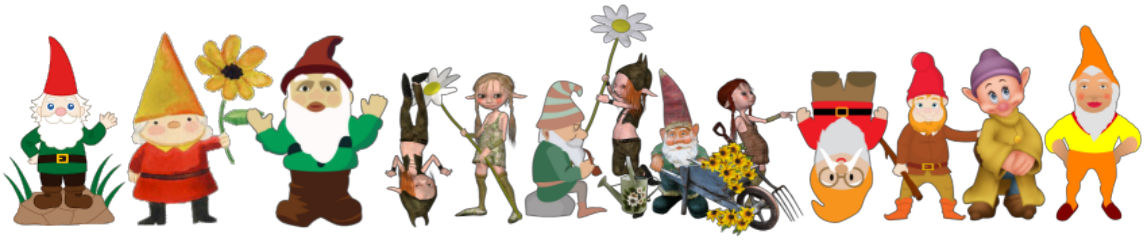
Naloga 1

Naloga 1 - urejanje vrtnih palčkov

Zvečer, tik preden se popolnoma stemni, se na vašem vrtu palčki in njim sorodna bitja naključno (uporabljajo kvantni vir naključnosti) postavijo v vrsto in uredijo po velikosti (včasih nepadajoče, drugič nenaraščajoče). V izogib pretiranemu prerivanju in zmedi se držijo tega postopka. Ko je ureditev popolna, palčki preverijo, če so vsi prisotni, kar lahko naredijo zelo hitro z dvojiškim iskanjem. Pri vsem tem jih lahko celo ugledate in opazujete, vendar le če ste pri študiju algoritmov zelo pridni in zelo pozorni. Seveda vas takoj pograbi radovednost, kako učinkovit je njigov postopek, zato preostanek noči porabite za implementacijo in primerjanje algoritmov urejanja, ki ste jih spoznali pri APS2, hkrati pa preberete še kaj zanimivega. Ker je svet palčkov diskreten, so velikosti palčkov cela števila, ki so sprva razvrščena v tabeli, tako kot se palčki najprej postavijo v vrsto. Včasih se kateri od palčkov postavi na glavo, takrat se njegova velikost upošteva kot negativno število.



Navodila

Glej tudi splošna navodila za izdelavo in oddajo domačih nalog.

Napišite program `Naloga1.java` za izpis sledi in štetja operacij izvajanja algoritmov za notranje urejanje tabele števil. Implementirajte naslednje algoritme:

- `bs` - urejanje z mehurčki (angl. bubble sort);
- `ss` - urejanje z izbiranjem (angl. selection sort);
- `is` - urejanje z vstavljanjem (angl. insertion sort);
- `hs` - urejanje s kopico (angl. heap sort);
- `qs` - hitro urejanje (angl. quicksort);
- `ms` - urejanje z zlivanjem (angl. merge sort);
- `cs` - urejanje s štetjem (angl. counting sort)
- `rs` - korensko urejanje (angl. radix sort).

Program naj podpira naslednje argumente:

- Prvi argument podaja način delovanja programa: `trace` - izpis sledi algoritma, `count` - izpis števila primerjav in premikov.

- Drugi argument programa podaja oznako (`bs`, `ss`, `is`, `hs`, `qs`, `ms`,`cs`,`rs`) izbranega algoritma.
- Tretji argument podaja smer urejanja: `up` - nepadajoče, `down` - nenaraščajoče.
- Četri argument podaja velikost tabele (v nekaterih testnih primerih ta argument ne bo prisoten, zato sami poskrbite za ustrezno povečevanje velikosti tabele).

Branje vhoda. Tabelo števil preberite s standardnega vhoda. Števila bodo med seboj ločena z belimi znaki (presledek, tab, newline). Uporabite `Scanner.nextInt()`! Branju po vrsticah se izognite. Če velikost tabele ni podana kot argument, najprej ustvarite tabelo velikost ena, ki jo potem dinamično povečujte (npr. dvakrat daljša od prejšnje).

Sled algoritma. Če je prvi argument enak `trace`, potem naj program izpiše sled delovanja algoritma. Sled naj vsebuje izpis elementov trenutne tabele. Mejo med posameznimi deli tabele (npr. med urejenim in neurejenim delom) označite z znakom "`|`". Sled izpišite znotraj zunanje zanke, po izvedbi notranje zanke. Za podrobnosti glej primere spodaj.

Štetje operacij. Če je drugi argument enak `count`, potem naj program izpiše število primerjav elementov in število prirejanj elementov (ena zamenjava šteje za tri prirejanja). Pozor: štejte le operacije nad elementi (primerjave indeksov ipd. ignorirajte)! Število operacij izpišite za tri različne primere:

- za urejanje podanega zaporedja števil v podani smeri;
- za urejanje podanega, vendar že urejenega zaporedja v podani smeri;
- za urejanje podanega, vendar že urejenega zaporedja v obratni smeri od podane.

Povedano drugače: najprej uredite podano zaporedje, nato ga (že urejenega) še enkrat uredite in na koncu še enkrat uredite, vendar v obratni smeri.

Po predavanjih, vajah in drugih virih samostojno **preštudirajte** dane algoritme. Ker obstaja več različic algoritmov, se pri izvedbi naloge **obvezno** držite tukaj podanih napotkov. Vaši programi bodo avtomatsko testirani.

Pozor: vse algoritme implementirajte sami, uporaba javanskih knjižnic ni dovoljena (z izjemo razreda `Scanner`).

Bubble sort

Na začetku je urejeni del prazen. Tekom algoritma urejeni del narašča od leve proti desni. Notranja zanka od desne proti levi menja dva zaporedna elementa, če sta med seboj neurejena. Nato se izpiše sled. Sled naj se izpiše tudi na začetku algoritma (glej primer).

Primer izvajanja:

Standardni vhod: `8 5 6 1 7 2`

Izpis sledi:

```
java Naloga1 trace bs up 6
| 8 5 6 1 7 2
1 | 8 5 6 2 7
1 2 | 8 5 6 7
1 2 5 | 8 6 7
1 2 5 6 | 8 7
1 2 5 6 7 | 8
```

Štetje operacij:

```
java Naloga1 count bs up 6
15 30
15 0
15 45
```

Selection sort

Na začetku je urejeni del prazen. Tekom algoritma urejeni del narašča od leve proti desni. Notranja zanka v neurejenem delu tabele poišče najmanjši element in ga zamenja s prvim elementom neurejenega dela tabele. Nato se izpiše sled. Sled naj se izpiše tudi na začetku algoritma (glej primer).

Primer izvajanja:

Standardni vhod: `8 5 6 1 7 2`

Izpis sledi:

```
java Naloga1 trace ss up 6
| 8 5 6 1 7 2
1 | 5 6 8 7 2
1 2 | 6 8 7 5
1 2 5 | 8 7 6
1 2 5 6 | 7 8
1 2 5 6 7 | 8
```

Štetje operacij:

```
java Naloga1 count ss up 6
15 15
15 15
15 15
```

Insertion sort

Na začetku urejeni del vsebuje prvi element tabele. Tekom algoritma urejeni del narašča od leve proti desni. Na vsakem koraku notranje zanke se prvi element v neurejenem delu potisne v urejeni del tako, da **menjamo** zaporedne elemente, dokler urejeni del tabele ne postane zopet urejen. Nato se izpiše sled. Sled naj se izpiše tudi na začetku algoritma (glej primer).

Primer izvajana:

Standardni vhod: `8 5 6 1 7 2`

Izpis sledi:

```
java Naloga1 trace is up 6
8 | 5 6 1 7 2
5 8 | 6 1 7 2
5 6 8 | 1 7 2
1 5 6 8 | 7 2
1 5 6 7 8 | 2
1 2 5 6 7 8 |
```

Štetje operacij:

```
java Naloga1 count is up 6
13 30
5 0
15 45
```

Heap sort

Urejanje s kopico poteka v dveh fazah: gradnja kopice in dejansko urejanje. Tekom postopka je kopica v levem delu table, v desnem pa je urejeni seznam. Izpis sledi naj sestoji iz izpisa dela table, ki vsebuje kopico. Sled izpišite takoj po gradnji kopice in nato po vsakem pogrezanju elementa (vsakič kot je drevo v tabeli kopica). Vsako sled oz. kopico izpišite v svojo vrstico, pri čemer nivoje drevesa ločite z navpično črto. Primer: `2 | 3 4 | 5 6 7`, predstavlja kopico, ki ima v korenu 2, na prvem nivoju števili 3 in 4, na zadnjem pa 5, 6 in 7.

Primer izvajanja:

Standardni vhod: `8 5 6 1 7 2 0 9`

Izpis sledi:

```
java Naloga1 trace hs up 8
9 | 8 6 | 5 7 2 0 | 1
8 | 7 6 | 5 1 2 0
7 | 5 6 | 0 1 2
6 | 5 2 | 0 1
5 | 1 2 | 0
2 | 1 0
1 | 0
0
```

Štetje operacij:

```
java Naloga1 count hs up 8
26 54
27 66
24 48
```

Quicksort

Deljenje tabele na dva oz. tri dele:

- Delitev izvedite po spodnji psevdokodi.
- Kot pivot vzemite sredinski element. V primeru dileme se *"nagnite*" na levo.
- Bodite pozorni, da lahko tabela razpade tudi na tri dele.
- Po vsaki delitvi izpišite sled, t.j. izpišite vse tri dele tabele z mejo "|".

Psevdokoda delitve (angl. partition) tabele:

```
i = levi;  j = desni
while i <= j do
    while table[i] < pivot do i = i + 1
    while table[j] > pivot do j = j - 1
    if i <= j then
        swap(i, j)
        i = i + 1
        j = j - 1
    endif
endwhile
```

Pri štetju operacij poleg `swap` ne pozabite šteti tudi premika, kjer shranite pivot.

Primer izvajanja:

Standardni vhod: `8 5 6 1 7 2 0 9`

Izpis sledi:

```
java Naloga1 trace qs up 8
0 1 | | 6 5 7 2 8 9
| 0 | 1
6 5 2 | | 7 8 9
2 | 5 | 6
7 | 8 | 9
```

Štetje operacij:

```
java Naloga1 count qs up 8
27 26
21 16
24 29
```

Merge sort

Deljenje tabele na dva dela:

- Na vsaki stopnji tabelo *in-place* razdelite (brez ustvarjanja novih tabel) na dve polovici, pri čemer je leva polovica enaka oz. kvečjemu za ena večja od desne.
- Razdeljevanje tabel izvedite do konca, t.j. dokler ne dobite tabel velikosti ena.
- Po vsaki delitvi izpišite sled, t.j. tabelo in mejo "|".

Sestavljanje oz. zlivanje dveh tabel v novo tabelo.

- Rezultat zlivanja je nova tabela, t.j. zlivanja ni potrebno implementirati *in-place*.

- Najmanjši element z začetka obeh tabel zaporedoma prepisujete v novo tabelo, v primeru enakosti ima prednost prva (*leva*) tabela.
- Po vsakem zlivanju dveh tabel izpišite sled, t.j. vsebino novo nastale tabele.

Animacija algoritma:

- http://www.cse.iitk.ac.in/users/dsrkg/cs210/applets/sortingII/mergeSort/mergeSort.html

Primer izvajanja:

Standardni vhod: `8 5 6 1 7 2 0 9`

Izpis sledi:

```
java Naloga1 trace ms up 8
8 5 6 1 | 7 2 0 9
8 5 | 6 1
8 | 5
5 8
6 | 1
1 6
1 5 6 8
7 2 | 0 9
7 | 2
2 7
0 | 9
0 9
0 2 7 9
0 1 2 5 6 7 8 9
```

Štetje operacij:

```
java Naloga1 count ms up 8
17 32
12 32
12 32
```

Counting sort

Algoritem izvedite v naslednjih korakih:

1. štetje elementov (vhodna števila bodo v intervalu od 0 do 255)
2. izračun kumulative
3. izpis tabele s kumulativo (v eni vrstici)
4. generiranje izhodne tabele
5. izpis izhodne tabele (v eni vrstici)

Funkcionalnost `count` (štetje operacij) pri tem algoritmu ignorirajte. Glej tudi korensko urejanje.

Primer izvajanja:

Standardni vhod: `5 1 6 1 6 2 0 2`

Izpis sledi:

```
java Naloga1 trace cs up 8
1 3 5 5 5 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8
0 1 1 2 2 5 6 6
```

V zgornjem izpisu sledi sta izpisani dve vrstici. Prva vsebuje 256 števil: 1 3 5 5 5 6 8 8 ..., druga pa je končna urejena tabela: 0 1 1 2 2 5 6 6.

Radix sort

Korensko urejanje - od najmanj pomembnega bajta do najbolj pomembnega. Kot stabilen podalgoritem uporabite urejanje s štetjem. Pri tem predlagamo, da metodo implementirate tako, da prejme tudi parameter, ki pove glede na kateri bajt elementa (število tipa int) se izvede urejanje - s tem bo izvedba korensko urejanja trivialna.

Predpostavite lahko, da bodo vhodna števila pozitivna. Vseeno pa lahko razmislite, kako bi v celoti podprli urejanje glede na Javanski tip `int` (negativna števila).

Kot sled se uporabi kar sled iz urejanja s štetjem, ki se sicer večkrat ponovi.

Primer izvajanja:

Standardni vhod: `256 123456 100 65536 300 42 7 16777216`

Izpis sledi:

```
java Naloga1 trace rs up
3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8
256 65536 16777216 7 42 300 123456 100
5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8
65536 16777216 7 42 100 256 300 123456
6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8
16777216 7 42 100 256 300 65536 123456
7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8
7 42 100 256 300 65536 123456 16777216
```

Status oddaje naloge

Status oddaje naloge	Pri tej nalogi vam ni treba oddati ničesar.
	Pri tej nalogi ne morete oddati prispevkov.
Stanje ocen	Neocenjeno

Rok za oddajo	nedelja, 8. april 2018, 23:55
Preostali čas	Rok za oddajo naloge je potekel
Zadnja sprememba	-
Komentar oddaje	Komentarji (0)

Odziv

Ocena	8,00 / 8,00
Ocenjeno v	četrtek, 31. maj 2018, 13:18

NAVIGACIJA

Pregledna plošča
- Prva stran
  - Strani spletnega mesta
  - Trenutni predmet
    - aps2uni
      - Sodelujoči
      - Priznanja
      - Splošno
      - O predmetu
      - Potek predmeta
      - Samostojno delo
      - Viri in literatura
      - Obvestila
      - Razprave
      - Opravljanje domačih nalog
      - Naloga 0
      - **Naloga 1**
      - Naloga 2
      - Naloga 3
      - Skupaj izzivi
      - Dodatne točke
      - teorija3-rok
      - 5. marec - 11. marec

12. marec - 18. marec
19. marec - 25. marec
26. marec - 1. april
2. april - 8. april
9. april - 15. april
16. april - 22. april
23. april - 29. april
7. maj - 13. maj
14. maj - 20. maj
21. maj - 27. maj
28. maj - 3. junij

Moji predmeti

Predmeti

NASTAVITVE

Skrbništvo predmeta