**6. praktiskais darbs. 2. semestris**

**1. uzdevums**

Realizēt 4 kārtošanas algoritmus, kas sakārto masīva elementus dilstošā (neaugošā) secībā (5 punkti):

* 1. Atspoles metode
  2. Ievietošanas metode
  3. Šella metode
  4. Hoara jeb ātrās kārtošanas metode

NB! Nedrīkst rakstīt algoritmu, kas vispirms sakārto masīvu augošā secībā un pēc tam to nodrukā apgrieztā secībā!

1. Uzdevumu izpildes nosacījumi:

* Uzsākot programmas izpildi, tiek izveidots masīvs, kas satur N + 1 elementu, kur skaitli N ievada lietotājs.
* Tiek izveidota funkcija, kas aizpilda šo masīvu pēc šādiem nosacījumiem: masīva elements ar indeksu 0 saņem vērtību 0, bet pārējie masīva elementi – gadījuma naturālos skaitļus intervālā no 1 līdz N, paredzot, ka tie masīvā neatkārtosies.
* Izveidot funkciju, kas doto masīvu pārvērš par simbolu virkni, kuras sākumā norādīts salīdzināšanas skaits, kam seko sakārtotie masīva elementi, kuri viens no otra ir atdalīti ar komatu, piemēram, ***17 salīdzināšanas – 7, 6, 5, 4, 3, 2, 1,0***
* Izveidotas četras funkcijas (procedūras), kas dotā masīva elementus sakārto dilstošā (neaugošā) un atgriež masīva elementu salīdzināšanas reižu skaits, kāds bija, realizējot šo kārtošanas metodi.
* Visas četras kārtošanas metodes tiek pielietotas vienam un tam pašam sākotnēji izveidotajam un aizpildītajam masīvam.
* Programmas izpildes rezultātā uz ekrāna tiek parādīti visi četri kārtošanas metožu izpildes rezultāti, sakārtojot 10, 1 000 un 100 000 skaitļus.

**Kods:**

# Programmas nosaukums: Četras kārtošanas algoritmi

# 1. uzdevums (1MPR06\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Realizēt 4 kārtošanas algoritmus, kas sakārto masīva elementus dilstošā (neaugošā) secībā (5 punkti):

# 1.1. Atspoles metode

# 1.2. Ievietošanas metode

# 1.3. Šella metode

# 1.4. Hoara jeb ātrās kārtošanas metode

# NB! Nedrīkst rakstīt algoritmu, kas vispirms sakārto masīvu augošā secībā un pēc tam to nodrukā apgrieztā secībā!

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

import random

import math

def is\_natural(n):

# Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav

# Ja ir naturāls skaitlis, tad True. Ja nav tad False.

# n - simbolu virkne, kuru pārbauda.

if str(n).isdigit() and float(n) == int(n) and int(n) > 0:

return True

else:

return False

def sort\_atspole\_dilstosa(a):

# Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu

# Kārtošanas tiek izmantota atspoles metode

# a - viendimensijas masīvs

n = len(a)

count = 0

for i in range(1, n):

if a[i - 1] < a[i]:

count += 1

for j in range(i, 0, -1):

if a[j - 1] < a[j]:

count += 1

x = a[j]

a[j] = a[j - 1]

a[j - 1] = x

else:

count += 1

break

else:

count += 1

return count

def sort\_ievietosana\_dilstosa(a):

# Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu

# Kārtošanas tiek izmantota ievietošanas metode (insertion sort)

# a - viendimensijas masīvs

n = len(a)

sk = 0

for i in range(1, n):

sk = sk + 1

if a[i - 1] < a[i]:

x = a[i]

j = i

sk = sk + 1

while a[j - 1] < x:

a[j] = a[j - 1]

j = j - 1

if j == 0:

break

sk = sk + 1

a[j] = x

return sk

def sort\_sella\_dilstosa(a):

# Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu

# Kārtošanas tiek izmantota Šellas metode (Shell sort)

# a - viendimensijas masīvs

sk = 0

n = len(a)

solis = (3\*\*math.floor(math.log(2 \* n + 1, 3)) - 1) // 2

while solis >= 1:

for k in range(0, solis):

for i in range(solis + k, n, solis):

sk = sk + 1

if a[i - solis] < a[i]:

x = a[i]

j = i

sk = sk + 1

while a[j - solis] < x:

a[j] = a[j - solis]

j = j - solis

if j == k:

break

sk = sk + 1

a[j] = x

solis = (solis - 1) // 3

return sk

def sort\_atrais\_dilstosa(a, sv, bv):

# Sakārto masīvu dilstoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu

# Kārtošanas tiek izmantota Hoara (ātrais) metode (quicksort)

# a - viendimensijas masīvs

# sv - sākuma vērtība

# bv - beigu vērtība

p = 0

if sv < bv:

i = sv

j = bv

solis = -1

lv = True

while i != j:

if lv == (a[i] < a[j]):

x = a[i]

a[i] = a[j]

a[j] = x

x = i

i = j

j = x

lv = not lv

solis = -solis

p += 1

j = j + solis

p1 = sort\_atrais\_dilstosa(a, sv, i - 1)

p2 = sort\_atrais\_dilstosa(a, i + 1, bv)

p = p + p1 + p2

return p

def izvadit\_masivu\_un\_salidzinasanas\_skaitu(x, sal):

# Izvada salīdzināšanas skaitu un izvada masīva elementus ar komatiem

# x - viendimensijas masīvs

# sal - int skaitlis

n = len(x)

s = str(x[0])

for i in range(1, n):

s = s + ", " + str(x[i])

return str(sal) + " salīdzināšanas - " + s

def izvadit\_masivu(x):

# Izvada masīva elementus ar komatiem

# x - viendimensijas masīvs

n = len(x)

s = str(x[0])

for i in range(1, n):

s = s + ", " + str(x[i])

return s

def samaisit(masivs):

# Samaisa masīva elementus (funkcija tiek izmantota, lai no sakārtota masīva numpy.arange(n + 1) izveidot nesakārtotu (unsort)

# masivs - viendimensijas masīvs

i = 0

while i < len(masivs) // 2:

x = random.randint(1, len(masivs) - 1)

y = random.randint(1, len(masivs) - 1)

temp = masivs[x]

masivs[x] = masivs[y]

masivs[y] = temp

i = i + 1

return masivs

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

n = input("Ievadiet masīva izmēru N ===> ")

while is\_natural(n) == False:

n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

n = int(n)

a = numpy.arange(n + 1)

samaisit(a)

print("\nSākotnējais masīvs:")

print(izvadit\_masivu(a))

y = numpy.copy(a)

z = numpy.copy(a)

k = numpy.copy(a)

p = numpy.copy(a)

sorted\_b = sort\_atspole\_dilstosa(y)

print("\nKārtošana ar atspoles metodi:")

print(izvadit\_masivu\_un\_salidzinasanas\_skaitu(y, sorted\_b))

sorted\_c = sort\_ievietosana\_dilstosa(z)

print("\nKārtošana ar ievietošanas metodi:")

print(izvadit\_masivu\_un\_salidzinasanas\_skaitu(z, sorted\_c))

sorted\_d = sort\_sella\_dilstosa(p)

print("\nKārtošana ar Šella metodi:")

print(izvadit\_masivu\_un\_salidzinasanas\_skaitu(p, sorted\_d))

sorted\_e = sort\_atrais\_dilstosa(k, 0, len(a) - 1)

print("\nKārtošana ar Hoara (ātrais) metodi:")

print(izvadit\_masivu\_un\_salidzinasanas\_skaitu(k, sorted\_e))

**Testa piemēri:**

1)

A picture containing text

Description automatically generated

2)

A picture containing text

Description automatically generated

3)

A picture containing arrow

Description automatically generated

4)

A picture containing text

Description automatically generated

5)

A computer screen capture

Description automatically generated with low confidence

6)

A computer screen capture

Description automatically generated with low confidence

7)

A picture containing background pattern

Description automatically generated

**2. uzdevums**

Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu. Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas mediānu. (1 punkts)

**Kods:**

# Programmas nosaukums: Kopas mediāna

# 2. uzdevums (1MPR06\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu. Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas mediānu. (1 punkts)

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

import math

def is\_natural(n):

# Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav

# Ja ir naturāls skaitlis, tad True. Ja nav tad False.

# n - simbolu virkne, kuru pārbauda.

if str(n).isdigit() and float(n) == int(n) and int(n) > 0:

return True

else:

return False

def izveidot\_masivu\_ar\_garumu(n):

# Izveido masīvu ar noradīto garumu n

# n - naturāls skaitlis

a = numpy.arange(n)

for i in range(n):

b = input("Ievadiet " + str(i) + ".elementu ===> ")

b = is\_whole(b, i)

a[i] = b

return a

def is\_whole(x, i):

# Pārbauda vai simbolu virkne ir vesels skaitlis un ja nē, tad paprasa ievadīt to vēlreiz (bezgalīgi daudz ievādes)

# Ja simbolu virkne ir vesels skaitlis, tad atgriež to kā int(x)

# x - pārbaudāma simbolu virkne

# i - i-tajs elements

while True:

try:

x = int(x)

except:

x = input("Kļūda! Ievadiet " + str(i) + ".elementu ===> ")

else:

return int(x)

def sort\_atrais\_augosa(a, sv, bv):

# Sakārto masīvu augoša secība un atgriež salīdzināšanas skaitu, lai sakartotu masīvu

# Kārtošanas tiek izmantota Hoara (ātrais) metode (quicksort)

# a - viendimensijas masīvs

# sv - sākuma vērtība

# bv - beigu vērtība

if sv < bv:

i = sv

j = bv

solis = -1

lv = True

while i != j:

if lv == (a[i] > a[j]):

x = a[i]

a[i] = a[j]

a[j] = x

x = i

i = j

j = x

lv = not lv

solis = -solis

j = j + solis

sort\_atrais\_augosa(a, sv, i - 1)

sort\_atrais\_augosa(a, i + 1, bv)

def is\_even\_masivs(x):

# Pārbauda vai masīva garums ir pāra skaitlis

# Ja masīva garums ir pāra skaitlis, tad atgriež True

# Ja masīva garums nav pāra skaitlis, tad atgriež False

# x - viendimensijas masīvs

masiva\_garums = len(x)

if masiva\_garums % 2 == 0:

return True

else:

return False

def mediana(x):

# Atgriež masīva x mediānu

# x - viendimensijas masīvs

if is\_even\_masivs(x):

return (x[len(x) // 2 - 1] + x[len(x) // 2]) / 2

else:

return x[len(x) // 2]

def izvade(x):

# Izvada masīva elementus pēc kārtas līdz pedējam

# x - viendimensijas masīvs

n = len(x)

s = str(x[0])

for i in range(1, n):

s = s + ", " + str(x[i])

print(s)

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

m = input("Ievadiet masīva izmēru N ===> ")

while is\_natural(m) == False:

m = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

m = int(m)

print("Ievadiet masīva skaitļus!")

t = izveidot\_masivu\_ar\_garumu(m)

print("\nIevadīta skaitļu kopa:")

izvade(t)

sort\_atrais\_augosa(t, 0, len(t) - 1)

print("\nSakārtota skaitļu kopa:")

izvade(t)

print("\nKopas mediāna ir:\n" + str(mediana(t)))

**Testa piemēri:**

1)

A picture containing text

Description automatically generated

2)

Text

Description automatically generated

3)

Calendar

Description automatically generated with medium confidence

4)

Text

Description automatically generated

5)

Text

Description automatically generated

6)

Text

Description automatically generated

7)

Text

Description automatically generated

**PU1. uzdevums**

Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu. Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas modu. Ja ir vairākas modas, tad jāpaziņo visas, bet, ja modas nav (t.i. visas atšķirīgas vērtības vienādā skaitā), tad jāpaziņo “moda netika atrasta” (2 punkti).

**Kods:**

# Programmas nosaukums: Modas noteikšana

# Papilduzdevums 1 (1MPR06\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Lietotājs ievada iepriekš zināma garuma nesakārtotu masīvu.

# Noskaidrot un izvadīt uz ekrāna šo skaitļu kopas modu. Ja ir vairākas modas, tad jāpaziņo visas, bet, ja modas nav

# (t.i. visas atšķirīgas vērtības vienādā skaitā), tad jāpaziņo “moda netika atrasta” (2 punkti).

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

def is\_natural(n):

# Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav

# Ja ir naturāls skaitlis, tad True. Ja nav tad False.

# n - simbolu virkne, kuru pārbauda.

if str(n).isdigit() and float(n) == int(n) and int(n) > 0:

return True

else:

return False

def izveidot\_masivu\_ar\_garumu(n):

# Izveido masīvu ar noradīto garumu n

# n - naturāls skaitlis

a = numpy.arange(n)

for i in range(n):

b = input("Ievadiet " + str(i) + ".elementu ===> ")

b = is\_whole(b, i)

a[i] = b

return a

def is\_whole(x, i):

# Pārbauda vai simbolu virkne ir vesels skaitlis un ja nē, tad paprasa ievadīt to vēlreiz (bezgalīgi daudz ievades)

# Ja simbolu virkne ir vesels skaitlis, tad atgriež to kā int(x)

# x - pārbaudāma simbolu virkne

# i - i-tajs elements

while True:

try:

x = int(x)

except:

x = input("Kļūda! Ievadiet " + str(i) + ".elementu ===> ")

else:

return int(x)

def izvade(x):

# Izvada masīva elementus pēc kārtas līdz pēdējam

# x - viendimensijas masīvs

n = len(x)

s = str(x[0])

for i in range(1, n):

s = s + ", " + str(x[i])

print(s)

def mode(masivs):

# Atrod masīva modu un atgriež masīvu ar modam (vai modu)

# Ja masīva visi elementi ir vienādi, tad atgriež False (moda netika atrasta)

# masivs - viendimensijas masīvs (kurš vel netika kārtots augoša secība)

'''

Paskaidrojums, kā šī funkcija darbojas:

Funkcija vispirms paņem, ka moda ir pirmais sakārtotā masīva elements.

Pēc tam, funkcija iziet cauri katram masīva elementam un atjaunina sarakstu ar modam un to biežumu katru reizi, kad tiek atrasta jauna mode ar augstāku biežumu (elements, kuru ir vairāk).

Ja tas saskaras ar modu ar tādu pašu biežumu (biežums - skaits, cik reizes paradās šis elements) kā pašreizēja moda, tad tas pievieno jauno modu, modu sarakstam.

Funkcija iterē katru elementu sakārtotajā masīvā un pārbauda, vai tas ir vienāds ar iepriekšējo elementu.

Ja tā ir, mainīgais "sk" tiek palielināts. Ja tā nav, funkcija pārbauda, vai pašreizējais skaits ir lielāks par līdz šim saglabāto maksimālo skaitu.

Ja ir lielāks, tad maksimālais skaits tiek atjaunināts un pašreizējais skaitlis (elements) tiek pievienots modas sarakstam.

Ja pašreizējais skaits ir vienāds ar maksimālo līdz šim saglabāto skaitu, modas sarakstam tiek pievienots arī šis skaitlis (elements) (vairākas modes).

Pēc tam kad tika "iziterēts" viss masīvs, tad pārbaudām, vai pēdējā elementa biežums ir lielāks par līdz šim redzēto maksimālo biežumu.

Ja tā ir, tad maksimālais skaits tiek atjaunināts un modas saraksts ir tas pēdējais elements.

Ja pēdējā elementa skaits ir vienāds ar maksimālo līdz šim redzēto (saglabāto) skaitu, modas sarakstam tiek pievienots arī pēdējais elements (vairākas modes).

Ja maksimālais biežums ir vienāds ar 1 (t.i., visi masīva elementi ir unikāli), funkcija atgriež vērtību False.

Pretējā gadījumā tas atgriež atrasto modas sarakstu.

'''

sort\_atrais\_augosa(masivs, 0, len(masivs) - 1) # Ievadītais masīvs tiek izkārtots augošā secībā, izmantojot sort\_atrais\_augosa

sk = 1 # seko pašreizējā apstrādājamā skaitļa (elementa) biežumam

max\_sk = 1 # saglabā jebkura masīva skaitļa (elementa) maksimālo biežumu

modas\_list = [masivs[0]] # līdz šim atrasto modu saraksts

if len(masivs) == 1: # Ja masīva garums ir 1, tad moda arī ir tas skaitlis (vienīgais elements)

modas\_list = [masivs[0]]

return modas\_list

for i in range(1, len(masivs)):

if masivs[i] == masivs[i - 1]: # Ja elements i ir vienāds ir i - 1, tad palielinām skaitītāju

sk = sk + 1

else:

if sk > max\_sk: # ja skaititajs kļūst lielāks nekā maksimālais skaitītājs

max\_sk = sk # tad maksimālais skaititajs atjaunojas

modas\_list = [masivs[i - 1]] # modas saraksts atjaunojas un paliek tas masīva elements, kurš vislielāko reizi paradījās pēc skaitītajā

elif sk == max\_sk: # Ja skaititajs ir vienāds ar pašreizējo maksimālo skaitītāju (max biežumu)

modas\_list.append(masivs[i - 1]) # tad pievienojam sarakstam jauno modu (papildus modu)

sk = 1 # pēc iterācijām sk atkal kļūst par 1

if sk > max\_sk: # Tas ir veidots pēdējam elementam. Ja lielāks skaititajs nekā pašreizējais maksimālais.

max\_sk = sk # tad atjaunojam maksimālo skaitītāju

modas\_list = [masivs[-1]] # atjaunojam modas sarakstu

elif sk == max\_sk: # Ja modas biežumi ir vienādi, tad pievienojam modas sarakstam

modas\_list.append(masivs[-1])

if max\_sk == 1: # Ja maksimālais biežums ir vienāds ar 1 (visi masīva elementi ir unikāli), atgriež vērtību False

return False

else:

return modas\_list # Atgriež atrasto modas sarakstu

def sort\_atrais\_augosa(a, sv, bv):

# Sakārto masīvu augoša secība un atgriež salīdzināšanas skaitu, lai sakārtotu masīvu

# Kārtošanas tiek izmantota Hoara (ātrais) metode (quicksort)

# a - viendimensijas masīvs

# sv - sākuma vērtība

# bv - beigu vērtība

if sv < bv:

i = sv

j = bv

solis = -1

lv = True

while i != j:

if lv == (a[i] > a[j]):

x = a[i]

a[i] = a[j]

a[j] = x

x = i

i = j

j = x

lv = not lv

solis = -solis

j = j + solis

sort\_atrais\_augosa(a, sv, i - 1)

sort\_atrais\_augosa(a, i + 1, bv)

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

m = input("Ievadiet masīva izmēru N ===> ")

while is\_natural(m) == False:

m = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

m = int(m)

print("Ievadiet masīva skaitļus!")

t = izveidot\_masivu\_ar\_garumu(m)

print("\nIevadīta skaitļu kopa:")

izvade(t)

sort\_atrais\_augosa(t, 0, len(t) - 1)

print("\nSakārtota skaitļu kopa:")

izvade(t)

modas = mode(t)

if modas == False:

print("\nModa netika atrasta.")

else:

print("\nSkaitļu kopas moda(s) ir:")

izvade(modas)

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated

3)

Text

Description automatically generated

4)

Text, calendar

Description automatically generated

5)

Text

Description automatically generated

6)

A picture containing shape

Description automatically generated

7)

Calendar

Description automatically generated

8)

Text

Description automatically generated

9)

A picture containing calendar

Description automatically generated

10)

Text

Description automatically generated

11)

Text

Description automatically generated

12)

Text

Description automatically generated