TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Raul Lehesalu

Sortimismeetodite testid ainele „Algoritmid ja andmestruktuurid“

Bakalaureusetöö (9 EAP)

Juhendaja: Ahti Peder, Phd

Tartu 2020

Sortimismeetodite testid ainele „Algoritmid ja andmestruktuurid“

Lühikokkuvõte:

Käesoleval sajandil on veebiõppe levik kiirendanud bakalaureusetaseme ülikoolihariduse standardiseerumist. Arvutiteaduse bakalaureuseõppe esimesel aastal kasutatakse tudengite hindamisel sageli teste, kuid kasvava tudengite arvu tõttu kuuluvad testid üha enam ka teise aasta kursuste õppevahendite hulka. Automaatse hindamisega testid kontrollivad tihti ainult faktiteadmisi. See bakalaureusetöö seab aga eesmärgiks luua ülesandeid, mille vastust õpiku mõisted või veebiotsing enamasti ei paku. Töö praktiline väljund on kaks sortimisalgoritmide testi Tartu Ülikooli „Algoritmide ja andmestruktuuride“ aine Moodle'i õpikeskkonnas. Kirjeldatakse valminud õppevahendi loomise protsessi alates lähteülesannete kogumisest kuni testide koostamiseni. Ülesannete ideed pärinevad peamiselt valitud ülikoolide avalikest õppematerjalidest. Teste plaanitakse õppetöös kasutama hakata 2020. aasta sügissemestril ja nende peamine eesmärk on pakkuda tudengitele enesehindamise võimalust.

Võtmesõnad:

sortimine, algoritmid ja andmestruktuurid, Moodle, test

CERCS: P175 Informaatika, süsteemiteooria; S270 Pedagoogika ja didaktika

Quiz set for the course “Algorithms and Data Structures”

Abstract:

The current century has witnessed a rapid growth in e-learning, which has facilitated the standardization of higher education at the bachelor level. Students in the first year of computer science studies are often graded by quizzes. However, the number of students is growing and this motivates using quizzes also in second-year courses. While automatically graded quizzes are used primarily to test factual knowledge, this thesis sets a goal to create quiz questions for which in most cases a textbook or a web search

would not provide an easy answer. As a result of this thesis two quizzes on the topic of sorting were created. The quizzes were developed for the course “Algorithms and Data Structures” at the University of Tartu. The process of gathering source material, developing quiz questions and creating quizzes in the study environment Moodle are described. Ideas for the quiz questions are derived primarily from public study materials used at selected universities. The main purpose of the quizzes is to offer the possibility of self-assessment for the students. It is planned to start using the quizzes during the autumn semester of 2020.

Keywords:

sorting, algorithms and data structures, Moodle, quiz

CERCS: P175 Informatics, systems theory; S270 Pedagogy and didactics

Sisukord

Sissejuhatus

Sorditud andmed on meie elu tavaline osa ning praktikas leiab sortimine rakendust äritarkvarast teadusarvutusteni. Sorditud andmeid saab tihti efektiivsemalt töödelda ning seega säästetakse kasvavate andmemahtude juures aega ja elektrienergiat. Sortimisalgoritmide võistluste veebilehel[1] kirjeldatud jõudlustestidest põhinebki üks – *JouleSort* [1] – energiakulu mõõtmisel. Sorditud andmeid peetakse sedavõrd oluliseks, et kiirmeetod valiti teaduse ja tehnika valdkonnas 20. sajandi kümne mõjukama algoritmi hulka [2]. Sortimismeetodeid on põhjalikult uuritud ja nad sobivad oma algoritmilise lihtsuse tõttu selgitama mitmeid arvutiteaduse põhimõtteid nagu näiteks rekursioon või ajaline keerukus.

Sortimist õpetatakse ka Tartu Ülikooli „Algoritmide ja andmestruktuuride“ kursusel, mille Moodle'i veebikeskkonnas selle bakalaureusetöö praktiline väljund – kaks sortimismeetodite testi – asub. Veebipõhine õpe on tänapäeval üha tavalisem ja selles kasutatakse hindamisel sageli teste. Eesti Infotehnoloogia Sihtasutuse e-Õppe Arenduskeskuse poolt välja antud „Kvaliteetse õpiobjekti loomise juhendis“ [3] öeldakse, et testid sobivad peamiselt faktiteadmiste kinnistamiseks ja kontrollimiseks. Selles bakalaureusetöös seati aga eesmärgiks luua testid, mis ei piirduks faktiküsimustega, vaid toetaksid kursuse sisu mõistmist. Valminud testid on mõeldud tudengitele eneseharimiseks ja -hindamiseks. Ülesannete loomisel lähtuti sortimise teema sisulisest küljest – sortimisest algoritmide kontekstis – ja didaktilisi meetodeid ei käsitleta.

Bakalaureusetöö on jagatud kolmeks peatükiks, millest esimene piiritleb teema ja annab ülevaate lähtematerjali kogumisest. Teine peatükk kirjeldab ülesannete liigitamist ning näidisküsimuste loomist. Kolmanda peatüki teemaks on ülesannete Moodle'i keskkonda sisestamine ja testide vormistamine. Töö sisaldab lisadena kahte testi (Lisa I) ning lähtematerjali kogumisel kasutatud ülikoolide ja kursuste nimekirja (Lisa II).

[1] https://sortbenchmark.org/

1 Eeltöö

Testide loomise eeltööl oli kaks eesmärki: esiteks tuli käsitletav teema piiritleda; teiseks sai selle piiritluse abil alustada näiteülesannete otsimisega. Kogutud näiteülesanded said lähtematerjaliks suuremale osale testide ülesannetest. Piiritlemisel uuriti Tartu Ülikooli „Algoritmide ja andmestruktuuride“ kursuse (LTAT.03.005) õppematerjale. Näiteid otsiti kursuse kohustuslikus ja soovituslikus kirjanduses leiduvate ülesannete hulgast. Veel uuriti valitud ülikoolide algoritme käsitlevaid bakalaureuseastme kursusi eesmärgiga täpsustada algset teema piiritlust ning leida näiteülesandeid.

1.1 Tartu Ülikooli kursuse loengud ja praktikumid

See jaotis refereerib Tartu Ülikooli „Algoritmide ja andmestruktuuride“ aine 2019. aasta sügissemestri loenguid ja praktikume ning põhineb kursuse Moodle'is asuvatel õppematerjalidel [4].

Sortimist käsitletakse kursuse alguses – esimeses loengus kasutatakse sortimist algoritmi ajalise keerukuse mõiste selgitamisel. Arvutipraktikumide esimene kodutöö nõuab kahe erineva keskmise ajalise keerukusega – $\Theta(n^2)$ ja $\Theta(n \log n)$ – sortimismeetodi programmeerimist ning nende tööaja katselist hindamist. Loengutes ja praktikumides puutuvad tudengid kokku järgmiste sortimismeetoditega:

1. Pistemeetodit (ingl *insertion sort*) tutvustatakse teises loengus ning tahvlipraktikumides antakse ülesandeks mängida läbi ahela ja massiivi järjestamine. Loengus tuuakse välja meetodi erinevus ahela ja massiivi puhul – kas elemendi saab pistekohale panna ühe „pistega“ või peab kõiki järgnevaid elemente edasi lükkama.
2. Ühildamis- ehk põimemeetodit (ingl *merge sort*) kasutatakse kolmandas loengus rekursiooni ning „jaga ja valitse“ (ingl *divide and conquer*) põhimõtete näitena. Tahvlipraktikumis harjutavad tudengid meetodi klassikalist varianti ahelal ja massiivil, optimeeritud varianti ahelal ning alt-üles varianti massiivil.

3. Kiirmeetodist (ingl *quicksort*) on tahvlipraktikumi materjalides esindatud klassikaline kaheharuline variant, kus lahkmeks valitakse esimene element. Selle variandi läbimänguslaididel selgitatakse ahela ja massiivi järjestamist. Kolmandas loengus esitatakse meetodi kolmeharuline variant (elemendid on: lahkmest väiksemad, lahkmega võrdsed või lahkmest suuremad), kus lahkmeks valitakse juhuslik element. Kirjeldatakse ka kiirmeetodi ajalise ruutkeerukuse juhtumit, mis võib põhjustada väljakutsete magasini ületäitumist (ingl *stack overflow*). Käsitletakse meetodi erinevust põimemeetodist rekursioonipuus liikumisel: kiirmeetodi korral ei pea infot üles tooma.
4. Mullimeetodit (ingl *bubble sort*) loengutes ei vaadelda, kuid tahvlipraktikumi slaididel kasutatakse meetodit massiivi järjestamisel.
5. Valikumeetod (ingl *selection sort*) puudub samuti loenguslaididelt ning läbimänguslaididel esitatakse meetodi töö ahelal ja massiivil.
6. Kuhjameetodit (ingl *heap sort*) tutvustatakse tudengitele viiendas loengus kahendkuhja (ingl *binary heap*) andmetüübi õpetamisel. Kahendkuhja näidatakse abstraktse andmetüübi – eelistusjärjekorra (ingl *priority queue*) – võimaliku realiseeringuna. Tahvlipraktikumis mängitakse läbi massiivi järjestamine kuhjameetodil. Arvutipraktikumi kuuendas kodutöös peavad tudengid kirjutama andmete pakkimise programmi Huffmani algoritmi kasutades.
7. Kimbumeetodit (ingl *bucket sort*) õpetatakse seitsmendas loengus koos paisktabeli (ingl *hash table*) andmestruktuuriga. Kursusel õpetatakse ühtlast paiskamist ja kimbumeetodit nimetatakse parimaks järjestamismeetodiks võtmete ühtlase jaotuse korral. Meetodi juures räägitakse tudengitele ka sortimise stabiilsuse mõistest. Seitsmendas ja kaheksandas arvutipraktikumis harjutavad tudengid kimbumeetodi programmeerimist võttes paiskfunktsiooniks ühtlase paiskamise. Tahvlipraktikumis harjutatakse järjestamise kimbumeetodit lihtahelal.
8. Positsioonimeetodil sortimist (ingl *radix sort*) õpetatakse seitsmendas loengus

arvusüsteemi positsioonide järgi. Praktikumis harjutatakse ahela ja massiivi järjestamist nii kahend-, kui ka kümnendsüsteemis.

9. Loendamismeetodi (ingl *counting sort*) puhul tuuakse seitsmendas loengus välja, et meetod eeldab võtmete kuulumist täisarvude lõiku.

Kursuse materjalides leidub ka kaks sortimismeetodit, mida detailsemalt ei käsitleta. Esimeses programmeerimise kodutöös lubatakse tudengitel lineaarlogaritmilise (ingl *linearithmic*, hinnatav funktsiooniga $n \log n$) keerukusega meetodina esitada Shelli meetod (ingl *shell sort*), mis tuleb tudengil aga iseseisvalt selgeks saada. Kaheksandas loengus kirjeldatakse, et kahendotsimispuu (ingl *binary search tree*) keskjärjestuses läbimine (ingl *in-order traversal*) on sisuliselt sortimine.

Teises loengus esitatakse sortimist näitena „lihtsast“ ülesandest: leidub algoritm, mille halvima juhu keerukus on polünomiaalne. Kolmandas loengus tuuakse näide sellest, kuidas sortimine võib probleemi lahendust lihtsamaks muuta: kas üks sõna on teisest saadav tähtede ümbertõstmise abil? Sortimise teema võetakse kokku seitsmendas loengus, kus soovitatakse sisendi omaduste põhjal otsustada, kas lineaarse ajalise keerukusega algoritm sobib kasutamiseks.

1.2 Ülesanded Tartu Ülikooli kursuse õpikutes

Kursuse kohustuslike ja soovituslike õppematerjalide [4] hulka kuuluvad ka mitmed raamatud, mida uuriti eesmärgiga leida sobivaid lähteülesandeid testide loomisel.

Esimesena uuriti „Ülesannete kogu" [5], mille ülesannete valik peaks katma kursuselt omandatavad oskused ja teadmised. Selles raamatus leidub üle viiesaja ülesande, sealhulgas nii teoreetilise suunaga (näiteks tõestamine), kui ka praktilisi (näiteks programmeerimine). Sortimisega seotud ülesandeid võib leida seitsmendas (Järjendi ümberkorraldamine), kaheksandas (Paisksalvestus), kümnendas (Otsimispuud) ja üheteistkümnendas (Kuhjad) peatükis. Õppevahendi lisas kirjeldadakse põimemeetodi optimeeritud ning alt-üles variante. Raamatus leidub ka lahendamise suuniseid ja vastuseid.

Soovitusliku õppematerjali hulka kuuluv Robert Sedgewicki ja Kevin Wayne'i õpik *Algorithms* [6] sisaldab peatükki „Sortimine", mille alajaotustes leidub umbes 150 ülesannet. Need jagunevad materjali omandamist kontrollivateks tavaülesanneteks, loovamateks ülesanneteks (näiteks algoritmi omaduse tõestamine) ning eksperimentideks (näiteks programmi kirjutamine ja tulemuste mõõtmine). Õpikus leidub ülesandeid (Näide 1.1), mis on sarnased „Ülesannete kogus" [5] leiduvatega (Näide 1.2). Mõlemad kontrollivad sortimismeetodi(te) tööpõhimõtte tundmist meetodi läbimängimise abil.

Näide 1.1. *Algorithms [6], ülesanne 2.1.1.*

2.1.1 Show, in the style of the example trace with Algorithm 2.1, how selection sort sorts the array E A S Y Q U E S T I O N

Näide 1.2. *Ülesannete kogu [5], ülesanne 7.21.*

7.21. Olgu antud massiivid

75	10	45	95	50	15	60			
62	63	67	68	65	60	59	77	69	70
80	88	92	97	90	96	81	85	95	99

Sooritada iga massiivi järjestamine

a) mullimeetodil;
b) vahetustega valikumeetodil;
c) pistemeetodil.

Näidata joonisel töö seis pärast välimise tsükli sisu iga täitmist.

Kurt Melhorni ja Peter Sandersi õpiku *Algorithms and Data Structures.* [7] viies peatükk „Sortimine ja otsimine" sisaldab enamasti ülesandeid, kus tuleb algoritmi omadusi tõestada, olemasolevat algoritmi muuta või raamatus esitatud algoritm mõnes programmeerimiskeeles valmis kirjutada. Testides kasutamiseks olid need liiga mahukad.

Veel uuriti soovituslikust kirjandusest Cormeni jt algoritmide õpikut *Introduction to Algorithms* [8], mille kaks esimest peatükki käsitlevad sortimismeetodeid. Meetodid illustreerivad üldisemaid põhimõtteid nagu algoritmide analüüs pistemeetodi näitel või „jaga ja valitse“ printsiip ning rekurrentsed võrrandid põimemeetodi näitel. Igas osajaotustes leidub 5-10 ülesannet, aga paljud neist eeldavad raamatus esitatud konkreetse algoritmi analüüsi. Testiülesannete eeskujuks sobisid neist vähesed, nagu (Näide 1.3).

Näide 1.3. *Introduction to Algorithms [8], ülesanne 6.3-2.*

6.3-2

Why do we want the loop index i in line 2 of BUILD-MAX-HEAP to decrease from $\lfloor A.length/2 \rfloor$ to 1 rather than increase from 1 to $\lfloor A.length/2 \rfloor$?

Valik õpikute ülesandeid moodustas esialgse andmehulga, mille põhjal teste looma hakati.

1.3 Sortimine valitud Euroopa ja Põhja-Ameerika ülikoolides

Sortimise õpetamist ja testide kasutamist uuriti Euroopa ning Põhja-Ameerika ülikoolide algoritme ja andmestruktuure käsitlevatel kursustel. Kursuste valik lähtus paljuski Tartu Ülikooli algoritmide kursuse parendamise eesmärgil kirjutatud bakalaureusetöödest [9, 10, 11, 12], milles leidus hulk viiteid algoritme käsitlevatele ülikoolikursustele. Valikusse võeti ülikoolid, mille loengute, praktikumide ja eksamite materjalid olid avalikult ligipääsetavad. Ülikoolide nimekirja koos veebiviidetega vaadeldud kursustele võib leida Lisas II.

Testivormis ülesandeid leiti viieteistkümnest uuritud ülikooli kursusest kolmes. Massachusettsi Tehnoloogiainstituudi (MIT) algoritmide sissejuhatavas kursuses kasutati neid vaheeksamitel ja lõpueksamil. Näiteks sisaldab 2017. aasta sügissemestri esimese vaheeksami [13] teine ülesanne kahendpuu joonist ja valikvastuseid, mille seast tudeng peab kehtivad väited valima. 2019. aasta kevadsemestri lõpuksami [14] esimeses ülesan-

des on kirjas väited, mille puhul tudengid peavad valima, kas need kehtivad või mitte. Selliseid ülesandeid on kasutatud veel Princetoni Ülikooli algoritmide kursuse eksamitel, näiteks 2019. aasta lõpueksamil [15]; ja Stanfordi Ülikooli algoritmide kursusel, näiteks 2020. aasta talvesemestri proovieksamil [16]. Valikvastustega ülesandeid on lisaks eksamitele kasutatud ka loengutes, näiteks esitatakse Stanfordi Ülikooli algoritmide kursuse teises loengus [17] „müstilise algoritmi“ ülesanne (Näide 1.4).

Näide 1.4. *„Müstiline algoritm“ Stanfordi Ülikooli algoritmide kursuse teises loengus [17].*

```
def mysteryAlgorithmOne(A):
    for x in A:
        B = [None for i in range(len(A))]
        for i in range(len(B)):
          if B[i] == None or B[i] > x:
            j = len(B)-1
            while j > i:
              B[j] = B[j-1]
              j -= 1
            B[i] = x
            break
        return B
```

What was the mystery sort algorithm?

1. MergeSort
2. QuickSort
3. InsertionSort
4. BogoSort

Kui võrrelda sortimise õpetamist Tartu Ülikooli kursusel ja uuritud ülikoolide algoritmide kursustel, siis ühe erinevusena saab esile tuua programmeerimiskeele kasutuse.

Tartu Ülikooli kursuse arvutipraktikumide juhend [18] kirjeldab programmeerimisoskuse täiendamise eesmärki, aga mitmetel kursustel õpetatakse algoritme ja andmestruktuure koos uue programmeerimiskeelega. Näiteks eeldatakse Tallinna Ülikooli algoritmide kursusel [19], et tudeng on võimeline varasemaid teadmisi uue keele kontekstis rakendama. Berkeley Ülikooli andmestruktuuride kursusel [20] õpetatakse samaaegselt nii algoritme ja andmestruktuure kui ka programmeerimist ning tudengitelt ei eeldata kursusel kasutatava keele tundmist. Programmeerimiskeelte kasutuse erinevus andis põhjuse küsida, kui palju võiks valmivates testides kasutada konkreetse programmeerimiskeele koodi. Selle küsimusega tegeletakse töö kolmandas peatükis ülesannete vormi käsitlemisel.

1.4 Järeldused

Eeltöö järel hinnati kogutud lähtematerjali kasutusvõimalusi. Kõige rohkem testidele eeskujuks sobivaid ülesandeid – ligikaudu kakskümmend – leiti „Ülesannete kogust“. Teiste allikate osa jäi oluliselt väiksemaks. Ülikoolide kursusi uurides jõuti seisukohale, et ülikoolikursused kujunevad aastate jooksul loomulikus arengus. Seetõttu tuleb „välise“ materjali lisamisel analüüsida, kuidas see kursusesse sobitub. Kokkuvõttes võib öelda, et sortimise testiülesannete jaoks leidub küll lähtematerjali, kuid napib selliseid, mida saaks kas otse või minimaalsete muudatustega kasutusele võtta.

2 Kavandamine

Kogutud materjalist parema ülevaate saamiseks kasutati ülesannete liigitamist. Seda lihtsustas sortimisprotsessist visandatud skeem, mida käsitletakse selle peatüki esimeses jaotises. Liigitamine aitas kaardistada, milliseid tüüpülesandeid saab sortimise kohta moodustada. Tüüpülesanne tähistab siin sellist ülesannet, millest saab väikeste muudatuste abil kas tekstis või vastustes uue ülesande luua. Liigitamise lõpliku versiooni kujunemise ajal alustati näidisülesannete loomist, mille üheks eesmärgiks oli paremini mõista, kuidas ülesandeid liigituses jaotada.

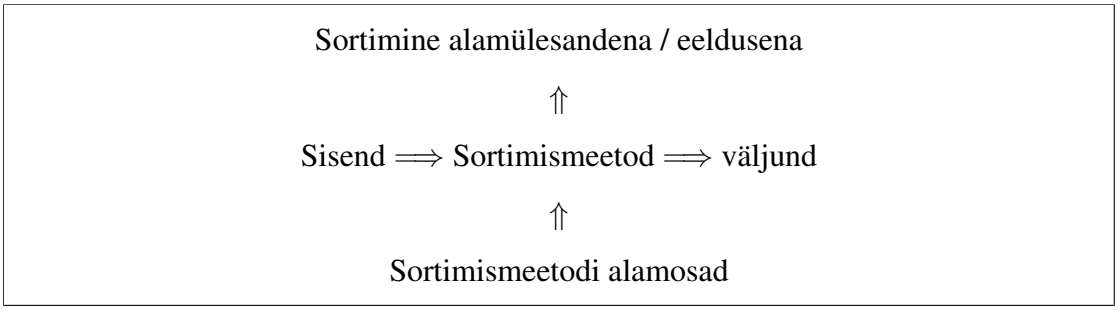
2.1 Ülesannete liigitus

Liigitamisel katsetati erinevaid meetodeid. Algul püüti lähteülesannete põhjal sortimisalgoritmide olulisi omadusi kirjeldada. Selle tulemusena valmis nimekiri, mis keskendus aga liigselt üksikasjadele ja ei toiminud ülesannete suurematesse rühmadesse koondamisel. Nimekirjale iseloomulikud kirjeldused olid näiteks:

- meetodi töö üldjuhul;
- meetodi töö erijuhul;
- arusaamine, et etteantud järjend on mingi meetodi jaoks väga hea juhtum;
- arusaamine, et etteantud järjend on mingi meetodi jaoks väga halb juhtum;
- ülesanded, kus tuleb sisendi omadustest lähtudes meetodi valikut põhjendada.

Toimiv liigitus leiti lõpuks üldiselt üksikule liikumise strateegiaga. Sortimisprotsessi vaadeldi joonisel 1 esitatud skeemi abil. Sisend tähistab selles skeemis andmestruktuuri (massiivi või ahelat), millega sorditavad elemendid meetodile ette antakse. Sortimismeetod on algoritm, mis sisendist etteantavad elemendid järjestab. Sortimismeetodi väljund saab olla korrektne või mitte. Meetodite alamosadena vaadeldakse iseseisva üksusena käsitletavaid sortimisalgoritmide osi, näiteks lahkme valik kiirmeetodi puhul.

Sortimismeetod saab olla ka probleemi lahendamise eeldus (võimaldab kahendotsingut) või alamosa (kuhjameetod prioriteedijärekorra osana).



Joonis 1. Sortimisprotsessi skeem.

Skeemi najal valmis liigitus, kus ülesanded jaotati järgnevalt:

1. Sisendist lähtuvad ülesanded. Nendes antakse ette sisendi omadus(ed), mis võivad olla näiteks:
   - sorditavate elementide arv;
   - sisendi andmestruktuur – ahel või massiiv;
   - elementide paiknemine – juhuslik, sorditud, mõned on „paigast ära“;
   - elementide jaotus – ühtlane jaotus, normaaljaotus, palju võrdseid elemente;
   - elementide omadused – kas saab ainult võrdlustel põhinevaid meetodeid kasutada või teame sisendist rohkem.
2. Sortimismeetodi omadustest lähtuvad ülesanded. Neis antakse ette meetod, mille omadusi või tööpõhimõtet kasutada tuleb. Selgemalt eristuvad:
   - ajaline keerukus – millised on meetodi ajalise keerukuse keskmine, hea ja halb juht? Millal need juhud realiseeruvad;
   - mälukeerukus – kas sortida saab sisendina antud andmestruktuuril? Sortimise funktsiooni rekursiivsete väljakutsete arvu sõltuvus sisendi elementide arvust;

- stabiilsus – kas meetod on stabiilne? Stabiilset meetodit vajava olukorra äratundmine;
- meetodi tööpõhimõte – kuidas algoritmi samme etteantud järjendile rakendatakse?

3. Ülesanded sortimismeetodite variantide kohta:
   - põimemeetodi variandid – klassikaline rekursiivne ja alt-üles;
   - kiirmeetod – kas jaotamisel jagatakse sisend kaheks või kolmeks;
   - hübriidmeetodid – mitme meetodi koos kasutamine. Näiteks sordime kiirmeetodil, aga lühikeste järjendite korral kasutame pistemeetodit.
4. Ülesanded sortimismeetodite alamosade kohta, näiteks jagamine ning lahkme valik kiirmeetodi puhul või kuhjastamine kuhjameetodi korral.
5. Sortimisele lähedaste teemade ülesanded. Siia võiks kuuluda näiteks valiku kiirmeetod (ingl *quickselect*) ja inversioonide arvu leidmine põimimisega.
6. Sortimine alamülesandena või rakendusena. Sortimismeetodit saab vaadelda teatud ülesannete alamosana, näiteks kuhjameetod prioriteedijärjekorra rakendamisel; või eeldusena teistele ülesannetele, näiteks kahendotsing.

Liigituse valmimist toetasid ka järgnevas jaotises käsitletavad näidisülesanded.

2.2 Näidisülesanded

Iga liigituse teema kohta tehti kaks või kolm näidisülesannet, mida kasutati edaspidi mallina ülesannete loomisel. Tüüpilise näidisülesande tööversioon (Näide 2.1) koosnes järgmistest osadest:

1) nimi – lühikirjeldus ülesande tekstis antu ja küsitava kohta;
2) tekst – ülesandes etteantava olukorra sõnastus;
3) vastuse liik – millist vastuse tüüpi kasutatakse. Valikvastuste korral kirjeldati võimalikke vastusevariante;

4) allikas – idee päritolu, näiteks: „Ülesannete kogu“ 7.6;

5) variandid – võimalikud muudatused uue ülesande loomiseks;

6) eesmärk – miks seda ülesannet küsitakse (õpiväljund).

Näide 2.1. *Sisendi omadustest lähtuva näidisülesande tööversioon (sisaldab ka tööversioonile omaseid vigu nagu kirjavead, anglitsismide kasutamine jt).*

Küsimuse nimi: A: sisendi kirjeldus – K: millisele meetodile halb juht

Tekst: Olgu sisendiks miljonite elementidega massiiv, millele iseloomulik lõik on selline: [..., 3, 1, 2, 3, 2, 3, 1, 3, 2, 2, 3, 3, 3, 3, 2, 1, 2, 2, 1, 1, 2, ...]. Millise sortimismeetodi jaoks on selline sisend ajalise keerukuse mõttes halb juhtum? Valige üks või mitu.

a) Valikumeetod
b) Loendamismeetod
c) Klassikaline (kaheks jagamisega) kiirmeetod
d) Alt-üles põimemeetod
e) Kuhjameetod
f) Kimbumeetod

Allikas: (Eeskujuks ÜL kogu 8.34, 8.44)

Variandid: muuta sisendit, küsida head juhtumit, muuta vastuste hulka, mitu õiget vastust vs üks õige. Kui stabiilsus oluline, siis kas merge sort (suuremal andmemahul) või pistemeetod

Eesmärk: Tudeng oskab massiivi valimi järgi sisendi omadusi hinnata, tunneb sortimismeetodi omadusi (sõltuvust sisendist).

Näidisülesannete tegemisel käsitleti ka seda, millised sisulised omadused peaksid ülesannetel olema. Ülesanded sisaldasid ainult neid sortimismeetodeid, mida kursuse loengud ja praktikumid lähemalt käsitlesid (näiteks ei küsita Shelli meetodi tööprintsiipi).

Tähtsaks peeti „Algoritmide ja andmestruktuuride“ kursuse mõistete järgimist ja kinnistamist. Kuna testid on vabatahtlikud, siis seati eesmärgiks, et ülesannete tekst oleks küll üheselt mõistetav, aga mitte väga formaalne. Kuigi testiülesanded peavad sobima kursuse konteksti, siis kaasata võib kursuse teemast väljaspool asuvaid teemasid. Näiteks õpetatakse kursusel kimbumeetodit ainult ühtlase paiskamisega ja seda piirangut tuli arvestada. Samas võib ülesande tagasisides kirjutada, et teades sisendi jaotust, saab kimbumeetodit kasutada ka muudel juhtudel kui sisendväärtuste ühtlane jaotus. Kui on näiteks teada, et sorditavaid andmeid iseloomustab normaaljaotus, siis saab jaotuse keskele rohkem kimpe luua. Testide vabatahtlikkust arvestati ka tehnilisemate ja loomingulisemate ülesannete osakaalu seadistamisel.

3 Rakendamine

Näidisülesannete eeskujul loodi ülejäänud ülesanded, mis sisestati „Algoritmide ja andmestruktuuride“ Moodle'i õpikeskkonda. Seejuures käsitleti nii ülesannete kui ka terve testikomplekti vormilist poolt.

3.1 Ülesannete sisestamine

Ülesannete sisestamisel kasutati Tartu Ülikooli elukestva õppe keskuse veebilehel asuvat Moodle'i testide loomise juhendit[2], mis pakub näiteks ülevaadet küsimuste[3] koostamisest ja testidesse lisamisest. Juhend sisaldab ka viite Moodle'i testide loomist tutvustavale e-lõuna videoloengule [21]. Valikvastustega ülesannete vormistamisel kasutati mitmeid loengus antud soovitusi:

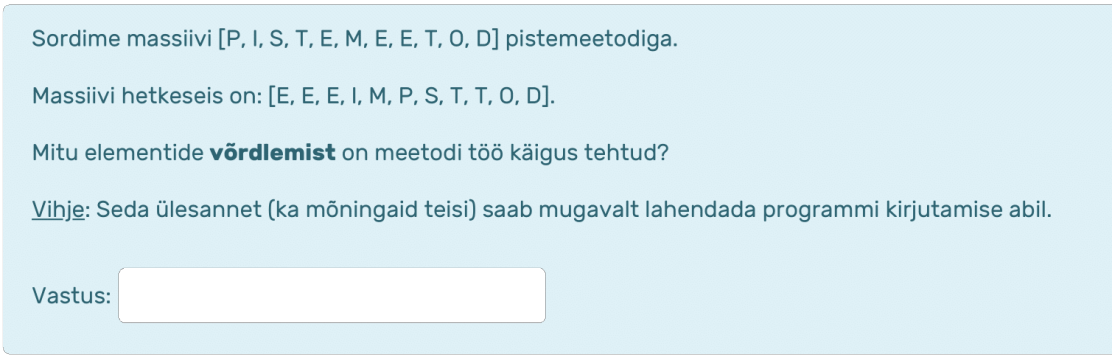
1. Soovitati hoiduda ilmselgelt valedest vastustest. Samas öeldi, et loomingulisematele ülesannetele on valesid, kuid usutavaid vastuseid keeruline kirjutada. Valminud testide lõppvariant sisaldab seetõttu vähem loomingulisemaid ülesanded, kui algul kavandati.
2. Tähtsustati vastusevariantide sarnast grammatilist ehitust ja stiili.
3. Valedele vastustele soovitati mitme õige vastuse korral miinuspunkte määrata. Vastasel korral saab kõiki vastuseid valides täispunktid.
4. Kirjeldati, et testides saab valikvastuste järjekorra juhuslikuks muuta. Seda soovitust kasutati enamike valikvastustega ülesannete puhul. Vastuste järjekord säilitati juhtudel, kus vastused olid loogilises (näiteks arvulises) järjestuses.

Suurem osa loodud ülesannetest kasutabki valikvastuseid, kuigi tehti ka arvulise vastusega ja arvutuslikke ülesandeid. Kuna teste kavatsetakse automaatselt hinnata, siis

[2] https://sisu.ut.ee/juhendid/test

[3] Siin töös on tudengile esitatava ülesande või küsimuse tähistamiseks kasutatud sõna *ülesanne*. Moodle'i keskkonnas vastab sellele mõistele sõna *küsimus*.

vabamas vormis vastuseid ei kasutatud. Näiteks muudeti massiivi elementide sisestamist nõudnud ülesannet ja kasutusele võeti valikvastused. Valikvastuste nummerdamisest loobuti selgema ülesande esituse eesmärgil. Samal põhjusel asuvad näiteks ülesandes etteantud olukord ja tudengile esitatav küsimus erinevates tekstijaotustes. Koodilõikude esiletõstmisel kasutati Javascripti võimalusi[4].



Joonis 2. Programmeerimise vihjega ülesanne.

Euroopa ja Põhja-Ameerika ülikoolide kursuseid Tartu Ülikooli omaga võrdlevas jaotises 1.3 kirjeldati, et teistes ülikoolides kasutatakse ülesannete esitamisel programmeerimiskeeli sagedamini. Selle töö käigus valminud testides otsustati koodilõike kasutada ainult kahes ülesandes. Küll aga julgustavad valminud testid tudengeid programmeerima. Kolmanda testi teises ülesandes (joonis 2) antakse vihje, et seda ja järgnevaid ülesanded võib olla mugavam programmi abil lahendada.

3.2 Testide valmimine

„Algoritmide ja andmestruktuuride“ kursusel kasutati enne seda bakalaureusetööd kahte Moodle’i testi: ajalise keerukuse ja rekursiooni teemadel. Neist võeti eeskuju küsimuste arvus: ülesanded jagati kaheks viieteistkümne küsimusega testiks. Peale töö autori

[4]Moodle’i „Kursuse failide“ „test“ kausta laaditud faili „kusimused_Raul.js“ teekond lisati samas kaustas asuvasse skriptide defineerimise faili: „kusimuste_link.js“

koostatud ülesannete kasutati kolme Küsimustepangas[5] leidunud ülesannet. Seejuures arvestati kursuse teemade järjestusega: kuna alguses läbitakse klassikalised sortimismeetodid (mulli-, valiku-, piste-, põime- ja kiirmeetod), siis sisaldab kolmas test peamiselt ülesandeid nende meetodite kohta. Neljandas testis lisanduvad loendamis-, positsiooni-, kimbu- ja kuhjameetod, mida õpetatakse koos paisktabeli ja kahendpuudega. Sellise jagamise miinuseks võib olla kolmanda testi teemade ja vastuste piiratum valik. Samas saab neljandas testis pakkuda ülesandeid, mis esitavad kolmandast testist tuttava olukorra uutel tingimustel.

Moodle'i õpikeskkonnas kannavad loodud testid pealkirju „Test 3" ja „Test 4" ning nende eelvaated asuvad töö lisades: Test 3 ja Test 4. Kumbki test sisaldab seitset tüüpülesannet, kus kasutati juhusliku küsimuse lisamist (küsimused vahelduvad testikatsetel). Kahes testis on kokku 51 ülesannet, mis asuvad Küsimustepanga kategooriates „3 - Sortimismeetodid (klassikalised)" ning „4 - Sortimismeetodid (klassikalised ja erimeetodid)". Nende alamkategooriad peegeldavad jaotises 2.1 loodud liigitust. Testide üldsätteid, nagu näiteks kirjeldus kursuse avalehel või hindamise lävend, pole seadistatud; kuid sisuliselt on testid kasutamiseks valmis.

[5]Moodle'i veebikeskkonnas hoitakse kõiki kursuse ülesandeid Küsimustepangas.

Kokkuvõte

Bakalaureusetöö eesmärk oli luua sortimismeetodite testid Tartu Ülikooli kursusele „Algoritmid ja andmestruktuurid“ (LTAT.03.005). Töö autor koostas kaks testi, mille näidised on lisas I. Töös antakse ülevaade testide loomise protsessist alates lähteülesannete kogumisest kuni Moodle'i õpikeskkonda sisestamiseni. Teste saab õppetöös kasutama hakata 2020. aasta sügissemestril.

Autori arvates saab seda tööd kasutada edasistes bakalauresetöödes, näiteks on võimalik:

1. Selle töö käigus valminud testide kvaliteedi hindamine tudengite tagasisidele ja/või testide statistikale toetudes. Näiteks võib statistika põhjal eemaldada või asendada sellised valesid valikvastuseid, mida tudengid valinud pole – need on tõenäoliselt liiga selgelt valed.
2. Töö loomise protsessist võib saada ideid uute testide loomisel „Algoritmide ja andmestruktuuride“ ainele, näiteks paisktabeli, kahendpuude või graafide teemal.
3. Testide loomisele lähenemise võrdlus: see töö lähtus sortimise sisulisest küljest, aga võimalik oleks ka alustada didaktilistest meetoditest, nagu näiteks Bloomi taksonoomia.

Autor loodab, et loodud õppevahend aitab tudengitel sortimise teemat paremini mõista ning toetab seeläbi „Algoritmide ja andmestruktuuride“ aine õpetamist.

Kasutatud kirjandus

[1] Rivoire S., Shah M., Ranganathan P., Kozyrakis C. JouleSort: A balanced energy-efficiency benchmark. *Proceedings of the ACM SIGMOD International Conference on Management of Data.* 2007, pp. 365-376. https://doi.org/10.1145/1247480.1247522.

[2] Sullivan F, Dongarra J. Guest Editors' Introduction: The Top 10 Algorithms. *Computing in Science & Engineering.* January/February 2000, pp. 22-23, vol. 2 https://doi.org/10.1109/MCISE.2000.814652.

[3] Villems A., Kusmin M., Peets M-L., Plank T., Puusaar M., Pilt L., Varendi M., Sutt E., Kusnets K., Kampus E., Marandi, T., Rogalevitš V. Juhend kvaliteetse õpiobjekti loomiseks. Tallinn: Eesti Infotehnoloogia Sihtasutus. 2012

[4] Tartu Ülikooli õppeaine „Algoritmid ja andmestruktuurid“ (LTAT.03.005) materjalid. 2019. a sügissemester. https://moodle.ut.ee/course/resources.php?id=182 (07.05.2020).

[5] Peder A., Kiho J., Nestra H. Algoritmid ja andmestruktuurid. Ülesannete kogu. Tartu Ülikooli Kirjastus, 2017.

[6] Sedgewick R., Wayne K. *Algorithms, Fourth Edition.* Addison-Wesley, 2011.

[7] Mehlhorn K, Sanders P. *Algorithms and Data Structures. The Basic Toolbox.* Springer-Verlag Berlin Heidelberg, 2008.

[8] Cormen T. H., Leierson C. E., Rivest R. L., Stein C. *Introduction to Algorithms, Third Edition* The MIT Press, 2009.

[9] Kesküla C. Rekursiooni käsitlemine selle õpetamisel. TÜ arvutiteaduse instituudi bakalaureusetöö. 2019. http://hdl.handle.net/10062/66212.

[10] Saidlo A. Paisktabeliprintsiibi õpetamine ülikoolikursustel. TÜ arvutiteaduse instituudi bakalaureusetöö. 2019. http://hdl.handle.net/10062/66275.

[11] Kalma K-J. Puude käsitlusviisid ja õpetamine. TÜ arvutiteaduse instituudi bakalaureusetöö. 2019. http://hdl.handle.net/10062/66256.

[12] Annuk A-A. Magasini ja järjekorra õpetamine ülikoolikursustel. TÜ arvutiteaduse instituudi bakalaureusetöö. 2019. http://hdl.handle.net/10062/66285.

[13] Massachusettsi Tehnoloogiainstituudi kursus "Introduction to Algorithms". Esimene vaheeksam. 2017. a sügissemester. https://learning-modules.mit.edu/service/materials/groups/186711... (07.05.2020).

[14] Massachusettsi Tehnoloogiainstituudi kursus "Introduction to Algorithms". Lõpueksam. 2019. a kevadsemester. https://learning-modules.mit.edu/service/materials/groups/243405... (02.05.2020).

[15] Princetoni Ülikooli kursus Älgorithms and Data Structures". Lõpueksam. 2019. a sügissemester. https://www.cs.princeton.edu/courses/archive... (07.05.2020).

[16] Stanfordi Ülikooli kursus "Design and Analysis of Algorithms". Lõpueksami näidis. 2020. a talvesemester. https://web.stanford.edu/class/cs161/Exams... (07.05.2020).

[17] Stanfordi Ülikooli kursus "Design and Analysis of Algorithms". Teine loeng. 2020. a talvesemester. https://web.stanford.edu/class/cs161/Lectures... (07.05.2020).

[18] Tartu Ülikooli õppeaine „Algoritmid ja andmestruktuurid“ (LTAT.03.005) arvutipraktikumide üldjuhend. 2019. a sügis. https://moodle.ut.ee/pluginfile.php/1263997... (07.05.2020).

[19] Tallinna Ülikooli aine „Algoritmid ja andmestruktuurid“ veebileht. 2018. a kevad. http://www.cs.tlu.ee/~inga/alg_andm_18/ (07.05.2020).

[20] Berkeley Ülikooli aine "Data Structures"veebileht. 2019. a sügis. http://inst.eecs.berkeley.edu/~cs61b/fa19/index.html (07.05.2020).

[21] Ly Sõõrd. Loenguvideo: Testide koostamine. Lõunatund e-õppega. https://panopto.ut.ee/Panopto... (07.05.2020).

Lisad

I. Testid

Test 3

Olgu sorditava massiivi elemendid mingist hulgast A.

Millised omadused peavad hulgal A olema, et massiivi elemente saaks järjestada?

Valige üks või mitu:

- [ ] Refleksiivsus
- [ ] Irrefleksiivsus
- [ ] Sümmeetrilisus
- [ ] Transitiivsus
- [ ] Antisümmeetrilisus

Sordime massiivi [P, I, S, T, E, M, E, E, T, O, D] pistemeetodiga.

Massiivi hetkeseis on: [E, E, E, I, M, P, S, T, T, O, D].

Mitu elementide **võrdlemist** on meetodi töö käigus tehtud?

Vihje: Seda ülesannet (ka mõningaid teisi) saab mugavalt lahendada programmi kirjutamise abil.

Vastus:

Sordime massiivi [P, I, S, T, E, M, E, E, T, O, D] pistemeetodiga. Meetodi implementatsioon eeldab, et massiivi sorditud osa on alguses tühi.

Mitmenda läbimise järel on võimalik selline massiivi seis: [E, E, E, I, M, P, S, T, T, O, D]?

Sisesta läbimiste arv.

Vastus:

Vaatleme optimeeritud mullimeetodit, kus massiivi iga läbimise järel kontrollitakse sorditud olekut.

Mitu korda läbitakse massiivi [M, U, L, L, I, M, E, E, T, O, D]?

Sisesta läbimiste arv.

Vastus:

Vaatleme klassikalist (kaheks jagamisega) kiirmeetodit massiivil.

Milline järgnevatest massiividest ajalise keerukuse mõttes **halb** sisend?

Valige üks või mitu:

- a. [2, 1, 1, 1, 2, 2, 2, 2, 2, 2]
- b. [2, 3, 4, 5, 7, 8, 9, 10, 13, 21]
- c. [15, 13, 8, 7, 6, 5, 4, 3, 2, 1]
- d. [2, 1, 4, 5, 7, 8, 9, 10, 13, 11]
- e. [-5, 143, 96, 125, 69, 63, -2, 90, 144, 20]

Vaatleme rekursiivset kiirmeetodit, mis valib lahkmeks ahela esimese elemendi ning jaotab elemendid kolmeks (väiksemad, võrdsed ja suuremad).

Millise ahela korral on kiirmeetodi väljakutsete arv **suurim**?

Valige üks:

- Tagurpidi sorditud reaalarvude järjend.
- Juhuslike reaalarvude järjend vahemikust (0, 10).
- Paljude korduvate elementidega täisarvude järjend vahemikust (0, 10).
- Peaaegu sorditud täisarvude järjend, kus vaid mõned elemendid on paigast ära.

Olgu kiirmeetodi sisendiks 25 elemendiga massiiv, kus ei esine korduvaid elemente. Lahkmeks valitakse juhuslikult üks massiivi element.

Milline on tõenäosus, et esimesel jaotamisel valitakse lahkmeks **halvim** võimalik element?

Vastamisel on lubatud eksida +/- 0.01.

Vastus:

Vaatleme klassikalist kiirmeetodit (kaheks jagamisega).

Millised pakutavatest muutustest saavutavad vastuses kirjeldatud eesmärgi?

Valige üks või mitu:

- [ ] Lahkmeks valitakse juhuslik element, et vähendada halvima juhu tõenäosust.
- [ ] Lahkme valikul kasutatakse lineaarse keerukusega meetodit mediaani leidmiseks, et garanteerida halvima juhu ajaline keerukus $O\left(n \log n\right)$.
- [ ] Sisendil kasutatakse segamist, et vähendada halvima juhu tõenäosust.
- [ ] Pikema haru väljakutse asendatakse tsükliga, et vähendada rekursiivsete väljakutsete ahelat.
- [ ] 15 elemendiga või väiksemad jaotused sorditakse pistemeetodiga, et vähendada rekursiivsete väljakutsete ahelat.

Millal tasub massiivi sortimisel eelistada klassikalist kiirmeetodit põimemeetodile?

Valige üks või mitu:

- [ ] Peame garanteerima ajalise keerukuse $O\left(n \log n\right)$.
- [ ] Andmed ei pruugi mällu ära mahtuda.
- [ ] Elementide võrdlemine on väga kulukas.
- [ ] Vajame stabiilset sortimist.
- [ ] Andmete liigutamine on väga kulukas.
- [ ] Sisend on "peaaegu" sorditud.

Vaatleme programmeerimiskeeles Python kirjutatud sortimismeetodeid.

Millised neist on stabiilsed?

Valige üks või mitu:

- [ ]
```
def sort1(a):
    N = len(a)
    for i in range(N):
        for j in range(N-1):
            if a[j+1] < a[j]:
                a[j], a[j+1] = a[j+1], a[j]
```

- [ ]
```
def sort2(a):
    N = len(a)
    for i in range(N):
        min = i
        for j in range(i+1, N):
            if (a[j] < a[min]):
                min = j
        a[i], a[min] = a[min], a[i]
```

- [ ]
```
def sort3(a):
    N = len(a)
    for i in range(N):
        for j in range(i, 0, -1):
            if (a[j] < a[j-1]):
                a[j-1], a[j] = a[j], a[j-1]
            else:
                break
```

- [ ]
```
def sort4(a):
    N = len(a)
    for i in range(N):
        for j in range(i, 0, -1):
            if (a[j] <= a[j-1]):
                a[j-1], a[j] = a[j], a[j-1]
            else:
                break
```

Logistikafirma ladu on raskeid kaste täis (vahetused on väga kulukad) ja lisaruumi on täpselt ühe kasti jagu. Igal kastil on peal silt väljastamise kuupäevaga. Firma laotöötaja peab kastid väljastamise kuupäeva järgi ümber järjestama.

Millise sortimismeetodi peaks laotöötja valima?

Valige üks:

- a. Alt-üles põimemeetod
- b. Pistemeetod
- c. Valikumeetod
- d. Mullimeetod

Arvutimuuseumis on eksponaadiks magnetlindil andmeid hoidev seade. Magnetlinti saab edasi liigutada nii, et korraga on võimalik töödelda (võrrelda ja vahetada) kahte andmekirjet. Veel on võimalik täiesti algusesse või lõppu kerimine.

Milline sortimismeetod sobib sellise seadmega andmete töötlemiseks?

Valige üks:

- Mullimeetod
- Pistemeetod
- Valikumeetod

Valikumeetodi parima juhu ajaline keerukus on $O($), mis realiseerub näiteks järjendil (valida kõige täpsem vastus):

- [1,3,5,7,9,11,13]
- [14,12,10,8,6,4,2]
- [7,3,1,5,11,9,13]
- Sobivad kõik eelnevad järjendid.
- Ükski eelnevatest järjenditest ei sobi.

Märgi meetodid, mille parima juhu ajaline keerukus on $\Theta(n)$, kus n on sorditavate elementide arv.

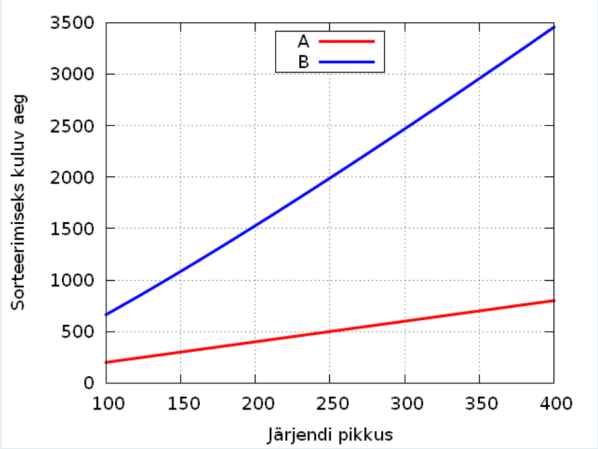
Valige üks või mitu:

- [ ] Mullimeetod
- [ ] Pistemeetod
- [ ] Kaheks jagav kiirmeetod
- [ ] Põime- ehk ühildusmeetod
- [ ] Kolmeks jagav kiirmeetod
- [ ] Valikumeetod

Graafikul on kujutatud sortimismeetodite A ja B ajakulu sõltuvalt sorditud sisendi pikkusest.

Millised järgnevatest paaridest sobivad meetoditeks A ja B?

Vastus anda kujul (A,B).



Valige üks või mitu:

- [ ] (mullimeetod, põime- ehk ühildusmeetod)
- [ ] (kiirmeetod, mullimeetod)
- [ ] (valikumeetod, kiirmeetod)
- [ ] (pistemeetod, põime- ehk ühildusmeetod)
- [ ] Ükski eelnevatest järjestatud paaridest ei sobi.

Test 4

Programm kasutab temperatuuride kirjete võrdlemiseks alljärgnevat funktsiooni

```
public int compareTo(Temperatuur teine) {
    double epsilon = 0.1;
    if (kraadid < teine.kraadid - epsilon)
        return -1;
    if (kraadid > teine.kraadid + epsilon)
        return 1;
    return 0;
}
```

Milline sortimiseks vajalik seose omadus **ei ole** selles meetodis täidetud?

Valige üks või mitu:

- [ ] Refleksiivsus
- [ ] Irrefleksiivsus
- [ ] Transitiivsus
- [ ] Sümmeetrilisus
- [ ] Antisümmeetrilisus
- [ ] Funktsioon täidab kõiki sortimiseks vajalikke nõudeid.

Vaatleme valiku kiirmeetodit massiivil.

Milline on selle algoritmi **halvima** juhu ajaline keerukus?

Valige üks:

- ○ $O(\log n)$
- ○ $O(n \log n)$
- ○ $O(n^2)$
- ○ $O(n)$

Vaatleme ühtlase paiskamisega kimbumeetodit ahelal.

Märgi ajalise keerukuse mõttes halvad sisendid.

Valige üks või mitu:

- [ ] 0.23, 0.29, 0.37, 0.38, 0.42, 0.46, 0.99, 0.59, 0.61, 0.69, 0.78
- [ ] -5, 143, 96, 125, 69, 63, -2, 90, -144, 20
- [ ] 49, 29, 10, 45, 25, 14, 53, 22, 34, 17, 15, 64
- [ ] 0.36, 0.29, 0.38, 0.31, 0.29, 0.36, 0.38, 0.27, 0.29, 0.37, 0.28
- [ ] 0.46, 0.69, 0.78, 0.61, 0.59, 0.42, 0.38, 0.27, 0.29, 0.37, 0.19, 0.02

Vaatleme algoritmi, kus n-elemendilisest massiivist k vähima elemendi massiivi algusesse ümberpaigutamiseks kasutatakse alamprotseduurina korduvalt lahkme järgi jaotamist.

Milline olukord on selle algoritmi jaoks ajalise keerukuse mõttes **halb** juht?

Valige üks või mitu:

- [ ] Lahkmeks valitakse mediaanväärtusega element
- [ ] Lahkmeks valitakse tööala keskmine element.
- [ ] Lahkmeks valitakse vähima väärtusega element.
- [ ] Lahkmeks valitakse juhuslik element.
- [ ] Lahkmeks valitakse tööala esimene element.
- [ ] Lahkmeks valitakse suurima väärtusega element.

Olgu sisendiks miljonite elementidega massiiv, millele iseloomulik lõik on selline:
[..., 3, 1, 2, 3, 2, 3, 1, 3, 2, 2, 3, 3, 3, 2, 1, 2, 2, 1, 1, 2, ...].

Millise sortimismeetodi jaoks on selline sisend ajalise keerukuse mõttes **hea** juhtum?

Valige üks või mitu:

- [ ] Positsioonimeetod
- [ ] Kuhjameetod
- [ ] Kiirmeetod (kolmeharuline variant)
- [ ] Loendamismeetod
- [ ] Kiirmeetod (kaheks jagamisega)
- [ ] Kimbumeetod

Millised väited põimemeetodi **alt-üles** variandi kohta on õiged?

Valige üks või mitu:

- [ ] Põimemeetodi alt-üles variant kasutab rohkem võrdlusi, kui klassikaline variant.
- [ ] Põimemeetodi alt-üles variandi halvima juhu ajaline keerukus on parem kui klassikalisel variandil.
- [ ] Põimemeetodi alt-üles variant kasutab rekursiooni.
- [ ] Nii alt-üles, kui klassikaline variant on stabiilsed.
- [ ] Põimemeetodi alt-üles variant kasutab klassikalise variandiga võrreldes vähem mälu.

Saja töötajaga firma kontorisse on öösel sisse murtud. Tuhanded dokumendikaustad on riiulitest maha tõmmatud ja laiali paisatud. Kahe tunni pärast saabuvad audiitorid ja dokumendid peavad selleks ajaks tähestikulises järjestuses olema. Kõik töötajad on võimalik appi kutsuda.

Millised sortimismeetod sobivad?

Valige üks või mitu:

- [ ] Pistemeetod
- [ ] Kuhjameetod
- [ ] Alt-üles põimemeetod
- [ ] Klassikaline kiirmeetod

Kuhjameetodi kuhjastamise alamosas liigutakse massiivis A esimesest elemendist alustades vasakult paremale kuni massiivi poole pikkuseni $\lfloor \frac{A.pikkus}{2} \rfloor$. Massiivi läbimisel viiakse töödeldavat elementi alla.

Millised väited kehtivad?

Valige üks või mitu:

- [ ] Meetod ei sobi, sellisel viisil kuhjastamisel peab ka lehti töötlema.
- [ ] Meetod ei sobi, sest kuhjatingimus pole täidetud.
- [ ] Meetod sobib ja tulemus on korrektne kuhi.
- [ ] Meetod ei sobi, sest kuhjas tuleb liikuda viitade abil.

Olgu teada sisendmassiiv ja mõned massiivi seisundid sortimise jooksul:

[6, 8, 9, 7, 8, 5, 6, 1]
[1, 5, 9, 7, 8, 8, 6, 6]
[1, 5, 6, 6, 8, 8, 9, 7]
[1, 5, 6, 6, 7, 8, 9, 8]
[1, 5, 6, 6, 7, 8, 8, 9]

Millist sortimismeetodit on kasutatud?

Valige üks:

- ( ) Mullimeetodit
- ( ) Valikumeetodit
- ( ) Pistemeetodit
- ( ) Klassikalist kiirmeetodit (lahkmeks valitakse esimene element)
- ( ) Kuhjameetodit

Massiiv sisaldab kahemõõtmelise ruumi punkte. Punktidel on defineeritud järjestus, mis võrdleb punkte x koordinaadi järgi ja võrdse tulemuse korral võtab arvesse y koordinaadi. Oletame, et me tahame leida ja eemaldada duplikaadid.

Milline sortimise strateegia on kõige **vähem** kasulik?

Valige üks:

- Sortida põimemeetodiga x koordinaadi järgi.
  Seejärel sortida põimemeetodiga y koordinaadi järgi.
- Sortida kiirmeetodiga x koordinaadi järgi.
  Seejärel sortida põimemeetodiga y koordinaadi järgi.
- Sortida kiirmeetodiga x koordinaadi järgi.
  Seejärel sortida kiirmeetodiga y koordinaadi järgi.
- Sortida kiirmeetodiga punktidel defineeritud järjestuse järgi.

Sortimismeetodi sisendiks on arvuti poolt ühe minuti jooksul juhuslikult genereeritud reaalarvud vahemikus (-5, 5).

Valige parima keskmise ajalise keerukusega sortimismeetod.

Valige üks:

- Kimbumeetod
- Klassikaline kiirmeetod
- Pistemeetod
- Loendamismeetod

Olgu kuhjameetodi sisendiks kuhjastatud massiiv.

Massiivi hetkeseis on [17, 17, 15, 9, 12, 17, 29].

Mitu kuhjaparanduse operatsiooni (alla viimist) sortimise algusest alates võib olla selleks hetkeks läbi viidud?

Valige üks:

- 1
- 1 või 2
- 2
- 2 või 3
- 3
- 3 või 4

Olgu programmi sisendiks N sõna, mille pikkus on 20 tähemärki. Rakendame ühte kindlat sortimismeetodit ja testime programmi tööaega erineva pikkusega sisendite puhul.

Millistel juhtudel saab püsitada hüpoteesi, et tegu on pistemeetodiga?

Valige üks või mitu:

- [ ] N = 10000, tööaeg 1 sekund
  N = 20000, tööaeg 4 sekundit
  N = 160000, tööaeg 256 sekundit
- [ ] N = 1000000, tööaeg 1 sekund
  N = 2000000 , tööaeg 4 sekundit
  N = 4000000, tööaeg 16 sekundit
- [ ] N = 10000, tööaeg 1 sekund
  N = 20000, tööaeg 2 sekundit
  N = 160000, tööaeg 16 sekundit
- [ ] N = 1000000, tööaeg 1 sekund
  N = 2000000, tööaeg 4 sekundit
  N = 4000000, tööaeg 8 sekundit
- [ ] N = 10000, tööaeg 1 sekund
  N = 20000, tööaeg 4 sekundit
  N = 80000, tööaeg 64 sekundit

Pistemeetodi sisendiks on massiiv [11, 2, 5, 9, 1, 8]. Olgu kahe elemendi asukoha vahetuse hind 5 eurot.

Kui suur on vahetuste kogumaksumus peale seda, kui element 1 jõuab massiivi algusesse?

Vastus:

Olgu teada nelja sortimismeetodi ajaline keerukus. Sorditavad järjendid sisaldavad sageli miljard või rohkem elementi.

Milline alljärgnevatest ajalistest keerukustest on parim?

Valige üks:

- $\frac{1}{3} \cdot n \cdot (\log n)^2$
- $3 \cdot n \cdot (\log n) \cdot (\log \log \log n)$
- $13 \cdot n \cdot (\log n)$
- $2 \cdot n \cdot (\log n) \cdot (\log \log n)$

II. Ülikoolide nimekiri ja kursuste viited

Kursuste veebiviited on viimati külastatud 08.05.2020.

1. Tallinna Ülikool

   Kursus: „Algoritmid ja andmestruktuurid“ IFI6083

   Kursuse veebileht: http://www.cs.tlu.ee/~inga/alg_andm_18/

2. Tallinna Tehnikaülikool TalTech

   Kursus: „Algoritmid ja andmestruktuurid“ IAS0090

   Kursuse ülevaade: http://www.tud.ttu.ee/im/Viktor.Leppikson/IAS0090%20ulevaade.pdf

3. Tallinna Tehnikaülikooli IT Kolledž

   Kursus: „Algoritmid ja andmestruktuurid“ ICD0001

   Kursuse veebileht: http://enos.itcollege.ee/~jpoial...

4. Ludwig-Maximilians-Universität München

   Kursus: „Algorithmen und Datenstrukturen“

   Kursuse veebileht: https://www.dbs.ifi.lmu.de/cms/studium...

5. Karlsruher Institut für Technologie KIT

   Kursus: „Algorithmen I“

   Kursuse veebileht: https://crypto.iti.kit.edu/index.php?id=799

6. The University of Edinburgh

   Kursus: “Informatics 2 - Introduction to Algorithms and Data Structures”

   Kursuse veebileht: https://www.learn.ed.ac.uk/webapps/blackboard...

7. Jacobs University

   Kursus: „Algorithms & Data Structures“ CH08-320201

   Kursuse veebileht: http://www.faculty.jacobs-university.de/llinsen/...

8. University of Cambridge

   Kursus: “Algorithms”

   Kursuse veebileht: https://www.cl.cam.ac.uk/teaching/1819/Algorithms/

9. Brown University

Kursus: “Introduction to Algorithms and Data Structures” CSCI0160

Kursuse veebileht: http://cs.brown.edu/courses/cs016/index.html

10. University of Houston

    Kursus: “Data Structures” COSC 2430

    Kursuse veebileht: http://www2.cs.uh.edu/~arjun/courses/ds/

11. Massachusetts Institute of Technology MIT

    Kursus: “Introduction to Algorithms” 6.006

    Kursuse veebileht: https://learning-modules.mit.edu/materials...

12. Stanford University

    Kursus: “Design and Analysis of Algorithms” CS161

    Kursuse veebileht: https://web.stanford.edu/class/cs161/index.html

13. University of California, Berkeley

    Kursus: “Data Structures” CS61B

    Kursuse veebileht: http://inst.eecs.berkeley.edu/~cs61b...

14. Princeton University

    Kursus: “Algorithms and Data Structures” COS226

    Kursuse veebileht: https://www.cs.princeton.edu/courses...

15. University of Waterloo

    Kursus: “Algorithms and Data Structures”

    Kursuse veebileht: https://ece.uwaterloo.ca/~dwharder/aads

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Raul Lehesalu**,

*(*autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Sortimismeetodite testid ainele „Algoritmid ja andmestruktuurid“,

   *(*lõputöö pealkiri)

   mille juhendaja on Ahti Peder,

   *(*juhendaja nimi)

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Raul Lehesalu

08.05.2020