TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Pihla Järv

Rakendus massiivialgoritmide läbimängude hindamiseks

Bakalaureusetöö (9 EAP)

Juhendajad: Ahti Põder, Kristo Väljako

Tartu 2024

Rakendus massiivialgoritmide läbimängude hindamiseks

Lühikokkuvõte:

Töö eesmärk oli valmistada aine „Algoritmid ja andmestruktuurid“ läbiviimise hõlbustamiseks rakendus, mis võimaldab automaatselt hinnata, kas tudengid oskavad massiivialgoritme õigesti samm-sammult rakendada. Valminud rakendus aitab vähendada õppejõudude koormust kontrolltööde hindamisel ning pakub tudengitele võimalust massiivialgoritmide läbimängu interaktiivselt harjutada. Rakendus arvestab läbimängu hindamisel nii tehtud vigade kui ka lõpliku lahenduse raskusastmega. Valminud rakendusel on lihtne tekstiline kasutajaliides. Lisaks said töö tulemusel defineeritud nõuded rakenduse graafilisele kasutajaliidesele, et rakendusele oleks võimalik teha graafiline kasutajaliides, kui see tõstetakse keskkonda DeepMOOC.

Võtmesõnad:

Algoritmid ja andmestruktuurid, sorteerimine, õpitarkvara, Java

CERCS: P175 Informaatika, süsteemiteooria

Application for Grading Array Algorithms

Abstract:

The goal of this thesis was to create an application to use in the subject Algorithms and Data Structures which would allow the automatic grading of working through the array algorithms by hand. The application helps to reduce the time needed by lecturers to grade tests. In addition, the program allows students to practice array algorithms interactively. The application grades the algorithm walkthroughs by looking at the mistakes made and the complexity of the final solution. The application has a simple text user interface. The thesis also defines the requirements for a graphical user interface for when the program gets integrated with the rest of the DeepMOOC platform.

Keywords:

Algorithms and data structures, sorting, educational software, Java

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Aine „Algoritmid ja andmestruktuurid“ on informaatika bakalaureuseõppekava kohustuslik aine, mis on mõeldud teise kursuse tudengitele [1]. Aines kasutatakse tudengite hindamiseks kodutöid ja kontrolltöid. Kodutööd koosnevad programmeerimisülesannetest ning kontrolltööd paberi peal lahendatavatest ülesannetest. Kui programmide hindamiseks kasutatakse automaatteste, siis kontrolltööde hindamisel peavad aine õppejõud praegu kõik ülesanded käsitsi üle vaatama. Tudengite arv kursusel on suur, näiteks 2023. aasta sügissemestril oli ainesse registreeritud 276 õppijat [2] ning seega on kontrolltööde hindamine küllaltki ajamahukas.

Aine „Algoritmid ja andmestruktuurid“ kontrolltöödes on ülesandeid kahte liiki. Esimesed on ülesanded, kus tuleb midagi põhjendada või tõestada. Selliseid ülesandeid ei ole võimalik automaatselt kontrollida, kuna õigeid vastuseid on palju erinevaid. Teist liiki ülesanded kontrolltöödes on sellised, kus tudeng peab mõne kursusel õpitud algoritmi tööd kindla sisendiga sammhaaval läbi tegema, et kontrollida, kas tudeng on algoritmi tööpõhimõttest aru saanud. Andmestruktuurid, mille peal tehakse kursusel läbimänguülesandeid, on massiivid, ahelad, paisktabelid ning graafid, s.h puud. Kuna läbimänguülesannete lahendused koosnevad piiratud arvust operatsioonidest, on neid võimalik automaatselt hinnata ning eesmärk on kõikide nende ülesannete kontrollimine automatiseerida. Kursuse „Algoritmid ja andmestruktuurid“ läbimänguülesannete hindamiseks valmis 2023. aastal Karolin Konradi bakalaureusetöö „Paisktabelialgoritmide läbimängu automaatse hindaja loomine“ [3] ning siinse tööga paralleelselt valmib Erik Presnovi lõputöö graafialgoritmide läbimänguülesannete hindamiseks. Kui kõigi kursusel käsitletavate läbimänguülesannete hindamise jaoks on rakendused olemas, on aine õppejõududel plaan need ühte keskkonda kokku tõsta ning hakata neid kasutama kontrolltööde automaatsel hindamisel. Samuti saaksid tudengid rakenduse abil algoritmide läbimängu harjutada. Läbimänguülesannete hindajad on plaanis panna arenduses olevale platvormile DeepMOOC. Platvormi arendusega tegeleti Märt Tenderi bakalaureusetöös „DeepMOOC platvormile tagarakenduse arendamine“ [4].

Siinse bakalaureusetöö eesmärk on koostada rakendus, mis võimaldaks automaatselt hinnata täisarvumassiividel teostatavate algoritmide läbi mängimist. Kui algoritmi läbimängul on tehtud viga, peaks rakendus suutma ka vea järel tehtud käike hinnata, kui see on võimalik. Samuti tuleb arvestada sellega, et vea tegemisel võib muutuda edasise lahenduse raskusaste.

Töö lisaeesmärgiks on panna kirja nõuded, millele peab vastama rakenduse lõplik graafiline kasutajaliides, ning teha sellest tekstiline prototüüp. Kasutajaliidese lõplikku versiooni ei saa selle töö käigus teha, kuna pole veel teada nõuded, mida on vaja täita, et kasutajaliidest oleks võimalik läbimänguülesannete ühisele platvormile integreerida ning et erinevat liiki läbimänguülesanded moodustaksid ühtse terviku.

Käesolevas töös on viis sisupeatükki. Esimeses peatükis kirjeldatakse läbimänguülesannete olemust ning nende hindamisel olulisi kriteeriume. Samuti antakse ülevaade töövahenditest. Teises peatükis kirjeldatakse tööprotsessi ning antakse ülevaade hindamisloogika kujunemisest. Kolmandas peatükis seletatakse lahti töös käsitletavad algoritmid ning kirjeldatakse, millele tuleb iga algoritmi hindamisel tähelepanu pöörata. Neljandas peatükis antakse ülevaade tööna valminud rakenduse lähtekoodi ülesehitusest ning rakenduse kasutamisest. Viimases peatükis on kirjeldatud võimalikud rakenduse edasiarendused. Lisas I on link valminud rakenduse lähtekoodile ning lisa II sisaldab valminud kasutajaliidese näidisväljundit.

1. Taustainfo

Selles peatükis on kirjeldatud, mis on läbimänguülesanded massiivil, milliseid kriteeriumeid tuleks läbimänguülesannete hindamisel kasutada ning milliseid töövahendeid kasutati rakenduse lähtekoodi valmimisel.

1.1 Läbimänguülesanded massiivil

Läbimänguülesannete puhul saab tudeng sisendiks massiivi ning eesmärk on rakendada etteantud algoritmi sellele massiivile, tehes igal sammul ühe operatsiooni massiiviga. Võimalikud operatsioonid läbimänguülesannetel on defineeritud peatükis 2.2. Iga õige sammu järel on tulemuseks massiiv, milles olevate elementide hulk on täpselt sama, nagu oli algses massiivis.

Töös käsitletakse nelja algoritmi: mullimeetod (ingl *bubble sort*), pistemeetod (ingl *insertion sort*), valikumeetod (ingl *selection sort*) ning valiku kiirmeetod (ingl *quickselect*). Esimesed kolm neist on sorteerimisalgoritmid massiividel ning viimane on algoritm massiivi teatud arvu väikseima elemendi leidmiseks. Sorteerimisalgoritmide puhul kitsendatakse ülesanne siin töös sorteerimisele mittekahanevas järjekorras.

1.2 Hindamisel arvestatavad kriteeriumid

Kui läbimänguülesande lahendamisel ei ole algoritmi täpselt järgitud, tuleb tehtud vigade eest punkte maha võtta. Kui paberi peal tehtud läbimängus on viga, siis üritavad kontrolltööde hindajad edasisest lahendusest aru saada, ning kui tundub, et tudeng on algoritmist põhimõtteliselt õigesti aru saanud, antakse talle peale viga tehtud käikude eest ikka punkte. Ka valmiv rakendus peaks hindama edasise lahenduse õigsust peale vea tegemist ning võimalusel selle eest punkte andma.

Sorteerimisalgoritmide juures aga võib vea tegemine muuta olulisel määral ülesande tehnilist keerukust. Näiteks kui võtta massiiv [7, 1, 2, 3, 4, 5, 6, 8], siis tuleb selle sorteerimiseks pistemeetodiga sooritada 6 pistet. Kui aga tehakse viga ning pistetakse esimesel sammul element 7 elemendi 5 taha, siis tuleks enne algoritmi läbimängu lõpetamist teha veel ainult üks operatsioon – pista element 6 elemendi 7 ette – ning ülesanne saaks kõigest kahe pistega lahendatud.

Seega tuleks lahenduse hindamisel arvesse võtta lisaks tehtud vigade arvule ka lahenduse keerukust võrreldes oodatud keerukusega.

1.3 Lahenduse raskusastme hindamine

Järgnevas lõigus räägitakse aastal 2021 Tartu Ülikoolis valminud Samuel Johannes Pitko bakalaureusetööst „Massiivialgoritmide sisendite genereerimine“ [5]. Pitko töö käigus leiti muuhulgas kõikide siinses töös käsitletavate algoritmide (mulli-, piste- ja valikumeetod ning valiku kiirmeetod) jaoks sobivad parameetrid, mille alusel hinnata algoritmi rakendamise keerukust massiivile, ehk nii-öelda raskusparameetrid. Ülesande keerukuse teadmine oli Pitko töös vajalik, et õppejõud saaks näiteks kontrolltöö jaoks teha ühest ülesandest mitu versiooni nii, et erinevate variantide raskusaste oleks võimalikult sarnane. Pitko töös käsitleti veel põime- , kiir- ja kuhjameetodit.

Käesolevas töös lähtutakse läbimängu raskusastme hindamisel Pitko töös leitud raskusparameetritest erinevate algoritmide korral.

1.4 Töövahendite valik

Programmeerimiskeeleks valiti Java, sest Java koodi on võimalik kasutada keeles Kotlin [6], ning just Kotlinis on kirjutatud platvormi DeepMOOC lähtekood [4]. Samuti on varasemad läbimänguülesannete hindamise automatiseerimisega seotud programmid kirjutatud Javas [3, 5]. Kuna rakenduse haldamise ja edasiarendusega tegelevad ilmselt edaspidi aine „Algoritmid ja andmestruktuurid“ õppejõud, oli Java ka loogiline valik, kuna aine õpetamisel kasutatakse samuti Javat, nii et võib eeldada, et aine läbiviijad on selle keelega tuttavad.

Rakenduse töö testimiseks kasutati raamistikku JUnit [7], mis võimaldab jooksutada automaatteste.

2. Tööprotsess ja hindamisloogika

Selles peatükis on kirjeldatud lahenduse valmimise tööprotsessi. Välja on toodud ka hindamisloogika olulised komponendid.

2.1 Tööprotsess

Otsus võimalike operatsioonide kohta

Kõigepealt tuli otsustada, kas kasutajaliides peaks pakkuma tudengile võimalikke operatsioone või peaks tudeng saama täiesti vabalt elemente massiivis ringi liigutada. Alguses tundus, et elemente peaks saama täiesti vabalt ringi liigutada, kuna nimekiri võimalikest käikudest muudaks ülesande lahendamise kasutaja jaoks liiga lihtsaks. Sellise lahenduse korral oleks rakendus pidanud numbrite järjekorra järgi massiivis aru saama, milline käik tehti, ning selle põhjal otsustama, kas see käik oli õige. Lahenduse planeerimisel aga jõuti järelduseni, et võimalikud operatsioonid peavad siiski olema ette antud, kuna isegi kui massiivi seis on peale käiku õige, on algoritmi tundmise seisukohalt oluline, milline operatsioon täpselt tehti. Nimekiri võimalikest käikudest tagab ka selle, et kasutaja teaks, milliseid operatsioone peab rakendus atomaarseteks.

Järgmised õiged käigud ja nende võimalikkus

Seejärel tuli välja mõelda, kuidas hinnata edasist lahendust, kui kasutaja on teinud algoritmi läbimängul vea. See osa oli oodatust tunduvalt keerukam. Alguses tundus, et õige oleks algoritmi läbimängu jätkata nii, et alati eeldatakse, et viimane tehtud käik oli õige, ning vastavalt viimase käigu indeksitele valida järgmine õige käik. Sellise lähenemisega aga oleks peale viga käikude tegemine võinud viia olukorrani, kus algoritm lõpetab töö, aga soovitud tulemust ei ole saavutatud. (Valiku kiirmeetodi puhul on soovitud tulemuseks see, et etteantud arv massiivi vähimaid elemente on massiivi alguses, ning muude meetodite puhul on soovitud tulemuseks see, et massiiv on mittekahanevalt sorteeritud). See tundus aga vale, kuna eelduspäraselt peaks algoritmi läbimäng ikkagi õige lahenduseni viima. Kuna leiti, et peale vea tegemist ei saa järgida algoritmi täpselt reeglite järgi, jõuti järeldusele, et tuleb iga võimaliku vale käigu jaoks leida, milline oleks järgmine õige käik, mis oleks samm edasi soovitud tulemuseni jõudmisel. Niimoodi kõikvõimalike õigete käikude kombinatsioone eraldi vaadates jõuti selgusele, et iga vea korral ei olegi võimalik leida käiku, mida võiks lugeda mõistlikuks järgmiseks käiguks. Seega otsustati, et teatud vigade korral ei peagi olema võimalik edasise lahenduse eest punkte saada.

Võimalikud operatsioonid ning vigade liigid

Algoritmide läbimängudes kasutatavad operatsioonid on kirjas peatükis 2.2. Iga operatsiooni hinnatakse tervikuna. Kui operatsioon ei ole selline, nagu peaks olema algoritmi õigel läbimängul, on see viga. Võimalikud vead otsustati liigitada kaheks: *olulised* ning *mitteolulised vead*.

Mitteoluline viga on viga, mille tegemise järel on võimalik algoritmi edasi rakendada ning sellega soovitud tulemuseni jõuda. *Mitteolulise vea* tegemisel küll selle käigu eest punkti ei anta, millal viga tehti, aga edasise lahenduse eest on ikkagi võimalik punkte saada.

Oluline viga on viga, mille tegemise järel ei leidu käiku, mida võiks lugeda järgmiseks loogiliseks käiguks ning mille tegemise järel oleks võimalik algoritmi järgides jõuda soovitud tulemuseni. Kuna algoritmi ei saa enam õigesti järgida, siis ei ole ka võimalik peale *olulise vea* tegemist edasise lahenduse eest punkte saada. Leiti, et selline otsus pole liiga karm, kuna algoritmi tundev kasutaja võiks aru saada, et algoritmi järgimine ei vii soovitud tulemuseni.

Üldistamine

Kui kõikvõimalikud õigete käikude ja nende asemel teha võidavate valede käikude kombinatsioonid olid kirja pandud, oli näha, et tegelikult ei peagi eraldi vaatama kõiki kombinatsioone. Selle asemel saab üldistada reegli, mille järgi saab leida järgmise õige käigu, nii, et see kehtib alati teatud operatsiooni tegemisel, ükskõik milline oleks olnud selle operatsiooni asemel õige operatsioon.

Algoritmide kirjeldused ja õiged käigud

Lõpuks kirjutati iga algoritmi kohta, milline on selle läbimäng käikhaaval. Iga operatsiooni kohta kirjeldati, millised on selle operatsiooni korrektsed argumendid, ning milline oleks operatsiooni järel järgmine õige käik. Algoritmide kirjapanekul lähtuti Pitko tööst [5], „Algoritmide ja andmestruktuuride“ kursuse läbimängu-slaididest [8] ning Jüri Kiho õpikust „Algoritmid ja andmestruktuurid“ [9]. Seejärel pandi kirja ka täpsustavad reeglid olukordade jaoks, mis tekivad ainult siis, kui läbimängus on olnud viga.

Iga algoritmi jaoks pandi kirja reeglid, kuidas otsustada, kas viga on *oluline* või *mitteoluline*. Eesmärgiks oli hindamisel olla võimalikult leebe: kui peale vea tegemist leidub mõni käik, mida võiks kuidagi moodi pidada järgmiseks õigeks käiguks, loeti viga *mitteoluliseks*.

Hindamisvalemi koostamine

Kui kõigi algoritmide jaoks oli hindamisloogika paigas, mõeldi välja hindamisvalem, mis arvestaks õigete ja valede käikude arvu ning ülesande suhtelist raskusastet.

Rakenduse valmimine

Kui kõik läbimänguülesannete hindamisel olulised komponendid olid välja mõeldud, kirjutati valmis rakendus, mis oskab hinnata käikude õigsust ning arvutada lahenduse eest punkte. Rakendusele kirjutati ka automaattestid ning tehti tekstiline kasutajaliides.

2.2 Võimalikud käigud

Selles peatükis on kirjeldatud kõik operatsioonid, mis esinevad töös käsitletavates algoritmides. Operatsioonide defineerimisel keskenduti sellele, et operatsioonid oleksid võimalikult üldised, et neid saaks erinevates algoritmides kasutada ning kasutajal oleks rohkem vabadust. See aitab tagada, et algoritmi mitteoskav kasutaja ei saa lahenduse eest täispunkte lihtsalt proovimise teel.

Tööala valimine

Tööala valimine on operatsioon, kus vaatluse all olev tööala muutub või see valitakse esmakordselt. Tööala valimiseks tuleb valida uue tööala esimene ning viimane element, kusjuures esimene element peab olema viimasest eespool.

Kuigi tööala valimise operatsioon ise massiivi elementide järjekorda ei mõjuta, otsustati selle operatsiooni kasutamist siiski kasutajalt küsida, kuna oskus õige tööala valida demonstreerib algoritmist aru saamist.

Elemendi pistmine

Elemendi pistmine on operatsioon, kus üks element liigutatakse kahe teise elemendi vahele või massiivi esimese elemendi ette või viimase järele. Elemendi asukoht peab pistmise tulemusel muutuma.

Kuigi reaalsuses massiivil pistmise tegemine tähendab mullina liigutamist ehk kõrvuti olevate elementide vahetamist, siis valmivas rakenduses lihtsustatakse see pistmiseks, kuna mullina liigutamine on nii triviaalne tegevus, et käsitletavate algoritmide läbimängude kontrollimisel ei ole mõtet eraldi kontrollida kasutaja oskust elementi mullina liigutada.

Kahe elemendi vahetamine

Kahe elemendi vahetamine on operatsioon, kus kahe elemendi positsioonid vahetatakse omavahel, kusjuures vahetatavad elemendid (aga mitte tingimata nende väärtused) peavad olema erinevad.

Lahkme järgi jaotamine

Lahkme järgi jaotamine on operatsioon, kus massiivi elementide asukohad muutuvad ning selle tulemusel on massiiv jagatud kaheks: massiivi vasakus otsast on elemendid, mille väärtus on lahkmest väiksem, ning paremas otsas elemendid, mille väärtus on lahkmest suurem või sellega võrdne. Lahkme järgi jaotamise tulemusel on märgitud ka kirjeldatud kahe osa vahelise piiri asukoht kas kahe elemendi vahel või massiivi alguses või lõpus.

Kuigi reaalsuses massiivi lahkme järgi kaheks jagamine tähendab elementide vahetamist, siis selles rakenduses hinnatakse lahkme järgi jagamist ühe tervikliku operatsioonina. Lahkme järgi jagamise protsessi läbimäng üksikuid vahetusi tehes võib minna ebamõistlikult pikaks, arvestades, et sisuliselt sama oskust saab kontrollida ainult ühe käiguga. Samuti on praegune variant leebem lahkme järgi jagamisel tehtavate vigade suhtes, näiteks vale lahkme valimise korral. Piiri märkimine on lahkme järgi jagamise operatsiooni osa, et lahkme järgi jagamise tulemuse järel oleks võimalik järgmist õiget käiku leida.

2.3 Raskusparameetri arvutamine

Selles ning järgmises lõigus viidatakse S. J. Pitko lõputöös [5] leitud raskusparameetritele. Pitko töös valiti mullimeetodi raskusparameetriks sorteerimisel tehtavate massiivi läbimiste arv, pistemeetodi raskusparameetriks tehtavate pistete arv, valikumeetodi raskusparameetriks tehtavate vahetuste arv ning valiku kiirmeetodi raskusparameetriks lahkme järgi jaotamiste arv.

Piste- ja valikumeetodi ning valiku kiirmeetodi raskusparameetri väärtused leiti siinses töös, lugedes kokku Pitko töös nimetatud operatsiooni esinemist terve läbimängu jooksul. Mullimeetodi puhul leiti Pitko töös raskusparameetriks massiivi läbimiste arv, aga siinses töös on mullimeetodi võimalikuks operatsiooniks elemendi pistmine. Seega leiti mullimeetodi raskusparameeter siinses töös nii, et loeti kordi, kui uue piste alguspunkt oli eelmise piste lõpp-punktist paremal.

2.4 Hindamisvalem

Lõpliku punktisumma arvutamiseks korrutatakse õigete käikude osakaal suhtelise raskusega, kuna mõlemad on olulised komponendid lahenduse hindamisel. Õigete käikude osakaal leitakse, jagades õigete käikude arvu kõikide käikude arvuga kuni *olulise veani*, kui see eksisteerib. Suhteline raskus leitakse, jagades tegeliku lahenduse raskusparameetri õige lahenduse raskusparameetriga. Kui tegeliku lahenduse raskusparameeter on suurem kui algse ülesande raskusparameeter, on suhteline raskus 1. Seega on lõplik punktisumma lõigus nullist üheni. Selline võimalik vahemik hinde väärtusele valiti, et rakendusest saadud tulemus oleks lihtsasti skaleeritav.

3. Algoritmid

Selles peatükis kirjeldatakse nelja massiivialgoritmi: mulli-, piste ja valikumeetod ning valiku kiirmeetod. Kõigi algoritmide jaoks on kirja pandud võimalikud operatsioonid. Iga operatsiooni kohta on kirjas, milline on selle operatsiooni õige sooritus kirjeldus ning milline oleks peale operatsiooni järgmine õige käik. Lisaks on iga algoritmi juures kirjas, millised vigu loetakse selle algoritmi puhul *olulisteks vigadeks*.

Kõigi algoritmide puhul on läbimängu alustamisel esimene õige käik tööala valimine.

3.1 Mullimeetod

Mullimeetod on keskmisel juhul ruutkeerukusega sorteerimisalgoritm, kus igal massiivi läbimisel viiakse vähim element massiivi etteotsa. Selle käigus liiguvad ka muud elemendid igal massiivi läbimisel oma õigetele kohtadele lähemale. Siin töös käsitletakse mullimeetodi varianti, kus algoritmi töö lõpetatakse, kui massiivi läbimisel ei muutunud ühegi elemendi asukoht.

Järgmisena kirjeldatakse võimalikke mullimeetodi operatsioone ning millised on läbimängul *olulised vead*.

Tööala valimine

Õige käigu kirjeldus

Kui tööala on valimata, siis on tööala valimisel õige valida tööalaks terve massiiv. Kui tööala on valitud, on õige käik valida uus tööala nii, et selle esimene element oleks vana tööala teine element ning uue ja vana tööala viimased elemendid kattuksid.

Järgmine õige käik

Kui peale tööala valimist leidub tööalas element, mille väärtus on väiksem kui tööalas temast vasakul oleva elemendi väärtus, siis on järgmine õige käik elemendi pistmine. Kui sellist elementi ei leidu, on järgmine õige käik läbimängu lõpetamine.

Elemendi pistmine

Õige käigu kirjeldus

Kui eelmine käik oli tööala valimine, siis on õige alustada pistetava elemendi otsimisega tööala kõige parempoolsest elemendist. Kui eelmine käik oli elemendi pistmine, siis on õige alustada otsimist eelmise piste lõpp-punktist vahetult vasakul olevast elemendist.

Elemendi pistmisel oleks õige võtta otsimise alguskohast (kaasa arvatud) alates vasakule liikudes esimene element, mis on endast vasakul olevast elemendist väiksem, ning pista ta kõigist järjestikustest vasakul endast suurematest elementidest vasakule.

Kui eelmine käik oli vale, on võimalik, et eelmine käik oli piste, mille lõpp-punkt oli tööalast väljas. Sellisel juhul on õige alustada pistetava elemendi otsimisega tööala kõige parempoolsemast elemendist.

Järgmine õige käik

Kui peale elemendi pistmist leidub tööalas piste lõpp-punktist vasakul element, mis on väiksem mõnest endast vasakul olevast elemendist, siis on järgmine õige käik elemendi pistmine. Kui sellist elementi ei leidu, on järgmine õige käik tööala valimine.

Kui eelmine käik oli vale, on võimalik, et tööala on valimata. Sellisel juhul on järgmine õige käik tööala valimine. Samuti on vale käigu korral võimalik, et tööala pikkus on üks, mille korral on õige käik läbimängu lõpetamine.

***Olulised vead* mullimeetodil**

Peale tööala valimist ja peale elemendi pistmist peavad nii tööalast eespool kui ka tagapool olevad elemendid, kui neid leidub, olema mittekahanevalt sorteeritud, sealjuures peavad tööalast eespool olevad elemendid olema massiivi vähimad elemendid ning tööalast tagapool olevad elemendid peavad olema massiivi suurimad elemendid. Kui see tingimus ei ole peale tööala valimist või elemendi pistmist täidetud, on see *oluline viga*, kuna tööalast väljas olevate elementide asukohta ei saa enam algoritmi järgides muuta.

Kui peale elemendi pistmist on tööala valitud, aga ei leidu elementi, mille väärtus oleks tööala vähim ning mis oleks piste lõpp-punktist vasakul või tööala esimene element, siis on see *oluline viga*, kuna tööala vähimat elementi ei ole võimalik enne tööala valimist algoritmi järgi tööala algusesse tuua ning seega kinnitab tööala valimine vale elemendi oma kohale.

Muul juhul on vale operatsiooni või operatsiooni valesti sooritamine *mitteoluline viga,* kuna peale eelmises lõigus kirjeldatud olukorra, on alati tööala võimalik algoritmi järgides sorteerida. Samuti on võimalik tööala valimine, kui see puudub.

3.2 Pistemeetod

Pistemeetod on keskmisel juhul ruutkeerukusega sorteerimisalgoritm, kus igal sammul suurendatakse tööala ning pistetakse tööala viimane element tööalas õigesse kohta.

Järgmisena kirjeldatakse võimalikke pistemeetodi operatsioone ning millised on läbimängul *olulised vead*.

Tööala valimine

Õige käigu kirjeldus

Kui tööala on valimata, on õige käik valida tööalaks massiivi esimene element. Kui tööala on valitud, on õige käik valida uus tööala nii, et vana ja uue tööala algused kattuksid, aga uue tööala lõpuindeks oleks vana tööala lõpuindeksist ühe võrra suurem.

Järgmine õige käik

Kui peale tööala valimist on tööala pikkus vähemalt kaks ning tööala eelviimase elemendi väärtus on suurem kui tööala viimase elemendi oma, siis on järgmine õige käik pistmine. Muul juhul on õige käik läbimängu lõpetamine, kui tööala ja massiivi viimased elemendid ühtivad, ning tööala valimine vastasel juhul.

Elemendi pistmine

Õige käigu kirjeldus

Õige käik on pista tööala viimane element tööalas paremalt esimesse võimalikku kohta nii, et pistetud elemendist vasakul oleva elemendi väärtus ei oleks pistetud elemendi väärtusest suurem.

Järgmine õige käik

Kui tööala viimane element on ühtlasi massiivi viimane element, on järgmine õige käik läbimängu lõpetamine. Muul juhul on õige käik tööala valimine.

Kui läbimängus on tehtud viga, on võimalik ka olukord, kus tööala on valimata. Sellisel juhul oleks järgmine õige käik tööala valimine.

***Olulised vead* pistemeetodil**

Kui tööala valimise või elemendi pistmise järel on olukord, kus tööala on valitud, aga tööalast eespool leidub elemente, mis pole massiivi vähimad elemendid või mis pole omavahel mittekahanevalt sorteeritud, siis on see *oluline viga*, kuna tööalast eespool olevaid elemente ei saa enam algoritmi järgides muuta.

Samuti on *oluline viga*, kui piste tegemise järel tööala on valitud, aga ei ole mittekahanevalt sorteeritud või tööala valimise järel ei ole enne tööala viimast elementi asuvad tööala

elemendid mittekahanevalt sorteeritud, kuna nende elementide omavahelist järjekorda ei saa enam algoritmi järgides muuta.

Muul juhul on vale operatsiooni või operatsiooni valesti tegemine *mitteoluline viga*, kuna kõiki tööalast tagapool olevaid elemente saab algoritmi järgides tööalas õigesse kohta liigutada. Kui viimane käik oli tööala valimine, kehtib see ka tööala viimase elemendi valimise kohta. Samuti on võimalik tööala valimine, kui see puudub.

3.3 Valikumeetod

Valikumeetod on keskmisel juhul ruutkeerukusega sorteerimisalgoritm, kus igal sammul vahetatakse tööala esimene element tööala vähima elemendiga ning seejärel vähendatakse tööala.

Järgmisena kirjeldatakse võimalikke valikumeetodi operatsioone ning millised on läbimängul *olulised vead*.

Tööala valimine

Õige käigu kirjeldus

Kui tööala on valimata, on õige valida tööalaks terve massiiv. Kui tööala on valitud, on õige valida uus tööala nii, et uue tööala esimene element oleks vana tööala teine element ning uue ja vana tööala viimased elemendid kattuksid.

Järgmine õige käik

Kui tööala pikkus on üks, on järgmine õige käik läbimängu lõpetamine. Kui tööala pikkus on üle ühe ja tööala esimene element on ühtlasi tööala vähim element, on järgmine õige käik tööala valimine. Kui tööala pikkus on üle ühe, aga tööala esimene element pole vähim, on järgmine õige käik kahe elemendi vahetamine.

Kahe elemendi vahetamine

Õige käigu kirjeldus

Õige käik on vahetada tööala minimaalne element tööala esimese elemendiga.

Järgmine õige käik

Järgmine õige käik on tööala valimine.

Kui läbimängus on tehtud viga, on võimalik, et tööala on valimata. Sellisel juhul oleks järgmine õige käik tööala valimine. Vea korral on ka võimalik, et tööala pikkus on peale elementide vahetamist üks, mille korral oleks järgmine õige käik läbimängu lõpetamine.

***Olulised vead* valikumeetodil**

Kui peale mõnda käiku ei ole tööalast väljas olevad elemendid mittekahanevalt sorteeritud või tööalast ees olevad elemendid pole massiivi vähimad või tööalast taga olevad elemendid pole massiivi suurimad, on see *oluline viga*, kuna neid elemente ei saa enam algoritmi järgides muuta.

Kui peale kahe elemendi vahetamist ei ole tööala esimene element tööala vähim element, siis on see *oluline viga*, kuna järgmisel käigul tehtav tööala valimine kinnitab tööala esimese elemendi oma valele kohale.

Muul juhul on vale operatsiooni või operatsiooni valesti sooritamine *mitteoluline viga,* kuna peale eelmises lõigus kirjeldatud olukorra on alati tööala võimalik algoritmi järgides sorteerida. Samuti on võimalik tööala valimine, kui see puudub.

3.4 Valiku kiirmeetod

Valiku kiirmeetod on keskmisel juhul lineaarse keerukusega algoritm, mille abil saab tuua etteantud arvul massiivi vähimaid elemente massiivi algusesse. Algoritm koosneb lahkme järgi jagamisest ning seejärel edasi rekursiivselt selle tulemuse ühe poole samamoodi töötlemisest, kuni lahkme järgi jagamine eraldab massiivi algusest nõutud arvul elemente.

Järgmisena kirjeldatakse võimalikke valiku kiirmeetodi operatsioone ning millised on läbimängul *olulised vead*.

Tööala valimine

Õige käigu kirjeldus

Kui tööala on valimata, on õige valida tööalaks terve massiiv. Kui tööala on valitud, siis algoritmi järgi oli eelmine käik lahkme järgi jaotamine ning seega märgiti eelmisel käigul piir, mis eraldab lahkme järgi jaotamisel leitud lahkmest väiksemad elemendid elementidest, mis on lahkmest suuremad või sellega võrdsed. Kui piir eraldab massiivi algusest rohkem elemente, kui ülesandes nõutud, on õige valida uus tööala nii, et selle algus kattuks vana tööala algusega, aga lõppeks täpselt enne piiri. Kui piir eraldab ülesandes nõutust vähem elemente, oleks õige valida tööala nii, et tööala algus oleks lahkme järgi jagamise piirist

vahetult paremal ning uue ja vana tööala lõpud kattuksid, välja arvatud juhul, kui piir on vahetult enne tööala esimest elementi, mille korral oleks õige valida uus tööala algus ühe koha võrra edasi, kuid uus tööala lõpp ikka samasse kohta, kui vana tööala lõpp.

Järgmine õige käik

Järgmine õige käik on lahkme järgi jaotamine.

Kui praegune käik oli vale, siis on võimalik olukord, kus ülesandes nõutud elementide piir jääb tööalast välja, mille korral on õige läbimängu lõpetamine.

Lahkme järgi jaotamine

Õige käigu kirjeldus

Lahkmeks on tööala esimese elemendi väärtus.

Lahkme järgi jaotamisel tuleb alates tööala otspunktidest otsida paremalt lahkmest väiksema väärtusega elementi ning vasakult lahkmest suurema või võrdse väärtusega elementi, kuni otsimiskohad kattuvad. Kui need elemendid leiduvad enne otsimiskohtade kattumist, tuleb need omavahel ära vahetada ning jätkata otsimist vahetatud elementidest seespool järgmistest elementidest ning niimoodi elementide vahetusi korrata, kuni otsimiskohad kattuvad.

Kui rohkem vahetusi ei saa teha, tuleb märkida lahkme järgi jaotamise piir tööalas kahe elemendi vahele, millest vasakpoolne on lahkmest väiksem ning parempoolne lahkmest suurem või sellega võrdne. Kui kõik tööala elemendid on lahkmest suuremad, tuleb piir märkida tööala esimese elemendi ette.

Järgmine õige käik

Kui lahkme järgi jaotamisel märgitud piir eraldab massiivi algusest ülesandes nõutud hulgal elemente, on järgmine õige käik läbimängu lõpetamine. Samuti on õige läbimäng lõpetada, kui lahkme järgi jaotamisel ühegi elemendi asukoht ei muutunud ning piir eraldab massiivi algusest ühe võrra vähem elemente, kui nõutud. Kui piir eraldab muu arvu elemente, on järgmine õige käik tööala valimine.

Kui läbimängul on tehtud viga, on võimalik, et peale lahkme järgi jaotamist on tööala valimata, sellisel juhul on järgmine õige käik tööala valimine.

***Olulised vead* valiku kiirmeetodil**

Kui peale tööala valimist jääb piir, mis eraldab ülesandes antud arvul elemente massiivi algusest, tööala esimesest elemendist ettepoole või tööala viimasest elemendist tahapoole,

aga massiiv ei ole sobivas järjestuses, on see *oluline viga*, kuna algoritmi järgi saab muuta ainult tööalas olevate elementide järjestust.

Peale tööala valimist peavad kõik elemendid, mis on enne tööala, kuuluma ülesandes antud arvu vähima massiivi elementide hulka. Kõik ülejäänud elemendid, mis kuuluvad vähimate elementide hulka, peavad olema tööalas. Kui need tingimused ei ole täidetud, on see *oluline viga*, kuna tööalast paremal olevad elemendid jäävad algoritmi järgides kindlasti ülesandes nõutud hulga esimese elemendi seast välja ning kõik tööalast vasakul olevad elemendid jäävad esimeste elementide hulka.

Kui lahkme järgi jaotamine tehakse nii, et järgmine õige käik on tööala valimine, aga see viiks eelmises lõigus kirjeldatud olukorrani, siis on see *oluline viga* eelmises lõigus kirjeldatud põhjusel. Kui peale lahkme järgi jaotamist oleks õige käik läbimängu lõpetamine, aga massiivi nõutud arvul vähimaid elemente ei ole ees, siis on see samuti *oluline viga,* kuna läbimängu tulemus on sellisel juhul vale.

Muul juhul on vale operatsiooni või operatsiooni valesti sooritamine *mitteoluline viga,* kuna tööala elementide järjekorda on võimalik algoritmi järgides muuta. Samuti on võimalik tööala valimine, kui see puudub.

4. Rakenduse ülesehitus

Selles peatükis on kirjeldatud rakenduse lähtekoodi ülesehitus. Rakenduse kirjeldamisel on tehtud mõningaid lihtsustusi ja jäetud paljud detailid mainimata, pidades silmas seda, et kirjeldus oleks lihtsasti jälgitav ning annaks rakenduse ülesehitusest üldise ülevaate.

Lisas I on link rakenduse lähtekoodile.

4.1 Hindamine ja järgmiste käikude leidmine

Massiivi seis

Klass *MassiiviSeis* hoiab massiivi seisu ühel ajahetkel. Seal on väljad täisarvumassiivi ning tööala alguse ja lõpu indeksite jaoks. Valiku kiirmeetodi puhul kasutatakse selle alamklassi, millel on lisaks veel väli, hoidmaks arvu, mis näitab, mitu vähimat elementi tuleb algoritmi tulemusena massiivi etteotsa tuua.

Massiivioperatsioon

Ühte võimalikku massiiviga tehtavat operatsiooni tähistab abstraktne klass *Massiivioperatsioon*. Selles klassis on väli, mis hoiab *MassiiviSeis*-tüüpi objekti. Lisaks on klassis abstraktsed meetodid *läbimänguOnVõimalikJätkata()*, mis tagastab tõeväärtuse, kas peale praegust operatsiooni on võimalik algoritmi läbimängu jätkata, ning *järgmineÕigeKäik()*, mis tagastab *Massiivioperatsioon*-tüüpi objekti, mis vastaks järgmisele õigele käigule, kui läbimängu jätkamine on võimalik.

Klassi *Massiivioperatsioon* laiendavad klassid, mis tähistavad konkreetset käiku, näiteks pistet ja tööala valimist. Seal on defineeritud, kuidas antud operatsioon massiivi seisu mõjutab. Neid klasse laiendavad omakorda klassid, mis tähistavad operatsiooni tegemist konkreetse meetodi korral, näiteks pistemeetodi pistet ja pistemeetodi tööala valimist. Seal on ülekaetud meetodid *läbimänguOnVõimalikJätkata()* ning *järgmineÕigeKäik()*.

Hindamistulemus ja läbimängu hindaja

Klassis *Hindamistulemus* hoitakse ühe läbimängu käikude statistikat, mida on vaja, et arvutada välja ülesande eest kujunenud punktisumma. Klassis on ka meetod, mis arvutab selle punktisumma välja.

Abstraktses klassis *LäbimänguHindaja* on meetod, mis võtab argumentiks listi *Massiivioperatsioon*-tüüpi objektidest ning tagastab *Hindamistulemuse* objekti. *LäbimänguHindaja*

klassi laiendavad igale meetodile spetsiifilised läbimängude hindajad, milles kõigis on ülekaetud ülemklassi meetod raskusparameetri arvutamiseks.

4.2 Tekstiline kasutajaliides

Kasutajaliidese jaoks on rakenduses abstraktne klass *Kasutajaliides*, milles on meetodid kolme erineva läbimänguviisi jaoks: harjutamine, kontrolltöö ja näide. Need kolm meetodit kasutavad enne kirjeldatud klasse ja meetodeid. Klassi *Kasutajaliides* laiendavad konkreetsetele algoritmidele mõeldud klassid, milles on ülekaetud meetodid kasutajale info kuvamiseks, kasutajalt saadud sisendi tõlgendamiseks ning konkreetsele algoritmile vastava alguskäigu ning läbimänguhindaja loomiseks.

Klassis *Kasutajaliides* on staatiline meetod *alusta()*, mis tegeleb sellega, et kasutajalt teada saada, millist algoritmi ning millisel viisil kasutaja soovib läbi mängida, ning loob vastava klassi *Kasutajaliides* isendi ning kutsub soovitud läbimänguviisi meetodi välja. Meetod *alusta()* kutsutakse välja rakenduse käivitamisel.

4.3 Rakenduse kasutusvoog

Rakenduse käimapanekul küsitakse kõigepealt kasutajalt, millise algoritmi läbimängu ta soovib harjutada, ning pakutakse võimalikud variandid. Kui kasutaja on sisestanud ühe meetodi nime, saab ta valida kolme läbimängu viisi vahel: harjutamine, kontrolltöö ning näide. Kui kasutaja valib näite, kuvatakse talle näide valitud meetodi läbimängust. Kui kasutaja valib harjutamise või kontrolltöö, kuvatakse talle massiiv ning nimekiri võimalikest käskudest. Võimalike käskude seas on kõik valitud meetodi rakendamisel vajalikud operatsioonid (s.h läbimängu lõpetamine) ning lisaks ka käsud viimase käigu tagasi võtmiseks ning võimalike käskude kuvamiseks. Seejärel saab kasutaja hakata järjest käske sisestama. Iga käsu sisestamise järel kuvatakse massiiv selle käsu täitmise järel. Kui kasutaja valis harjutamise formaadi, siis teavitatakse teda veast kohe peale vea tegemist ning palutakse tal uuesti proovida, kuni ta lõpuks õige käigu sisestab. Kui läbimäng on läbi, kuvatakse kasutajale kõik tehtud käigud. Kontrolltöö formaadi korral kuvatakse ka info valede käikude kohta ning läbimängu eest saadud punktid ja info nende kujunemise kohta. Kui läbimäng on lõpetatud, saab kasutaja valida, kas soovib läbimänge jätkata. Kui kasutaja vastab jaatavalt, algab kogu protsess otsast peale, ning kui eitavalt, suletakse programm.

Näited rakenduse kasutamisest on lisas II.

4.4 Testimine

Valminud lahenduse kontrollimiseks kirjutati kaks automaattesti.

Esimene test kontrollib, kas peale algoritmi alustamist alati järgmist õiget käiku tehes jõutakse algoritmi soovitud tulemuseni ning kas hindamise tulemuseks on üks punkt.

Teine test vaatab läbi kõik võimalused ühel massiivil, kus rakendatakse algoritmi, aga seejuures tehakse täpselt üks vale käik. Kontrollitakse, et kui läbimängus esineb *oluline viga*, jõutakse alati valesti järjestatud massiivini, ning kui *oluline viga* puudub, siis õigesti järjestatud massiivini. Veel kontrollitakse, kas kokku loetud vigade arv on igal läbimängul võrdne ühega ning vähemalt mõnel juhul saadakse rohkem kui null punkti. Samuti kontrollitakse, kas läbimängude seas esinevad nii *olulise* kui ka *mitteolulise veaga* läbimäng.

5. Rakenduse võimalikud edasiarendused

Tööna valminud lahendus on piisav, et harjutada nelja algoritmi läbimängu. Valminud lahendust ei saa aga veel praegusel kujul aines „Algoritmid ja andmestruktuurid“ tööde hindamiseks kasutusele võtta, kuna rakendus pole ühendatud teiste algoritmide läbimänguülesandeid hindavate rakendustega ning pole võimalust saadud punkte õppejõududele edastada. Samuti on puudu kolm kursusel olulist algoritmi ning tekstiline kasutajaliides ei ole kasutamiseks eriti mugav. Selles peatükis on kirjeldatud valminud lahenduse võimalikke edasiarendusi, mis tagaks et rakendus oleks kasutatav kursuse hindamisel ning oleks ka mugav kasutada.

Põime-, kiir- ja kuhjameetod

Algselt oli plaanis töö tulemusena valmistada rakendus, mis võimaldab läbimänguülesannete hindamist kokku kõigi seitsme Pitko töös [5] käsitletud algoritmi jaoks, aga kuna sobiva hindamisloogika välja mõtlemine osutus oodatust mahukamaks ülesandeks, jäid põime-, kiir- ja kuhjameetod tööst välja. Töö osana välja mõeldud üldine hindamise põhimõte ilmselt lihtsustab ülejäänud kolme algoritmi hindamisreeglite välja mõtlemist. Samuti on rakendus üles ehitatud nii, et uue algoritmi lisamisel ei tule palju koodi juurde kirjutada ning uute algoritmide kontrollimiseks saab kasutada juba praegu olemasolevaid automaatteste.

Põime- ja kiirmeetod on siin töös käsitletavatest algoritmidest keerulisemad, kuna nende puhul tuleb välja mõelda, kuidas leida binaarse rekursiooni puhul, milline oleks järgmine õige tööala, kui varasemal tööala valimisel on olnud viga. Kiirmeetodi puhul tuleb veel arvestada, et kui muude algoritmide jaoks leiti Pitko töös [5] sobiv raskusparameeter, siis kiirmeetodi puhul seda seal ei leitud.

Kuhjameetod on muudest käsitletud algoritmidest erinev, kuna meetodi käigus ei kujutata massiivil mitte järjestikuseid numbreid, vaid massiivil kujutatakse kuhja. Samuti on teistest algoritmidest erinev see, et kuhjameetod on jagatud kaheks eraldiseisvaks etapiks: massiivi kuhjastamine ning kuhjast elementide eemaldamine, mis teeb võimalike operatsioonide välja mõtlemise keerulisemaks. Kuhjameetodi kuvamisel kasutajaliideses tasub kaaluda, kas kuhja on mõistlik kasutajale massiivina kuvada, kuna kuhja esitamine puu kujul oleks kasutajale palju lihtsamini jälgitav.

Kasutajaliides

Kui ka teiste aine „Algoritmid ja andmestruktuurid“ läbimänguülesannete hindamiseks on rakendused valmis, on plaanis tõsta need arendatavasse keskkonda DeepMOOC. Kasutajaliidese lõplikku versiooni ei saanud selle töö käigus teha, kuna pole veel teada nõuded, mida on vaja täita, et kasutajaliidest oleks võimalik läbimänguülesannete ühisele platvormile integreerida ning et erinevat liiki läbimänguülesanded moodustaksid ühtse terviku. Seega tehti töö käigus tekstiline kasutajaliides, kuna selle tegemine oli vähem ajamahukas.

Nõuded graafilisele kasutajaliidesele

Järgnevalt on kirjeldatud tegevused, mida graafiline kasutajaliides peab võimaldama.

1. Tööala valimine. Kasutaja peab valima tööala esimese ja viimase elemendi kõigi massiivi elementide seast. Tööala esimene element ei tohi olla viimasest elemendist tagapool.
2. Elemendi pistmine. Kasutaja peab valima kõigi massiivi elementide seast elemendi, mida tuleb liigutada, ning koha kahe elemendi vahel või massiivi alguses või lõpus, kuhu valitud element liigub. Elemendi uus ja vana asukoht peavad olema erinevad.
3. Kahe elemendi vahetamine. Kasutaja peab valima massiivis kaks erinevat elementi, mille asukohad vahetatakse. Elementide valimise järjekord pole tähtis.
4. Lahkme järgi jaotamine. Kasutajal palutakse sisestada uue massiivi numbrite järjekord. Uue ja vana massiivi pikkused peavad olema samad. Lisaks palutakse kasutajal valida kahe elemendi vaheline koht või massiivi esimene või tagumine ots, mis on piiriks lahkmest suuremate ja väiksemate arvude vahel.
5. Käigu tagasi võtmine. Kui kasutaja annab teada, et soovib käiku tagasi võtta, peab rakendus viimase tehtud massiivioperatsiooni tühistama.
6. Läbimängu lõpetamine.

Kui kasutaja proovib teha mittelubatud operatsiooni, peab kasutajaliides teda keelama. Näiteks, kui kasutaja proovib valida tööala nii, et selle esimene element on viimasest elemendist tagapool.

Kasutajaliideses peavad olema näha kõik varem tehtud operatsioonid ning millised elemendid nende tulemusel muutusid. Iga operatsiooni puhul peab kasutajaliides kuvama tulemuseks saadud massiivi ning sellel tööala. Valiku kiirmeetodi puhul peab kasutajaliides

kuvama piiri, mis eraldab massiivi algusest nii palju elemente, kui ülesandes nõutud. Lahkme järgi jaotamise operatsiooni korral tuleb kuvada ka jaotamise tulemuse piir.

Käesolevas töös jääb lahendamata küsimus, kas lõplik kasutajaliides peaks kuvama kõiki võimalikke käike või ainult neid, mis on antud algoritmi puhul lubatud. Ühest küljest oleks hea, kui kasutajale pakutakse kõiki käike ning ta peab ise aru saama, millised neist üldse on antud algoritmi puhul vajalikud, et lihtsalt nuppude klõpsimisel oleks väiksem tõenäosus läbimängu eest häid punkte saada. Teisest küljest aga võib kõikide käikude olemasolu tekitada segadust ka algoritmi tegelikult tundvale inimesele: näiteks pisteoperatsioon realiseeritakse massiivis tegelikkuses järjestikuste elementide vahetusena. Kui tahta, et rakendus loeks õigeks ka piste asemel järjestikuste elementide vahetusi, tuleks rakenduse koodi täiendada nii, et iga algoritmi jaoks, kus kasutatakse piste operatsiooni, oleks realiseeritud ka vahetuse operatsioon ning sellele oleks määratud järgmine õige käik ning tingimused, mis peavad olema täidetud, et peale vahetust oleks võimalik läbimängu jätkata. Kui tahta alati lubada kasutajal kõiki käike teha, tuleb defineerida iga algoritmi jaoks ka need käigud, mida tegelikult algoritm ei kasuta.

Kokkuvõte

Töö tulemusena valmis analüüs, mida sorteerimismeetodite läbimängudel hinnata, ja rakendus, mille abil saab hinnata mulli-, piste- ja valikumeetodi ning valiku kiirmeetodi läbimängu massiivil. Hindamisel arvestatakse käikude õigsust ning kui läbimängudel on tehtud vigu, antakse ka edasise lahenduse eest punkte, kui see on võimalik. Samuti arvestatakse hindamisel lõpliku lahenduse raskusastet, kuna vigade tegemine võib lõplikku raskusastet muuta. Algselt oli plaanis mõelda välja hindamisloogika ka kiir-, põime- ja kuhjameetodi jaoks, aga kuna hindamisloogika välja mõtlemine osutus oodatust mahukamaks ülesandeks, siis jäi see teostumata.

Rakendusel on lihtne tekstiline kasutajaliides, mis võimaldab kolme erinevat läbimänguvarianti: harjutamine, kontrolltöö ja näide. Näite variant kuvab kasutajale algoritmi näidisläbimängu, harjutamise ja kontrolltöö puhul palutakse kasutajal ise käike sisestada. Harjutamise variandi puhul kontrollitakse käikude õigsust jooksvalt, kontrolltöö variandi puhul kontrollitakse tulemust alles kõige lõpus ning väljastatakse kasutajale ka saadud punktid.

Töö lõpus toodi välja võimalused rakenduse edasiarenduseks: uute algoritmide lisamine ning graafilise kasutajaliidese tegemine.

Viidatud kirjandus

[1] Informaatika bakalaureuseõppekava (2023/2024). Tartu Ülikooli arvutiteaduse instituut. https://cs.ut.ee/sites/default/files/2023-12/InfBak2023_2024.pdf (15.05.2024).

[2] Algoritmid ja andmestruktuurid (6 EAP) LTAT.03.005. ÕIS II. https://ois2.ut.ee/#/courses/LTAT.03.005/version/c8a59cf3-f5bc-3bff-f628-687d9855048b/details (14.05.2024).

[3] Konrad K. Paisktabelialgoritmide läbimängu automaatse hindaja loomine. TÜ arvutiteaduse instituudi bakalaureusetöö. 2023. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=77834 (15.05.2024).

[4] Tender M. DeepMOOC platvormile tagarakenduse arendamine. TÜ arvutiteaduse instituudi bakalaureusetöö. 2023. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=77057 (15.05.2024).

[5] Pitko S. J. Massiivialgoritmide sisendite genereerimine, TÜ arvutiteaduse instituudi bakalaureusetöö. 2021. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=72137 (15.05.2024).

[6] Get started with Kotlin. Kotlin Programming Language. https://kotlinlang.org/docs/getting-started.html (15.05.2024).

[7] JUnit 5 User Guide. JUnit 5. https://junit.org/junit5/docs/current/user-guide/ (15.05.2024).

[8] Massiiv. Läbimänguslaidid. Algoritmid ja andmestruktuurid (LTAT.03.005). Moodle - Tartu Ülikool. https://moodle.ut.ee/mod/book/view.php?id=754639&chapterid=47476 (15.05.2024).

[9] Kiho J. Algoritmid ja andmestruktuurid. Tartu: Tartu Ülikooli Kirjastus. 2003.

Lisad

I. Valminud rakenduse lähtekood

Valminud rakenduse lähtekood on leitav veebikeskkonnas GitHub aadressil https://github.com/Pihla/MassiivialgoritmideHindaja/.

II. Kasutajaliidese näidisväljundid

Paksus kirjas on kasutaja sisend ning tavalises kirjas programmi väljund.

```
Vali algoritm. [mullimeetod/pistemeetod/valikumeetod/valiku kiirmeetod]: valiku
kiirmeetod
Millist lahendusviisi soovid? [harjutamine/kontrolltöö/näide]: kontrolltöö

Alustame valiku kiirmeetodi läbimängu massiivil [5, 5, 16, 16, 8], tuua esimesed
3 elementi massiivi algusesse.
Massiiv algab indeksilt 0. Võimalikud käigud:
jaota a b _ c d e - muudab massiivi seisu, alakriips märgib jaotamise lahkme-
kohta
tööala <algusindeks> <lõpuindeks> - muudab tööala, lõpuindeks on tööalast välja
arvatud
lõpeta - lõpetab läbimängu
tagasi - võtab viimase käigu tagasi
abi - kuvab võimalikud käigud
-----------
Sisesta järgmine käik: tööala 2 5
Tööala valimine 2, 5. Uus tööala: [5 5 | 16 16 8|].
Sisesta järgmine käik: jaota 5 5 8 _ 16 16
Lahkme järgi jaotamine. Massiivi seis peale lahkme järgi jaotamist: [5 5 | 8 16
16|], lahe on indeksil 3, vastuse piir on indeksil 3.
Sisesta järgmine käik: lõpeta
Läbimängu lõpetamine.
-----------
Tehtud käigud:
Valiku kiirmeetodi läbimängu alustamine massiivil [5 5 16 16 8].
1. Tööala valimine 2, 5. Uus tööala: [5 5 | 16 16 8|]. Vale käik. Õige oleks ol-
nud Tööala valimine 0, 5. Uus tööala: [| 5 5 16 16 8|].
2. Lahkme järgi jaotamine. Massiivi seis peale lahkme järgi jaotamist: [5 5 | 8
16 16|], lahe on indeksil 3, vastuse piir on indeksil 3.
3. Läbimängu lõpetamine.
-----------
Tulemus: Raskusparameeter 1/3 ehk 0,33, õigeid käike 2/3 ehk 0,67, oluline viga
puudub, kokku punkte 0,22.

Kas soovid veel läbimänge? [jah/ei]: ei
Aitäh programmi kasutamast. Kohtumiseni!
```

```
Vali algoritm. [mullimeetod/pistemeetod/valikumeetod/valiku kiirmeetod]: piste-
meetod
Millist lahendusviisi soovid? [harjutamine/kontrolltöö/näide]: kontrolltöö

Alustame pistemeetodi läbimängu massiivil [13, 19, 17, 1, 9].
Massiiv algab indeksilt 0. Võimalikud käigud:
piste <algusindeks> <lõpuindeks> - teeb massiivil piste
tööala <algusindeks> <lõpuindeks> - muudab tööala, lõpuindeks on tööalast välja
arvatud
lõpeta - lõpetab läbimängu
tagasi - võtab viimase käigu tagasi
abi - kuvab võimalikud käigud
-----------
Sisesta järgmine käik: piste 0 1
Piste indeksilt 0 indeksile 1. Massiivi seis peale pistet: [19 13 17 1 9].
Sisesta järgmine käik: tööala 0 1
Tööala valimine 0, 1. Uus tööala: [| 19 | 13 17 1 9].
Sisesta järgmine käik: tööala 0 2
Tööala valimine 0, 2. Uus tööala: [| 19 13 | 17 1 9].
Sisesta järgmine käik: piste 1 0
Piste indeksilt 1 indeksile 0. Massiivi seis peale pistet: [| 13 19 | 17 1 9].
Sisesta järgmine käik: tööala 0 3
Tööala valimine 0, 3. Uus tööala: [| 13 19 17 | 1 9].
Sisesta järgmine käik: piste 2 1
Piste indeksilt 2 indeksile 1. Massiivi seis peale pistet: [| 13 17 19 | 1 9].
Sisesta järgmine käik: tööala 0 4
Tööala valimine 0, 4. Uus tööala: [| 13 17 19 1 | 9].
Sisesta järgmine käik: piste 3 0
Piste indeksilt 3 indeksile 0. Massiivi seis peale pistet: [| 1 13 17 19 | 9].
Sisesta järgmine käik: tööala 0 5
Tööala valimine 0, 5. Uus tööala: [| 1 13 17 19 9|].
Sisesta järgmine käik: piste 4 1
Piste indeksilt 4 indeksile 1. Massiivi seis peale pistet: [| 1 9 13 17 19|].
Sisesta järgmine käik: lõpeta
Läbimängu lõpetamine.
-----------
Tehtud käigud:
Pistemeetodi läbimängu alustamine massiivil [13 19 17 1 9].
1. Piste indeksilt 0 indeksile 1. Massiivi seis peale pistet: [19 13 17 1 9].
Vale käik. Õige oleks olnud Tööala valimine 0, 1. Uus tööala: [| 13 | 19 17 1
9].
2. Tööala valimine 0, 1. Uus tööala: [| 19 | 13 17 1 9].
3. Tööala valimine 0, 2. Uus tööala: [| 19 13 | 17 1 9].
4. Piste indeksilt 1 indeksile 0. Massiivi seis peale pistet: [| 13 19 | 17 1
9].
5. Tööala valimine 0, 3. Uus tööala: [| 13 19 17 | 1 9].
6. Piste indeksilt 2 indeksile 1. Massiivi seis peale pistet: [| 13 17 19 | 1
9].
7. Tööala valimine 0, 4. Uus tööala: [| 13 17 19 1 | 9].
8. Piste indeksilt 3 indeksile 0. Massiivi seis peale pistet: [| 1 13 17 19 |
9].
9. Tööala valimine 0, 5. Uus tööala: [| 1 13 17 19 9|].
10. Piste indeksilt 4 indeksile 1. Massiivi seis peale pistet: [| 1 9 13 17
19|].
11. Läbimängu lõpetamine.
-----------
Tulemus: Raskusparameeter 5/3 ehk 1,00, õigeid käike 10/11 ehk 0,91, oluline
viga puudub, kokku punkte 0,91.

Kas soovid veel läbimänge? [jah/ei]: ei
Aitäh programmi kasutamast. Kohtumiseni!
```

```
Vali algoritm. [mullimeetod/pistemeetod/valikumeetod/valiku kiirmeetod]: valiku-
meetod
Millist lahendusviisi soovid? [harjutamine/kontrolltöö/näide]: harjutamine

Alustame valikumeetodi läbimängu massiivil [16, 17, 4, 11, 0].
Massiiv algab indeksilt 0. Võimalikud käigud:
vaheta <indeks1> <indeks2> - vahetab 2 elementi
tööala <algusindeks> <lõpuindeks> - muudab tööala, lõpuindeks on tööalast välja
arvatud
lõpeta - lõpetab läbimängu
tagasi - võtab viimase käigu tagasi
abi - kuvab võimalikud käigud

Sisesta järgmine käik: tööala 0 5
Tööala valimine 0, 5. Uus tööala: [| 16 17 4 11 0|].
Sisesta järgmine käik: vaheta 0 4
Elementide vahetus indeksitel 0 ja 4. Uus massiiv: [| 0 17 4 11 16|].
Sisesta järgmine käik: vaheta 1 2
Elementide vahetus indeksitel 1 ja 2. Uus massiiv: [| 0 4 17 11 16|].
Vale käik. Proovi uuesti. Massiivi seis on: [| 0 17 4 11 16|]
Sisesta järgmine käik: tööala 1 5
Tööala valimine 1, 5. Uus tööala: [0 | 17 4 11 16|].
Sisesta järgmine käik: vaheta 1 2
Elementide vahetus indeksitel 1 ja 2. Uus massiiv: [0 | 4 17 11 16|].
Sisesta järgmine käik: tööala 2 5
Tööala valimine 2, 5. Uus tööala: [0 4 | 17 11 16|].
Sisesta järgmine käik: vaheta 2 3
Elementide vahetus indeksitel 2 ja 3. Uus massiiv: [0 4 | 11 17 16|].
Sisesta järgmine käik: tööala 3 5
Tööala valimine 3, 5. Uus tööala: [0 4 11 | 17 16|].
Sisesta järgmine käik: vaheta 4 5
Mõlema vahetatava indeksid peavad olema massiivi sees. Pead sisestama käigu
uuesti.
Sisesta järgmine käik: vaheta 3 4
Elementide vahetus indeksitel 3 ja 4. Uus massiiv: [0 4 11 | 16 17|].
Sisesta järgmine käik: lõpeta
Läbimängu lõpetamine.
Vale käik. Proovi uuesti. Massiivi seis on: [0 4 11 | 16 17|]
Sisesta järgmine käik: tööala 4 5
Tööala valimine 4, 5. Uus tööala: [0 4 11 16 | 17|].
Sisesta järgmine käik: lõpeta
Läbimängu lõpetamine.
Palju õnne! Oled läbimängu harjutamise edukalt läbinud.
--------
Tehtud käigud:
Valikumeetodi läbimängu alustamine massiivil [16 17 4 11 0].
Tööala valimine 0, 5. Uus tööala: [| 16 17 4 11 0|].
Elementide vahetus indeksitel 0 ja 4. Uus massiiv: [| 0 17 4 11 16|].
Tööala valimine 1, 5. Uus tööala: [0 | 17 4 11 16|].
Elementide vahetus indeksitel 1 ja 2. Uus massiiv: [0 | 4 17 11 16|].
Tööala valimine 2, 5. Uus tööala: [0 4 | 17 11 16|].
Elementide vahetus indeksitel 2 ja 3. Uus massiiv: [0 4 | 11 17 16|].
Tööala valimine 3, 5. Uus tööala: [0 4 11 | 17 16|].
Elementide vahetus indeksitel 3 ja 4. Uus massiiv: [0 4 11 | 16 17|].
Tööala valimine 4, 5. Uus tööala: [0 4 11 16 | 17|].
Läbimängu lõpetamine.

Kas soovid veel läbimänge? [jah/ei]: ei
Aitäh programmi kasutamast. Kohtumiseni!
```

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Pihla Järv,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Rakendus massiivialgoritmide läbimängude hindamiseks“ mille juhendajad on Ahti Põder ja Kristo Väljako, 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.

Pihla Järv

15.05.2024