TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kaspar Saarem

Programmide automaattestide kasutusuuring

Bakalaureusetöö (9 EAP)

Juhendaja: Reimo Palm

Tartu 2022

Programmide automaattestide kasutusuuring

Lühikokkuvõte:

Kursus “Programmeerimine” (6 EAP, LTAT.03.001) on osalejate arvu poolest üks kõige populaarsemaid kursusi Tartu Ülikoolis. Sellel kursusel on kokku 13 kodutööd ning nende kontrollimise lihtsustamiseks on kursuse läbiviijad loonud Moodle’i keskkonda automaattestid. Bakalaureusetöö eesmärk on luua kursuse juhendajatele statistiline ülevaade kodutööde lahendamise kohta ning kirjeldada tüüpilisi tudengite profiile, kes õpivad kursusel “Programmeerimine”. Lisaks sellele on koostatud loetelu kõige tüüpilistematest vigadest, mida tudengid kodutööde lahendamise juures teevad. Vigade kirjeldamise eesmärk on juhtida kursuse juhendajate tähelepanu võimalikele automaattestide poolt tuvastamata jäänud vigadele, et saada automaatteste vastavalt sellele täiendada.

Võtmesõnad: automaattestid, programmeerimise didaktika, kodutööd

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Study on the use of automated tests of programs

Abstract:

The course “Computer Programming” (6 ECTS, LTAT.03.001) is one of the most popular courses at the University of Tartu in terms of the number of participants. There are a total of 13 homework assignments in this course and the course organizers have created automated tests in the Moodle environment to make them easier to check. The purpose of this thesis is to give a statistical overview to the course instructors about the homework solving and to describe the typical profiles of students studying in the course “Computer Programming”. In addition, a list of the most common mistakes students make when solving homework has been compiled. The purpose of describing the mistakes is to draw the attention of the course instructors to possible errors not detected by the automated tests in order to supplement the automated tests accordingly.

Keywords: automated tests, programming didactics, homeworks

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

Sisukord

Sissejuhatus

Kursus “Programmeerimine” (6 EAP, LTAT.03.001) [1] on Tartu Ülikoolis Arvutiteaduse Instituudi poolt läbi viidav kursus, mis toimub iga õppeaasta sügissemestril ja kevadsemestril. See kursus on kohustuslik kõikidele informaatika õppekava täitvatele üliõpilastele. Lisaks informaatika erialale on kursus “Programmeerimine” kohustuslik veel mitmes loodus- ja täppisteaduste valdkonna õppekavas. Kursus on osutunud populaarseks vabaaineks ka väljaspool loodus- ja täppisteaduste valdkonna erialasid ning kuna loodus- ja täppisteaduste valdkond on üks kõige suurema vastuvõtukohtade arvuga valdkond Tartu Ülikoolis [2], siis on kursus “Programmeerimine” hetkel osalejate arvu poolest ka üks kõige populaarsemaid kursuseid, mida Tartu Ülikoolis on võimalik läbida. Kursust võetakse peamiselt sügissemestril, kus osalejate arv algab alates 200-st, viimastel aastatel on sügissemestri osalejate arv olnud isegi üle 300, ning õppekeel on eesti keel. Vähem on kursusel osalejaid kevadsemestril, kus osalejate arv on olnud seni viiekümne ringis ning õppekeel on inglise keel.

Bakalaureusetöö eesmärk on kursuse “Programmeerimine” kodutööde automaattestide kvaliteedi uurimine, et teada saada, kas kursusel kasutusel olevad automaattestid jätavad katmata mõningaid olulisi aspekte või annavad tudengitele liiga vähe tagasisidet. Lisaks on uurimise all ka kodutööde ülesanded. Töö autor toob lahendatud kodutööde statistika põhjal välja, millised kodutöö ülesanded olid lahendajate jaoks kõige raskemad, millised olid lihtsamad ning milline on kodutöö lahendamise aktiivsus ehk kui kaua enne tähtaega hakatakse kodutöid lahendama ja millal valmis saadakse. Kõige viimasena luuakse kokku kogutud andmete põhjal tüüpiliste tudengite profiilid ning tehakse ülevaate tüüpilistest tudengite käitumismustritest programmeerimise kursusel.

Bakalaureusetöö esimeses peatükis kirjeldatakse õppekorraldust kursusel “Programmeerimine”, automaattestide olemust ning bakalaureusetöö raames kasutatud andmestikku. Teises peatükis teostatakse kursuse “Programmeerimine” 2021. aasta sügissemestril osalejate kodutööde lahendamiste põhjal statistiline analüüs iga kodutöö iga ülesande kohta. Kolmandas peatükis luuakse teises peatükis kokku kogutud andmete põhjal ülevaade andmete jaotusest ja omavahelistest seostest ning kirjeldatakse k-keskmiste klasterdamismeetodit kasutades tudengite profiile. Neljandas peatükis kirjeldatakse tudengite kõige tüüpilisemaid vigu kodutööde lahendamisel, mida kursuse korraldajad saaksid

tulevikus automaatkontrollide täiendamise jaoks aluseks võtta. Viiendas peatükis tehakse põgus kokkuvõte eelnevatest tehtud uuringutest tudengite käitumismustrite kohta programmeerimise õppimisel. Viies peatükk võiks olla heaks aluseks programmeerimise kursusega seotud edasise uurimistöö kavandamisel.

1 Automaattestimine kursusel “Programmeerimine”

Selles peatükis käsitletakse automaatkontrollide kasutamisega seotud nüansse programmeerimise õpetamisel. Lisaks on peatükis kirjeldatud bakalaureusetöö raames analüüsitava kursuse “Programmeerimine” õppekorraldust ning bakalaureusetöö teostamiseks saadud andmestikku.

1.1 Programmeerimise õpetamine

Kuna programmeerimise puhul on tegu oskusega, siis vajab see pidevat harjutamist ja konstruktiivset tagasisidet. Matthíasdóttir toob oma uuringus [3] välja, et koolis programmeerimise õpetamise puhul peaks tasakaal olema praktika poole kaldu, mis lühidalt öeldes tähendab, et vähem teoreetilisi loenguid ja rohkem praktikume ehk harjutamist. Õppejõu poolt läbiviidavate praktikumide käigus tuleks luua tudengitega arutelu koodi kirjutamise teemal, mille käigus õppijad kirjeldavad enda kirjutatud koodi laiemale publikule ning üheskoos antakse teineteisele tagasisidet. Selline pidev arutelu loob õppijale toetava õpikeskkonna ning julgustab rohkem end väljendama ning aruteludest osa võtma. Lisaks suurendab see tõenäoliselt ka tudengi ettevalmistust praktikumide jaoks, kuna ta hakkab eeldama endapoolset aktiivset osavõttu igast praktikumi sessioonist.

Programmeerimise õpetamiseks kasutatakse üha rohkem ka automaatteste, mis on abiks tudengite poolt tehtud programmide kontrollimisel. Automaatkontrollide eeliseks õppejõudude ees on kiirus - automaattestidelt saab tagasisidet kohe pärast programmi esitamist. Barra jt [4] poolt tehtud uuringus tõdesid õppijad, et automaattestid hindasid esitatud programme õiglaselt ning tänu automaattestidele suutsid nad saavutada parema hinde ning kulutasid ülesannete lahendamisele rohkem aega, sest võimalus esitada oma lahendus piiramatu arv kordi suurendas neis soovi saavutada maksimaalne võimalik tulemus. Lisaks olid õppijad rahul ka automaattestidelt saadud tagasisidega, mis ei olnud ainult numbriline hinne, vaid sisaldas ka lühikest sõnumit selle kohta, mis oli esitatud lahenduses valesti. Barra jt arvates on õppijate poolt saadud tagasiside kergelt üllatav, sest automaattestid on näiteks trükkimisvigade osas veel küllaltki tundlikud, piisab väikesest süntaksiveast koodis ja automaattest võib tagastada tulemuseks 0, isegi kui kõik ülejäänud nõuded olid õigesti täidetud.

1.2 Automaattestid

Automaattest on inimese poolt kirjutatud programm, milles on kirjeldatud erinevaid aspekte käsitlevad funktsioonid, mis omakorda kontrollivad automaattestile etteantud programmide korrektsust. Ala-Mutka [5] toob esile, et automaattestide kasutamine pakub kiiret ja objektiivset tagasisidet kirjutatud programmi korrektsuse kohta ning seda nii tudengitele kui ka õppejõududele. Lisaks on automaattestide eeliseks nende kättesaadavus. Automaattestide kasutusaeg ei ole piiratud ning tudeng saab oma esitatud lahendusele tagasisidet kohe pärast automaattesti käivitamist, ja seda iga päev ööpäevaringselt. Kuid automaattestid ei ole võimelised andma tagasisidet terviklikult. See tähendab, et automaattestide abil on võimalik saada tagasisidet selle kohta, kas automaattestile etteantud programm tagastab õige tulemuse või kas programmis on olemas nõutud funktsioonid ja muutujad, kuid automaattestid ei anna tagasisidet tudengi kirjutatud koodi optimaalsuse ja stiili kohta.

1.3 Kursus “Programmeerimine” Tartu Ülikoolis

Bakalaureusetöö on tehtud kursuse “Programmeerimine” 2021. aasta sügissemestri andmete põhjal. Kursuse sügissemestri õppekorraldus näeb ette iganädalaselt tehtavaid teooriateste ning kodutööülesandeid ehk programmeerimisülesandeid. Programmeerimise ülesandeid lahendatakse programmeerimiskeeles Python. Kodutöid ja teooriateste ei tehta kahel nädalal, mil toimub kontrolltöö, kus kontrollitakse kursusel õppijate omandatuid teadmisi. Kodutöid ei tehta ka kursuse viimasel nädalal, mil toimuvad projektide esitlused, kus kursusel osalejad näitavad ja kirjeldavad paaristööna välja arendatud programmi, mida nad on teinud semestri teise poole vältel. Kursuse läbimiseks tuleb positiivsele tulemusele sooritada ka eksam, mis kontrollib kursuse jooksul omandatud teoreetilisi teadmisi ning oskust neid praktikas rakendada [6].

Kursuse auditoorne osa koosneb iganädalastest loengutest ja praktikumidest. Loengute eesmärk on juhatada tudeng sisse käesoleva nädala teemasse ning loengutest saadud teadmiste kontrollimiseks tuleb kursusel osalejatel lahendada vastava nädala teooriatest. Praktikumides toimub saadud teooria kinnistamine praktiliste lahenduste ehk programmide koostamise kaudu. Praktikumides lahendatakse ülesandeid, mis on abiks ka kodutööde lahendamisel.

1.4 Kodutööd kursusel “Programmeerimine”

Programmeerimise kodutöid on sügissemestril kokku 13. Iga kodutöö on erineva temaatikaga ning iga tudeng saab kodutööd esitada nii mitu korda kui ta soovib. Lahenduse õigsuse kohta annavad tudengile tagasisidet automaattestid. Ülevaate kodutööde esitamise nädalatest, iga kodutöö temaatikast ning kodutöödes olevate ülesannete arvust annab allolev tabel 1.

Tabel 1. Kodutööde teemad kursusel “Programmeerimine” [6].

Nädal	Teema	Ülesannete arv
1	Sissejuhatus	1
2	Avaldised ja lihtlaused	3
3	Tingimus- ja korduslaused	4
4	Funktsioonid	3
5	Algoritmid	3
7	Järjendid	3
8	Järjendid II	3
9	Järjendite muutmine	2
10	Andmestruktuurid	2
11	Andmestruktuurid II	2
13	Rekursioon	2
14	Rekursioon II	2
15	Mitmesuguseid algoritme	3

1.5 Kodutööde automaattestid kursusel “Programmeerimine”

Lahendatud kodutööd laaditakse üles Moodle’i keskkonda [7] ning neid kontrollitakse automaattestide abil. Lisaks kursusele “Programmeerimine” on automaattestid kasutusel veel näiteks kursustel “Objektorienteeritud programmeerimine” (6 EAP, LTAT.03.003) [8], “Automaadid, keeled ja translaatorid” (6 EAP, LTAT.03.006) [9], “Programmeerimise alused” (3 EAP, MTAT.03.236) [10] ning “Algoritmid ja andmestruktuurid” (6 EAP, LTAT.03.005) [11]. Automaattestide eesmärk on kontrollida tudengi poolt esitatud koodi õigsust ning anda vahetut tagasisidet leitud probleemidest. Iga nädala jaoks on mitu ülesannet, millest omakorda igaühe kontrollimiseks on koostatud mitu automaattesti. Tudengitele kuvatakse tagasisidena, kui mitu testi läbiti korrektselt ning mis mitteläbitud testide puhul valesti oli. Kui esitatud kood ei läbi kõiki automaatteste, siis on tudengitel võimalik kuni kodutöö

esitamise tähtajani oma koodi parandada ja esitada kodutöö uuesti, kuniks kõik testid õnnestuvad.

Joonis 1. Kursuse “Programmeerimine” automaattestide väljund 13. nädala kodutöö näitel.

Näiteks joonisel 1 on näha, et ülesande “kodu2” kontrollimiseks oli ettenähtud üks automaatkontroll ning ülesande “kodu1” kontrollimiseks kaheksa automaatkontrolli. Ülesande “kodu2” esitus läbis testi, kuid ülesande “kodu1” lahendus läbis kaheksast testist neli. Läbikukkunud testide tagasisidest on võimalik lugeda, mis läks valesti ning seejärel on võimalus oma esitust parandada ning esitada kodutöö uuesti. Näiteks ülesande “kodu1” viies test kukkus läbi, sest ülesandes nõutud funktsioon ei tagastanud väärtust nõuetekohaselt.

1.6 Andmestiku kirjeldus

Bakalaureusetöös kasutatav andmestik on saadud kursuse “Programmeerimine” Moodle’i õpikeskkonnast [7], kuhu on salvestatud kõikide kursusel osalejate kõikide esituste kaustad koos kellaaegadega. Moodle´i keskkonnas on võimalik zip-failina alla laadida iga kodutöö kaust. Kodutöö kaust sisaldab kõikide selle kodutöö lahendajate kaustasid ning kodutöö lahendaja kaust sisaldab tema esituste kaustasid koos kellaaegadega. Iga esituse kaust sisaldab kõiki selle esitusega esitatud kodutöö ülesandefaile. Lisaks on töös kasutatud eraldi andmestikku ka kodutööde automaattestide tulemuste kohta, mis on Exceli formaadis. Selline

Exceli fail on olemas iga kodutöö kohta ning iga fail sisaldab kodutöö jaoks kasutatud automaattestide tulemusi, kus tulemused on kirja pandud kujul 1 ja 0. Kui tulemus on 1, siis see tähendab, et automaattest õnnestus, ning kui tulemus on 0, siis märgib see ebaõnnestunud automaattesti. Exceli failides on veerg tudengi nimega ning veerud automaattestide märgistamiseks kujul x-y, kus x tähistab kodutöö ülesande järjekorranumbrit ja y tähistab selle ülesande kontrollimise tarbeks loodud automaattesti järjekorranumbrit.

Mõlemas andmestikus on kirjed kõikide kursusel tehtavate kolmeteistkümne kodutöö kohta ning mõlema andmestiku analüüsimiseks koostati vastavad programmid, mis on ära kirjeldatud lisas 1. Igas Moodle´i keskkonnast alla laaditud kodutöö kaustas on olemas ka fail report.txt, milles on kirjeldatud kodutöö esitajate arv ning esitatud lahenduste arv. Kirjeldatud Exceli failides on olemas kõik automaattestide tulemused iga kodutöö iga ülesande kohta, välja arvatud 3. nädala kodutöö 2. ülesanne, millel puudus automaatkontroll ning seetõttu ka automaattestide tulemused. Kõik kirjeldatud kodutööde kaustad, automaattestide tulemuste Exceli failid ning ka ülesannete kontrollimise jaoks loodud automaattestide failid on talletatud Tartu Ülikooli Owncloud´i keskkonnas ning nendele ligipääsu saamiseks tuleks pöörduda kursuse “Programmeerimine” vastutavate õppejõudude poole. Privaatsuse tagamiseks neid faile selles töös avalikustada ei saa, kuna nad sisaldavad õppijate isikuandmeid.

2 Kodutööde lahendamise statistiline analüüs

Töö autor kogus kokku andmed ning koostas diagrammid kursuse “Programmeerimine” 2021. aasta sügissemestril esitatud kodutööde kohta. Andmete põhjal arvutati iga kodutöö iga ülesande kohta järgnevate näitajate keskmised väärtused: esituste arv, esitatud failisuurus, muudatuse suurus kahe erineva esitatud versiooni vahel (failisuuruse erinevusena ja Levenshteini kauguse [12] järgi), lahenduse õigsus ehk automaattestide läbimisprotsent. Seda tehti lisas 1 toodud Pythoni programmide abil, kus anti sisendiks kodutöö nimi ja ülesande nimi sõnena ning programm väljastas seejärel kirjeldatud info selle kodutöö ülesande kohta. Lisaks sellele leiti ka Pythoni mooduli NumPy abil keskmised tulemused kodutööde esitamisaegade kohta: esituse aeg enne tähtaega, kui kaua enne tähtaega alustati kodutöö esitamisega ja lõpetati kodutöö esitamine ning millal oli kodutööde esitamise kõrghetk ehk mediaanaeg. Kõrghetke puhul on mõeldud hetke, mil täpselt pooled tudengid on töö esitanud.

Muudatuse suuruse arvutamisel kasutati kahte meetodit: failisuuruse erinevus ja Levenshteini kaugus. Failisuuruse ühikuks on baidid ning failisuuruse erinevust kirjeldavad tabeli 2 veerud E ja F. Levenshteini kaugust kirjeldab veerg G, kus faili on käsitletud kui ühte pikka sümbolitest koosnevat järjendit ning kahe erineva ülesande versiooni puhul arvutatud Levenshteini kaugus näitab, kui mitu sümbolit tuleks ühes failis muuta, eemaldada ja/või lisada, et see fail oleks võrreldava failiga identne. Sümbolite hulka on lisaks tähtedele, numbritele ja kõikidele kirjavahemärkidele arvestatud ka näiteks tühikuid ja reavahetusi. Tabelis 2 on näidatud eespool kirjeldatud andmed iga nädala kodutöö iga ülesande kohta ning neid ülesandeid oli kokku 33. Tabelis olevate ülesannete nimedes on selge muster, kus iga ülesande eesliiteks on ‘kodu’ ning sellele järgneb number, mis näitab, mitmenda ülesandega on tegu. 2. nädala kodutöö ülesannete hulgas algavad ülesandefailid numbriga 2 ning puudub ülesandefail “kodu1”, mis tähendaks justkui puuduks sellel kodutööl esimene ülesanne või oleks see aine korraldajate poolne viga ülesannete nimetamisel. Tegelikult oli 2. nädala kodutöö esimeseks ülesandeks tutvuda Pythoni dokumentatsiooniga, kus failina midagi esitada ei tulnud. 8. nädala kodutöö teise ülesande nimi on “kodu2” asemel “film” ning seda seetõttu, et ülesande koostaja otsustas, et kuna see ülesanne on vaja “kodu3” ülesande lahendamisel moodulina sisse lugeda, siis peaks 8. nädala kodutöö teise ülesandefaili nimeks jääma “film”.

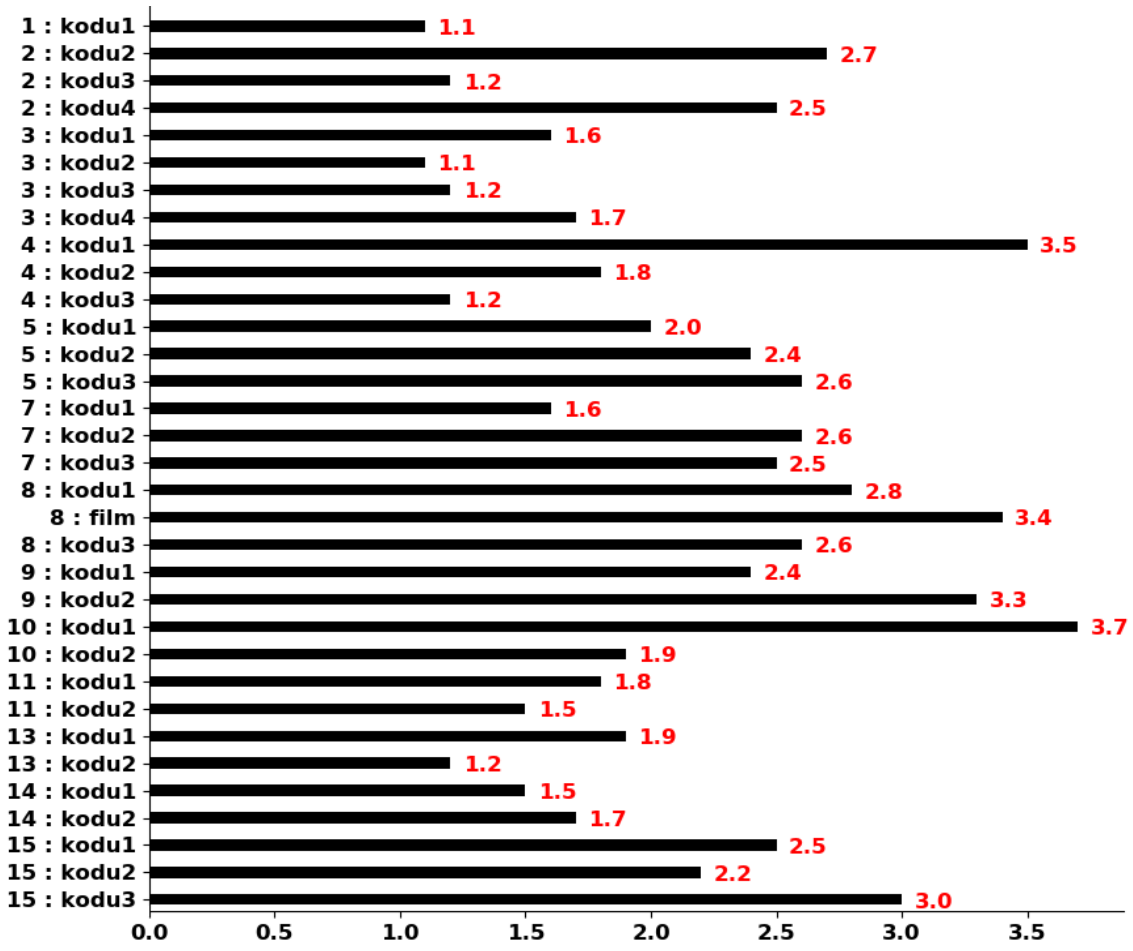
Tabel 2. Andmed iga kodutöö iga ülesande kohta.

A - kodutöö nädal, B - ülesandefaili nimi, C - keskmine versioonide arv, D - keskmine failisuurus baitides, E - keskmine muudatuse suurus baitides kahe erineva versiooni vahel, F - keskmine muudatuse suurus protsentides (E/D), G - keskmine Levenshteini kaugus kahe erineva versiooni vahel, H - keskmine muudatuse suurus protsentides (G/D), I - keskmine automaattestide läbimise protsent iga esitatud versiooni kohta, J - ülesande esitajate arv.

A	B	C	D	E	F	G	H	I	J
1	kodu1	1.1	533	16	3%	20	4%	100%	368
2	kodu2	2.7	327	26	8%	31	9%	77%	353
2	kodu3	1.2	206	16	8%	16	8%	90%	353
2	kodu4	2.5	366	35	10%	45	12%	30%	348
3	kodu1	1.6	450	18	4%	21	4%	85%	334
3	kodu2	1.1	484	35	7%	35	7%	-	332
3	kodu3	1.2	302	25	8%	25	8%	71%	324
3	kodu4	1.7	320	28	9%	33	10%	60%	314
4	kodu1	3.5	766	39	5%	66	9%	40%	321
4	kodu2	1.8	461	45	10%	54	12%	67%	315
4	kodu3	1.2	466	41	9%	43	9%	71%	304
5	kodu1	2.0	926	40	4%	51	6%	89%	331
5	kodu2	2.4	487	42	9%	58	12%	56%	326
5	kodu3	2.6	601	59	10%	70	12%	60%	316
7	kodu1	1.6	377	26	7%	32	8%	93%	315
7	kodu2	2.6	640	48	8%	58	9%	65%	310
7	kodu3	2.5	894	58	6%	63	7%	61%	306
8	kodu1	2.8	328	37	11%	43	13%	46%	291
8	film	3.4	933	85	9%	122	13%	66%	282
8	kodu3	2.6	1050	100	10%	114	11%	22%	249
9	kodu1	2.4	862	79	9%	102	12%	58%	291
9	kodu2	3.3	297	37	12%	44	15%	52%	286
10	kodu1	3.7	1171	46	4%	56	5%	61%	286
10	kodu2	1.9	1882	220	12%	238	13%	53%	265
11	kodu1	1.8	1050	55	5%	72	7%	60%	270
11	kodu2	1.5	439	42	10%	49	11%	59%	260
13	kodu1	1.9	193	10	5%	14	7%	79%	280
13	kodu2	1.2	463	55	12%	57	12%	79%	262
14	kodu1	1.5	322	21	7%	26	8%	80%	236
14	kodu2	1.7	795	80	10%	91	11%	45%	204
15	kodu1	2.5	330	15	5%	19	6%	64%	213
15	kodu2	2.2	494	47	10%	49	10%	75%	203
15	kodu3	3.0	983	98	10%	119	12%	33%	190

2.1 Keskmine versioonide arv

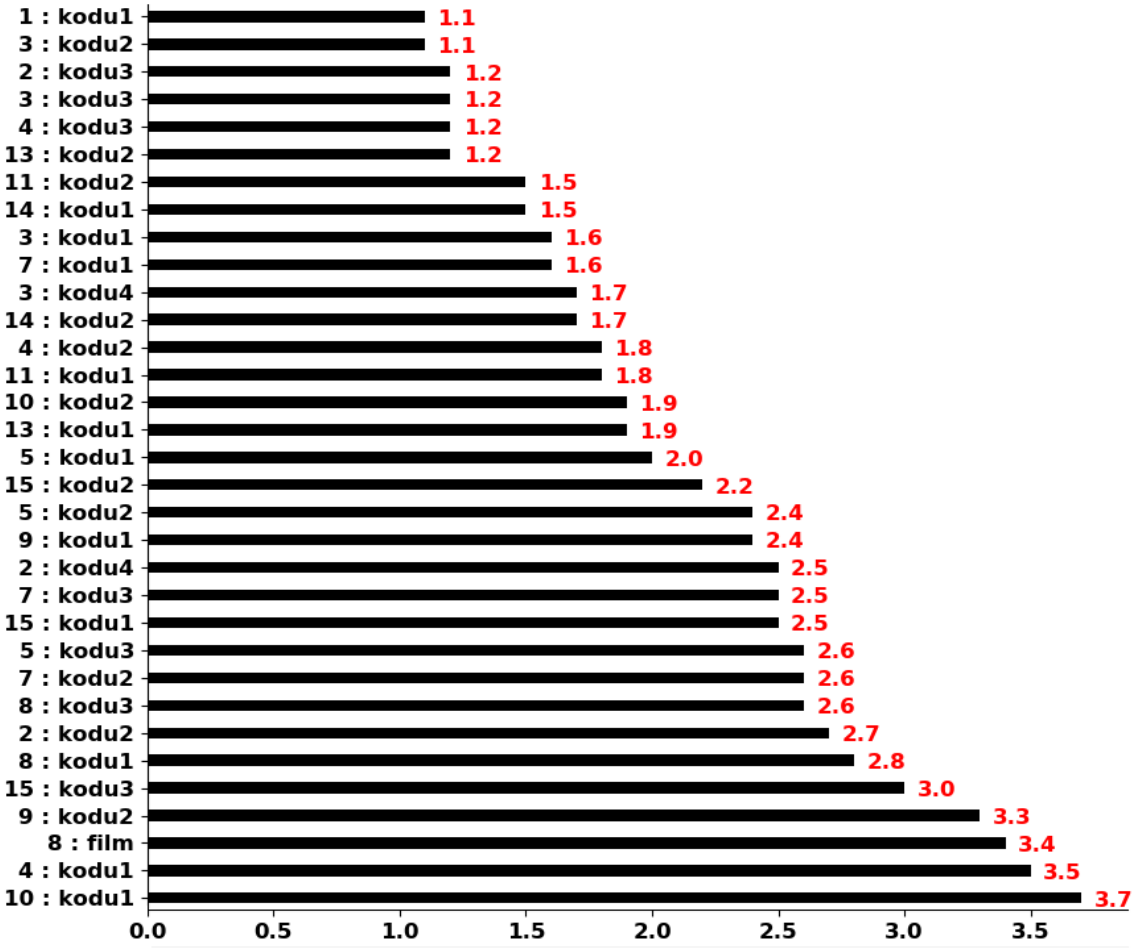
Keskmine versioonide arv näitab, kui mitu korda esitati keskmiselt ühte ülesannet, sealjuures arvestades vaid selliste esitustega, mis erinesid eelmisest esitusest. Näiteks, kui tudeng esitas kodutööd, milles oli kokku kaks ülesannet, kokku kolm korda, kuid esimest ülesannet peale esimest esitamist enam ei muutnud, vaid muutis vaid teist ülesannet, siis läheb arvesse, et see tudeng esitas esimest ülesannet kokku ühe korra ning teist ülesannet kokku kolm korda.



Joonis 2. Kodutöö ülesannete keskmine versioonide arv nädala lõikes.

Joonisel 2 on kujutatud kursuse “Programmeerimine” 2021. aasta sügissemestri kodutööde ülesannete keskmine versioonide arv kodutööde lahendamise järjekorras. Joonise 2 põhjal võib järeldada, et kodutöö ülesannete raskused kõikusid läbi kursuse üles-alla. Kursus algas ühe lihtsa kodutööga, seejärel olid 2. kodutöö ülesanded jällegi tunduvalt raskemad (“kodu2”, “kodu4”), siis tuli jällegi veidike lihtsam kodutöö, peale mida ülesannete keerukus taas suurenes ja vähenes. Jooniselt 2 paistab silma, et keskmiste esituste arvu järgi kodutöö

raskusi hinnates nõudsid kõige rohkem tööd kursuse teise kolmandiku kodutööd. Üks kõige raskematest ülesannetest tundus olevat 15. kodutöö 3. ülesanne, kus lisaks kõrgemale keskmiste versioonide arvule oli väike ka joonisel 8 kujutatud vastav õigsuse protsent.



Joonis 3. Kodutöö ülesannete keskmine versioonide arv kasvavas järjekorras.

Joonisel 3 on kujutatud sama info nagu joonisel 2, kuid keskmiste versioonide arv on kasvavas järjekorras. Joonisel 3 on näha, et kõige raskem oli 10. nädala kodutöö ülesanne "kodu1", mida esitati keskmiselt 3.7 korda. 10. nädala teema oli "Andmestruktuurid", kus käsitleti erinevaid andmetüüpe, mille abil saab infot esitada. Tudengitele anti ülesandeks koostada kolm funktsiooni, kõik erinevate andmetüüpide peale [13]. Kuna see ülesanne oli lahendamise järjekorras määratud esimeseks ülesandeks ning seal oli palju detaile, mida tuli tähelepanelikult järgida, siis võisid niivõrd suure keskmiste esituste arvu põhjuseks olla raskus sõnastiku ja muude andmestruktuuride mõistetest aru saamisel ning tudengite tähelepanematusest tingitud näpuvead.

Kõige kergemateks kodutöödeks võib joonise 3 põhjal pidada 1. nädala kodutöö ülesannet, mille eesmärgiks oli tutvustada Pythoni põhilisi töövahendeid ning anda lühiülevaade programmeerimise olemusest [14] ning 3. nädala kodutöö ülesannet kodu2, mille sisuks olid tingimus- ja korduslaused [15]. Üldiselt jääb mulje, et kui hinnata kodutööde raskust keskmiste esitatud versioonide arvu põhjal, siis see varieerub üsna ühtlaselt. Ei paista välja liiga suuri hüppeid ja eristuvaid ülesannete rühmi, vaid raskuste skaalal on eri raskused võrdlemisi ühtlaselt esindatud.

2.2 Kodutöö esitamise aeg

Töö autor uuris ülesannete esitamise aega neljas aspektis: kodutöö kõige esimese versiooni esitamise aeg enne tähtaega ehk millal kodutöö tegemist alustati, kodutöö keskmine esitamise aeg enne tähtaega, kodutöö kõige viimase versiooni esitamise aeg enne tähtaega ehk kui kaua enne tähtaega lõpetatakse kodutöö esitamine ning kodutöö esitamise mediaanaeg ehk kui kaua enne tähtaega toimus kodutöö esitamise kõrghetk. Saadud tulemused koondati tabelisse 3, kus on lisaks esitatud ka iga kodutöö esitamise tähtaeg.

Lisaks tabelile 3 koostati eelpool mainitud nelja aspekti näitamiseks tulpdiagrammi (joonis 4), kus kõik kirjeldatud kodutööde esitamiste ajad enne tähtaega on arvutatud tundides. Tulpdiagrammil olevad tulbad kasvavad paremalt vasakule, diagrammi parem äär tähistab kodutööde esitamise tähtaega ning tulba pikkus näitab esituse aega enne tähtaega.

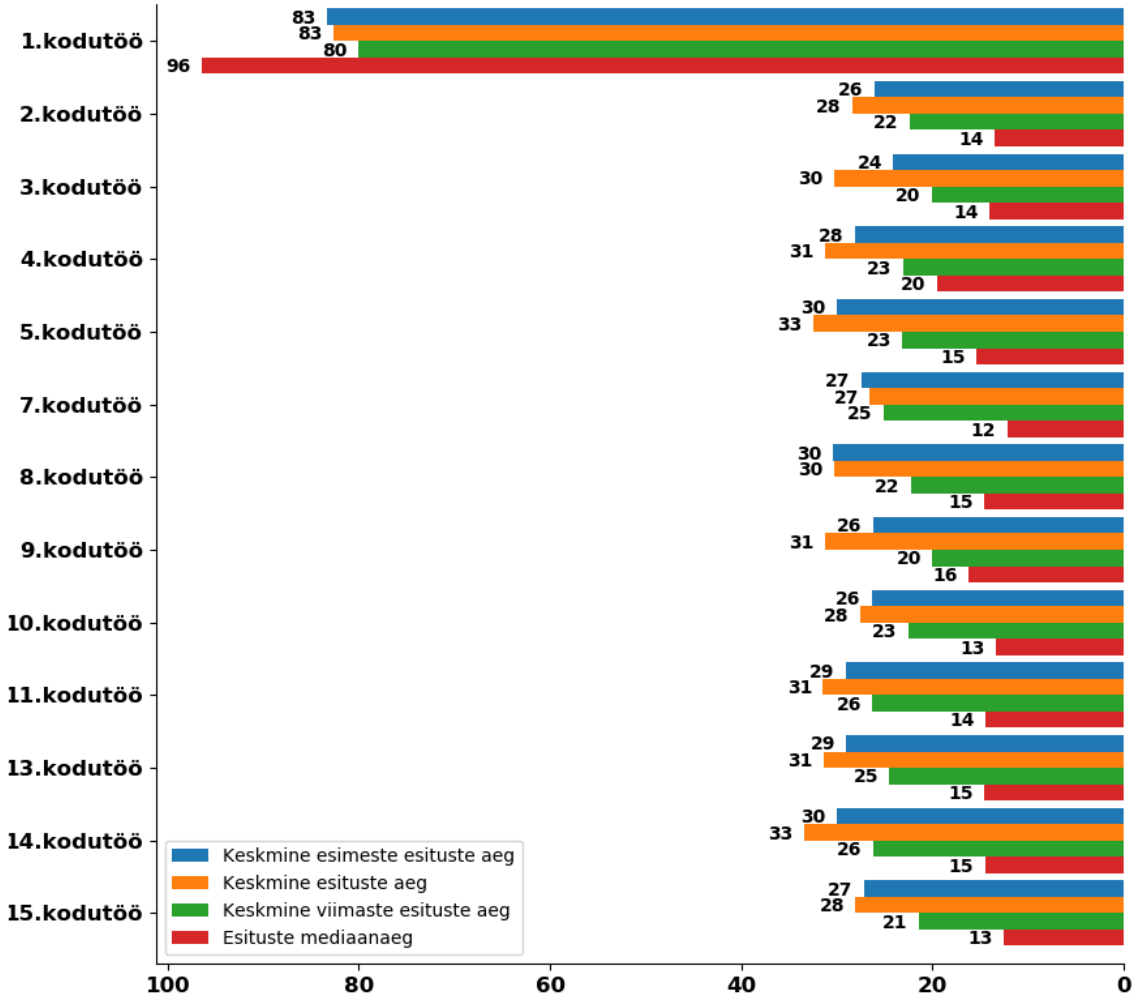
Tabel 3. Kodutööde esitamiste ajad kursusel “Programmeerimine”.
A - kodutöö nädal, B - kodutöö esitamise tähtaeg, C - kodutöö keskmine esitusaeg enne tähtaega, D - esituste mediaanaeg enne tähtaega (esitamise kõrghetk), E - keskmiselt kui kaua enne tähtaega alustatakse kodutöö esitamisega (esimene versioon), F - keskmiselt kui kaua enne tähtaega lõpetatakse kodutöö esitamine (viimane versioon). Veergudes C, D, E ja F on: p = päevad, t = tunnid, m = minutid.

A	B	C	D	E	F
1	05.09.2021 23:59	3p 10t 59m	4p 0t 39m	1p 11t 38m	3p 8t 6m
2	09.09.2021 08:00	1p 4t 37m	0p 13t 50m	1p 2t 11m	0p 22t 38m
3	16.09.2021 08:00	1p 6t 31m	0p 14t 9m	1p 0t 13m	0p 20t 10m
4	23.09.2021 08:00	1p 7t 28m	0p 19t 50m	1p 4t 11m	0p 23t 13m
5	30.09.2021 08:00	1p 8t 52m	0p 15t 43m	1p 6t 1m	0p 23t 15m
7	14.10.2021 08:00	1p 2t 56m	0p 12t 22m	1p 3t 41m	1p 1t 12m
8	21.10.2021 08:00	1p 6t 35m	0p 14t 55m	1p 6t 47m	0p 22t 30m
9	28.10.2021 08:00	1p 7t 20m	0p 16t 24m	1p 2t 21m	0p 20t 1m
10	04.11.2021 08:00	1p 3t 55m	0p 13t 39m	1p 2t 39m	0p 22t 55m
11	11.11.2021 08:00	1p 7t 46m	0p 14t 41m	1p 5t 11m	1p 2t 31m
13	25.11.2021 08:00	1p 7t 39m	0p 14t 57m	1p 5t 0m	1p 0t 54m
14	02.12.2021 08:00	1p 9t 49m	0p 14t 54m	1p 6t 3m	1p 2t 18m
15	09.12.2021 08:00	1p 4t 6m	0p 12t 54m	1p 3t 15m	0p 21t 45m

Joonisel 4 tuleb selgelt esile 1. nädala kodutöö, mille esitamisega alustati ja ka lõpetati võrreldes teiste kodutöödega ligi kaks päeva varem. See võis olla tingitud sellest, et tegemist oli kõige esimese kodutööga ning tudengid asusid seda innukalt lahendama kohe pärast kursusele ligipääsu saamist. Üldjoontes on kõikide ülejäänud kodutööde lahendamisega tegeletud päev kuni poolteist päeva enne tähtaega, samas esituste kõrghetk on olnud peamiselt kodutöö tähtajale eelnenud päeva õhtutundidel.

Enamuste kodutööde puhul on märgata, et keskmine esituse aeg on varasem kui keskmine esimese versiooni esitamise aeg. Seda juhtub kõikide kodutööde puhul, välja arvatud 1., 7. ja 8. kodutöö. Seda võivad põhjustada need tudengid, kes alustavad kodutöö tegemisega juba varakult, näiteks samal päeval või päev pärast kodutöö ülesande avalikustamist ning teevad kodutöö lahendamise juures väga palju esitusi, samas kui teised tudengid ei ole veel kodutöö ülesannete lahendamisega alustanud. Selliste õppijate ja nende poolt tehtavate paljude esituste mõjul nihkub keskmine esituse aeg kodutöö tähtajast aina kaugemale, samas

kui keskmist esimese versiooni esitamise aega korduvad esitused ei mõjuta, sest esimesi esitusi on igal kodutöö lahendajal ainult üks.



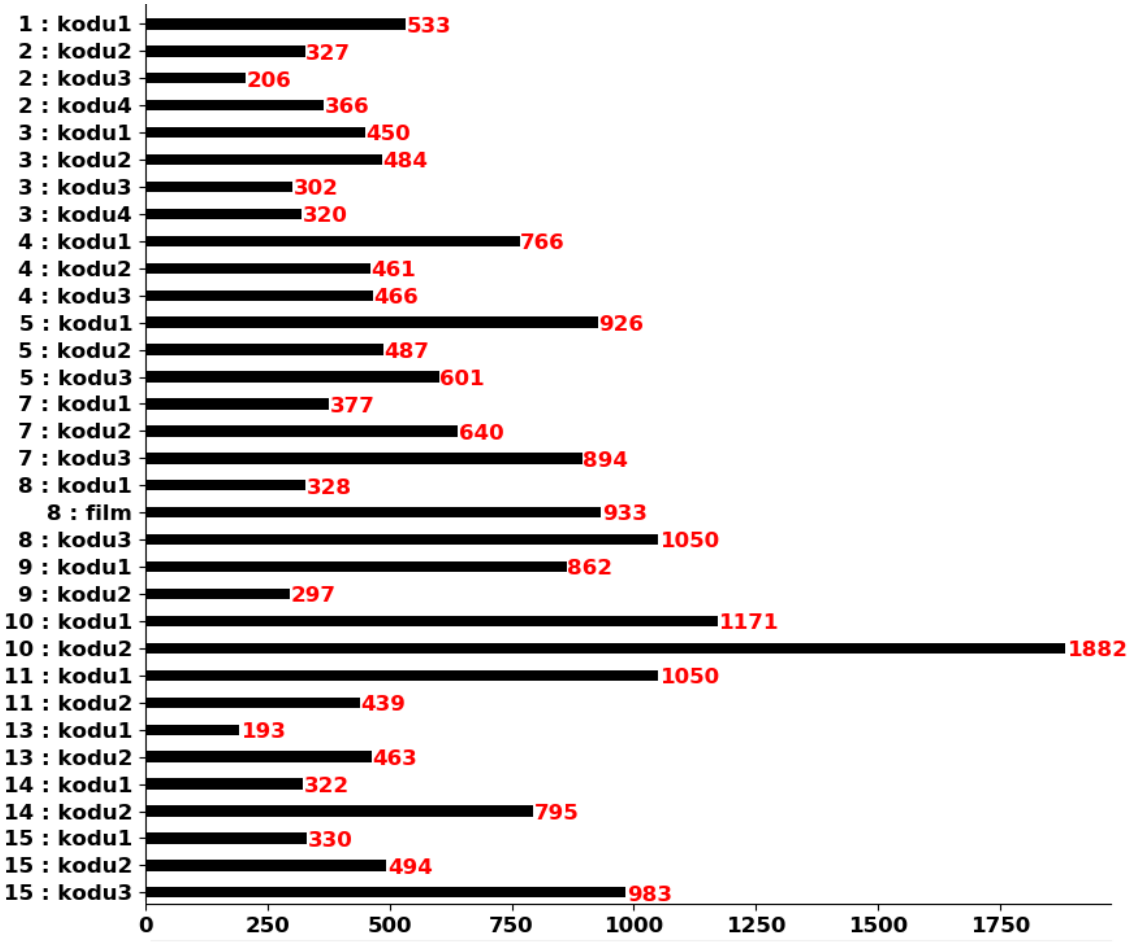
Joonis 4. Kodutöö esitamiste aegade võrdlus enne tähtaega (tundides).

2.3 Keskmine esitatud failisuurus

Keskmine esitatud failisuurus näitab iga kodutöö iga esitatud ülesandefaili keskmist suurust baitides. Joonis 5 on heaks aluseks jaotises 2.4 kirjeldatud keskmise muudatuse suuruse hindamiseks.

Joonisel 5 paistab teiste ülesannete seast selgelt välja 10. nädala kodutöö ülesanne “kodu2”, mille failisuurus baitides on kõikidest teistest ülesannetest peajagu üle. Selline suur ülekaalukus tuleneb ilmselt sellest, et 10. nädala kodutöö “kodu2” ülesandes koostasid paljud

tudengid ülesande lahendamise jaoks mitu erinevat pikka tingimuslauset, kuigi seda ülesannet oleks olnud võimalik sooritada ka palju lihtsamalt [13].

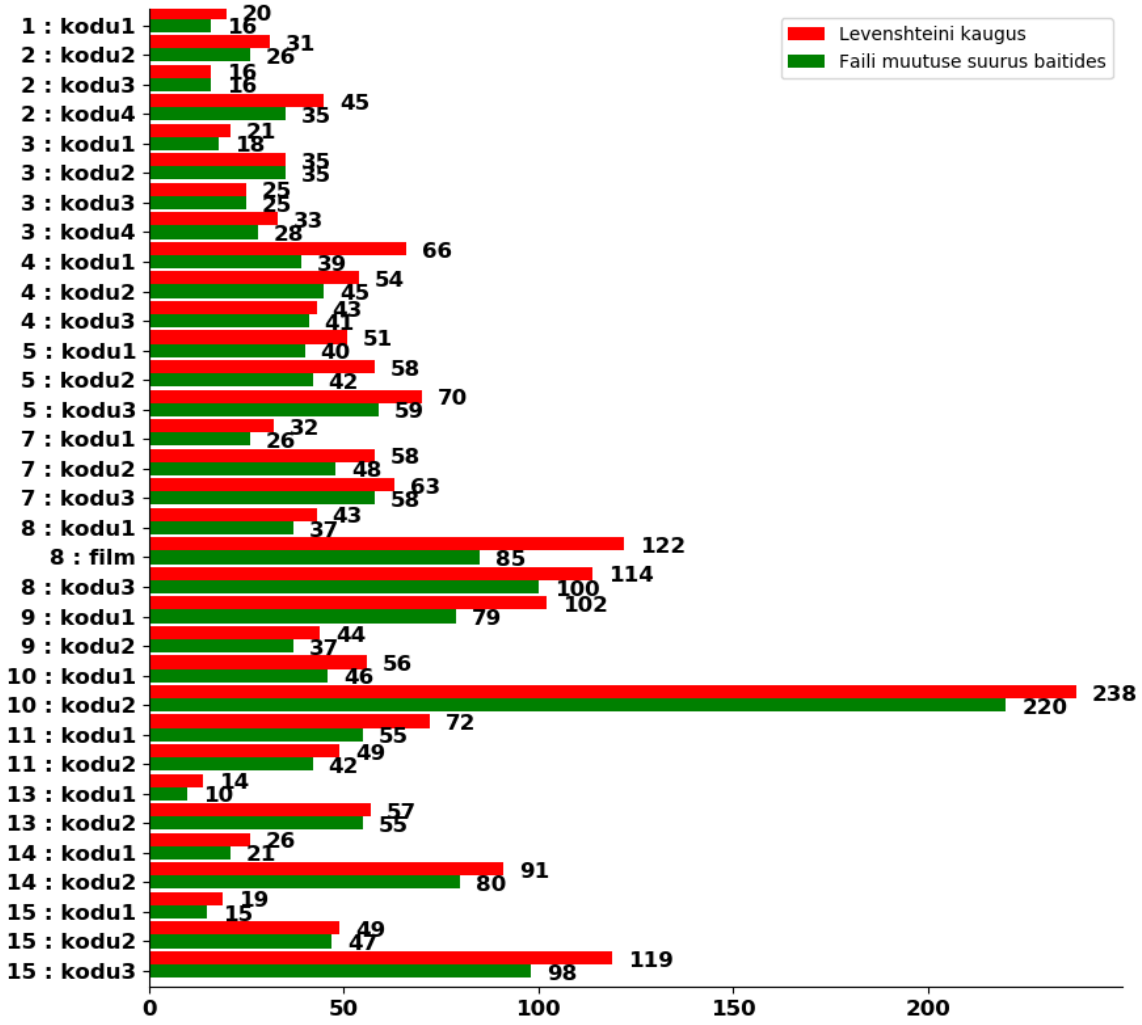


Joonis 5. Keskmine kodutöös esitatud failisuurus.

2.4 Ülesannete keskmine muudatuse suurus üle versioonide

Keskmine muudatuse suurus näitab, kui suures mahus tegi tudeng parandusi ülesande uuesti esitamisel. Töö autor kogus andmeid Levenshteini kauguse [12] kohta kahe erineva esitatud versiooni vahel, kus kodutöö ülesandefaili sisu on käsitletud kui ühte pikka sümbolite järjendit, ning muudatuse suuruse kohta baitides ehk kui suur oli tehtud muudatuse suurus uuesti esitatud faili ja eelmise esitatud versiooni vahel baitides. Levenshteini kaugus näitab, mitu sümbolit on vaja failis muuta, et selle faili sisu oleks identne võrreldava failiga. Keskmine muudatuse suurus üle versioonide baitides ja Levenshteini kauguse mõistes on arvutatud absoluutväärtusena. See tähendab, et joonis 6 näitab ainult muudatuse suurust kahe

erineva esitatud versiooni vahel ning ei näita, kas uuesti esitatud versioon oli eelmisest versioonist faili mahu mõistes suurem või vastupidi.

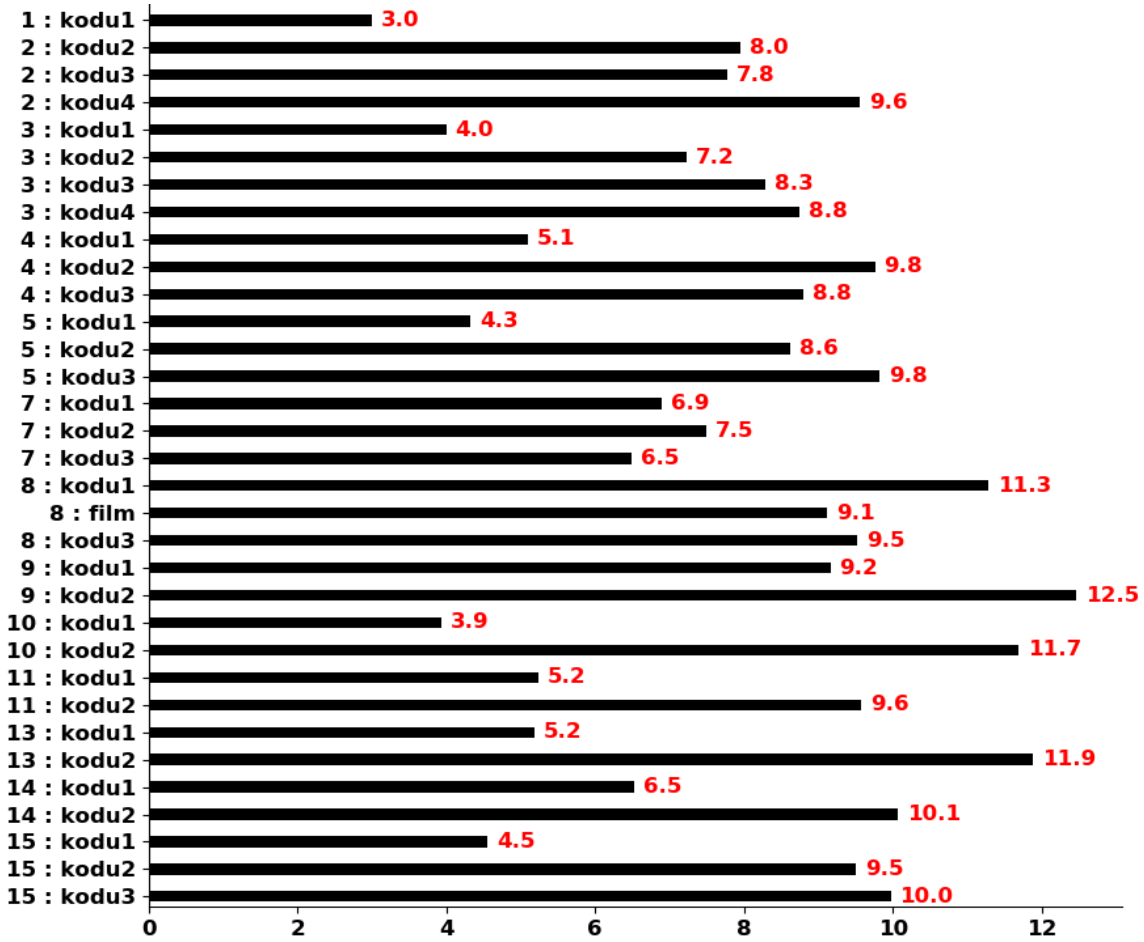


Joonis 6. Keskmine muudatuse suurus üle versioonide.

Joonise 6 põhjal saab välja tuua, et enamikus ülesannetest tehti parandusi üsna väikeses mahus. Seda võib põhjendada selliselt, et enamasti tegid tudengid enda arvates pisemaid loogikavigu, neid siis kas sümbolite kirjeldamisel või muutujate defineerimisel. Märgatavalt suurem on muudatuse suurus nii baitides kui ka Levenshteini kaugusena 10. nädala kodutöö ülesande “kodu2” esitamisel. Sama kodutöö sama ülesanne tõuseb esile ka joonisel 2.3, milles oli näidatud esitatud failide suurused baitides ja pikkused sümbolite järjendina. 10. nädala kodutöö ülesande “kodu2” sisuks oli mäng trips-traps-trull laual mõõtmetega 4*4, kus tudengitel tuli koostada funktsioon, mis võtaks argumendina maatriksina esitatud trips-traps-trulli mängu seisu ning tagastab sõnastiku, milles on võtmed “X” ja “O” ning

võtmete väärtuseks on nende esinemissagedus kolm korda järjest vertikaalselt, horisontaalselt või diagonaalselt [13]. Tundub, et selle ülesande esitamisel on tehtud suuremaid parandusi kui lihtsalt sümbolite muutmine, sest ülesande teksti lugemisel võis tudengitel jääda mitmeid olulisi nüansse tähelepanuta.

Lisaks arvutati ka keskmise muudatuse suurus protsentides ehk mitu protsenti moodustab keskmine muudatuse suurus üle versioonide baitides keskmise esitatud faili suurusest baitides. Tulemused on näidatud joonisel 7.

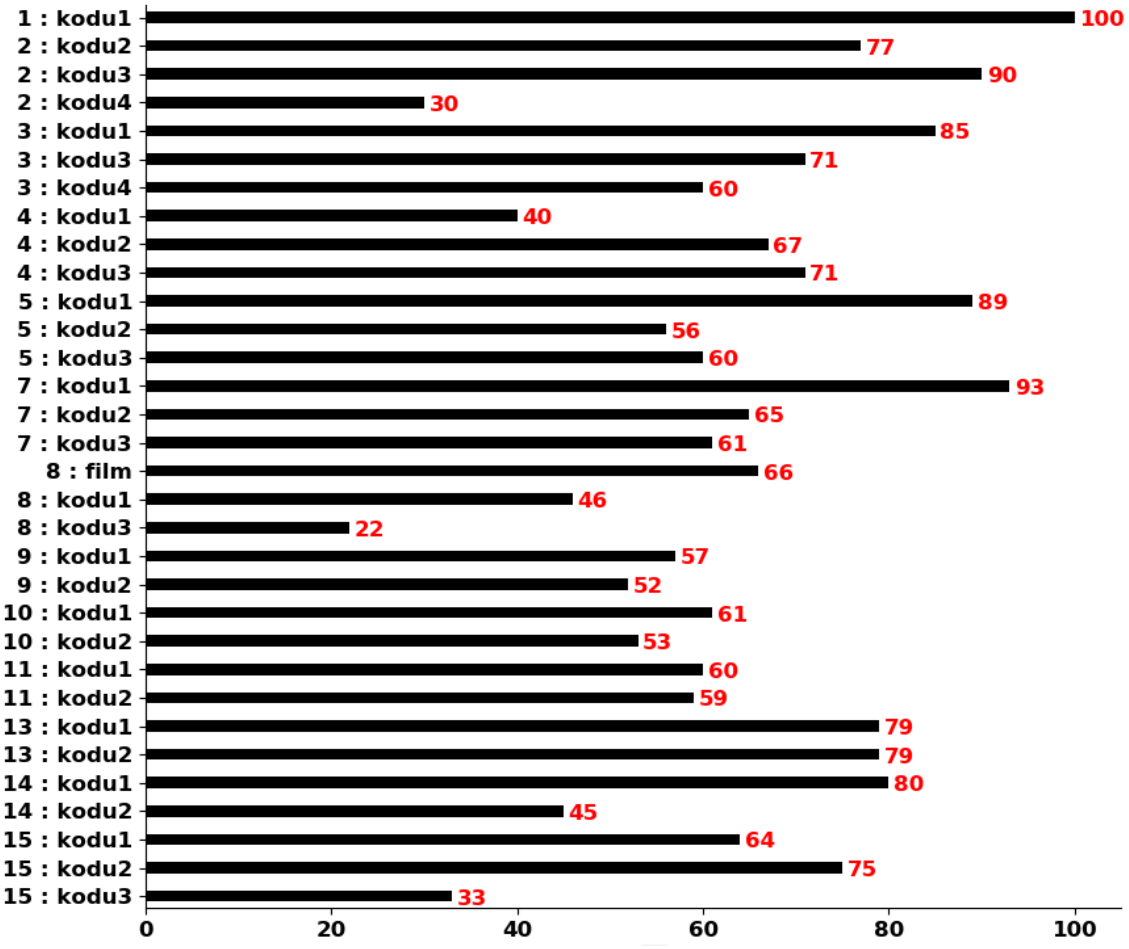


Joonis 7. Keskmine muudatuse suurus üle versioonide (%).

Joonisel 7 olevate andmete põhjal võib oletada, et kui kodutöö ülesanne ei läbinud kõiki automaatteste, siis tehti vastavad parandused, mille tulemusel muutus faili sisu 3.0 - 12.5 protsenti. Võib öelda, et peamiselt oli tegemist pisemate muudatustega, mis võisid tuleneda tudengite tähelepanematusest ülesande teksti lugemisel.

2.5 Lahenduse õigsus

Lahenduse õigsus näitab, kui suur oli keskmiselt automaattestide õnnestumise protsent iga kodutöö igal ülesandel. Keskmise lahenduse õigsuse leidmisel koguti kokku kõikide esituste automaattestide arvu (õnnestunud ja ebaõnnestunud) ning õnnestunud automaattestide arvu. Kasutades protsendi leidmist ristkorrutise abil (x% y-st = y*x/100) arvutati lahenduse õigsuse protsent iga kodutöö iga ülesande kohta.



Joonis 8. Keskmine lahenduse õigsus iga esituse kohta (%).

Joonisel 8 on kõikide kodutööde ülesannete lahenduse õigsus protsentides, välja arvatud 3. nädala kodutöö ülesanne “kodu2”, millel puudusid andmed automaattestide tulemuste kohta. Maksimaalse lahenduse õigsuse protsendiga oli ainult 1. nädala kodutöö ülesanne “kodu1”, mille puhul oli tegemist lihtsa olemasoleva programmi modifitseerimisega ning kuna programmi väljundiks oli pilt, siis automaatkontroll kontrollis ainult vajalike konstruktsioonide olemasolu [14]. 3. nädala kodutöö ülesanne “kodu2” oli Pykkari ülesanne,

kus lahendajatel tuli tutvust teha virtuaalse robotiga nimega Pykkar ning luua virtuaalne maailm, kus siis vastavaid käsklusi kasutades saaks robotiga ringi liikuda, värvida ja asju tõsta [15].

Kõige madalama õigete lahenduste protsendiga oli 8. nädala kodutöö ülesanne “kodu3”, milles kasutatakse moodulina ülesannet “film” [16]. Üllataval kombel oli lahenduse õigsuse protsent väike ka näiteks 2. nädala kodutöö ülesandel “kodu4”. Tegemist oli üsna lihtsa ülesandega, kuid on võimalik, et selle ülesande juures valmistas tudengitele raskusi failide töötlemine, mis sel hetkel oli veel uus teema ja varasema programmeerimise kogemusteta tudengitele tehniliselt keerukas [17]. Lisaks saab madala lahenduse õigsuse protsendi puhul välja tuua ka 15. nädala kodutöö ülesande “kodu3”, milles sai eeldatavasti määravaks õppijate tähelepanelikkus ning sõnastusest ja tegevuste loogikast arusaamine [18].

3 Tudengite profiilid

Klasterdamine on peamiselt andmeteaduses ning masinõppes kasutusel olev meetod, mille abil on võimalik rühmitada objekte erinevatesse gruppidesse, arvestades selle juures objekte iseloomustavaid parameetreid [19]. Klasterdamise meetodeid on erinevaid, kuid töö autor otsustas selle peatüki tarbeks kasutada ühte kõige enam levinumat klasterdamise meetodit, milleks on k-keskmiste klasterdamine.

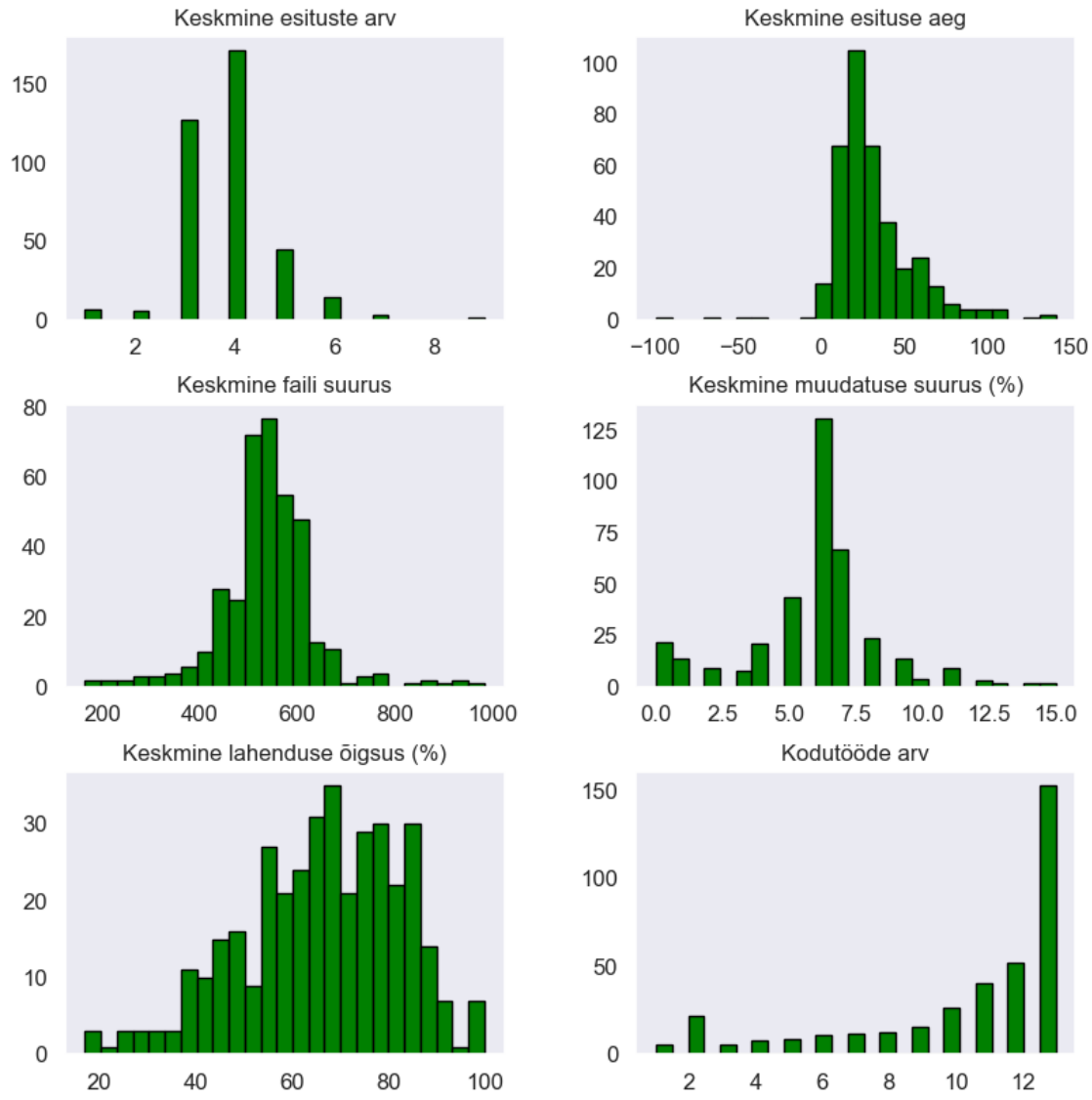
K-keskmiste klasterdamisalgoritmi idee on jagada objektid erinevatesse gruppidesse nii, et iga objekt kuuluks sellesse klastrisse, mille keskpunktini on tal kõige väiksem eukleidiline kaugus. Tegemist on peamiselt puhtalt katsetamise meetodiga. Algoritmi jooksutatakse kokku kogutud andmete peal erinevate klastrite arvuga ning saadud tulemuste põhjal otsustatakse klastrite arv, mille puhul oli tulemus kõige sobivam. K-keskmiste algoritmi on kõige parem kasutada väiksemate andmekogumite peal, sest mida rohkem on andmekogumis andmeid, seda kauem võtab aega andmepunktide klassifitseerimine [20].

K-keskmiste klasterdamisalgoritm kuulub tsentroidipõhiste klasterdamismeetodite hulka, mis on kõige sagedamini kasutatav klastrite tüüp. Tsentroidipõhise rühmitamise eesmärgiks on lähtuda andmepunktide kaugusest klastri keskpunktini, kus andmepunkt määratakse sellesse klastrisse, mille keskpunktile see kõige lähemal on. Lisaks tsentroidipõhisele meetodile on olemas ka tiheduspõhine meetod, jaotuspõhine meetod ning hierarhiline meetod. Tiheduspõhine meetod määrab andmed klastritesse tiheduspunktide alusel. Algoritmi eesmärk on üles leida kõik graafikul olevad punktide kogumid, mis on üksteisega tihedalt koos ning neid kogumeid kutsutaksegi edaspidi klastriteks. Jaotuspõhise meetodi aluseks on klastrite keskpunktid ning klastrisse kuulumine otsustatakse tõenäosuse alusel. Mida kaugemal on andmepunkt klastri keskpunktist, seda väiksem on tõenäosus, et ta kuulub sellesse klastrisse. Hierarhilise klasterdamise puhul luuakse andmetest justkui sugupuu, kus igale andmepunktile leitakse seos mingisuguse teise andmepunktiga. Tekkinud puu harusid nimetatakse klastriteks [21].

3.1 Tunnuste jaotused

Töö autor kogus kokku andmed iga tudengi kohta, kes esitas 2021. aasta sügissemestri kursusel “Programmeerimine” vähemalt ühe kodutöö, ning koondas andmed kokku ühte tabelisse (tabel on esitatud lisas 2). Kodutööde esitajaid oli kokku 376 ning keskmiselt esitas iga tudeng 13-st kodutööst 10. Loodud tabeli algandmeteks olid kursuse vältel Moodle’i keskkonda salvestatud kõikide kursusel osalejate kõigi esituste failid koos kellaaegadega. Tabelis on esitatud iga tudengi statistilised andmed esitatud kodutööde üldkeskmise kohta: keskmine esituste arv, keskmine esituse aeg enne tähtaega tundides, keskmine esitatud failisuurus, keskmine muudatuste suurus protsendina ning keskmine lahenduse õigsus protsendina. Tulemused on arvutatud vastavalt sellele, mitu kodutööd tudeng esitas ehk kui tudeng sooritas 13-st kodutööst näiteks 5, siis tema keskmise esituste arvu arvutamiseks jagati kõikide kodutööde esituste keskmine viiega, mitte kolmeteistkümnega.

Kokku kogutud andmete põhjal koostati andmete jaotuste diagrammid, kus on näha erinevate parameetrite jaotumine tudengite vahel. Joonisel 9 on iga parameetri jaoks oma tulpdiagramm, milles on näha, kuidas selle parameetri väärtused tudengite vahel jagunevad. Tulpade väärtusi ümardatakse joonisel üles ehk kui keskmine esituste arv on 3.5, siis see loetakse väärtuseks 4, mitte 3, ning lisatakse seejärel vastavasse tulpa. Tulpdiagrammide x-telgedel on kirjeldatud parameetrite väärtused ning y-teljed kirjeldavad tudengite arvu. Keskmise esituse aja puhul on x-teljel olevad väärtused arvestatud tundides, kus väärtus 0 tähistab kodutöö esitamise tähtaega. Väärtused, mis on suuremad kui 0, tähistavad tudengi keskmist kodutöö esitamise aega tundides enne kodutöö tähtaega ning väärtused, mis on väiksemad kui 0, tähistavad kodutööde esitamise aega tundides peale kodutöö esitamise tähtaega. Üksikutel juhtudel esitati töid Moodle’isse ka pärast tähtaega, näiteks võis olla olukord, kus tudeng ei jõudnud oma kodutööd õigeks ajaks valmis ning esitas siis selle e-kirja vahendusel oma praktikumijuhendajale, kes siis seejärel töö õppija nimel esitas. Lisaks kokku kogutud tudengite statistilistele andmetele on joonisel 9 diagramm tudengite esitatud kodutööde arvu kohta, mis näitab, mitu kodutööd tudengid üldiselt kursuse jooksul sooritasid. Joonisel 9 kirjeldatud keskmine failisuurus on arvutatud baitides.

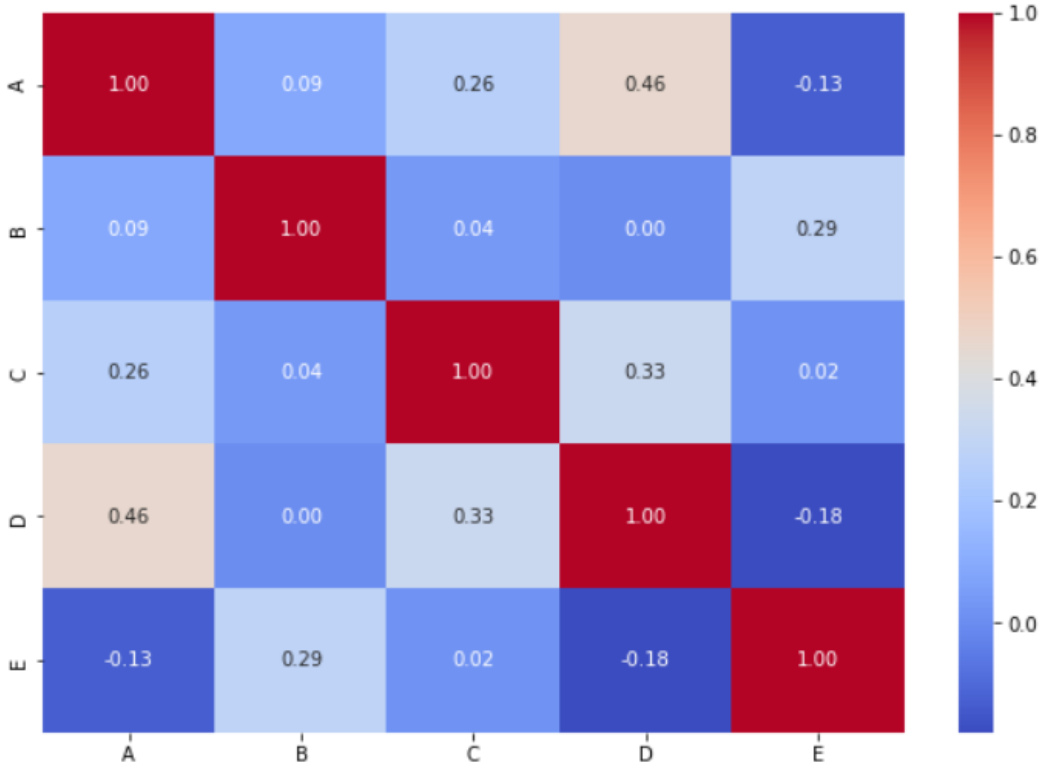


Joonis 9. Andmete jaotus tudengite vahel.

3.2 Tunnusepaaride vahelised seosed

Kokku kogutud andmete põhjal koostati soojuskaart (ingl k. - *heatmap*, joonis 10), millel on näha seoste tugevused kõikide kirjeldatud parameetrite vahel. Täpsemate seoste leidmise tarbeks skaleeriti enne klasterdamise algoritmi rakendamist andmed võrreldavale skaalale, skaleerimiseks kasutati Pythoni mooduli *sklearn.preprocessing* funktsiooni *normalize*. Lisaks koostati kõikide parameetrite omavaheliste seoste kirjeldamise tarbeks ka hajuvusdiagrammide maatriks, parameetrite omavahelised seosed on visualiseeritud joonisel 11, kus eri värvi punktid vastavad eri klastritele.

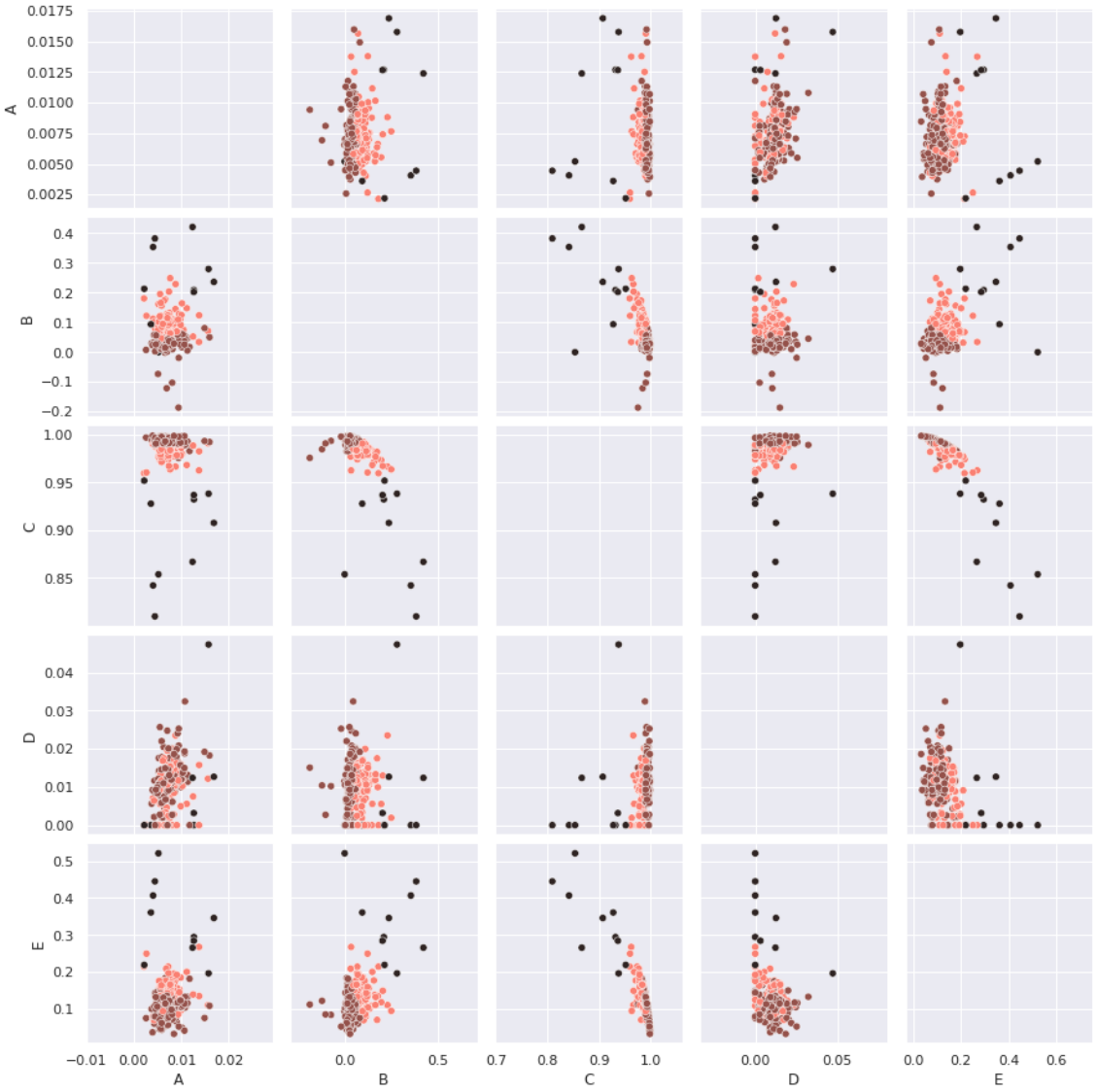
A - keskmine esituste arv, B - keskmine esituse aeg, C - keskmine failisuurus baitides, D - keskmine muudatuse suurus (%), E - keskmine lahenduse õigsus (%).



Joonis 10. Parameetrite omavahelised seosed.

Joonisel 10 on näha, et kõige tugevamas seoses on keskmine esituste arv ja keskmine muudatuse suurus protsentides. Lisaks sellele on omavahel seoses ka keskmine failisuurus baitides ja keskmine muudatuse suurus protsentides, keskmine esituse aeg ja keskmine lahenduse õigsus ning keskmine esituste arv ja keskmine failisuurus baitides.

A - keskmine esituste arv, B - keskmine esituse aeg, C - keskmine failisuurus baitides, D - keskmine muudatuse suurus (%), E - keskmine lahenduse õigsus (%).



Joonis 11. Normaliseeritud kujul olevate andmete hajuvusdiagrammide maatriks.

Joonise 11 põhjal võib öelda, et mida varem esitatakse kodutöö, seda rohkem automaatteste esitatud programm korraga läbib. Seda näitab seos veergude B ja E vahel. Hilinenud kodutööde puhul on lahenduse õigsus märgatavalt madalam kui näiteks kõige varakumalt tehtud kodutöödel. Kuigi on olemas ka erandeid, saab siiski öelda, et mida hiljem esitatakse kodutöö, seda väiksem on kodutöö ülesannete korrektsus.

Lisaks saab joonise 11 põhjal järeldada, et mida väiksem on failisuurus (veerg C), seda kõrgem on lahenduse õigsuse protsent (veerg E). Seda võib tõlgendada ka selliselt, et mida vähem nüansse ja nõudmisi on kodutöö ülesandel, seda lihtsam on tudengitel seda lahendada.

Mida rohkem antakse kodutöö ülesandes tudengitele selliseid aspekte, mida tuleb mitmel erineval moel kontrollida ja töödelda, seda suuremaks kujuneb tudengi poolt esitatav failisuurus ehk seda rohkem on kodutöö lahendaja failis koodiridu, ning mida rohkem on koodiridu, seda rohkem on ka võimalusi vigade tegemiseks.

Lisaks on jooniselt näha, et lahenduse õigsuse paranemine ei sõltu sellest, kui suures mahus tehti kodutöö ülesande uuesti esitamisel koodiridades muudatusi ja mitu versiooni tudeng esitas. Parameetrite A ja E ning D ja E vaheliste seoste kokkuvõttena võib öelda, et tudengid on tihtipeale üritanud lahendada mitu probleemi korraga ning see on võibolla omakorda toonud juurde mingeid lisaprobleeme ebaõnnestunud testide näol, mis ei paranda lahenduse õigsust. Võib oletada, et tihtipeale arvasid tudengid, et ebaõnnestunud automaattestide põhjuseks on pisemad loogikavead ning neid asuti siis korraga parandama, kuigi tegelikult võis programm olla juba alguses nii vale, et targem oleks olnud see otsast peale ümber kirjutada. Psühholoogiast on tuntud nähtus, et inimesed kipuvad hindama oma oskusi paremaks, kui need tegelikult on. Seda nähtust nimetatakse Dunningi-Krugeri efektiks [22].

3.3 Tudengite klastrid

Jaotises 3.1 mainitud tabelile rakendati k-keskmiste klasterdamisalgoritmi (ingl k. - *k-means clustering*) [23], et saada aimu, millised on kõige tüüpilisemad kursusel õppinud tudengite profiilid ning mustrid kodutööde lahendamisel. Katsetades erinevate klastrite arvudega, osutus kõige sobivamaks klastrite arvuks kolm, mille puhul eristusid klastrite andmed üksteisest kõige enam.

Tabel 4. Klasterdamise tulemus.
A - klaster, B - keskmine esituste arv, C - keskmine esituse aeg enne tähtaega tundides, D - keskmine failisuurus baitides, E - keskmine muudatuse suurus (%), F - keskmine lahenduse õigsus (%), G - keskmine lahendatud kodutööde arv, H - tudengite arv klastris.

A	B	C	D	E	F	G	H
1	3.8	18	555	6.1	60	11	216
2	3.9	48	529	5.7	74	10	150
3	2.5	69	257	2.0	90	2	10
ÜLDINE	**3.8**	**31**	**537**	**5.8**	**66**	**10**	**376**

Tuginedes tabelis 4 olevatele andmetele, saab kursusel õppinud tudengid jagada kolme erinevasse gruppi, kes üksteisest veidikene erinevad. Esimeses grupis on tudengid, kes tegelevad kodutööga kodutöö tähtajale eelnenud päeval. Need tudengid on tõenäoliselt kodutööde lahendamisel tormakad ja tähelepanematud, kuna nende keskmine esituste arv 3.8 ei too tulemust keskmise lahenduse õigsuse näol, mille näitaja on esimesel grupil alla üldise keskmise. Nende keskmisest madalam lahenduse õigsuse protsent võib olla ka tingitud sellest, et nad alustavad kodutöö lahendamisega liiga hilja, seda sama seost kirjeldab ka joonis 11.

Teise grupi tudengid tegelevad kodutööga umbes kaks päeva enne kodutöö tähtaega ning nende lahenduse õigsuse protsent on tunduvalt kõrgem võrreldes esimese klastriga. Teise grupi tudengid alustavad ülesannete lahendamisega tunduvalt varem kui seda teeb keskmine kursusel õppiv tudeng. Selle põhjal võib arvata, et tegu on keskmisest motiveeritumate ja sihikindlamate tudengitega. Teist gruppi iseloomustav kõige kõrgem keskmiste esituste arv väljendab eelkõige püüdlikkust parema tulemuse saavutamiseks.

Kolmanda grupi tudengite puhul võib olla tegemist selliste tudengitega, kes alustavad kursust õigeaegselt koos kõikide teiste õppijatega, kuid jõuavad enne ainele registreerimise tähtaja lõppemist ennast kursusel osalejate nimekirjast maha võtta. Seda hüpoteesi kinnitab kõige rohkem kolmanda grupi keskmine esitatud kodutööde arv, mis on 2. Kuna esimeses kodutöös oli automaattestide läbimise protsent kõikidel kodutöö esitajatel maksimum ehk 100, siis on ka see heaks aluseks kõrge lahenduse õigsuse protsendi, mis on ka võrreldes teiste klastritega kõige kõrgem, säilitamiseks. Kodutööde keskmine esitatud versioonide arv on kolmandal grupil kõige madalam, mis kinnitab töö autori arvamust, et kolmanda grupi tudengid on pärast kergemat 1. nädala kodutööd tundnud märgatavalt suurenenud ülesannete keerukust juba 2. kodutöö lahendamise juures ning otsustanud end seejärel kursusel osalejate nimekirjast maha võtta. Kolmandas grupis võivad olla ka need õppijad, kes on lihtsalt loobunud kodutööde tegemisest või on sooritanud kursuse alguses positiivsele tulemusele kursuse eeleksami ning saanud tänu sellele kursuse ainepunktid kirja juba kursuse algfaasis. Seda võimalikkust toetavad ka väga varajane kodutöödega tegelemise aeg ja teistest ligemale kolm korda väiksem keskmise muudatuse suuruse protsent, mis kirjeldavad üldiselt õppijaid, kellel on piisavalt tugev varasema programmeerimise kogemus olemas ning kellele kursusel käsitletavad programmeerimise teemad ja ülesanded ei valmista erilist raskust.

4 Automaatkontrollide kvaliteedi pisteline uuring

Bakalaureusetöö raames uuriti ka põhjalikumalt kursusel “Programmeerimine” osalenud tudengite kodutöid, et tuvastada ja üles märkida kodutööde lahendamisel tehtud vigu. Selle eesmärgiks oli kategoriseerida kõige tüüpilisemad vead ning koostada vigade loetelu, mille kursuse juhendajad saaksid tulevikus automaattestide täiendamisel aluseks võtta. Kodutööde põhjalikum analüüs toimus läbi pistelise kontrolli, kus genereeriti kõikide 376 lahendaja seast nimekiri kahekümnest tudengist, kelle kõikide kodutööde kõik ülesanded läbi vaadati. Genereerimine toimus juhuslikkuse alusel ning selleks kasutati Pythoni programmi (lisa 1), millele anti sisendiks vahemik 1-377 ning kasutades moodulit *random* väljastas programm järjendi, milles oli 20 etteantud vahemikust suvaliselt genereeritud arvu. Järjendis olevad arvud tähistasid indekseid, mille alusel otsiti tabelist (lisa 2) sellel indeksil paikneva tudengi nime.

Järgnevas loetelus on tudengite poolt tehtud kõige tüüpilisemate vigade nimekiri, kus vead on liigitatud erinevatesse kategooriatesse vea tüübi järgi. Näiteks, kui tudengi poolt esitatud programmi väljund ei olnud see, mida ülesandes küsiti, või programmi väljund oli vigane, siis on need mõlemad vead märgitud programmi väljastusel esinenud vigade kategooriasse. Kategoriseerimise eesmärk on pakkuda lugejale paremat arusaama tehtud vigade temaatikast.

1. Võimalikud valed andmetüübid.
    a. Kasutajalt küsitakse sisendit, aga ei kontrollita, kas kasutaja poolt antud sisend on õiges/loogilises vahemikus. Näiteks kui ülesandes on nõutud positiivset arvu, mida kasutajalt sisendina küsitakse, siis ei kontrollita, kas kasutaja poolt sisestatud arv on positiivne.
2. Vead Pythoni moodulite importimisel.
    a. Imporditakse mooduleid, mida programmis ei kasutata.
    b. Jäetakse importimata programmi töös vajaminevad ja ülesandes nõutud moodulid - näiteks ei impordita moodulit *math*, et kasutada selle mooduli konstanti pi, vaid pi väärtus määratakse ise: 3.14.
3. Vead programmi esitamisel.
    a. Esitatakse tühi programm - esitatud programmil puudub sisu.
    b. Esitatakse vale ülesanne - sama kodutöö mingisugune teine ülesanne.

4. Vead muutujate kasutamisel.
    a. If-lausetes kasutatakse mittedefineeritud muutujaid.
    b. Eelnevalt defineeritud muutuja modifitseerimisel eksitakse muutuja nime õigekirjaga. Näiteks kui programmis on kirjeldatud muutuja koos väärtusega, siis hiljem muutujale väärtust juurde lisades tehakse muutuja nime kirjutamisel kirjavigu - programm käsitleb seda kui mittedefineeritud muutujale väärtuse lisamist.
    c. Funktsioonis omistatakse muutuja väärtus iseendale: n = n.
5. Vead arvutuskäikudes.
    a. Programmis kasutatavad valemid on valed või vigased. Näiteks 4. nädala kodutöö “kodu2”, kus on ülesandes kasutatava erirelatiivsusteooria valemis on tehete järjekord kirja pandud valesti.
    b. If-lausetes olevate “<” ja “>” kasutamisest tingitud vead - valed vahemikud. Näiteks 3. nädala kodutöö “kodu1”: aastatulude vahemikud on paika pandud ebatäpsetes vahemikes, kuna võrratus on valet pidi või puudub lisaks võrratusele ka “=” märk.
6. Programmi väljastusel esinenud vead.
    a. Väljastatud tulemust ei ole ümardatud nii nagu nõutud.
    b. Ülesande teksti kohaselt peaks programm midagi väljastama, aga programm ei väljasta midagi.
    c. Programm väljastab midagi muud, kui oli ülesandes nõutud.
    d. Programmi väljund on puudulik. Näiteks 10. nädala kodutöö ülesandes “kodu1” peab programmi funktsioon tagastama argumendina saadud sõnes esinevate kõikide erinevate sümbolite hulga - osad sümbolid on jäänud hulka lisamata või ei ole eristatud suur- ja väiketähti.
7. Ülesande tekstist arusaamisega tekkinud vead.
    a. Ülesandes pole asju selgelt sõnastatud: 1) 5. nädala kodutöö ülesandes “kodu2” peab funktsioon tagastama parameetrina saadud sõne modifitseeritud kujul, kuid pole öeldud, kas see sõne peab olema sama, mis saadi parameetrina või võib tagastada täiesti uue sõna - mitmetel kordadel muutsid tudengid oma programmis just seda nüanssi, 2) 7. nädala kodutöö ülesandes “kodu3” tuleb väljastada isikukoodi põhjal genereeritud sünnikuupäev, kuid ülesande tekstis pole täpsustatud, kas kirjutada näiteks 1. jaanuar või 01. jaanuar.

b. Ülesanne nõuab sisendi küsimist, aga sisendit ei küsita.
c. Visatud erindid ei ole selle tekstiga, mis ülesandes nõutud.
d. Ülesanne eeldab, et programm modifitseerib ja seejärel tagastab etteantud muutuja, kuid programm tagastab täiesti uue muutuja.
e. Programm tagastab, mitte ei väljasta. Probleemid "return" ja "print" käskudel vahet tegemisel. Üks näide paljudest: 7. nädala kodutöö ülesandes "kodu1" peab funktsioon poisse_ja_tüdrukuid() tagastama poiste ja tüdrukute arvu, kuid programm väljastab funktsiooni tulemuse ekraanile.
f. Sisendväärtusi töödeldakse globaalselt, mitte funktsiooni siseselt. Näiteks 4. nädala kodutöö "kodu1", kus kasutajalt küsitakse ühe sisendina koogi nime ning kui koogi nimi ei ole kolme etteantud variandi hulgas, peab funktsioon viskama erindi, kuid sisendi kontrollimise samm ja sellele reageering tehakse juba enne funktsiooni välja kutsumist.
g. Ülesanne nõuab programmis ühe funktsiooni defineerimist, kuid programmis on rohkem funktsioone.

8. Programmi koostamisel arvestatakse ainult ülesande kirjelduses olnud näidistulemiga/andmetega.
    a. Näiteks lähtutakse ainult ülesandes näitena näidatud failist ning keskendutakse ainult sellele, et selle konkreetse faili puhul programm töötaks ehk ei arvestata erijuhtudega, kus näiteks fail võib olla tühi või faili sisu teises mahus.
9. Lokaalsete ja globaalsete muutujate kasutamine.
    a. Luuakse globaalne muutuja, kuid kasutatakse seda ainult mingi funktsiooni siseselt.
10. Programmide ebaoptimaalsus.
    a. Liigne if-lausete kasutamine, selle asemel et kasutada elif-lauset - kontrollitakse, millisele tingimusele mingi muutuja väärtus vastab, kuid kui see tingimus on leitud, siis kontrollitakse edasi.
    b. While-tsükli või for-tsükli kasutamise asemel korratakse ühte käsku n korda. Näiteks 4. nädala kodutöö "kodu3", kus tsükli kasutamise asemel korratakse funktsiooni väljakutsumist näiteks 30 korda.
    c. Programm väljastab liiga palju asju/vahetulemusi, mida ei ole ülesande teksti kohaselt nõutud.
    d. Tsüklit jooksutatakse liiga palju kordi.
    e. Programmis teisendatakse kasutajalt saadud sisendväärtus string-tüüpi

väärtuseks - see pole vajalik, sest “input” tagastab alati string-tüüpi väärtuse.

11. Faili sisse lugemisel tehtud vead.
    a. Ei tehta vahet käskudel readlines() ja readline(). Näiteks 3. nädala kodutöö ülesandes “kodu4” oleva faili sisse lugemisel loetakse readline() abil sisse ainult faili esimene rida ning seejärel teostatakse for-tsüklit kasutades sellele reale vajaminevaid operatsioone, kuigi tegelikult oleks vaja for-tsükliga läbi käia terve faili sisu ning neid operatsioone rakendada faili kõikidele ridadele.
    b. Avatakse ilma failiõigusteta - faili kirjutamise jaoks ei lisata faili avamisel vastavaid õigusi.
    c. Faili käsitlemisel ei arvestata sellega, et tegemist võib olla tühja failiga.
12. While-tsükli kasutamisega tekkinud vead.
    a. While-tsüklil puudub lõpptingimus - while-tsükkel jääb lõputult kestma.
    b. Programmis on while-true-tsükkel, mida ülesande teksti järgi ei tohiks olla.
13. Vead rekursiivse funktsiooni kasutamisel.
    a. Rekursiivsel meetodil puudub baasjuhtum - funktsioon töötab lõputult.
    b. Rekursiivset funktsiooni kutsutakse uuesti välja täpselt samade argumentidega.
14. Väärtuste tüüpide käsitlemise probleemid.
    a. String-tüüpi muutujaga üritatakse teha arvutustehteid.
    b. Arvutustehteid ei tehta sama tüüpi muutujatega - tehteid üritatakse sooritada string ja float või string ja int muutujatüüpide vahel.
15. Defineeritud funktsioon on ebakorrektne.
    a. Funktsioonil puuduvad ülesandes nõutud parameetrid.
    b. Funktsioonil on parameetrid, mida pole nõutud/vaja.
    c. Programmis puudub nõutud funktsioon.
    d. Funktsiooni nimes esineb kirjavigu.
16. Programmi ei kirjutata Pythoni süntaksile kohaselt.
    a. Funktsiooni tagastatav väärtus on sulgudes.
    b. If-lause tingimus on sulgudes.
    c. Funktsiooni nimi (ülesanne nõuab funktsiooni, aga funktsiooni nime ei ole ette antud ehk see on ülesande lahendaja vaba valik) on suure algustähega.
    d. Kui muutuja või funktsioon koosneb mitmest sõnast, siis ei ole need eraldatud alakriipsuga, vaid suurte tähtedega - muutuja_näide asemel muutujaNäide.

17. Pythoni moodulite vale kasutamine.
    a. Meetodi väljakutsumisel ei kasutata sulge - readline() asemel readline, islower() asemel islower, count() asemel count jms.
18. Programmis olevaid parameetreid kontrollitakse vales järjestuses. Näiteks 5. nädala kodutöö ülesandes “kodu1” on ülesande tekstis kindel järjekord kasutajaga suhtlemiseks ja operatsioonide tegemiseks, kuid seda ei järgita.
19. Koodi stiililised vead. Kursuse raames võiksid juhendajad anda õppijatele soovitusi ja juhiseid puhtama ja loetavama koodi kirjutamiseks - näiteks PEP8 dokumentatsiooni aluseks võttes.
    a. Asjasse mittepuutuvad kommentaarid.
    b. Muutujate nimed ei ole defineeritud vastavalt muutuja olemusele - näiteks muutuja, mis kirjeldab auto hinda, nimi ei ole auto_hind vaid a või x.
    c. Liiga palju tühje ridu.
20. Pythoni moodulite importimine tehakse funktsiooni sees - tavaliselt asuvad import-käsud funktsioonist väljaspool ja kohe programmi alguses.
21. Pärast erindi viskamist üritatakse teha veel tegevusi.
22. Pärast “return” käsku üritatakse teha veel tegevusi.

5 Õpilaste käitumismustrid

Selles peatükis leitakse allikate põhjal õpilaste tüüpilised käitumismustrid programmeerimise kursusel. Saadud andmete põhjal teeb töö autor programmeerimise kursust alustavate või kursusega algfaasis olevate õpilaste tausta ja käitumise info põhjal ennustused nende kursuse lõppresultaadi kohta. Tulevastes uurimustes on võimalik seda peatükki aluseks võttes uurida tudengite käitumuslikke erinevusi ka kursusel "Programmeerimine". Kui tulevastel uurijatel on ligipääs ka kursusel osalejate hinnetele, siis saab tudengite käitumismustreid analüüsides hinnata ka selles peatükis tehtava hinnete ennustamise täpsust.

5.1 Tüüpilised käitumismustrid programmeerimise kursusel

Koolis programmeerimise õppimisel on õpilase õppimistahe ja panus iseseisvasse õppimisse tugevas seoses lõpphinde kujunemisega. Pereira jt [24] poolt tehtud uuringus järeldati, et õpilaste lõpphinde kujunemist mõjutavad kõige rohkem nende tegutsemine esimese saadud tagasiside järgselt. Kui õpilane on esimese tööga või esimeste ülesannetega ebaõnnestunud ja ei kuluta vajalikku aega oma vigade analüüsimisele ja vigade parandamisele, siis suure tõenäosusega on sellel väga negatiivsed tagajärjed lõpphinde kujul.

Sarnasele tulemusele jõuti ka Alzaid'i ja Hsiao [25] poolt tehtud uuringus, kus täheldati, et kursust paremini alustanud õpilased säilitavad kauem positiivset suhtumist ja neil tekib vähem allaandmise momente, kuid esimese programmeerimise tööga halvemini hakkama saanud õpilased näitasid üles ebakindluse jätkumist. Lisaks järeldati, et stabiilne ajapanustamine programmeerimise õppimisse on oluline aspekt parema lõpphinde saamiseks. Kõige halvemaks mustriks hinnati harjutamise puudulikkust.

Mõlema uuringu tulemused toovad esile arusaama, et õppimise tulemuslikkuse kujunemisega saab määravaks osaks iga inimese enda töötahe. Väga oluline osa on ka harjutamisel, mille käigus saab inimene õppida ja ammutada uusi teadmisi, seega ainult teoreetilisest lähenemisest ei piisa.

5.1.1 Varasema programmeerimise kogemuse mõju tulemustele

Kooli poolt antavate kodutööde lahendamisel mängib rolli ka õpilaste varasem kogemus programmeerimises. Kogemus annab tihtipeale eelise, sest ajalist ressurssi võib kuluda

vähem ning seetõttu võib kodutöö tegemisega alustada palju hiljem kui need õpilased, kellel puudub varasem kogemus ja kes lahendavad seda tüüpi ülesandeid esimest korda. Albluwi ja Salteri [26] poolt tehtud uuringus jõuti tulemusele, et varasema programmeerimise kogemuse ja programmeerimisülesannetes tehtavate vigade arvu vahel puudub märgatav seos. See tähendab, et õpilased, kellel on varasem programmeerimise kogemus suurem, ei tee eranditult vähem vigu kui õpilased, kellel on varasem kogemus väiksem või puudulik. Varasema programmeerimise kogemusteta õpilased tegid küll üldise keskmisega võrreldes rohkem vigu ning kogemusega õpilased jällegi vähem vigu, kuid mõlemate tulemused olid üldisele keskmisele väga lähedal, mistõttu ei saa väita, et varasema kogemuse maht ja tehtavate vigade arv oleksid omavahel märkimisväärses seoses.

5.1.2 Soolise erinevuse mõju tulemustele

Programmeerimise ülesannete lahendamisel eristuvad omavahel ka mehed ja naised. Albluwi ja Salteri [26] uuringus leiti, et kuigi varasema programmeerimise kogemusega mehi oli protsentuaalselt rohkem kui naisi, tegid naised ülesannete lahendamisel vähem vigu kui mehed. Ülesannete lahendamisel tehtud vigade parandamine on samuti naistel ajaliselt kiirem protsess kui meestel ning lisaks toodi välja naiste puhas stiil koodi kirjutamisel ja dokumentatsiooni koostamise põhjalikkus. Nende andmete põhjal võib kursust alustades oodata, et naiste keskmine tulemus tuleb parem kui meeste keskmine tulemus. Naiste dokumenteerimise oskus ja puhas stiil hakkavad kursuse arenedes aina rohkem kasu tootma, sest kui varasemad ülesanded on korrektselt ja loetavalt lahendatud, siis võib lahenduskäikude uuesti analüüsimine olla abiks ka edasiste ülesannete lahendamisel. Meeste lohakus ülesannete lahendamisel võib pärssida nende motivatsiooni ning järjepidevust tulevaste ülesannete lahendamisel, mis omakorda kajastub ka kursuse lõpptulemuses.

5.1.3 Individuaalsed ülesanded ja paarisülesanded

Albluwi ja Salter [26] uurisid ka individuaalselt tehtud ülesannete ja paaristööna tehtud ülesannete seoseid tehtud vigade arvuga. Leiti, et paaristööna tehtavatele ülesannete lahendamisele kulus märkimisväärselt vähem aega kui individuaalselt tehes. Paaristööna tehti ka vähem vigu ning vigade parandamisele kuluv aeg oli samuti väiksem. Saadud tulemus on igati loogiline, sest probleemide koos arutamine toob esile rohkem nurki ja kitsaskohti. Üksinda tehes on abiks peamiselt ainult veebist leitavad materjalid ning õppetöö käigus konspekteeritud materjalid.

5.2 Tulemuste ennustamine käitumismustrite põhjal

Kõikidest uurimistöödest saadud tulemuste kokkuvõttena on võimalik ennustada programmeerimise kursuse õpilaste lõpptulemusi. Pereira ning Alzaid'i ja Hsiao [24, 25] uurimistööde põhjal on kõige olulisem etapp kursuse algus. Mida paremate tulemustega kursust alustada, seda meeldivam tõotab tulla ka kursuse jätk, mis omakorda päädib meeldiva resultaadiga kursuse hinde näol. Seetõttu võib kõrget kursuse hinnet oodata ka õpilastelt, kes panustavad juba varakult ja järjepidevalt oma aega harjutusülesannete lahendamisele ja teoreetilise osa analüüsimisele. Seevastu on kõige kehvemaid tulemusi oodata õpilastelt, kes tegelevad kursuse materjalidega ainult kodutööde lahendamisel ja ei pane erilist rõhku teoreetilise osa omaksvõtmisele ning koduülesannete lahendamisele. Albluwi ja Salteri [26] uuringu põhjal saab eeldada, et kõikide kursust alustanud tudengite hulgast lõpetab kursuse kõige positiivsema tulemusega naisterahvas, kellel on olemas ka varasem programmeerimise kogemus. Kõige kehvemaid tulemusi on oodata meesterahvalt, kes ei ole programmeerimisega varasemalt kokku puutunud.

Kui programmeerimise kursuse hinnatavate ülesannete seas on ka paaristööna tehtavaid ülesandeid, siis võivad need osutuda oluliseks just nendele õpilastele, kelle üldine hindeline seis ei ole kõige parem. Kuna Albluwi ja Salteri uuringu [26] tulemustes täheldati, et paaristöö tulemuslikkus on parem kui individuaalse töö tulemuslikkus, siis võib paaristöid pidada nii-öelda päästerõngaks, mis on kursuse parema lõppresultaadi kujunemise juures suureks abiks.

Kokkuvõte

Bakalaureusetöö eesmärk oli analüüsida kursusel “Programmeerimine” 2021. aasta sügisel osalenud tudengite kodutöid ning leida nende põhjal, millised kodutööde ülesanded olid kõige lihtsamad ja millised raskemad. Töö raames arvutati ja analüüsiti tulemusi iga kodutöö iga ülesande keskmise esituste arvu kohta, kodutöö ülesannete parandamisel tehtud keskmise muudatuse suuruse kohta, iga kodutöö ülesande keskmise automaattestide läbimise protsendi kohta ning iga kodutöö esitamise aegade kohta. Lisaks kirjeldati kursusel osalejate andmete põhjal tüüpilisemad tudengite profiilid ning koostati loetelu kõige enam levinumatest vigadest, mida tudengid kodutööde lahendamisel tegid.

Kõige kergemad kodutööde ülesanded olid 1. nädala kodutöö ülesanne “kodu1” ja 3. nädala kodutöö ülesanne “kodu2” ning kõige raskem ülesanne oli 10. nädala kodutöö ülesanne “kodu1”. Keskmiselt esitati igat kodutöö ülesannet 2.1 korda ning ülesannete parandamisel muutus failisuurus keskmiselt 6 protsenti. Kodutööde ülesannete keskmine lahenduse õigsus oli 66%, kodutöid esitati keskmiselt 31 tundi ning esitamise kõrghetk oli keskmiselt 21 tundi enne kodutöö esitamise tähtaega. Kursusel osalevad tudengid jagunevad kolmeks erinevaks tüübiks: 1) kodutöödega liiga hilja tegelev tudeng, 2) keskmisest püüdlikum tudeng, 3) kursuse algfaasis kursuse läbimisest loobunud, kodutööde lahendamisest loobunud või kursuse eeleksami positiivsele tulemusele sooritanud ning seetõttu kursusel osalemisest vabastatud tudeng. Kõige sagedasemad vead, mida tudengid kodutööde lahendamisel tegid, olid seotud andmete valesti kasutamisega, programmi töö tulemusega seotud aspektidest arusaamisega, ülesande tekstis olevate nüansside tähelepanuta jätmisega, õigekirjavigadega ning loogikavigadega.

Edasiarendusena saaks ülevaadet automaatkontrollide kasutamise kohta arvesse võttes täiendada olemasolevaid automaatkontrolle, kodutööde ülesannete kirjeldusi ning jagada õppijatele soovitusi kodutööde lahendamiseks. Selle bakalaureusetöö raames loodud metoodikat ja töövahendeid saaks kasutada ka kursuse “Programmeerimine” järgmiste toimumiskordade andmete analüüsimiseks ning neid tulemusi omavahel võrrelda. Soovi korral võib edaspidi proovida ka muid klasterdamise algoritme. Saab täpsemalt uurida tüüpilisemaid vigu, neid kvantifitseerida ning pakkuda välja ülesandeid ja tegevusi tehtavate vigade vähendamiseks.

Viidatud kirjandus

[1] Tartu Ülikooli õppeinfosüsteem. Kursus “Programmeerimine”. https://ois2.ut.ee/#/courses/LTAT.03.001/version/37ca53ac-440c-032f-1c5e-125c983bb04b/details (06.03.2022)

[2] Tartu Ülikooli koduleheküg. Tartu Ülikooli õppekavade loend. https://ut.ee/et/oppekavad (06.03.2022)

[3] Matthíasdóttir, Á. How To Teach Programming Languages To Novice Students? Lecturing or Not?, 2006. https://www.researchgate.net/profile/Asrun-Matthiasdottir/publication/251812193_How_to_teach_programming_languages_to_novice_students_Lecturing_or_not/links/548990140cf268d28f0afce4/How-to-teach-programming-languages-to-novice-students-Lecturing-or-not.pdf (30.04.2022)

[4] Barra jt. Automated Assessment in Programming Courses: A Case Study during the COVID-19 Era, 2020. https://www.mdpi.com/2071-1050/12/18/7451/htm (03.05.2022)

[5] Ala-Mutka, K. A Survey of Automated Assessment Approaches for Programming Assignments, 2005. https://cormack.uwaterloo.ca/papers/l6u1l75017112972.pdf (30.04.2022)

[6] Tartu Ülikooli Arvutiteaduse Instituudi kursused. Kursuse “Programmeerimine” ülesehitus nädalate kaupa. https://courses.cs.ut.ee/2021/programmeerimine/fall (08.03.2022)

[7] Tartu Ülikooli Moodle’i õpikeskkond. Kursus “Programmeerimine” https://moodle.ut.ee/enrol/index.php?id=500 (26.04.2022)

[8] Tartu Ülikooli õppeinfosüsteem. Kursus “Objektorienteeritud programmeerimine” https://ois2.ut.ee/#/courses/LTAT.03.003/version/416b088b-3102-ea07-6f2a-8a87c4407953/details (13.03.2022)

[9] Tartu Ülikooli õppeinfosüsteem. Kursus “Automaadid, keeled ja translaatorid” https://ois2.ut.ee/#/courses/LTAT.03.006/version/439833f2-d451-99b4-1c73-4f766c29a89f/details (13.03.2022)

[10] Tartu Ülikooli õppeinfosüsteem. Kursus “Programmeerimise alused” https://ois2.ut.ee/#/courses/MTAT.03.236/version/f4d0c873-df35-7fff-084c-3f20ddc046d7/details (06.04.2022)

[11] Tartu Ülikooli õppeinfosüsteem. Kursus “Algoritmid ja andmestruktuurid” https://ois2.ut.ee/#/courses/LTAT.03.005/version/ea9a3a40-eec8-f78a-2ee2-9b5f98ac3565/details (06.04.2022)

[12] Vikipeedia. Levenshteini kaugus. https://et.wikipedia.org/wiki/Levenshteini_kaugus (03.05.2022)

[13] Kursuse “Programmeerimine” 10. nädala kodutöö. https://courses.cs.ut.ee/2021/programmeerimine/fall/Main/Kodu10 (29.03.2022)

[14] Kursuse “Programmeerimine” 1. nädala kodutöö. https://courses.cs.ut.ee/2021/programmeerimine/fall/Main/Kodu1 (29.03.2022)

[15] Kursuse “Programmeerimine” 3. nädala kodutöö. https://courses.cs.ut.ee/2021/programmeerimine/fall/Main/Kodu3 (29.03.2022)

[16] Kursuse “Programmeerimine” 8. nädala kodutöö. https://courses.cs.ut.ee/2021/programmeerimine/fall/Main/Kodu8 (29.03.2022)

[17] Kursuse “Programmeerimine” 2. nädala kodutöö. https://courses.cs.ut.ee/2021/programmeerimine/fall/Main/Kodu2 (29.03.2022)

[18] Kursuse “Programmeerimine” 15. nädala kodutöö. https://courses.cs.ut.ee/2021/programmeerimine/fall/Main/Kodu15 (29.03.2022)

[19] Vikipeedia. Klasteranalüüs. https://et.wikipedia.org/wiki/Klasteranalüüs (12.04.2022)

[20] Vikipeedia. K-keskmiste klasterdamine. https://et.wikipedia.org/wiki/K-keskmiste_klasterdamine (12.04.2022)

[21] McGregor, M. 8 Clustering Algorithms in Machine Learning that All Data Scientists Should Know, 2020. https://www.freecodecamp.org/news/8-clustering-algorithms-in-machine-learning-that-all-data-scientists-should-know/ (12.04.2022)

[22] Vikipeedia. Dunningi-Krugeri efekt. https://et.wikipedia.org/wiki/Dunningi-Krugeri_efekt (30.04.2022)

[23] Kunal, N. Sklearn K-means example, 2020. https://www.kaggle.com/code/funxexcel/p1-sklearn-k-means-example/notebook (06.04.2022)

[24] Pereira jt. Explaining Individual and Collective Programming Students' Behavior by Interpreting a Black-Box Predictive Model, 2021. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9517104 (11.11.2021)

[25] Alzaid, M. ja Hsiao, I.-H. Behavioral Analytics for Distributed Practices in Programming Problem-Solving, 2019. https://ieeexplore-ieee-org.ezproxy.utlib.ut.ee/stamp/stamp.jsp?tp=&arnumber=9028583&tag=1 (11.11.2021)

[26] Albluwi, I. ja Salter, J. Using Static Analysis Tools for Analyzing Student Behavior in an Introductory Programming Course, 2020. https://www.researchgate.net/publication/342347638_Using_Static_Analysis_Tools_for_Analyzing_Student_Behavior_in_an_Introductory_Programming_Course (07.12.2021)

Lisad

I Bakalaureusetöö teostamise tarbeks loodud programmid

Kõik programmid on kirjutatud programmeerimiskeeles Python.

Zip-fail: programmid.zip

Peatükk 2

(eelduseks on, et Moodle'st alla laadidud kodutööde zip-failid on lahti pakitud)

1) failide_keskmise_suuruse_leidmine.py - programm leiab keskmise esitatud failisuuruse.
2) keskmise_esituse_aja_leidmine.py - programmis on funktsioonid, mille eesmärgiks on leida:
    a) kodutööde keskmine esituse aeg,
    b) esituste mediaanaeg,
    c) vahe keskmise esituse aja ja kodutöö esitamise tähtaja vahel tundides,
    d) vahe mediaanaja ja kodutöö esitamise tähtaja vahel tundides,
    e) keskmine esimese esituse aeg,
    f) keskmine viimase esituse aeg,
    g) vahe keskmise esimese esituse aja ja kodutöö tähtaja vahel tundides,
    h) vahe keskmise viimase esituse aja ja kodutöö tähtaja vahel tundides.

    Aegade leidmiseks on kõigepealt leitud a,b,e,f ning seejärel lisatud tulemused tabelisse (käsitsi). Seejärel on leitud tabelis olevate andmete c,d,g,h. Lisaks peaks olema alla laetud ka andmestik kodutööde automaattestide tulemuste kohta.
3) keskmise_muudatuse_suuruse_protsentides_leidmine.py - eelduseks on, et tekitatud on tabel kodutööde statistiliste andmete kohta ehk käivitatud on programm kodutöö_versioonide_arvu_ja_suuruse_erinevuse_leidmine.py. Programmis on funktsioon, mis koostab töös kirjeldatud joonise nr.7.
4) kodutöö_versioonide_arvu_ja_suuruse_erinevuse_leidmine.py - programm koostab tabeli, milles on kirjeldatud kodutöö number, ülesandefaili nimi, keskmine versioonide arv, keskmine muudatuste suurus üle versioonide, keskmine Levenshteini kaugus.
5) kodutöö_ülesannete_lahendajate_arvu_leidmine.py - programm peab olema samas kaustas, kuhu on alla laadidud kodutööde zip-failid. Programmi käivitamisel küsib

programm kasutajalt kodutöö kausta nime ja kodutöö ülesandefaili nime ning väljastab selle ülesande esitajate arvu.

6) lahenduse_õigsuse_leidmine.py - programmis on funktsioon, mis tagastab keskmise automaattestide läbimise protsendi kirjeldatud kodutöö kõikide ülesannete kohta
7) tabelid_ja_graafikud_esituste_aegade_kohta.py - programm koostab töös kirjeldatud joonise nr.4.
8) tabelid_ja_graafikud_üldine.py - programm töös kirjeldatud jooniste koostamiseks - joonised 2,3,5,6 ja 8.

Peatükk 3

1) klasterdamise_tabelite_tegemine.py - programmile antakse muutuja väärtusena ette kodutöö nädal, mille kohta koostab programm csv faili (tabeli), kus on kirjeldatud statistilised andmed selle kodutöö kõikide lahendajate kohta eraldi.
2) klasterdamise_tabelitest_ühe_üldise_tabeli_tegemine.py - eelduseks on, et programm klasterdamise_tabelite_tegemine.py on käivitatud iga kodutöö esitustega (13 korda). Programmi tulemuseks on tabel, mis on kirjeldatud lisas 2.
3) klasterdamine.py - eelduseks on, et käivitatud on programm klasterdamise_tabelitest_ühe_üldise_tabeli_tegemine.py. Programm töös kirjeldatud jooniste ja tabelite koostamiseks - joonised 9,10,11 ja tabel 4.

Peatükk 4

1) automaattestide_pisteline_kontrollimine.py - programm peab olema samas kaustas, kuhu on alla laadidud kodutööde zip-failid. Programmi käivitamisel küsib programm kasutajalt kodutöö kausta nime, kodutöö ülesandefaili nime ning tulemuste faili nime. Seejärel salvestab programm kausta etteantud nimega tulemuste faili, mille sisuks on kõikide eelnevalt juhuslikult välja valitud tudengite poolt esitatud ülesanded.
2) juhuslik.py - genereerib ja seejärel väljastab järjendi, mis sisaldab 20 juhuslikku ja üksteisest erinevat arvu etteantud vahemikust.

II Peatükis 3 kirjeldatud tabel

Exceli fail: tudengite tulemused kodutööde lahendamisel.xlsx

Tabelis olevad veerud:

1) Keskmine esituste arv - tudengi keskmine kodutööde esituste arv.
2) Keskmine esituse aeg - arvutatuna tundides enne või pärast kodutöö esitamise tähtaega.
3) Keskmine failisuurus - keskmine kodutöö lahendusena esitatud faili suurus baitides.
4) Keskmine muudatuse suurus (%) - keskmine muudatuse suurus kahe erineva esitatud versiooni vahel.
5) Keskmine lahenduse õigsus (%) - keskmine automaattestide läbimise protsent.
6) Kodutööde arv - lisainformatsioon selle kohta, mitu erinevat kodutööd tudeng kursuse jooksul esitas.

Tabelis olevad veerud on arvutatud selle põhjal, mitu kodutööd konkreetne tudeng esitas, mitte kõikide kursusel olnud kodutööde arvestuses kokku.

III Litsents

Mina, **Kaspar Saarem**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   Programmide automaattestide kasutusuuring,

mille juhendaja on Reimo Palm,

   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.

Kaspar Saarem
08.05.2022