TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristjan Mänd 206360

Sudoku lahendaja

Bakalaureusetöö

Juhendaja: Peeter Ellervee
PhD

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kristjan Mänd

05.10.2023

Annotatsioon

Käesoleva lõputöö eesmärgiks oli mugava kasutajaliidesega sudoku lahendaja veebirakendus. Mugavaks teeb selle võimalus telefoniga skaneerida ja automaatselt ära täita sudoku ruudud olemasolevate numbritega.

Lõputöös on kirjeldatud sudoku ajalugu ja reeglid. Lisaks uuriti erinevaid sudoku lahendamise meetodeid ja otsiti sobiv algoritm.

Enne töö tegemist tutvuti olemasolevate lahendustega ning jõuti järeldusele, et sellisel kujul sudoku lahendajat ei eksisteeri ning teema on ennast õigustav.

Töö käigus uuriti ka erinevaid *JavaScript*'i teeke optilise märgituvastuse jaoks. Võrreldi plusse ja miinuseid ning valiti rakenduse jaoks sobiv.

Nii sudoku lahendaja algorim kui ka numbrituvastus ja ruutude täitmine testiti vastavalt ühiktestidega ja manuaalselt.

Töö tulemusena valmis veebirakendus, kus kasutaja saab mugavalt telefoniga skaneerides või käsitsi ära täita olemasolevad sudoku ruudud ja seejärel küsida vihjet või kogu lahendust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 7 peatükki, 25 joonist, 1 tabelit.

Abstract

Sudoku solver

The goal of this thesis was a sudoku solver web application with a convenient user interface. What makes this web application convenient is the ability to scan the sudoku with your phone and automatically fill in the sudoku squares with existing numbers.

In the thesis there is described the history of sudoku and the rules. In addition, various sudoku solving methods were studied and a suitable algorithm for solving the sudoku was researched for.

Before the work was done, the existing solutions were examined, and the conclusion was reached that there is no sudoku solver in this form and the topic is self-explanatory.

Various JavaScript libraries for optical character recognition were also studied during the work. The decision for choosing the best library for the application was made after comparing the pros and cons.

Both the sudoku solver algorithm and the number recognition with filling the correct sudoku squares were tested with unit tests and manually, respectively.

As a result of the work, a web application was completed, where the user can comfortably scan sudoku with the phone to fill in the sudoku squares automatically or fill them by hand. After that the user can ask for a hint or the entire solution.

The thesis is in Estonian and contains 28 pages of text, 7 chapters, 25 figures, 1 table.

Lühendite ja mõistete sõnastik

JSON	*JavaScript Object Notation*. Lihtne andmevahetusvorming.
Kolmik	Kolm ruutu veerus, reas või sektsioonis, kus on samade kandidaatide kombinatsioon kolmest väärtusest (*Naked Triple*).
Nelik	Neli ruutu veerus, reas või sektsioonis, kus on samade kandidaatide kombinatsioon neljast väärtusest (*Naked Quad*).
Paar	Kaks ruutu veerus, reas või sektsioonis, kus on samad kandidaadid (*Naked Pair*).
Peidetud üksikkandidaat	Unikaalne väärtus veerus, reas või sektsioonis, kus samalajal on ka teisi kandidaate (*Hidden Single*).
Sektsioon	9x9 sudoku väiksem 3x3 eraldatud osa.
Teek	Funktsioonide komplekt teatud tüüpi ülesannete lahendamiseks.
Ühiktest	Alamprogrammi test.
Üksikkandidaat	Ainuke võimalik väärtus, mis sobib antud ruutu (*Naked Single*).

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva lõputöö teemaks oli luua sudoku lahendaja rakendus. Olemasolevates lahendustes on sudoku ruutude täitmine üsna ebamugav. Kõik ruudud tuleb ise ükshaaval käsitsi sisse trükkida. Seetõttu oli eesmärgiks võimalikult mugav teadaolevate sudoku ruutude täitmine.

Ülesandeks oligi teha sudoku lahendaja veebirakendus, kus on kasutajal mugav ära täita teadaolevad sudoku ruudud ja siis hakata vihjeid või lahendust küsima.

Rakenduse tingimusteks olid põhilised suduoku lahendajas vajaminevad funktsionaalsused. Nendeks olid:

- Sudoku laua käsitsi täitmine.
- Vihje küsimine.
- Lahenduse küsimine.
- Sudoku laua puhastamine.
- Vea nägemine, kui sisestatud number ei vasta reeglitele.
- Kaamera abil sudoku ruutude täitmine.

Töös tuleb esialgu juttu sudoku ajaloost ja sudoku reeglitest. Seejärel on ülevaade olemasolevatest sudoku veebi- ja mobiilirakendustest. Saame aimu sudoku lahendamise meetoditest ja vaatame, milline näeb välja valmisolev rakendus välja ja kuidas see töötab ning lõpetuseks väike ülevaade ka testimisest.

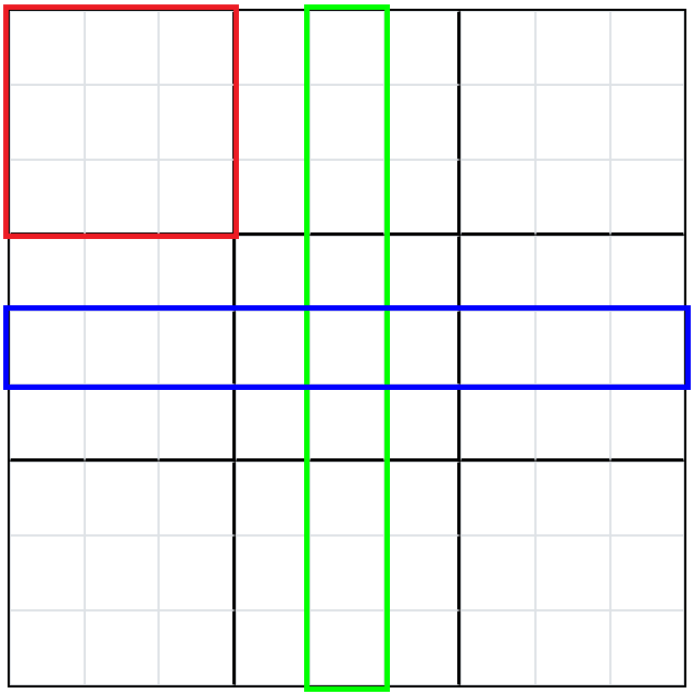
2 Sudoku mängu kirjeldus

Sudoku on laialt levinud loogika mäng, mida leiab tihti ajalehtede viimastelt lehekülgedelt. Seetõttu on see leidnud palju huvilisi. Mäng on sobilik kõigile, kes soovivad lõbusat väljakutset ning arendada mõtlemisoskust. Reegleid on küll mitmeid, kuid need on lihtsad.

2.1 Ajalugu

Sudoku ajalugu ulatub tagasi 18. sajandisse, kui Šveitsi matemaatik Leonhard Euler mõtles välja mängu nimega “Latin Squares”. Tema kontseptsioon oli küll Sudokust kaugel, aga siiski peetakse selle mängu põhimõtteid inspiratsiooni allikaks Sudoku mängu loomisel. [1]

Esimene tänapäevasele Sudokule sarnane numbrimõistatus avaldati 1979. aastal ajakirjas “Dell Pencil Puzzles and Word Games”. Toona sai mäng endale nimeks “Number Place”. Selles mängus tuli ära täita 9x9 ruudustik numbritega 1 kuni 9, kusjuures ühes reas, veerus ega ka väiksemas 3x3 sektsioonis ei tohtinud ükski number korduda. (Joonis 1) Mängu loojaks peetakse Howard Garns’i. [1]



Joonis 1. 9x9 sudoku väli, kus on märgitud punasega 3x3 sektsioon, rohelisega veerg ja sinisega rida

1984. aastal avaldati mäng ka Jaapani ajakirjades, seal sai see nimeks “Suuji wa dokushin ni kaguru”, mis tähendab, et numbrid peavad esinema ainult korra. Mängus olid samad põhimõtted, mis varem, aga lisati kaks uut reeglit. Esimene neist oli selleks, et muuta mõistatus inimestele atraktiivsemaks. Nimelt pidid sudoku alguses olemas olevad numbrid moodustama erinevaid mustreid. (Joonis 2) Teine reegel oli, et kõigist 81st ruudust võib olla alguses täidetud maksimaalselt 32 ruutu. Nii on mäng piisavalt väljakutset esitav ka kõige kergema raskusastme puhul. Mäng läks Jaapanis massidesse ja tänu sellele lühendati nime. Jaapani keeles pandi kokku sõnad number (数 - Su) ja üksik (独 - Doku). Nimeks saigi Sudoku. [1]

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Joonis 2. Sudoku mustriga [2]

2.2 Reeglid

Sudokusid on tänapäeval erinevaid. Levinuim on klassikaline variant, ehk 9x9 ruudustikuga mäng.

Klassikalise sudoku reeglid:

- Sudoku ruudustik on 9x9.
- Tohib kasutada ainult numbreid 1 kuni 9.
- Igas veerus, reas, või 3x3 regioonis võib igat numbrit kasutada ainult ühe korra.
- Mäng on läbi, kui kõik sudoku ruudud on numbritega õigesti täidetud.

Nendele reeglitele tuginedes on arendatud ka minu sudoku lahendaja veebirakendus. Samamoodi kasutavad neid reegleid ka järgnevas peatükis uuritud olemasolevad sudoku lahendajad.

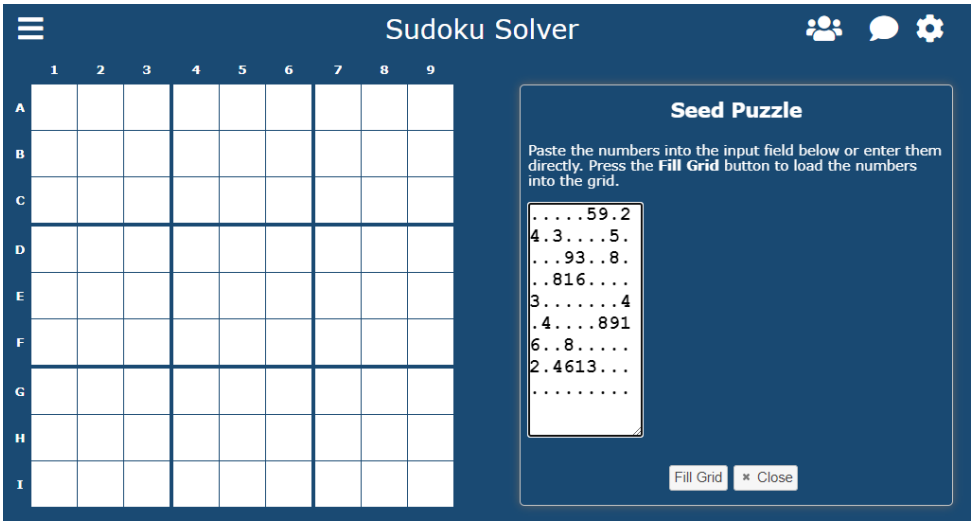
3 Olemasolevate sudoku lahendajate analüüs

Erinevaid sudoku lahendamise rakendusi on tehtud mitmeid. Enne enda versiooni tegemist tuli uurida, millised on eksisteerivad lahendused ja kas äkki juba on mõni selline tehtud, nagu mina teha plaanin. Otsisin neist mõningad üles ja uurisin neid.

3.1 Veebi lahendused

Sudoku Spoiler – saab käsitsi ära täita olemasolevad sudoku ruudud, saab küsida lahendust ja kindla ruudu lahendamist. Kui on sudokul mitu lahendust, siis kuvatakse maksimaalselt 10 lahendust. Üsna vähe võimalusi, kuid kõik vajalik olemas, et sudoku saaks lahendatud. [3]

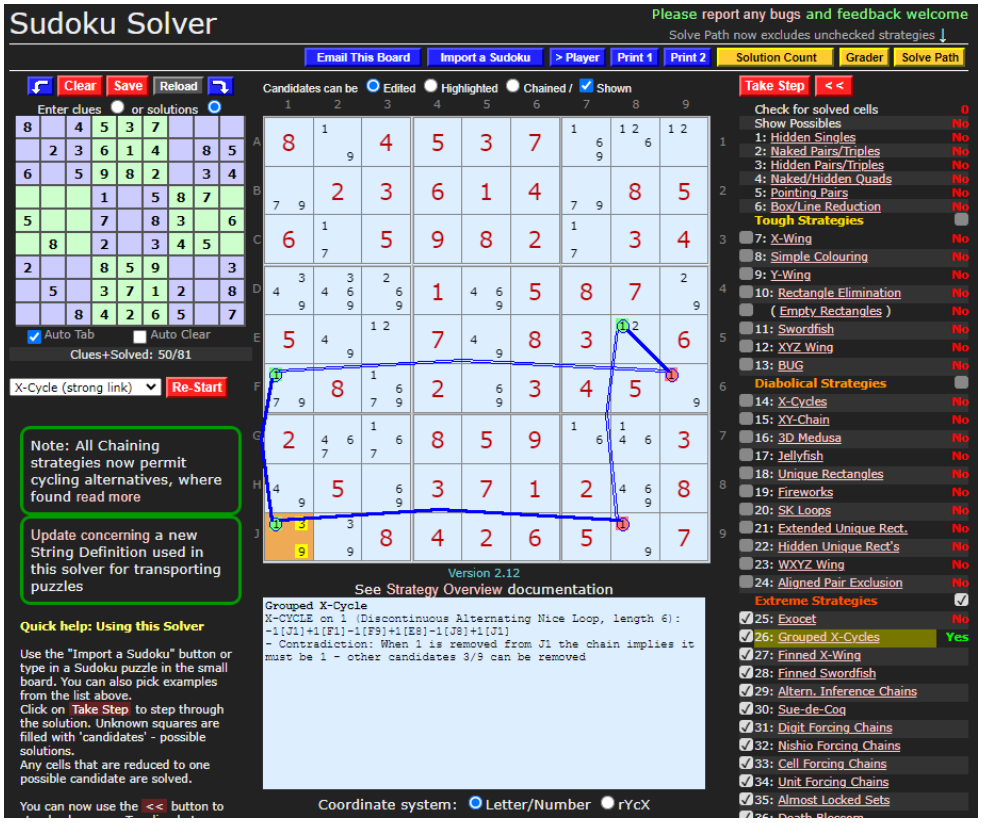
Sudoku Solutions – saab samuti täita, küsida lahendust ja kindla ruudu lahendamist. Ruutude käsitsi täitmise kõrval on võimalus ka laadida juba eelnevalt salvestatud pusle unikaalse id koodi järgi ja ka kleepida sudoku sellisel kujul, kus "." tähendab lahendamata ruutu (Joonis 3). Lisaks on palju muid erinevaid võimalusi. Saab lasta osaliselt lahendada, selleks tuleb seadedest määrata eelistused. Erinevad võimalused analüüsimiseks: *Check* – vaatab, kas pusle on valiidne, lahendatav ja omab ühte lahendust. Kui sudokul on mitu lahendust, siis on võimalik neid näha pärast analüüsi. Saab hinnata raskusastet ja küsida vihjet. Veel on võimalik genereerida kõikidele ruutudele kandidaadid ja neid ka ise muuta. Kokkuvõtteks võiks öelda, et väga palju erinevaid võimalusi, et sudoku lahendamine saaks olema lihtne ja huvitav. [4]



Joonis 3. Sudoku täitmine numbrite ja punktide jada abil

Sudoku.com – olemuselt lihtne. Ainuke võimalus lahendada ära terve sudoku. [5]

SudokuWiki – esmamulje oli, et kõik on väga kirju ja mitte midagi aru ei saa. Natuke katsetades leiab jällegi palju erinevaid võimalusi ja tundub huvitav. Kuid on ka mõned puudujäägid. Näiteks lubab ühte ruutu kirjutada kahekohalist numbrit, mis on sudoku reeglite vastane. Ei leidnud ka võimalust küsida korraga terve sudoku lahendust. Sai ainult sammu kaupa edasi liikuda, aga huvitav on see, et näidatakse, mis strateegiat kasutatakse. (Joonis 4) Pole kohandatud kasutamiseks mobiilis. [6]



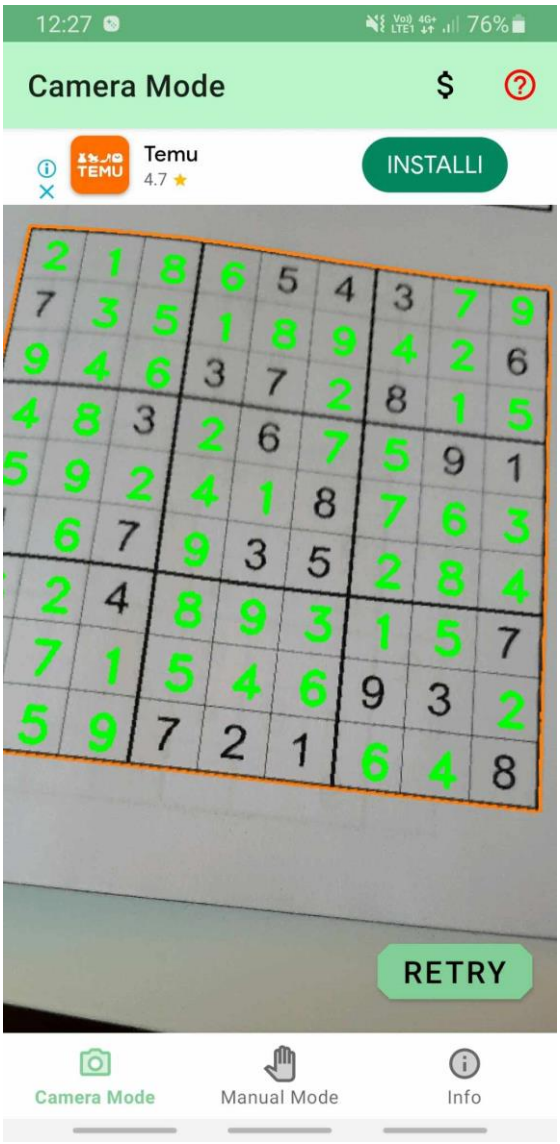
Joonis 4. Sudoku lahendamise strateegia kuvamine

AnySudokuSolver – väga lihtne ja algne. Täidad teadaolevad ruudud ära ja saad küsida lahendust. Kui sudokul on mitu lahendust, siis öeldakse seda, aga kõiki võimalike lahendusi näha pole. [7]

3.2 Mobiilirakendused

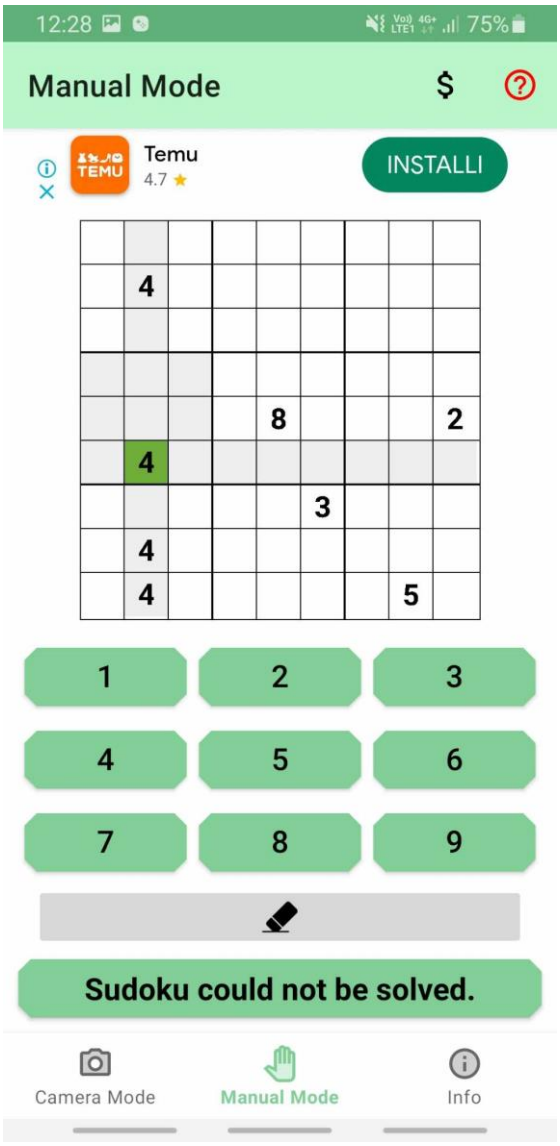
Sudoku Solver (Shai Alkoby) – Tavaline. Lihtne kasutada. Saab sudoku ruute täita ja lahendust küsida. [8]

Sudoku Solver (Camera) (Robinson Industries) – Kõikides eelnevates lahendustes tuli ruudud käsitsi täita. Selles rakenduses saab sarnaselt minu lahendusele sudokut telefoni kaameraga skaneerida. Tuvastab sudoku suurepäraselt, aga liialt kiirustab, st. lahendab sudoku selliselt ära, et kasutaja osa lahendust ei näegi (Joonis 5). [9]



Joonis 5. Kaamera tuvastusega sudoku lahendamine

On võimalus ka käsitsi ruudud ära täita, aga seal olev valideerimine ei ole minu meelest piisavalt hästi lahendatud. Sudokut valideeritakse alles siis, kui vajutatakse nuppu. (Joonis 6)



Joonis 6. Vea mitte kuvamine

3.3 Kokkuvõte

Olemasolevad veebirakendused pakuvad väga palju erinevaid võimalusi sudoku lahendamiseks. Kahjuks ükski neist ei paku võimalust teadaolevaid sudoku ruute täita kasutades telefoni kaamerat. Mobiilirakenduste hulgas leidub üks rakendus, mis kasutab telefoni kaamerat, kuid sellel on ka omad miinused. Näiteks sudoku kohene lahendamine. Lisaks on tegemist mobiilirakendusega, mis tähendab, et see tuleb endale telefonisse installeerida. Kui kasutada rakendust harva, siis võtab see mõttetult ruumi.

4 Sudoku lahendamise meetodid

Selleks, et sudokut oleks võimalik lahendada, kasutades loogikat, on välja mõeldud erinevad meetodid. Toon neist välja mõned, mis on lihtsamad. Lisaks neile on olemas veel keerukamad meetodid nagu näiteks “X-Wing”, “Swordfish” ja “Forcing Chains”.

4.1 Üksikkandidaat

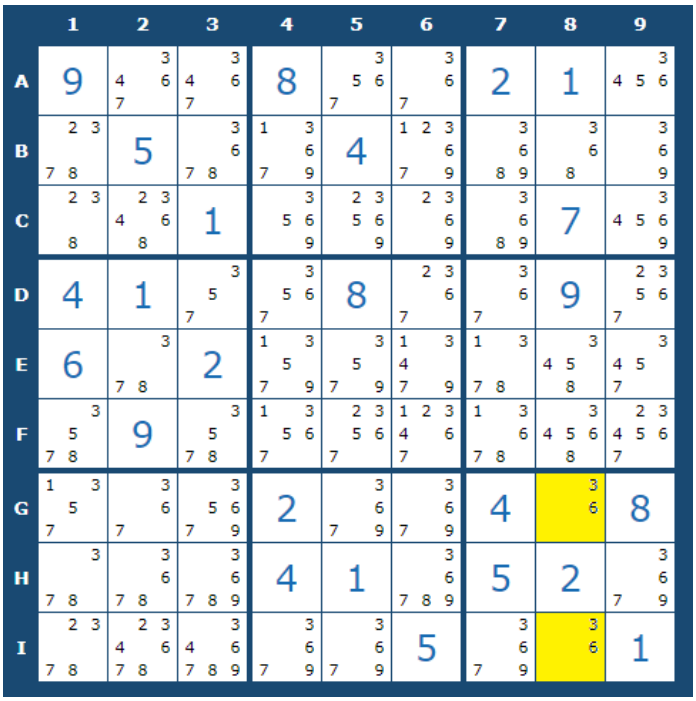
Üksikkandidaatide leidmiseks, tuleks ära täita kõik sudoku ruudud kõikide sobivate kandidaatidega ja seejärel on näha, millisesse ruutu sobib ainult üks väärtus ning need ruudud saab kohe ära täita. (Joonis 7) [10]

	1	2	3	4	5	6	7	8	9
A	3 4 9	8	2 3 4 6 9	3 4 5 6	2 3 4 6	1	2 6 7 9	2 6	2 6 7 9
B	7	5	1 2 4 6 9	4 6	2 4 6 8	4	1 2 6 8 9	1 2 6 8	3
C	1 3	3 6	1 2 3 6	9	2 3 6 8	7	1 2 6 8	5	4
D	3 4 5 9	2	3 4 5 6 7 9	1 3 4	1 3 4 9	8	1 4 5 6 7	1 6	1 5 6 7
E	4 8 9	1	4 8 9	7	5	6	2 4 8	3	2
F	3 4 5 8	3 4 6	3 4 5 6 7 8	2	1 3 4	3 4	1 4 5 6 7 8	9	1 5 6 7
G	6	7	1 3 4 5 8 9	1 3 4 5	1 3 4 9	2	1 3 5 9	1	1 5 9
H	2	3 4 9	1 3 4 5 9	1 3 4 5 6	1 3 4 6 9	3 4 5 9	1 3 5 6 9	7	8
I	1 3 5 9	3 9	1 3 5 9	8	1 3 6 7 9	3 5 9	1 2 3 5 6 9	4	1 2 5 6 9

Joonis 7. Üksikkandidaatidega ruudud

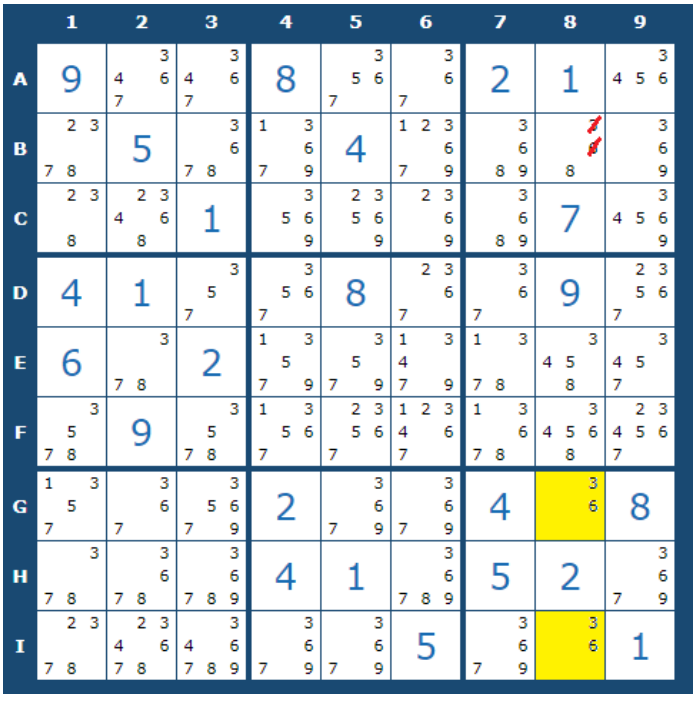
4.2 Paar

Kui kõik sobivad kandidaadid on kirjas ja sudokul laual leidub ühes reas või veerus või sektsioonis selline paar, kus mõlemas ruudus on kaks kandidaati ja need on samad, siis võib kõikidest teistest selles veerus, reas või sektsioonis olevatest ruutudest need kandidaadid eemaldada, kuna need väärtused peavad kindlasti olema leitud paaris. (Joonis 8, Joonis 9) Selle tulemusel võivad välja ilmuda näiteks üksikkandidaadid. (Joonis 10) [10]



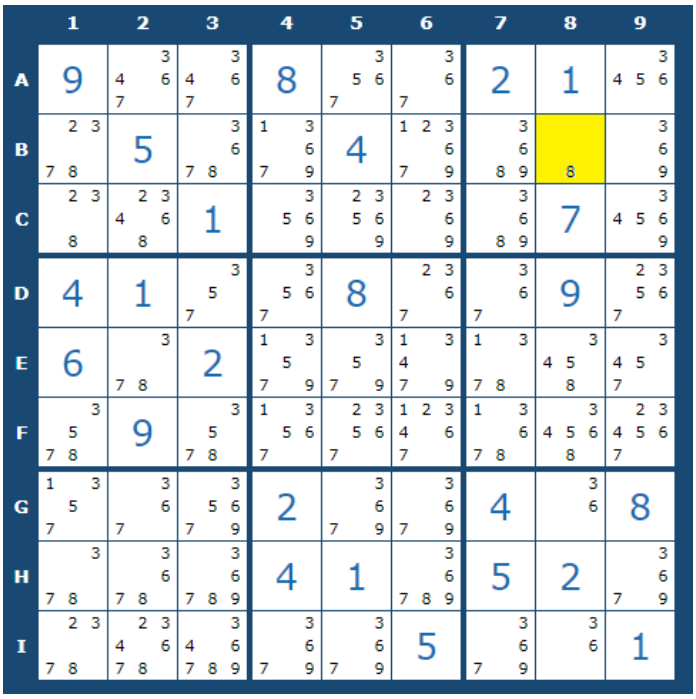
	1	2	3	4	5	6	7	8	9
A	9	3 6 4 7	3 6 4 7	8	3 5 6 7	3 6 7	2	1	3 4 5 6
B	2 3 7 8	5	3 6 7 8	1 3 6 9 7	4	1 2 3 6 9 7	3 6 9 8 9	3 6 8	3 6 9
C	2 3 8	2 3 6 4 8	1	3 6 9 5	2 3 6 9 5	2 3 6 9	3 6 9 8 9	7	3 6 9 4 5
D	4	1	3 5 7	3 6 5 6 7	8	2 3 6 7	3 6 7	9	2 3 6 5 6 7
E	6	3 7 8	2	1 3 5 7 9	3 5 7 9	1 3 4 7 9	1 3 7 8	3 4 5 8	3 4 5 7
F	3 5 7 8	9	3 5 7 8	1 3 6 5 6 7	2 3 6 5 6 7	1 2 3 6 4 7	1 3 6 7 8	3 4 5 6 8	2 3 6 4 5 6 7
G	1 3 5 7	3 6 7	3 6 5 6 9 7	2	3 6 9 7 9	3 6 9 7	4	3 6	8
H	3 7 8	3 6 7 8	3 6 7 8 9	4	1	3 6 7 8 9	5	2	3 6 9 7
I	2 3 7 8	2 3 6 4 7 8	3 6 4 7 8 9	3 6 7 9	3 6 7 9	5	3 6 7 9	3 6	1

Joonis 8. Paari leidmine



	1	2	3	4	5	6	7	8	9
A	9	3 6 4 7	3 6 4 7	8	3 5 6 7	3 6 7	2	1	3 4 5 6
B	2 3 7 8	5	3 6 7 8	1 3 6 9 7	4	1 2 3 6 9 7	3 6 9 8 9	3 6 9 8	3 6 9
C	2 3 8	2 3 6 4 8	1	3 6 9 5	2 3 6 9 5	2 3 6 9	3 6 9 8 9	7	3 6 9 4 5
D	4	1	3 5 7	3 6 5 6 7	8	2 3 6 7	3 6 7	9	2 3 6 5 6 7
E	6	3 7 8	2	1 3 5 7 9	3 5 7 9	1 3 4 7 9	1 3 7 8	3 4 5 8	3 4 5 7
F	3 5 7 8	9	3 5 7 8	1 3 6 5 6 7	2 3 6 5 6 7	1 2 3 6 4 7	1 3 6 7 8	3 4 5 6 8	2 3 6 4 5 6 7
G	1 3 5 7	3 6 7	3 6 5 6 9 7	2	3 6 9 7 9	3 6 9 7	4	3 6	8
H	3 7 8	3 6 7 8	3 6 7 8 9	4	1	3 6 7 8 9	5	2	3 6 9 7
I	2 3 7 8	2 3 6 4 7 8	3 6 4 7 8 9	3 6 7 9	3 6 7 9	5	3 6 7 9	3 6	1

Joonis 9. Kandidaatide eemaldamine teistest ruutudest

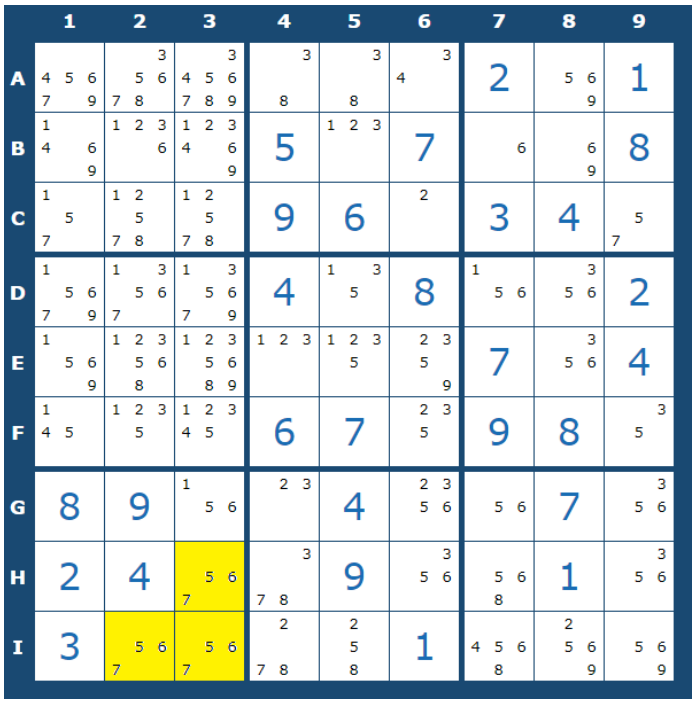


Joonis 10. Üksikkandidaadi ilmumine pärast „paari“ meetodi rakendamist

4.3 Kolmik

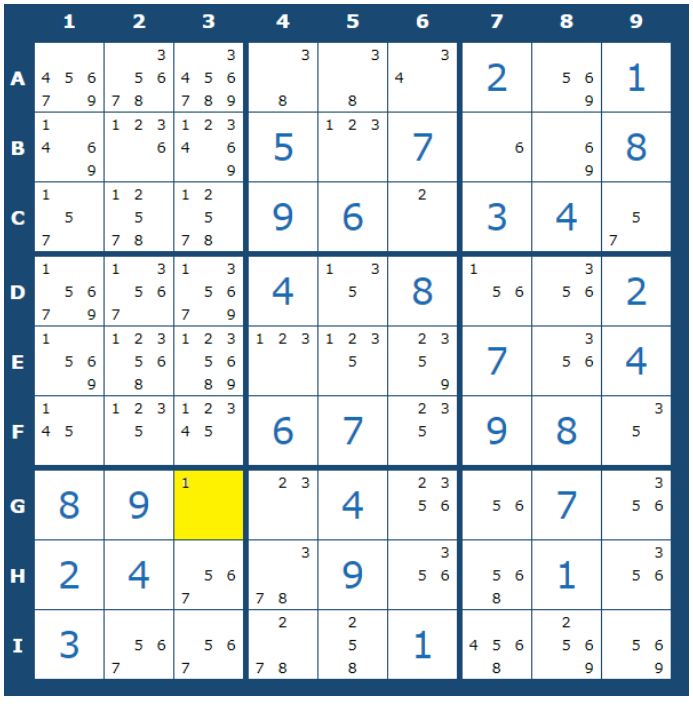
Sarnaselt paariga on sama metoodikat võimalik rakendada ka kolmiku puhul. Sel juhul tuleb kahe ruudu ja kandidaadi asemel üles otsida ruudust, veerust või sektsioonist kolm sellist ruutu, kus kõigis on mingisugune kombinatsioon kolmest kandidaatist. Igas ruudus ei pea olema kolm kandidaati, võib olla ka kaks. Näiteks kui ühes ruudus on kandidaatideks 5, 8 ja 9, siis kolmiku jaoks sobivad samas reas, veerus või sektsioonis ruudud, kus on samad kolm kandidaati või on nendeks 5 ja 8 või 5 ja 9 või 8 ja 9.

Kui sobivad ruudud on leitud, siis võib vastavalt leitud kohale, kas veerus, reas või sektsioonis, need kandidaadid teistest ruutudest ära eemaldada, sest need kolm väärtust peavad esinema just neis ruutudes. (Joonis 11) Selle tulemusel ilmub jälle esile näiteks üksikkandidaat. (Joonis 12) [10]



	1	2	3	4	5	6	7	8	9
A	4 5 6 7 9	3 5 6 7 8	3 4 5 6 7 8 9	3 8	3 8	3 4	2	5 6 9	1
B	1 4 6 9	1 2 3 6	1 2 3 4 6 9	5	1 2 3	7	6	6 9	8
C	1 5 7	1 2 5 7 8	1 2 5 7 8	9	6	2	3	4	5 7
D	1 5 6 7 9	1 3 5 6 7	1 3 5 6 7 9	4	1 3 5	8	1 5 6	3 5 6	2
E	1 5 6 9	1 2 3 5 6 8	1 2 3 5 6 8 9	1 2 3	1 2 3 5	2 3 5 9	7	3 5 6	4
F	1 4 5	1 2 3 5	1 2 3 4 5	6	7	2 3 5	9	8	3 5
G	8	9	1 5 6	2 3	4	2 3 5 6	5 6	7	3 5 6
H	2	4	5 6 7	3 7 8	9	3 5 6	5 6 8	1	3 5 6
I	3	5 6 7	5 6 7	2 7 8	2 5 8	1	4 5 6 8	2 5 6 9	5 6 9

Joonis 11. Kolmiku leidmine

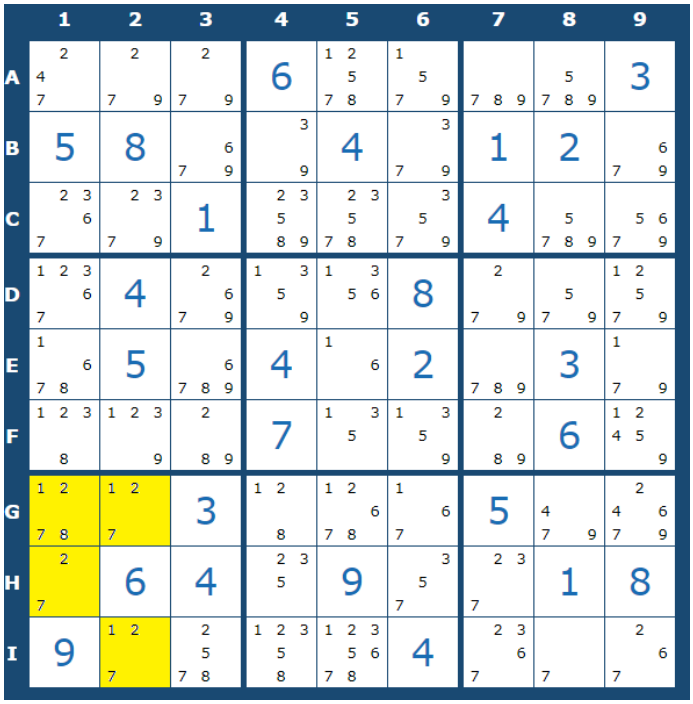


	1	2	3	4	5	6	7	8	9
A	4 5 6 7 9	3 5 6 7 8	3 4 5 6 7 8 9	3 8	3 8	3 4	2	5 6 9	1
B	1 4 6 9	1 2 3 6	1 2 3 4 6 9	5	1 2 3	7	6	6 9	8
C	1 5 7	1 2 5 7 8	1 2 5 7 8	9	6	2	3	4	5 7
D	1 5 6 7 9	1 3 5 6 7	1 3 5 6 7 9	4	1 3 5	8	1 5 6	3 5 6	2
E	1 5 6 9	1 2 3 5 6 8	1 2 3 5 6 8 9	1 2 3	1 2 3 5	2 3 5 9	7	3 5 6	4
F	1 4 5	1 2 3 5	1 2 3 4 5	6	7	2 3 5	9	8	3 5
G	8	9	1	2 3	4	2 3 5 6	5 6	7	3 5 6
H	2	4	5 6 7	3 7 8	9	3 5 6	5 6 8	1	3 5 6
I	3	5 6 7	5 6 7	2 7 8	2 5 8	1	4 5 6 8	2 5 6 9	5 6 9

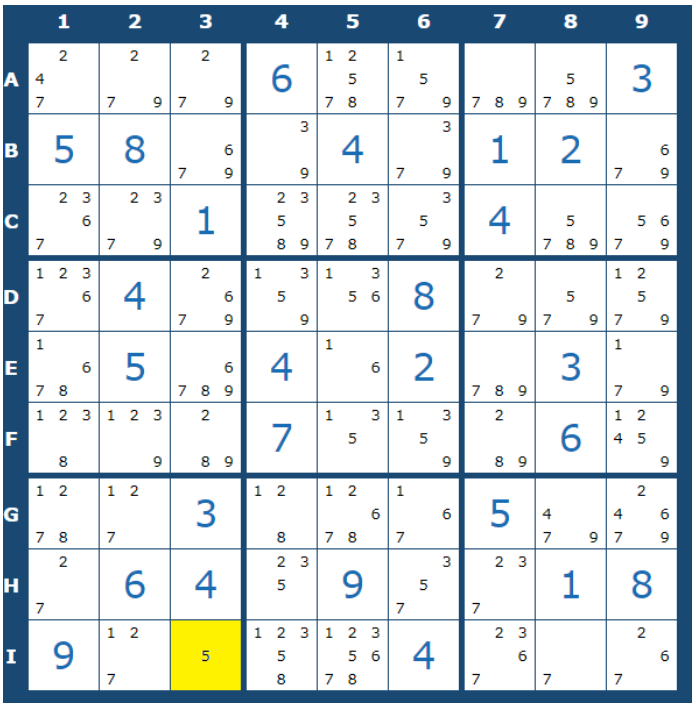
Joonis 12. Üksikkandidaadi ilmumine pärast „kolmiku“ meetodi rakendamist

4.4 Nelik

Neliku puhul tuleb otsida üles neli ruutu, mis kõik sisaldavat mingit kombinatsiooni neljast võimalikust kandidaadist ja ei sisalda ühtegi muud kandidaati. Kui sellised ruudud on leitud, siis võib jällegi vastavalt leitud reale, veerule või sektsioonile need kandidaadid teistest ruutudest eemaldada, kuna saab kindel olla, et need neli väärtust on just nendes leitud ruutudes. (Joonis 13) Kui need kandidaadid on teistest ruutudest eemaldatud, siis ilmub jälle nähtavale üksikkandidaat. (Joonis 14) [10]



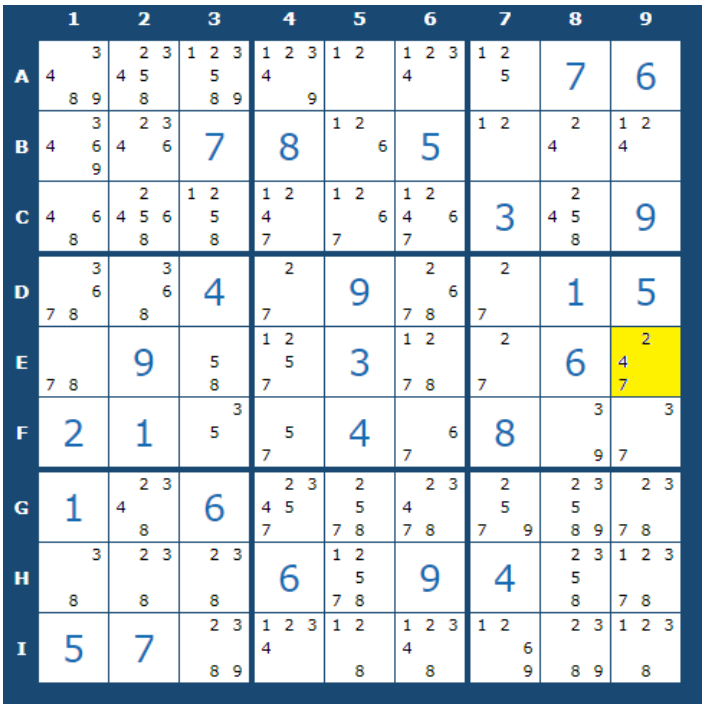
Joonis 13. Neliku leidmine



Joonis 14. Üksikkandidaadi ilmumine pärast „neliku“ rakendamist

4.5 Peidetud üksikkandidaat

Kui üks mitmest ruudus olevast kandidaadist on ainus reas, veerus või sektsioonis, siis seda kandidaati võib nimetada peidetud üksikkandidaadiks. Peidetud seetõttu, et ta ei ole ainuke kandidaat selles ruudus, aga kuna see väärtus esineb reas, veerus või sektsioonis ainult selles ruudus, siis ta on üksikkandidaat ja võib ruudu selle väärtusega ära täita. (Joonis 15) Toodud näites asub number 4 selles sektsioonis ainult ühes ruudus, seetõttu võib teisi ruudus olevaid kandidaate eirata. [11]



Joonis 15. Peidetud üksikkandidaat

Samamoodi nagu eelmiste strateegiatega, saab ka seda rakendada paari, kolmiku ja neliku puhul.

4.6 Tagurdamismeetod

Tagurdamismeetod on meetod, mis ei põhine loogikal, sudokut lahendatakse jõuga. Selle meetodi puhul proovitakse läbi erinevaid väärtusi kuni leitakse sobiv lahendus. Lahendamist alustatakse ühest ruudust, kuhu proovitakse erinevaid arve, kui leitakse esimene sinna ruutu sobiv arv. Seejärel liigutakse järgmise ruudu juurde ja tehakse sama, mis esimese ruuduga, proovitakse väärtusi, kuni leitakse esimene sobiv. Kui leidub sobiv, siis minnakse kolmanda ruudu juurde ja nii edasi. Kui sobivat ruutu ei leitud, siis minnakse eelmise ruudu juurde tagasi ja leitakse sinna järgmine sobiv arv.

See meetod on väga laialt kasutuses just sudoku lahendaja algoritmi kirjutamisel.

4.7 Meetodi valik

Valituks osutus tagurdamismeetod, sest see on üks populaarsemaid sudoku lahendamise algoritme programmeerimises. Kuna see on niivõrd populaarne, siis leidub ka internetis

erinevaid lahendusi. Ka minu rakendus kasutab internetist leitud lahendust, kuna lõputöö peamine eesmärk ei olnud enda algoritmi realiseerimine, vaid võimalikult mugavalt kasutatav sudoku lahendaja.

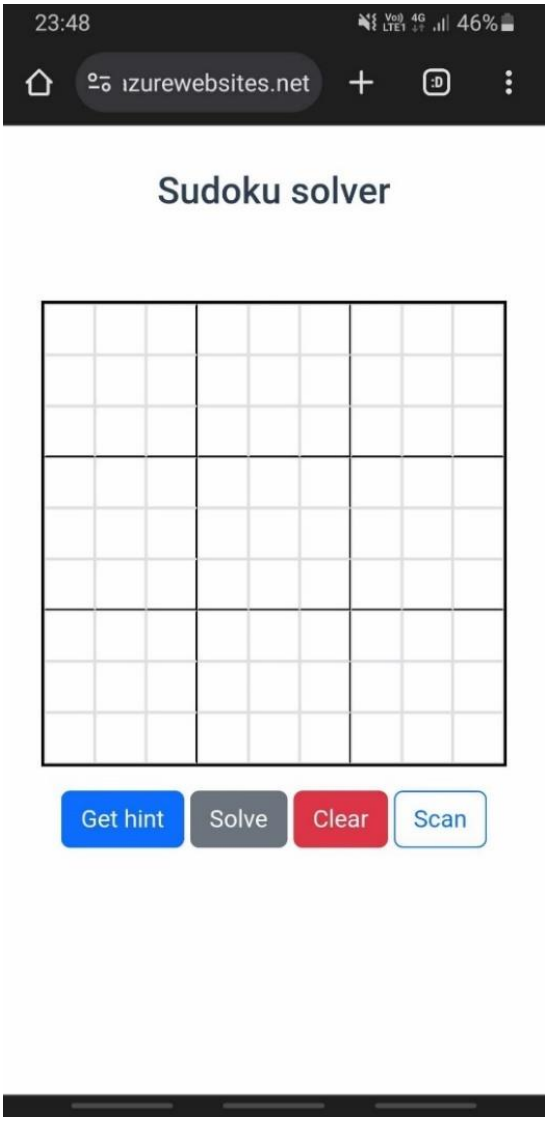
5 Sudoku lahendaja realisatisoon

Kui olemasolevad lahendused said analüüsitud ja sai tutvutud erinevate sudoku lahendamise meetoditega ning valitud ka sobiv meetod oma rakenduse jaoks, siis oli aeg alustada sudoku lahendaja realisatsiooniga.

5.1 Kasutajaliides

Kasutajaliides sai tehtud võimalikult lihtne ja intuitiivne. Avades veebirakenduse on näha sudoku ruudustikku, kuhu saab kohe hakata numbreid sisestama.

Sudoku ruudustiku all on 4 nuppu: *Get hint*, *Solve*, *Clear*, *Scan*. (Joonis 16)



Joonis 16. Sudoku rakenduse kasutajaliides

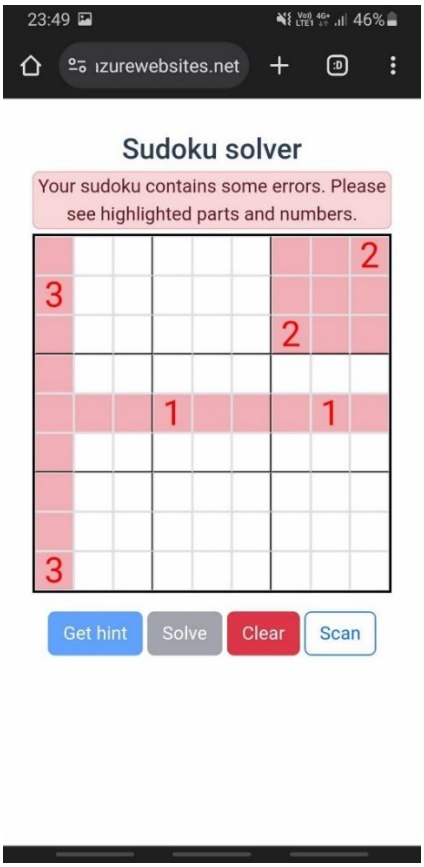
Get hint – täidab ära ühe võimaliku sudoku ruudu

Solve – lahendab ära terve sudoku

Clear – puhastab mänguvälja

Scan – nupule vajutades avatakse kasutaja telefoni kaamera ja ilmub 2 uut nuppu: *Close camera* ja *Snap*. Vajutades nuppu *Close camera*, pannakse kaamera kinni ja on näha jälle sudoku ruudustik. Vajutades nuppu *Snap* tehakse pilt, mida programm hakkab töötlema ja mille põhjal täidab ära sudoku ruudud.

Selleks, et kasutaja teaks, kas tema sisestatud numbrid lähevad vastuollu sudoku reeglitega, jooksutame pärast igat muutust valideerimist. Kui sudoku ei ole korrektne, kuvatakse kasutajale vea teksti, et sudoku sisaldab vigu ja palun vaadata esile tõstetud osi ja numbreid. Kui viga on näiteks mingis reas, siis rakendus tõstab vastava rea ja vastavad numbrid esile. Samamoodi toimitakse, kui viga esineb veerus või väiksemas 3x3 sektsioonis. (Joonis 17)

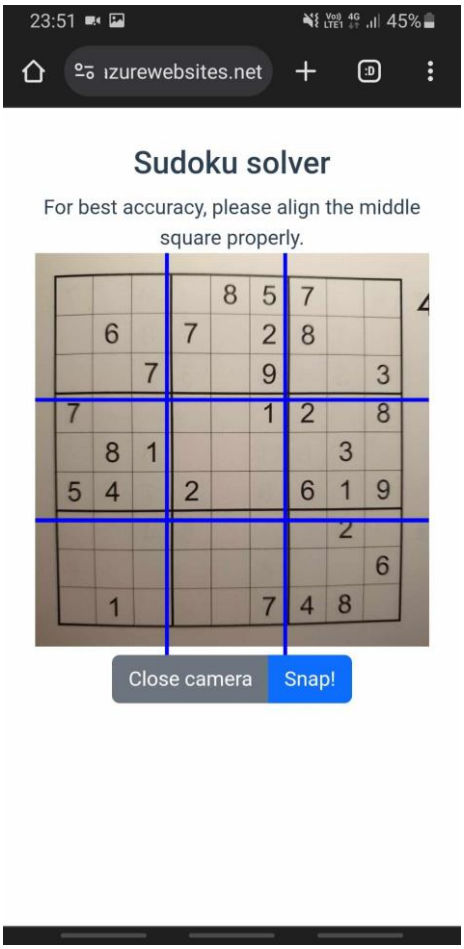


Joonis 17. Veateate kuvamine sudoku rakenduses

5.2 Sudoku väljade täitmine

Sudoku ruute saab täita käsitsi nagu ka igas teises sudoku lahendaja rakenduses. Vajutades ruudule, kuhu on soov mingit numbrit kirjutada, avaneb telefonis numbrite klaviatuur.

Lisaks on võimalik sudoku ruute lasta rakendusel endal täita. Vajutades nuppu *Scan* avaneb kaamera pilt. Selleks, et programm täidaks ruudud võimalikult täpselt, tuleks joondada sudoku ruudud vastavate joontega kaamera pildi peal. Kui jooned on omavahel hästi joondatud, tuleb vajutada nuppu *Snap*, mis teeb sudokust pildi ja hakkab seda töötlema. (Joonis 18)



Joonis 18. Kaamera vaade sudoku rakenduses

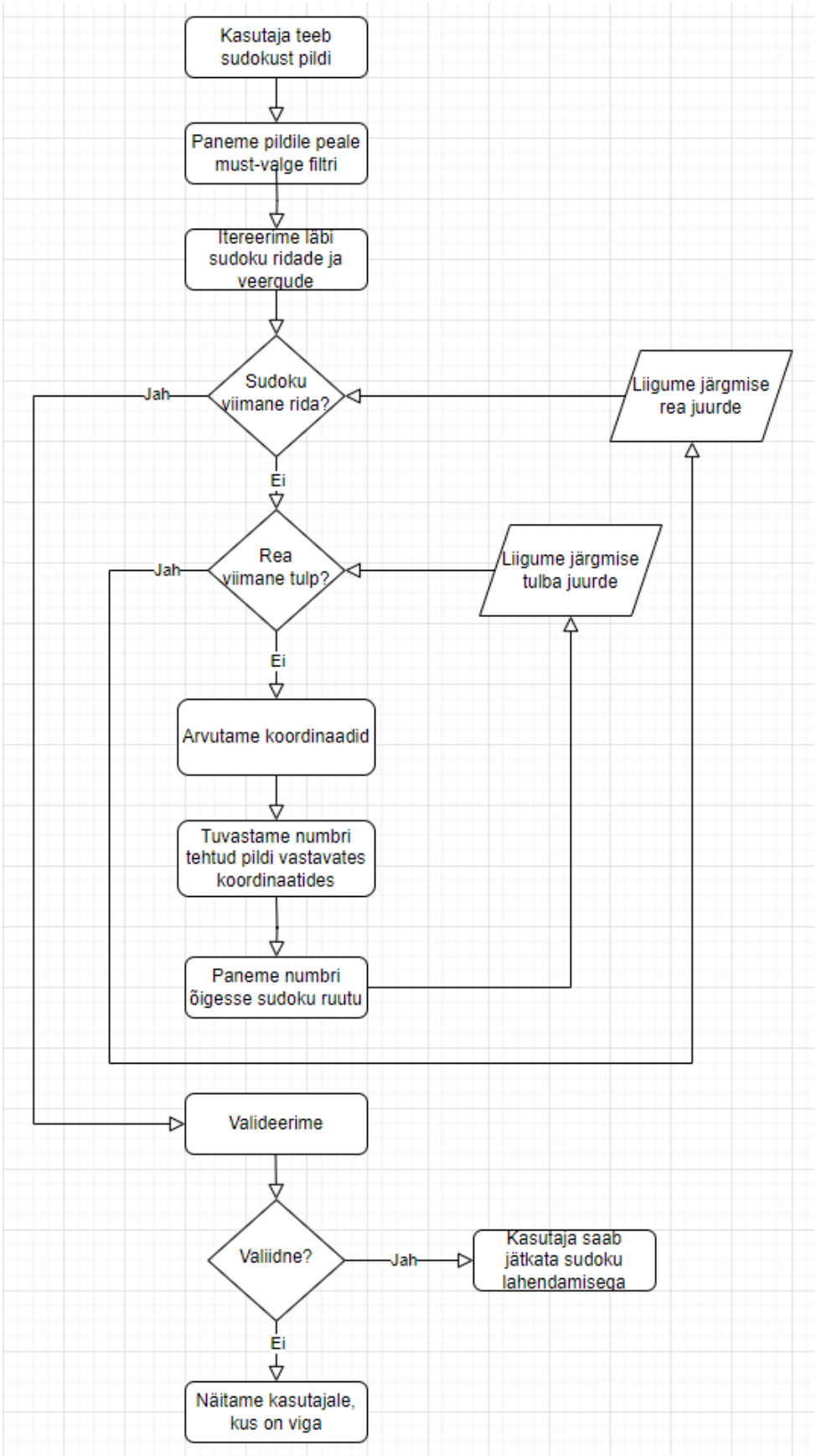
Automaatse täitmise jaoks otsisin sobiva optilise märgituvastuse teegi. Leidsin kaks populaarset teeki, milleks on OCRad.js ja Tesseract.js. Mõlemal on omad plussid ja miinused ning nende põhjal siis tuli valik teha. (Tabel 1) [12]

Tabel 1. OCRad.js ja Tesseract.js plussid ja miinused

OCRad.js		**Tesseract.js**	
+	-	+	-
Lihtne kasutada ja vähemahukas	Madal täpsus	Kõrge täpsus	Aeglane, mis ei võimalda reaalajas kiiret töötlust
Reaalajas tuvastus	Vähene dokumentatsioon	Tugev dokumentatsioon	Mahukam, mis võib kaasa tuua rakenduse pikema laadimisaja
Paljude keelte tugi	Sõltub OCRopus'est, mis on vähem populaarne, mistõttu väiksem tugi.	Eeltreenitud mudelid, mis suurendab täpsust	Vajab rohkem ressursse (protsessori võimsust ja mälu)

Kuna minu rakenduses on väga oluline, et numbrid tuvastataks võimalikult suure täpsusega, siis osutus valikuks Tesseract.js teek. Muud plussid ja miinused nagu näiteks reaalajas töötlus, teegi mahukus, keelte tugi ning ressursinõudlikkus, ei olnud niivõrd olulised, et need oleks valikut mõjutanud.

Järgnevalt voodiagrammilt on näha, kuidas käib sudoku numbrite tuvastamise protsess. Kui kasutaja teeb pildi ära, siis programm paneb pildile pealt must-valge filtri, et oleks kontrastsem ja tänu sellele lihtsam numbreid tuvastada. Seejärel käime läbi kõik 81 ruutu vastavalt koordinaatidele ja tuvastame neis numbri ning paneme õigesse ruutu. Kui kõik ruudud on läbi käidud, siis valideerime sudoku vastavalt sudoku reeglitele. Kui sudoku ei vasta reeglitele, siis kuvame kasutajale, kus viga on. (Joonis 19)



Joonis 19. Sudoku numbrite tuvastamise voodiagramm

5.3 Sudoku lahendaja programm

Sudoku lahendaja algoritm on võetud internetist ja vastavalt vajadustele muudetud.

Esmalt on sudoku lahendaja meetod, mis saab sisendiks sudoku laua ja käib seejärel rekursiivselt läbi sudoku ruudud ja otsib neisse sobivaid väärtusi. Väärtuse sobivust vaadatakse *getErrors()* meetodi abil, mis tagastab vead, kui number ei sobi ruutu. Kui tagastatud vigade arv on 0, siis number sobib ruutu ja liigutakse edasi. (Joonis 20)

```
export function solveSudoku(board) {
  let row = -1
  let col = -1
  let isEmpty = true
  for (let i = 0; i < 9; i++) {
    for (let j = 0; j < 9; j++) {
      if (board[i][j] === '') {
        row = i
        col = j
        isEmpty = false
        break
      }
    }
    if (!isEmpty) {
      break
    }
  }

  if (isEmpty) return true

  for (let num = 1; num <= 9; num++) {
    if (getErrors(board, row, col, num).length === 0) {
      board[row][col] = num
      if (solveSudoku(board)) {
        return true
      } else {
        board[row][col] = ''
      }
    }
  }
  return false
}
```

Joonis 20. *solveSudoku* meetod [13]

Järgmisena on originaalis *isSafe* meetod, mis sai ümber nimetatud *getErrors* meetodiks, sest meetod sai muudetud tagastama vigu, mis esinevad reas, veerus või sektsioonis. See

kõik oli vajalik selleks, et kasutada meetodit ka sudoku valideerimisel ja vigade kuvamiseks.

```
export function getErrors(board, row, col, num) {
  let errors = new Set()

  let sqrt = Math.floor(Math.sqrt(board.length))
  let boxRowStart = row - row % sqrt
  let boxColStart = col - col % sqrt

  for (let r = boxRowStart; r < boxRowStart + sqrt; r++) {
    for (let c = boxColStart; c < boxColStart + sqrt; c++) {
      if (r === row && c === col) continue
      if (board[r][c] === num) {
        errors.add({row: r, col: c, error: 'section'})
      }
    }
  }

  for (let d = 0; d < board.length; d++) {
    if (board[row][d] === num && d !== col) {
      errors.add({row: row, col: col, error: 'row'})
    }
    if (board[d][col] === num && d !== row) {
      errors.add({row: row, col: col, error: 'column'})
    }
  }

  return Array.from(errors)
}
```

Joonis 21. *getErrors* meetod [13]

5.4 Versioonihaldus, koodihoidla ja publitseerimine

Selleks, et erinevad katsetused ja töötav versioon omavahel sassi ei läheks ning töötavast variandist ei saaks mittetöötav, oli kasutuses versioonihaldustarkvara git. Sellega saab tehtud muudatused salvestada, millega luuakse uus versioon, ja kui edaspidiste katsetuste käigus läheb programm katki, siis on lihtne eelmine versioon taastada. [14]

Et maandada riski, kus programmi kood on ainult minu arvutis ja selle arvutiga midagi juhtub ning kood pole enam kättesaadav, kasutasin TalTech'i GitLab koodihoidlat[1]. Programmi kood on seal kättesaadav kõigile autentitud kasutajatele.

Rakendus on kättesaadav aadressilt https://sudoku-resolver.azurewebsites.net/.

[1] https://gitlab.cs.ttu.ee/kriman/sudoku-solver

6 Testimine

Sudoku lahendajat sai testitud kahel viisil. Ühiktestidega ja manuaalselt testides. Ühiktestimine hõlmas endas ainult lahendaja algoritmi testimist. Manuaalne testimine sisaldas endas ka numbrite tuvastamise testimist.

6.1 Manuaalne testimine

Selleks, et teada, kui hästi töötab sudoku numbrite tuvastus ja õigete ruutude täitmine, tuli läbi viia ka manuaalne testimine. Selle jaoks kasutasin sudoku raamatut, kust leiab erineva raskusastmega sudokusid. Skaneerisin neist 15 erinevat ja panin kirja, mitu ruutu õnnestus rakendusel korrektselt tuvastada kõigist võimalikest.

Kahjuks ühegi katsetuse tulemus ei olnud 100% korrektne. Kõige parem tulemus oli 79 õiget ruutu 81-st ehk 97,5%, mis on üsna hea tulemus. Keskmine tulemus oli 75 õiget ruutu ehk 92,6%.

Automaatse numbrituvastuse tulemusi mõjutavad mitmed faktorid. Näiteks valgus. Kui on liiga pime või valgus peegeldab vastu või tekib vari sudoku peale, siis täpsus langeb. Veel mõjutab minu rakenduses täpsust see, kui täpselt on joondatud sudoku keskmine ruut rakenduse kaamera vaate peal oleva sinise ruuduga.

6.2 Ühiktestimine

Selleks, et leitud algoritmi kiirust ja õigsust testida, genereerisin internetist leitud sudoku generaatoriga erinevate raskusastmetega sudokusid. [15]

Genereerisin iga raskusastme jaoks 50 erinevat sudokut koos lahendusega. Edasi hoidsin neid JSON failis objektide massiivis, kus igal objektil oli kaks välja. Väli *sudokuString* hoidis väärtusena lahendamata sudoku mõistatust ja *solutionString* selle sama sudoku lahendust. (Joonis 22)

```
[
  {
    "sudokuString":
    ".....59.2\n4.3....5.\n...93..8.\n..816....\n3.......4\n.4....891\n6..8..
    ...\n2.4613...\n.........",
    "solutionString":
    "861475932\n493286157\n725931486\n978164325\n312598764\n546327891\n657849
    213\n284613579\n139752648"
  }
]
```

Joonis 22. Sudokude hoidmise formaat JSON failis

Kuna generaatorist saadud formaat ei olnud sobiv minu sudoku lahendaja sisendiks, siis tegin abimeetodi, mis konverteeris failis oleva sõne minu sudoku lahendaja jaoks sobivasse formaati ja mida ma siis hiljem testides kasutasin. Selleks kõigepealt jagati pikk sõne ridadeks. Edasi itereeriti kõiki ridu ja määrati need väärtused sudokule. Lisaks kuna mina kasutasin juba oma koodis tühja ruudu tähistamiseks tühja sõne, aga generaator kasutas selleks punkti, siis tuli need punktid ära vahetada tühja sõnega. (Joonis 23)

```
export function convert(sudokuString) {
  let sudoku = []
  const rows = sudokuString.split('\n');
  for (let i = 0; i < rows.length; i++) {
    sudoku[i] = rows[i].split('')
  }
  return sudoku.map(row => row.map(element => element === '.' ? '' :
parseInt(element)))
}
```

Joonis 23. *convert()* meetod genereeritud sudoku sõne muutmiseks sobivale kujule

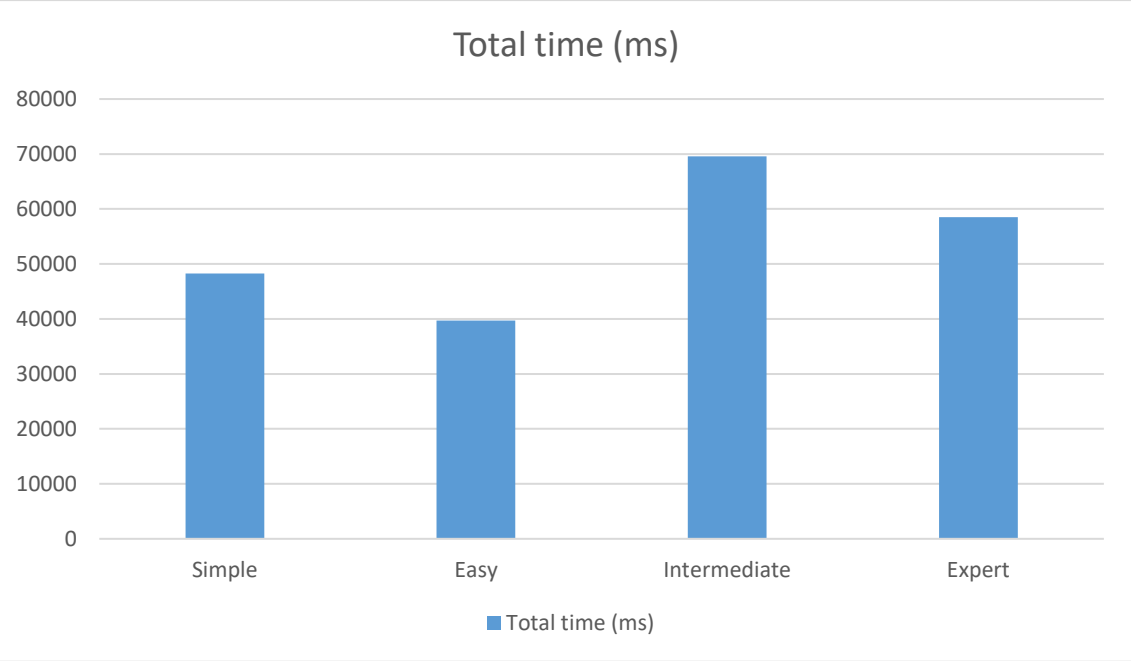
Testides importisin vajalikud JSON failid ja itereerisin läbi kõikide failis olevate objektide. Kõigepealt konverteerisin sudoku oma lahendajale õigesse formaati, seejärel käskisin lahendajal selle ära lahendada. Kui sudoku sai lahendatud, siis konverteerisin ka generaatori poolt saadud lahenduse samale kujule ja võrdlesin neid omavahel. Lisaks võtsin ma aega, palju kõikide failis olevate sudokude lahendamiseks kulus. (Joonis 24)

```
import {convert} from '../converter'
import solver from '../../src/solver/solver'
import simpleSudokus from './simpleSudokus.json'

describe('solve', function () {
  it('simple sudokus', function () {
    let start = Date.now()
    for (let sudoku of simpleSudokus) {
      const convertedSudoku = convert(sudoku.sudokuString)
      solver.solveSudoku(convertedSudoku)
      const convertedSolution = convert(sudoku.solutionString)
      expect(convertedSudoku).toEqual(convertedSolution)
    }
    const time = Date.now() - start
    console.log(`solved simple sudokus in ${time} milliseconds`)
  })
})
```

Joonis 24. Test lihtsate sudokude testimiseks

Testimise tulemused näitasid, et leitud algoritm töötab õigesti. Kõik etteantud sudokud said õige lahenduse. Algoritmi lahenduskiirus ei sõltunud nii väga sudoku keerukusest, mida oli ka arvata, kuna sudokut ei lahendata loogilist, vaid jõuga ehk tagurdusmeetodiga. 200 sudoku keskmine lahendamise aeg oli 1 sekund, mis minu arvates ei ole aeglane ning seetõttu ei näinud vajadust otsida muud algoritmi. (Joonis 25)



Joonis 25. Sudoku lahendaja kiirus vastavalt raskusastmele

7 Kokkuvõte

Töö eesmärk sai saavutatud ning selle tulemusena valmis sudoku lahendaja veebirakendus. Rakenduses on lihtne kasutajaliides, kus ei ole liiga palju funktsionaalsust, kuid parasjagu, et saada abi sudoku lahendamisel. Sudoku ruutude täitmise teeb mugavaks võimalus kasutada selleks telefoni kaamera abi.

Kaameraga tehtud pildilt tuvastab rakendus olemasolevad numbrid ära ning paneb õigesse kohta. Kuigi numbrituvastusega ei pruugi saavutada alati 100% täpsust, mida oli näha ka manuaalse testimise käigus, siis nende mõne numbri käsitsi parandamine on siiski mugavam kui kõikide teadaolevate numbrite käsitsi sisse trükkimine.

Projekti on võimalik edasi arendada, lisades uut funktsionaalsust. Näiteks kõikide võimalike kandidaatide kuvamise funktsionaalsus või kasutaja poolt valitud ruudu lahendamine. Kindlasti saab ka olemasolevat paremaks teha. Näiteks optimeerida lahendaja algoritmi. Lisaks võiks töötada numbrite tuvastamise kallal, et täpsuse protsent oleks veel suurem.

Töö käigus õppisin kasutama optilise märgituvastuse raamistikku. Samuti sain juurde teadmisi sudoku ajaloo ja erinevate lahendusmeetodite kohta.

Kasutatud kirjandus

[1] "History of Sudoku," [Võrgumaterjal]. Saadaval: https://www.sudokuVõrgumaterjal.io/tips/history-of-sudoku. [Külastatud 7.11.2023].

[2] F. D. Jorge, "Sudoku Rules," [Võrgumaterjal]. Saadaval: https://community.fico.com/s/sudoku-rules-episode-1. [Külastatud 22.11.2023].

[3] "Sudoku Solver," [Võrgumaterjal]. Saadaval: https://sudokuspoiler.com/. [Külastatud 3.12.2023].

[4] "Sudoku Solver," [Võrgumaterjal]. Saadaval: https://www.sudoku-solutions.com/. [Külastatud 22.11.2023].

[5] "Sudoku.com," Easybrain, [Võrgumaterjal]. Saadaval: https://sudoku.com/sudoku-solver. [Külastatud 3.12.2023].

[6] A. Stuart, "Sudoku Solver," [Võrgumaterjal]. Saadaval: https://www.sudokuwiki.org/sudoku.htm. [Külastatud 3.12.2023].

[7] A. Kumar, "AnySudokuSolver," [Võrgumaterjal]. Saadaval: https://anysudokusolver.com/. [Külastatud 3.12.2023].

[8] S. Alkoby, "Sudoku Solver," [Mobiilirakendus]. Saadaval: https://play.google.com/store/apps/details?id=com.alkobyshai.sudokusolver. [Külastatud 3.12.2023].

[9] "Sudoku Solver (Camera)," Robinson Industries, [Mobiilirakendus]. Saadaval: https://play.google.com/store/apps/details?id=com.RobinsonIndustries.SudokuSolver. [Külastatud 3.12.2023].

[10] "Solving Naked Subsets," [Võrgumaterjal]. Saadaval: https://www.sudoku-solutions.com/index.php?section=solvingNakedSubsets. [Külastatud 22.11.2023].

[11] "Solving Hidden Subsets," [Võrgumaterjal]. Saadaval: https://www.sudoku-solutions.com/index.php?section=solvingHiddenSubsets. [Külastatud 22.11.2023].

[12] B. Ruan, "OCR JavaScript – Extract text from image effortlessly," [Võrgumaterjal]. Saadaval: https://bensonruan.com/ocr-javascript-extract-text-from-image/#ocradjs_vs_tesseractjs. [Külastatud 17.11.2023].

[13] GeeksforGeeks, "Algorithm to Solve Sudoku," [Võrgumaterjal]. Saadaval: https://www.geeksforgeeks.org/sudoku-backtracking-7/#:~:text=Sudoku%20using%20Backtracking%3A. [Külastatud 22.11.2023].

[14] "git," [Võrgumaterjal]. Saadaval: https://git-scm.com/. [Külastatud 4.12.2023].

[15] S. Ostermiller, "QQWing Sudoku," [Võrgumaterjal]. Saadaval: https://qqwing.com/generate.html. [Külastatud 16.11.2023].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Kristjan Mänd

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Sudoku lahendaja", mille juhendaja on Peeter Ellervee
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

27.11.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.