UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
NOVI SAD
Departman za računarstvo i automatiku
Odsek za računarsku tehniku i računarske komunikacije

ISPITNI RAD

Kandidat: Lazar Nagulov
Broj indeksa: SV61/2022

Predmet: Objektno orijentisano programiranje 2
Tema rada: Sudoku

Mentor rada: dr Miodrag Đukić

Novi Sad, januar, 2024.

Sadržaj

Spisak slika

Spisak tabela

1 Uvod

1.1 Sudoku

Sudoku je logička zagonetka najčešće u obliku 9×9 tabele (matrice). U prazna polja tabele se upisuju cifre, tako da se svaka broj mora pojaviti tačno jednom u svakom redu, svakoj koloni i svakoj 3×3 podmatrici (bloku).

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

Slika 1: Primer sudoku zagonetke

Zagonetka ne mora da ima jedno rešenje, ali je standard da ga ima. Primer rešenje zagonetke sa slike 1:

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

Slika 2: Primer rešenja sudoku zagonetke

1.2 Zadatak

Realizovati konzolnu aplikaciju koja omogućava rešavanje i generisanje sudoku zagonetki. Korisnik unosi dateteke kroz argumente komandne linije. Primer

pokretanja programa:
`./sudoku` *input.txt* *output.txt*
Argumenti:

1. *input.txt* - Datoteka iz koje se čita zagonetka.
2. *output.txt* - Datoteka u koju se upisuje zagonetka.

Nakon pokretanja programa, korisniku se prikazuje početni meni koji nudi opcije:

- `Generate new Sudoku puzzle` - Generisanja nove zagonetke.
- `Load Sudoku puzzle from file` - Učitavanja zagonetke.
- `Exit` - Izlaska iz igre.

Nakon generisanja ili učitavanja zagonetke, korisnik može da:

- `Import solution` - Učita rešenje.
- `Solve` - Dopusti programu da reši zagonetku.
- `Exit` - Izlađe iz igre.

Na konzolnoj aplikaciji, posle rešavanja koje je generisao program ili učitao iz datoteke, prikazuju se statistički podaci igre, uključujući broj dobro postavljenih polja, broj grešaka i brojač odigranih igara i spisak svih pronađenih grešaka. Nakon završetka igre, korisnik ima opciju da odabere ponovno igranje, što pokreće novu iteraciju igre.

2 Analiza problema

3 Koncept rešenja

3.1 Algoritmi za rešavanje zagonetke

3.1.1 Obrnuta pretraga

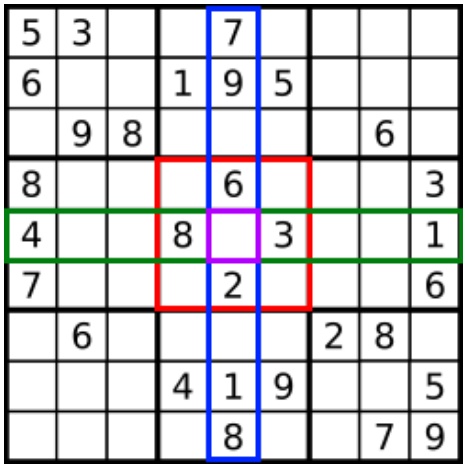
Najtrivijalniji algoritam za rešavanje sudoku zagonetke je obrnuta pretraga (eng. Backtracking). Ovo je algoritam grube sile (eng. Brute force) koji isprobava sve moguće kombinacije. Dakle, potrebno je da se prođe kroz svako polje u tabeli. Ukoliko je polje prazno, upisujemo cifru koja u trenutnoj tabeli ispunjava sva pravila. Nakon upisivanje cifre, rekurzivno pozivamo funkciju - pokušavamo da pronađe rešenje sa novom tabelom. Ukoliko rešenje nije pronađeno, vraćamo se nazad i upisujemo drugu cifru.

Vremenska složenost ovog algoritma je $\mathcal{O}(n^m)$, gde je n dimenzija tabele, a m broj polja koja trebaju da se popune. (U našem slučaju je složenost $\mathcal{O}(9^n)$). Minimalan broj polja koja moraju biti popunjena je 17, dakle, u najgorem slučaju se provera 9^{64} mogućnosti!

3.1.2 Optimizacija obrnute pretrage

Način na koji možemo optimizovati algoritam obrnute pretrage je da ubrzamo proveru da li se broj može postaviti na zadatoj poziciji. To ćemo postići tako što ćemo pamtiti koji broj se našao u redu, koloni i bloku. Za to ćemo koristiti `std::bitset<>` iz zaglavlja `bitset` gde, ako se za $i \in [0, 8]$ na i-toj poziciji nalazi 1, znači da se cifra $i + 1$ nalazi u redu, koloni ili bloku.

Pre samog ulaska u rekurzivnu funkciju obrnute pretrage, moramo proći kroz tabelu i zapisati svaki broj koji se nalazi u tabeli u nizove bitova. Potrebna su 3 niza bitova - za red, kolonu i blok. Za proveru da li je cifru moguće upisati




```
Polje (4,4)
RowSet:     010001101
ColSet:     111100011
BlockSet:   010100110
```

Slika 3: Primer konverzije polja u binaran broj

koristimo bitnu operaciju ili (eng. bitwise or):

`std::bitset<> contain = rows[row] | cols[col] | blocks[block];`
Novi niz bitova `contain` ima 0 na i-toj poziciji ako je moguće postaviti cifru $i+1$ na poziciju '(row, col)'. Na slici 3 vidimo da se u redu nalaze brojevi 1,3,4,8 (010001101), koloni 7,9,6,2,1,8 (111100011) i u bloku 6,3,2,8 (010100110). Iz ovoga zaključujemo da je u polje (4,4) moguće upisati broj 5 (111101111).

Vremenska složenost je i dalje $\mathcal{O}(n^m)$, gde je n dimenzija tabele, a m broj polja koja trebaju da se popune, stim da je su sve provere da li se broj može upisati u polje svedene na $\mathcal{O}(m)$ za razliku od prethodnog algoritma koji ima složenost $\mathcal{O}(nm)$.

	Maksimum	Medijana	Srednja vrednost
Gruba sila	6.20147s	0.001222s	0.04691s
Optimizacija	2.14567s	0.00072s	0.02301s

Tabela 1: Optimizacija obrnute pretrage - 1000 zagonetaka sa 25 popunjenih polja.

3.2 Algoritmi za generisanje zagonetke

3.2.1 Koraci u generisanju zagonetke

Generisanje zagonetke se može opisati u dva koraka:

1. Popuniti čitavu tabelu slučajnim vrednostima [1, `BOARD_SIZE`].

2. Nasumično obrisati brojeve iz tabele.

Primetimo da se blokovi na glavnoj dijagonali mogu zasebno popuniti jer ne utiču jedan na drugog. Dakle, prvi korak u generisanju se svodi na popunjavanja glavne dijagonale i rešavanja tako generisane zagonetke. Zatim brišemo brojeve iz tabele. Koliko brojava trebamo obrisati zavisi od težine zagonetke koju želimo generisati (Tabela 2).

Težina	Broj popunjenih polja
Lako	32 - 38
Srednje	26 - 31
Teško	22 - 25
Veoma teško	17 - 21

Tabela 2: Težina zagonetke u odnosu na broj popunjenih polja

3.2.2 Generisanje pseudoslučajnih brojeva

4 Opis rešenja

4.1 Modul glavnog programa

`int main(int argc, char** argv);`
Glavna funckija programa.

4.2 Modul Sudoku

4.2.1 Članovi

`int currentRound;`	Trenutna runda
`int correctCount;`	Broj tačnih cifara
`int wrongCount;`	Broj pogrešnih cifara
`Board board;`	Sudoku tabela

4.2.2 Enumeracije

`enum Difficulty;`
Određuje težinu Sudoku zagonetke na osnovu broja izbrisanih polja.
Vrednost: `EASY, MEDIUM, HARD, VERY_HARD`

4.2.3 Funkcija članica za pokretanje (Run)

`void Run();`
Pokreće aplikaciju i kreira početni meni za korisnika. Nudi korisniku mogućnost da generiše ili učita zagonetku iz datoteke. U njoj se nalazi glavna petlja igre.

4.2.4 Funcija članica za unos rešenja (SolvingOptions)

`void SolvingOptions();`
Nudi korisniku mogućnost da učita rešenje ili da dopusti programu da sam reši zagonetku. Poziva se iz `Run()` funkcije nakon generisanja ili učitavanja zagonetke.

4.2.5 Funkcija članica za rešavanje zagonetke (Solve)

`void Solve();`
Rešava zagonetku koristeći algoritam obrnute pretrage.

4.2.6 Funckija članica za proveru rešenja (Check solution)

`void CheckSolution(std::ifstream& in);`
Učitava zagonetku iz datoteke i proverava da li je validna. Ispisuje sve greške koje pronađe, njihov broj i broj tačno upisanih brojeva.
Parametri:

- (`std::ifstream&`) in - datoteka iz koje se čita rešenje.

4.2.7 Funkcija članica za gerenisanje zagonetke (Generate)

void Generate(Difficulty difficulty);
Generiše Sudoku zagonetku sa zadatom težinom.
Parametri:

- (`Sudoku::Difficulty`) difficulty - enumeracija koja označava težinu zagonetke.

4.3 Modul Tabela

4.3.1 Konstante

`int BOARD_SIZE = 9;`	Veličina tabele.
`int BLOCK_SIZE = 3;`	Veličina bloka.
`int EMPTY = 0;`	Oznaka za prazno polje.
`char EMPTY_CHAR = '_';`	Oznaka za prazno polje prilikom ispisa.

4.3.2 Članovi

`int board[BOARD_SIZE * BOARD_SIZE];`	Niz koji predstavlja tabelu.

4.3.3 Alijas BitArray

```
using BitArray =
std::array<std::bitset<Board::BOARD_SIZE>, Board::BOARD_SIZE>;
```

Niz od `Board::BOARD_SIZE` setova bitova dužine `Board::BOARD_SIZE`.

4.3.4 Funkcija članica za proveru poteza (IsPossibleMove)

`bool IsPossibleMove(int row, int col, int number) const;`
Proverava da li je moguće postaviti broj 'number' na poziciju '(row, col)'.
Parametri:

- (`int`) row - red u tabeli
- (`int`) col - kolona u tabeli
- (`int`) number - broj koji se pokušava staviti

Povratna vrednost:

- (`bool`) - true ako je moguće postaviti broj, false ako nije.

4.3.5 Funckija članica za proveru validnosti tabele (IsValid)

`bool IsValid() const;`
Proverava da li trenutna tabela ispunjava sva pravila sudoka.

4.3.6 Funkcija članica za pronalaženje grešaka (CountErrors)

`int CountErrors(const Board& original) const;`
Prebrojava i ispisuje sve greške u tabeli.
Povratna vrednost:

- (`int`) Broj grešaka u tabeli.

4.3.7 Funkcija članica za dobavljanje elementa tabele (At)

`int& At(int row, int col);`
Dobavlja element na poziciji '(row, col)'. Proverava granice.

4.3.8 Statična funkcija članica za dobavljanja blocka (GetBlock)

`static constexpr int GetBlock(int row, int col);`
Vraća blok u kome se nalazi polje '(row, col)'.
Parametri:

- (`int`) row - Red u tabeli.
- (`int`) col - Kolona u tabeli.

Povratna vrednost:

- (`int`) Broj bloka u kome se nalazi polje (row, col)

4.3.9 Funkcija članica za generisanje elemenata na glavnoj dijagonali (GenerateDiagonal)

`void GenerateDiagonal();`
Generiše nasumično elemente na glavnoj dijagonali.

4.3.10 Funcija članica za generisanje ostalih elemenata (GenerateOther)

`bool GenerateOther(int row, int col);`
Rekurzivno generiše nasumično elemente koji se ne nalaze na glavnoj dijagonali.
Parametri:

- (`int`) row - Početan red (uglavnom 0).
- (`int`) col - Početna kolona (uglavnom 0).

Povratna vrednost:

- (`bool`) Zaustavlja rekurzivno generisanje tabele.

4.3.11 Funkcija članica za popunjavanje blokova (FillBlock)

`void FillBlock(int row, int col);`
Rekurzivno generiše nasumično elemente u bloku.
Parametri:

- (`int`) row - početni red (gornje levo polje u bloku).
- (`int`) col - početna kolona (gornje levo polje u bloku).

4.3.12 Funkcija članica za brisanje elemenata iz tabele (RemoveNumber)

`void RemoveNumber(int count);`
Nasumično briše *count* elementa iz tabele.
Parametri:

- (`int`) count - broj elemenate koliko se briše iz tabele.

4.3.13 Funkcija članica koja implementira algoritam obrnute pretrage (Backtrack)

`bool Backtrack(BitArray& rSet, BitArray& cSet, BitArray& bSet);`
Funkcija implementira algoritam obrnute pretrage. Rekurzivno popunjava tabelu. Poziva se iz funkcije `Solve()` nakon generisanja pomoćnih nizova bitova.
Parametri:

- (`BitArray`) rSet - Pomoćni niz koji prati pojavljivanje brojeva u redovima.
- (`BitArray`) cSet - Pomoćni niz koji prati pojavljivanje brojeva u kolonama.
- (`BitArray`) bSet - Pomoćni niz koji prati pojavljivanje brojeva u blokovima.

Povratna vrednost:

- (`bool`) Zaustavlja rekurziju kada pronađe rešenje.

4.3.14 Funkcija članica za pronalaženje prvog praznog polja (FindEmpty)

`bool FindEmpty(int& row, int& col);`
Pronalazi prvo prazno polje od pozicije '(row, col)'. Prazno polje se nalazi u *row* i *col* promenljivi nakon završetka funkcije.
Parametri:

- (`int&`) row - referenca na početan red koji se pretražuje.
- (`int&`) col - referenca na početnu kolonu koja se pretražuje.

4.3.15 Funckija članica za brisanje tabele (Clear)

`void Clear();`
Postavlja sve elemente u tabeli na 0.

4.3.16 Operatori upisa i ispisa

```
std::istream& operator>>(std::istream& in, Board& b);
std::ostream& operator<<(std::ostream& out, const Board& b);
std::ofstream& operator<<(std::ofstream& out, const Board& b);
```

4.3.17 Operator dobavljanja

```
int& operator()(int row, int col);
```

5 Testiranje

6 Uočeni problemi i ograničenja

7 Zaključak