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 cifra 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.

2 Algoritmi za rešavanje zagonetke

2.1 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!

2.2 Optimizovana obrnuta pretraga

Način na koji možemo optimizovati algoritam obrnute pretrage je da ubrzamo proveru da li se cifra može postaviti na zadatoj poziciji. To ćemo postići tako što ćemo pamtiti koja cifra se našla 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 svaku cifru koja 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 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)'.

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 provera da li se cifra može upisati u polje svedena na $\mathcal{O}(1)$ za razliku od prethodnog algoritma koji ima složenost $\mathcal{O}(n)$.

2.3 Poređenje algoritama

3 Algoritmi za generisanje zagonetke

4 Opis rešenja

4.1 Modul glavnog programa

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

4.2 Modul Tabela

4.2.1 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.

4.2.2 Članovi

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

4.2.3 Funkcija članica za proveru poteza

`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.

4.2.4 Funckija članica za proveru validnosti tabele

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

4.2.5 Funkcija članica za pronalaženje grešaka

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

- (`int`) Broj grešaka u tabeli.

4.2.6 Funkcija članica za dobavljanje elementa tabele

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

4.2.7 Funkcija članica za generisanje elemenata na glavnoj dijagonali

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

4.2.8 Funcija članica za generisanje ostalih elemenata

`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`) Ignorisati, potrebno samo zbog rekurzivnih poziva.

4.2.9 Funkcija članica za popunjavanje blokova

`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).

4.2.10 Funkcija članica za brisanje elemenata iz tabele

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

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

4.2.11 Funkcija članica za pronalaženje prvog praznog polja

`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.

4.2.12 Funckija članica za brisanje tabele

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

4.2.13 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);
```

4.2.14 Operator dobavljanja

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

4.3 Modul Sudoku

4.3.1 Članovi

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

4.3.2 Enumeracije

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

4.3.3 Statična funkcija članica za pokretanje

`static void Run();`
Pokreće aplikaciju i kreira početni meni za korisnika.

4.3.4 Funkcija članica za rešavanje zagonetke

`void Solve();`
Rešava zagonetku.

4.3.5 Funkcija članica za gerenisanje zagonetke

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

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

5 Testiranje

6 Uočeni problemi i ograničenja

7 Zaključak