***Elektrotehnicki fakultet Univerziteta u Beogradu - Katedra za RTI***

# Domaći zadatak iz predmeta Ekspertni sistemi za junski rok školske 2014/15. godine

*Toplica Tanasković*

Crossword game

1. ***Prikaz implementiranog algoritma*** 
   1. **Definicija problema**

Metodom zadovoljenja ograničenja (eng. *constraint satisfaction problem* ) treba rešiti problem ukrštenih reči, tj. pronaći sva moguća rešenja pri čemu su početna tabla kao i skup reči ulazni parametri. CPA se definiše kao uređena trojka (X, D, C), gde je:

X = {X1, …, Xn} je skup promenljivih,

D = {D1, …, Dn} je skup domena,

C = {C1, …, Cn} je skup ogranicenja.

Definišemo skup D skupom svih reči, a skup X skupom mogucih slotova za reči na tabli. Svaki slot xi će imati svoj domen skup Di (Di ⊂ D). Elementi skupa Di su sve reči skupa D koje zadovoljavaju “Za sve reci r iz D rec r pripada skupu Di akko je duzina reci r jednaka duzini slota xi”. Skup C ce biti definisan u sledećem poglavlju (1.2).

* 1. **Prikaz strukture podataka**

Elemente domena reči predstavljamo klasom Word. Za svaku reč čuvamo podatak o njenom tekstu i o tome da li se trenutno koristi.

**public class Word {**

**private final String text;**

**private boolean used;**

**public Word(String text);**

**public String getText();**

**public boolean isUsed();**

**public void setUsed(boolean isUsed);**

**}**

Elemente skupa promenljivih predstavljamo klasom Space.

**public class Space {**

**private final Set<Word> wordSet = new HashSet<>();**

**private final Stack<Set<Word>> wordStackRemoved = new Stack<>();**

**private final Set<SlotsLimitation> slotsLimitationsSet = new HashSet<>();**

**private final Point start;**

**private final Point direction;**

**private final int length;**

**private boolean used;**

**public Space(Point start, Point direction, int length);**

**public Point getStart();**

**public Point getDirection();**

**public int getLength();**

**public Set<SlotsLimitation> getSlotsLimitationsSet();**

**public void addSlotLimitation(SlotsLimitation slotsLimitation);**

**public boolean isUsed();**

**public void setUsed(boolean used);**

**public void addWords(Set<Word> words);**

**public Set<Word> getWordSet();**

**public int removeWords(char ch, int index);**

**public void addWords();**

**}**

Za svaki slot čuvamo početak slota ((x, y) koordinate), smer slota može biti (0,1) ili (1,0) u zavisnosti od toga da li se kreće po x ili po y osi, duzina slota i podatak o tome da li je trenutno zauzet. U promenjivoj wordSetčuvaju se reči koje trenutno mogu da se stave u slot. Promenljiva wordStackRemovedse koristi za čuvanje skupa izbačenih reči pre ulaska u rekurziju da pi se posle rekurzije *pop-ovala* nazad u wordSet. Takodje se cuva skup ograničenja slota.

Ogranicenja predstavljamo klasom SlotLimitation***.***

**public class SlotsLimitation** {

**private final Space** **slot**;

**private final int slotIndex**;

**private final int index**;

**public SlotsLimitation**(**Space** **slot**, **int slotIndex**, **int index**);

**public Space getSlot();**

**public int getIndex();**

**public int getSlotIndex();**

}

Za svaki slot čuvamo skup ogranicenja, a svako ogranicenje ima 3 podatka. Slot, sa kojim se slot koji čuva ovo ogranicenje seče tj. sadrže zajedničko polje, kao i indekse tog polja u tim slotovima.

* 1. **Prikaz algoritma**

Rekurzivna metoda fillCrosswordModeTwo prima dva parametra, slotUsed i slotIndex. Parametar slotUsed meri dubinu rekurzije, kada njena vrednost bude jednaka sa ukupnim brojem slotova to znaci da smo nasli resenje.

**private void fillCrosswordModeTwo(int slotUsed, int slotIndex) {**

**if (slotUsed == slots.length) {**

**results.add(copyBoard(crossword));**

**return;**

**}**

**slots[slotIndex].setUsed(true);**

**for (Word word : slots[slotIndex].getWordSet()) {**

**if (!word.isUsed()) {**

**ArrayList<Space> slotsIndex = new ArrayList<>();**

**int step = 0;**

**for (SlotsLimitation s : slots[slotIndex].getSlotsLimitationsSet()) {**

**if (!s.getSlot().isUsed()) {**

**int wordsNumber = s.getSlot().removeWords(word.getText().charAt(s.getIndex()), s.getSlotIndex());**

**slotsIndex.add(s.getSlot());**

**if (wordsNumber == 0) {**

**step = -1;**

**break;**

**}**

**}**

**}**

**if (step == 0) {**

**putWordInSlot(word, slots[slotIndex]);**

**int space = 0;**

**for (int i = 0; i < slots.length; i++) {**

**if (!slots[i].isUsed()) {**

**space = i;**

**if (slots[i].getWordSet().size() == 1) {**

**break;**

**}**

**}**

**}**

**fillCrosswordModeTwo(slotUsed + 1, space);**

**removeWordFromSlot(word, slots[slotIndex]);**

**}**

**for (Space s : slotsIndex) {**

**s.addWords();**

**}**

**}**

**}**

**slots[slotIndex].setUsed(false);**

**}**

Parametar slotIndex oznacava index slota na koji se trenutno ubacuju reči. Pri ulasku u metodu slot postaje zauzet postavljanjem odgovarajućeg polja klase Space na true. Posle toga se prolazi kroz skup svih dozvoljenih reči i za svaku proverimo da li je već upotrebljena u ukrtenici ako jeste uzimamo sledeću reč. Kada smo odabrali reč koja još nije upotrebljena ažuriramo ostale skupove po ograničenjima. Ako je sve prošlo kako treba tj. ako ne postoji skup dozvoljenih reči nekog slota koji je prazan onda stavljamo tu reč na tablu i pronalazimo novi slot, ako postoji prazan skup brišemo sve promene koje smo napravili kad smo ažurirali ograničenja i uzimamo novu reč.

Kada smo stavili novu reč na tablu sledeći slot biramo tako što proverimo da li postoji neki slot u čijem skupu reči je ostala samo jedna reč ako postoji ulazimo dublje u rekurziju sa tim slotom, ako ne ulazimo sa bilo kojim drugim slobodnim slotom. Kad izađemo iz rekurzije vratimo sve promene koje smo napravili kad smo ažurirali ograničenja i trazimo sledeću reč. Ako više nema reči oslobađamo slot i izlazimo iz rekurzije jedan stepen ka gore.

**private void putWordInSlot(Word word, Space slot) {**

**Point position = new Point(slot.getStart());**

**for (int i = 0; i < slot.getLength(); i++) {**

**crossword[position.x][position.y] = word.getText().charAt(i);**

**letterUsage[position.x][position.y]++;**

**position.x += slot.getDirection().x;**

**position.y += slot.getDirection().y;**

**}**

**word.setUsed(true);**

**}**

**private void removeWordFromSlot(Word word, Space slot) {**

**Point position = new Point(slot.getStart());**

**for (int i = 0; i < slot.getLength(); i++) {**

**letterUsage[position.x][position.y]--;**

**if (letterUsage[position.x][position.y] == 0) {**

**crossword[position.x][position.y] = CrosswordApp.BLANK;**

**}**

**position.x += slot.getDirection().x;**

**position.y += slot.getDirection().y;**

**}**

**word.setUsed(false);**

**}**

Metode za dodavanje i izbacivanje reči iz slota. Promenljiva letterUsage pamti koliko puta je neko slovo upisano u određeno polje i tek kad padne na nulu slovo možemo obrisati.

1. ***Opisi ostalih klasa***

Klasa **StartPanel:**

**public class StartPanel extends JPanel {**

**private JTextField xField;**

**private JTextField yField;**

**private StartPanel();**

**public static void main(String[] args) ;**

**private void initUI();}**

**private int getXFiled() throws NumberFormatException;**

**private int getYFiled() throws NumberFormatException;**

**}**

Ova klasa je jedna od dve početne klase (sadrži main() metod) u ovoj aplikaciji. Pokretanjem nje otvara vam se dialogi traze da unesete dimenzije table. Posle toga se pogreće glavni frame aplikacije (klasa CrosswordApp).

**public class CrosswordApp extends JFrame {**

**public static final char *FILLED* = '#';**

**public static final char *BLANK* = ' ';**

**private JList list;**

**private DefaultListModel model;**

**private int xField;**

**private int yField;**

**private ArrayList<Set<Word>> words;**

**private char[][] crossword;**

**private Space[] slots;**

**private int mode;**

**private boolean go;**

**private JTextArea description;**

**private JButton btnStart;**

**private JLabel lblTime;**

**private CrosswordPanel crosswordPanel;**

**private ArrayList<char[][]> results;**

**private JPanel south;**

**public CrosswordApp(int xField, int yField);**

**public static void main(String args[]);**

***// PUBLIC***

**public void descriptionAdd(String s);**

**public void setTime(long time);**

**public boolean isGo();**

**public void setGo(boolean go);**

**public void setBoard(char[][] board);**

***//PRIVATE***

**private void initUI();**

**private void addWordsToSlot();**

**private int checkMode();**

**private void addPanelMode1();**

**private void addPanelMode2();**

**private int initWordsSets()**

**private void addWordsDialog();**

**private void makeCrosswordSlot();**

**private void makeSlotLimitation();**

**}**

Ova klasa takođe ima main() metodu, pokretanjem pokreće se glavni frejm aplikacije (klasa CrosswordApp) sa dimenzijama 5x5.

**public void descriptionAdd(String s);** - Dodaje string s na kraj teksa promenljive description

**public void setTime(long time);** - U labelu lblTime upisuje vreme izvršavanja algoritma

**public void setBoard(char[][] board);** - upisuje matricu karaktera na tablu.

**private void addWordsToSlot();** - Dodaje reci u slot koje su njegove dužine.

**private int checkMode();** - Dialog na kom korisnik bira način izvršavanja algoritma.

**private void addWordsDialog();** - Dialog za unos reči

**private void makeCrosswordSlot();** - Na osnovu table pravi niz slotova

**private void makeSlotLimitation();** - Dodaje ograničenja u slotove

Klasa koja predstavlja tablu ukrštenice:

**public class CrosswordPanel extends JPanel {**

**private int x;**

**private int y;**

**private JButton[][] btns;**

**private JPanel[][] box;**

**private JLabel[][] top;**

**private JLabel[][] bottom;**

**private JLabel[][] center;**

**public CrosswordPanel(int x, int y);**

**private void initGui();**

**public void disableAll();**

**public char[][] initCrossword();**

**public void fillBoard(char[][] crossword);**

**public void setTopLabel(int index, int i, int j);**

**public void setBottomLabel(int index, int i, int j);**

**}**

**public void disableAll();** - Briše dugmad i dodaje labele.

**public char[][] initCrossword();** - Napravi matricu karaktera na osnovu crnih polja.

**public void fillBoard(char[][] crossword);** - Upisuje karaktere u tablu na osnovu matrice karaktera.

Klasa CrosswordGridLayoutje custom layout koji forsira svoje elemente da budu kvadratnog oblika u ovom slucaju board dugmad i labele.

1. ***Moguća unapređenja aplikacije***

Trebalo bi zameniti sve anonimne klase lambda izrazima zbog popoljšanja čitljivosti. Takođe, koristiti forEach() metod, kao i stream() metod koji su sastavni deo java kolekcija a takođe poboljšavaju čitljivost. Na funkcionalnosti aplikacije moze još puno da se radi, npr. omogućiti da se mogu uneti minimalna i maksimalna duzina reči i na osnovu toga napraviti slotove na tabli, npr. omogućiti brisanje slotova koje ne želimo, npr. omogućiti naknadno unošenje reči. Većina Set-ova, ArrayList-i i drugih kolekcija koje se koriste u implementaciji algoritma su stalne velicine, što omogućava laku konverziju u java Array. Sto bi poboljšalo performance.