**Rīgas Tehniskas Universitātes**

**Informācijas tehnoloģiju katedras**

Priekšmēta:

**Mākslīgā intelekta pamati**

Praktiskais darbs Nr.1

**“**Divpersonu spēles ar pilnu informāciju izstrāde**”**

**GAME DOWNLOAD LINKL:  
https://github.com/ArtjomsBreitmanis/131RMC272-AI-GAME.git**

Students: Artjoms Breitmanis

131RMC272

2.kurss, 3.g

Pasniedzējs: Doktors Alla Anohina-Naumeca

Rīga 2023g.

Saturs:

Darba Saturs.

[Ievads: 3](#_Toc132219549)

[Spēles apraksts: 4](#_Toc132219550)

[**Lietošanas instrukcija un izskats:** 5](#_Toc132219551)

[**Spēles algoritma apraksts:** 6](#_Toc132219552)

[**Python kods:** 7](#_Toc132219553)

# Ievads:

Šajā praktiskajā darbā students izveido vienkāršu spēli diviem spēlētājiem, no kuriem viens ir dators. Izstrādājot šo spēli, es izmantoju Minimax algoritmu, kas ļāva datorspēlētājam aprēķināt viņam izdevīgu gājienu un veikt to.

# Spēles apraksts:

Šī spēle ir elementāra matemātiska spēle. Lietotājs var izvēlēties, vai iet pirmais, viņš vai dators. Atkarībā no izvēles spēle gaida, kad lietotājs izdarīs gājienu, vai arī pati veic gājienu, izvērtējot labāko gājienu un to izdarot. Spēles laikā lietotājs var redzēt šādu informāciju:

* "spēles numurs", kas dinamiski mainās spēles gaitā.
* "Gājieni", atkarībā no tā, cik gājienu ir izdarīts, mainās arī to skaits.
* Norādes, brīdinājumi un paziņojumi par uzvarētāju.

Lietotāja rīcībā ir spēles vadības pogas:

* "Ģenerēt nejaušu skaitli".
* "Veikt gājienu, apstiprinot savu izvēli ar izvēlēto dalītāju 2 vai 5".
* "Izvēlēties sākuma spēlētāja pogu".
* "Poga, ar kuru datoram signalizē, ka spēlētājs ir gatavs un var sākt.

Spēle ir veidota kā grafisks logs ar tekstu, pogām un nolaižamo izvēlni ar izvēles iespējām.

# **Lietošanas instrukcija un izskats:**

Šie norādījumi lietotnē parādīsies vairāk nekā vienu reizi. Lai sāktu spēli, ir nepieciešams veikt dažas vienkāršas darbības, ja spēlētājs nespēj veikt vienkāršas darbības, programma sniedz ziņojumus, kas ir paredzēti, lai palīdzētu spēlētājam veikt vienkāršas darbības.

1. Pirms spēles sākšanas izvēlieties spēlētāju, kurš sāks gājienu, noklikšķiniet uz nolaižamajā izvēlnē un izvēlieties vienu no divām iespējām "User" vai "Computer" (1)

2. Pēc tam noklikšķiniet uz pogas, lai ģenerētu nejaušo skaitli (2), kas tiks izspēlēts. Jūs varat spiest pogu, līdz skaitlis atbilst jūsu vēlmēm.

3. Ja lietotājs izdara pirmo gājienu, tad ir jānospiež poga "GO!"(3), uzrakstot loga(5) izvelēto ciparu 2 vai 5. Gadījumā, ja spēlētājs apžēlojas par datoru un dod tam tiesības veikt pirmo gājienu, tad ir jāļauj datoram to zināt, nospiežot pogu "START"(4).

4. Bezizejas situācijas gadījumā (ja pirmā gājiena rezultātā ģenerētais skaitlis novedīs pie uzvaras/zaudējuma) ir jānospiež no bezizejas izejas poga "Random Number"(2).

Diagram

Description automatically generated with medium confidence

# **Spēles algoritma apraksts:**

Šī spēle neuzglabā gājienu koku atmiņā, jo paradoksālā kārtā spēle katru gājienu ģenerē jaunu koku un apstrādā to, lai izvēlētos visizdevīgāko gājienu, izmantojot Minimax apstaigāšanas algoritmu.

# **Python kods:**

import tkinter as tk

from tkinter import messagebox

import random

**class** AIGAME:

**def** \_\_init\_\_(self):

        self.root = tk.Tk()

        self.root.title("2 vai 5 dalijuma spele")

        self.existing\_numChar = 0

        self.move\_P = 1

        self.number\_label = tk.Label(self.root, text="Current Number: ")

        self.number\_label.pack()

        self.showNumVal = tk.Label(self.root, text="0")

        self.showNumVal.pack()

        self.player\_label = tk.Label(self.root, text="User's Turn: ")

        self.score\_P = tk.StringVar(value="1")

        self.player\_value\_label = tk.Label(self.root, textvariable=self.score\_P)

        self.player\_label.pack()

        self.player\_value\_label.pack()

        self.playthisnnumber = tk.StringVar()

        self.playthisnnumber.set("Computer")

        self.starting\_player\_menu = tk.OptionMenu(self.root, self.playthisnnumber, "Player", "Computer")

        self.starting\_player\_menu.pack()

        self.user\_input\_label = tk.Label(self.root, text="Enter number 2 or 5: ")

        self.user\_input\_label.pack()

        self.user\_input\_entry = tk.Entry(self.root)

        self.user\_input\_entry.pack()

        self.submit\_button = tk.Button(self.root, text="GO!", command=self.acceptstake)

        self.submit\_button.pack()

        self.generate\_button = tk.Button(self.root, text="Random number", command=self.randgenbutton)

        self.generate\_button.pack()

        self.begindiv\_button = tk.Button(self.root, text="Start", command=self.begindiv)

        self.begindiv\_button.pack()

        self.starting\_player\_label = tk.Label(self.root, text="Starting Player: ")

        self.starting\_player\_label.pack()

        self.info\_button = tk.Button(text="Game Instruction!!!!", height=5,width=20, command=self.FAQ\_agree)

        self.info\_button.pack()

        self.number\_label = tk.Label(self.root, text="Instructions: ")

        self.number\_label.pack()

        self.number\_label = tk.Label(self.root, text="1. To start game choose the starting player. ")

        self.number\_label.pack()

        self.number\_label = tk.Label(self.root, text="2. Choose starting number, enter number by hand in window, or click 'Random number'")

        self.number\_label.pack()

        self.number\_label = tk.Label(self.root, text="3. If User is chosen to start then press 'GO!', if Computer - press 'Start' ")

        self.number\_label.pack()

        self.number\_label = tk.Label(self.root, text="4. If the generated number is not dividable with 2 or 5, you can proceed and and game instantly, or regenerate pressing 'Random Number'")

        self.number\_label.pack()

        """

        self.number\_label = tk.Label(self.root, text="5. Due to limit if tree depth numbers over 6000 will cause the loop error (regenerate)")

        self.number\_label.pack()"""

**def** FAQ\_agree(self):

        win=tk.Toplevel()

        win.title('Terms')

        message="To start game You should accept terms of agreement, press 'Accept', or close the game.\n\n\n\n\nJust kidding, no one cares!"

        tk.Label(win, text=message).pack()

        tk.Button(win,text='Accept',height=5,width=10, command=self.FAQ).pack()

        messagebox.showinfo(title="Lecince",message="To start game You should accept terms of agreement, press 'Accept', or close the game.", command=self.FAQ)

**def** FAQ(self):

         messagebox.showinfo(title="Game Instructions", message="1. To start game, firsh select who starts the game 'User' or 'Computer'"+"\n"+"2. Click 'Random Number' button to generate number in box"+"\n"+"3. If the User starts press'GO!' button, if Computer starts - 'Start'"+"\n"+"4. If the generated number is not dividable with 2 or 5, you can proceed and and game instantly, or regenerate pressing 'Random Number'")

**def** randgenbutton(self, doComputerMove : bool = True):

        self.existing\_numChar = random.randint(1000, 1995) \*5

        self.showNumVal.config(text=str(self.existing\_numChar))

        if(doComputerMove):

            self.makeAImove()

**def** restart(self):

        self.randgenbutton(False)

        self.tree = move\_koks(self.existing\_numChar)

        self.move\_P = 1

        self.score\_P.set(self.move\_P)

        self.user\_input\_entry.delete(0, tk.END)

**def** begindiv(self):

        if(self.existing\_numChar > 0):

            self.tree = move\_koks(self.existing\_numChar)

        else:

            messagebox.showinfo(title="Attention",message="Not valid move, You should enter YOUR number, Or press Random to generate number.")

        self.move\_P = 1

        self.score\_P.set(self.move\_P)

        self.user\_input\_entry.delete(0, tk.END)

        self.makeAImove()

**def** acceptstake(self):

        num\_gen\_in = self.user\_input\_entry.get()

        if num\_gen\_in not in ["2", "5"]:

            messagebox.showinfo(title="Attention",message="Not valid move, only 2 or 5 are accepted")

            return

        num\_gen\_in = int(num\_gen\_in)

        if self.existing\_numChar % num\_gen\_in != 0:

            messagebox.showerror(title="User-looser", message= "Cannot be divided, You loose!")

            self.restart()

            return

        self.existing\_numChar //= num\_gen\_in

        self.showNumVal.config(text=str(self.existing\_numChar))

        self.move\_P = self.move\_P + 1

        self.score\_P.set(self.move\_P)

        if self.existing\_numChar == 1:

            messagebox.showinfo("Congratulations! Player {} wins!".format(self.move\_P))

            self.restart()

        else:

            self.makeAImove()

**def** makeAImove(self):

        if (self.move\_P % 2 == 0 and self.playthisnnumber.get() == "Player")\

        or (self.move\_P % 2 != 0 and self.playthisnnumber.get() == "Computer"):

            optimal\_divisor = self.algominimax()

            if(optimal\_divisor == None):

                messagebox.showwarning(title="Victory",message="You beat primitive AI algorithm, \nNumber is not divisible nor by 5 nor by 2. \nYou win!")

                return self.restart()

            self.existing\_numChar //= optimal\_divisor

            self.showNumVal.config(text=str(self.existing\_numChar))

            self.move\_P = 3 - self.move\_P

            self.score\_P.set(self.move\_P)

            self.player\_value\_label.config(text=str(self.move\_P))

            if self.existing\_numChar == 1:

                messagebox.showinfo("Player {} wins!".format(self.move\_P))

                self.restart()

**def** algominimax(self):

**def** eval\_big(number):

            if number == 1:

                return 0

            if number % 2 == 0:

                return 1 + eval\_big(number // 2)

            else:

                if number < 5:

                    return 2

                else:

                    return 1 + eval\_big(number // 5)

**def** eval\_small(number):

            if number == 1:

                return 0

            if number %2 == 0:

                return 1 + eval\_small(number // 2)

            else:

                if number < 5:

                    return 2

                else:

                    return 1 + eval\_small(number // 5)

        divisors = [2, 5]

        bestnum = None

        numhigh = float('-inf')

        for divisor in divisors:

            if self.existing\_numChar % divisor == 0:

                eval = eval\_small(self.existing\_numChar // divisor)

                if eval > numhigh:

                    numhigh = eval

                    bestnum = divisor

        return bestnum

**def** move\_koks(number):

    root = Node(number)

    queue = [root]

    while queue:

        node = queue.pop(0)

        if node.value > 1:

            for move in [2, 5]:

                if canItBeDivided(node.value // move):

                    child = Node(node.value // move)

                    node.children.append(child)

                    queue.append(child)

    return root

**def** canItBeDivided(number):

    return number % 2 == 0 or number % 5 == 0

**class** Node:

**def** \_\_init\_\_(self, value):

        self.value = value

        self.children = []

**def** moveLeaf(self):

        return len(self.children) == 0

if \_\_name\_\_ == "\_\_main\_\_":

    AI\_GAME = AIGAME()

    AI\_GAME.root.mainloop()