Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Дискретна математика

Лабораторна робота №1

«Бінарні відношення та їх основні властивості, операції над відношеннями»

Виконав: студент групи ІО-64

Андрійчук Д. А.

Залікова книжка № 6401

Перевірив Новотарський М. А.

Київ

2017 р.

**Тема:** «Бінарні відношення та їх основні властивості, операції над відношеннями».

**Мета:** вивчити основні властивості бінарних відношень та оволодіти операціями над бінарними відношеннями.

**Загальне завдання:**

1. Написати в окремому модулі функцію для формування несуперечливих бінарних відношень.

2. Написати в окремому модулі функції виконання логічних операцій над бінарними відношеннями.

3. Пояснити правило формування несуперечливих відношень відповідно до Вашого варіанту.

3. Варіант виразу відповідно до індивідального завдання.

Z=(1+64%60)%30+1 = 6

4. Короткі теоретичні відомості по темі, які відображають правила виконання логічних операцій, застосованих при виконанні лабораторної роботи.

**Бінарним** (або *двомісним*) відношенням *R* називають підмножину

впорядкованих пар, тобто множину, кожен елемент якої є впорядкованою парою.

Якщо *R* є деяким відношенням, це записують як *x*, *y R* або *xRy* .

Один з типів відношень − це множина всіх таких пар *x*, *y* , що *x* є

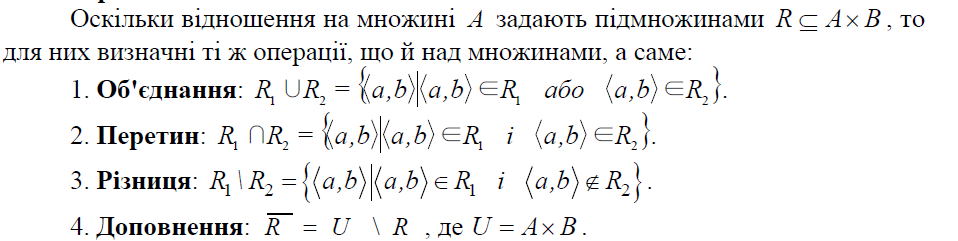
елементом деякої фіксованої множини *X* , а *y* − елементом деякої фіксованої

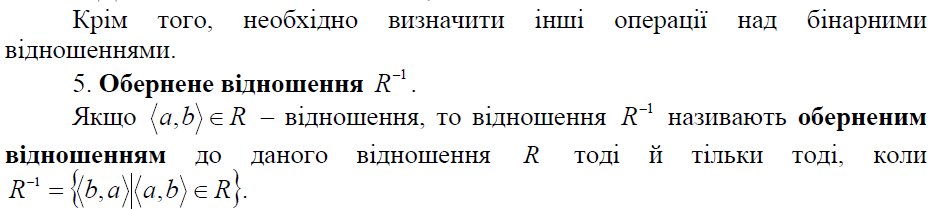
множини *Y* . Таке відношення називають *прямим* або *декартовим добутком*.

**Бінарним** відношенням *R* називають підмножину пар *x*, *y* *R* прямого

добутку *X Y* , тобто *R*  *X Y* .

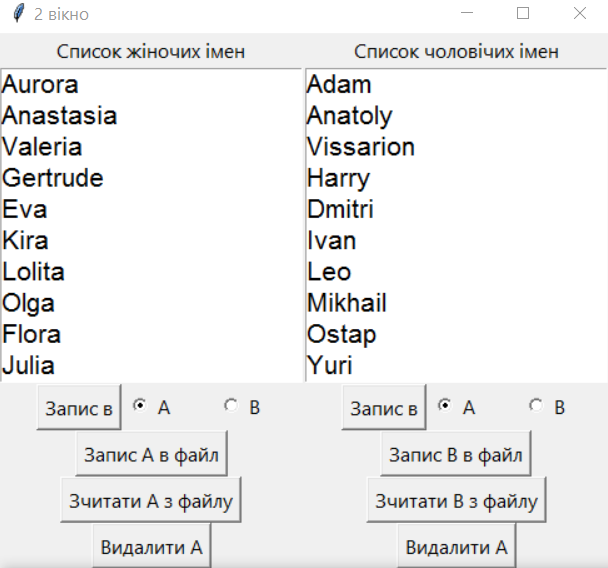
**Операції над відношеннями**

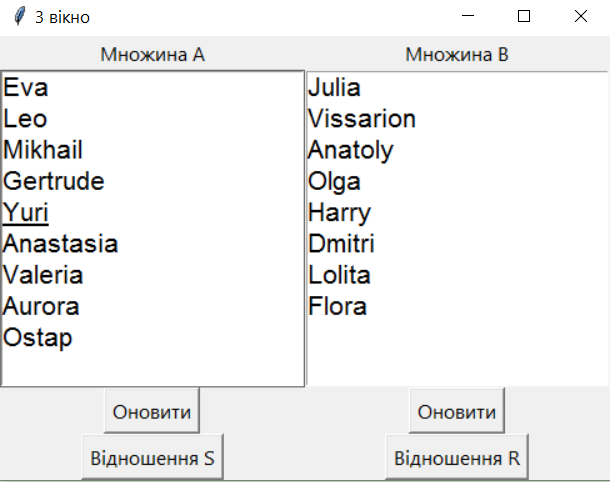




5.Роздруківка того фрагменту тексту програми, який написаний індивідуально.

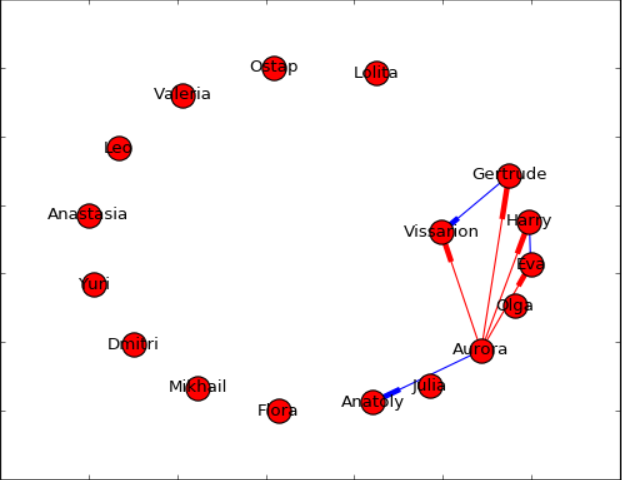
**from** tkinter **import** \*  
**from** random **import** randint, sample  
**import** networkx **as** nx  
**import** pylab **as** plt  
  
**class** Main:  
 A = set()  
 B = set()  
 R = list()  
 S = list()  
  
 **def** \_\_init\_\_(self):  
 menubar = Menu(root)  
 filemenu = Menu(menubar, tearoff=0)  
 filemenu.add\_command(label=**"2 вікно"**, command=self.window2)  
 filemenu.add\_command(label=**"3 вікно"**, command=self.window3)  
 filemenu.add\_command(label=**"4 вікно"**, command=self.window4)  
 menubar.add\_cascade(label=**"Вікнa"**, menu=filemenu)  
  
 l1 = Label(root, text=**"Моя група: ІО -"**, font=(**"Arial"**, 20))  
 l1.grid(row=0, sticky=W)  
 l2 = Label(root, text=**"Мій номер у групі:"**, font=(**"Arial"**, 20))  
 l2.grid(row=1, sticky=W)  
 l3 = Label(root, text=**"Мій варіант:"**, font=(**"Arial"**, 20))  
 l3.grid(row=2, sticky=W)  
  
 self.e1 = Entry(root, width=10, font=(**"Arial"**, 20))  
 self.e2 = Entry(root, width=10, font=(**"Arial"**, 20))  
 self.t1 = Text(root, width=5, height=1, wrap=WORD, font=(**"Arial"**, 20))  
 b1 = Button(root, width=7, height=1, command=self.calc\_variant, text=**"Варіант"**, font=(**"Arial"**, 10))  
 self.e1.grid(row=0, column=1)  
 self.e2.grid(row=1, column=1)  
 b1.grid(row=2, column=1, sticky=E)  
 self.t1.grid(row=2, column=1, sticky=W)  
  
 root.config(menu=menubar)  
  
 **def** window2(self):  
 self.win2 = Toplevel(root)  
 self.win2.title(**"2 вікно"**)  
  
 **def** set\_w():  
 **if** womens\_set.get() == **"A"**:  
 **for** i **in** list\_of\_women.curselection():  
 self.A.add(self.list\_of\_women\_text[i])  
 **elif** womens\_set.get() == **"B"**:  
 **for** i **in** list\_of\_women.curselection():  
 self.B.add(self.list\_of\_women\_text[i])  
  
 **def** set\_m():  
 **if** mens\_set.get() == **"A"**:  
 **for** i **in** list\_of\_men.curselection():  
 self.A.add(self.list\_of\_men\_text[i])  
  
 **elif** mens\_set.get() == **"B"**:  
 **for** i **in** list\_of\_men.curselection():  
 self.B.add(self.list\_of\_men\_text[i])  
  
 **def** save\_a():  
 **with** open(**r"F:\Dima\дискретна математика\Lab\_2\saved\_a.txt"**, **"w"**) **as** w:  
 **for** i **in** self.A:  
 w.write(str(i)+**"\n"**)  
 print(self.A)  
  
 **def** save\_b():  
 **with** open(**r"F:\Dima\дискретна математика\Lab\_2\saved\_b.txt"**, **"w"**) **as** w:  
 **for** i **in** self.B:  
 w.write(str(i)+**"\n"**)  
 print(self.B)  
  
 **def** read\_a():  
 **with** open(**r"F:\Dima\дискретна математика\Lab\_2\saved\_a.txt"**, **"r"**) **as** r:  
 **for** line **in** r:  
 self.A.add(line[:-1])  
  
 **def** read\_b():  
 **with** open(**r"F:\Dima\дискретна математика\Lab\_2\saved\_b.txt"**, **"r"**) **as** r:  
 **for** line **in** r:  
 self.B.add(line[:-1])  
  
 **def** clear\_a():  
 self.A.clear()  
  
 **def** clear\_b():  
 self.B.clear()  
  
 **with** open(**r"F:\Dima\дискретна математика\Lab\_2\saved\_a.txt"**, **"w"**) **as** w:  
 **with** open(**r"F:\Dima\дискретна математика\Lab\_2\saved\_b.txt"**, **"w"**) **as** w:  
 **pass** l\_w = Label(self.win2, text=**"Список жіночих імен"**)  
 l\_w.grid(row=0, column=0)  
 list\_of\_women = Listbox(self.win2, font=(**"Arial"**, 16), selectmode=EXTENDED)  
 self.list\_of\_women\_text = [**"Aurora"**, **"Anastasia"**, **"Valeria"**, **"Gertrude"**, **"Eva"**, **"Kira"**, **"Lolita"**, **"Olga"**, **"Flora"**, **"Julia"**]  
 **for** i **in** self.list\_of\_women\_text:  
 list\_of\_women.insert(END, str(i))  
 list\_of\_women.grid(row=1, column=0)  
 womens\_set = StringVar()  
 womens\_set.set(**"A"**)  
 womens\_a = Radiobutton(self.win2, text=**'A'**, variable=womens\_set, value=**"A"**)  
 womens\_b = Radiobutton(self.win2, text=**'B'**, variable=womens\_set, value=**"B"**)  
 womens\_a.grid(row=2, column=0, padx=30)  
 womens\_b.grid(row=2, column=0, sticky=E, padx=30)  
 b\_w = Button(self.win2, text=**"Запис в"**, command=set\_w)  
 b\_w.grid(row=2, column=0, sticky=W, padx=30)  
  
 l\_m = Label(self.win2, text=**"Список чоловічих імен"**)  
 l\_m.grid(row=0, column=1)  
 list\_of\_men = Listbox(self.win2, font=(**"Arial"**, 16), selectmode=EXTENDED)  
 self.list\_of\_men\_text = [**"Adam"**, **"Anatoly"**, **"Vissarion"**, **"Harry"**, **"Dmitri"**, **"Ivan"**, **"Leo"**, **"Mikhail"**, **"Ostap"**, **"Yuri"**]  
 **for** i **in** self.list\_of\_men\_text:  
 list\_of\_men.insert(END, str(i))  
 list\_of\_men.grid(row=1, column=1)  
 mens\_set = StringVar()  
 mens\_set.set(**"A"**)  
 mens\_a = Radiobutton(self.win2, text=**'A'**, variable=mens\_set, value=**"A"**)  
 mens\_b = Radiobutton(self.win2, text=**'B'**, variable=mens\_set, value=**"B"**)  
 mens\_a.grid(row=2, column=1, padx=30)  
 mens\_b.grid(row=2, column=1, sticky=E, padx=30)  
 b\_m = Button(self.win2, text=**"Запис в"**, command=set\_m)  
 b\_m.grid(row=2, column=1, sticky=W, padx=30)  
  
 save\_a\_b = Button(self.win2, text=**"Запис A в файл"**, command=save\_a)  
 save\_a\_b.grid(row=3, column=0)  
 save\_b\_b = Button(self.win2, text=**"Запис B в файл"**, command=save\_b)  
 save\_b\_b.grid(row=3, column=1)  
  
 read\_a\_b = Button(self.win2, text=**"Зчитати A з файлу"**, command=read\_a)  
 read\_a\_b.grid(row=4, column=0)  
 read\_b\_b = Button(self.win2, text=**"Зчитати B з файлу"**, command=read\_b)  
 read\_b\_b.grid(row=4, column=1)  
  
 clear\_a\_b = Button(self.win2, text=**"Видалити A"**, command=clear\_a)  
 clear\_a\_b.grid(row=5, column=0)  
 clear\_b\_b = Button(self.win2, text=**"Видалити A"**, command=clear\_b)  
 clear\_b\_b.grid(row=5, column=1)  
  
 **def** window3(self):  
 self.win3 = Toplevel(root)  
 self.win3.title(**"3 вікно"**)  
  
 **def** update\_a\_func():  
 list\_a.delete(0, END)  
 **for** i **in** list(self.A):  
 list\_a.insert(END, str(i))  
  
 **def** update\_b\_func():  
 list\_b.delete(0, END)  
 **for** i **in** list(self.B):  
 list\_b.insert(END, str(i))  
  
 l\_a = Label(self.win3, text=**"Множина А"**)  
 l\_a.grid(row=0, column=0)  
 list\_a = Listbox(self.win3, font=(**"Arial"**, 16), selectmode=EXTENDED)  
 update\_a\_func()  
 list\_a.grid(row=1, column=0)  
 update\_a = Button(self.win3, text=**"Оновити"**, command=update\_a\_func)  
 update\_a.grid(row=2, column=0)  
  
 l\_b = Label(self.win3, text=**"Множина B"**)  
 l\_b.grid(row=0, column=2)  
 list\_b = Listbox(self.win3, font=(**"Arial"**, 16), selectmode=EXTENDED)  
 update\_b\_func()  
 list\_b.grid(row=1, column=2)  
 update\_b = Button(self.win3, text=**"Оновити"**, command=update\_b\_func)  
 update\_b.grid(row=2, column=2)  
  
 rel\_s\_b = Button(self.win3, text=**"Відношення S"**, command=self.relations\_S)  
 rel\_s\_b.grid(row=3, column=0)  
  
 rel\_r\_b = Button(self.win3, text=**"Відношення R"**, command=self.relations\_R)  
 rel\_r\_b.grid(row=3, column=2)  
  
 **def** window4(self):  
 **def** union():  
 g = nx.DiGraph()  
 g.add\_nodes\_from(list(self.A | self.B))  
 g.add\_edges\_from(self.S, color=**'b'**)  
 g.add\_edges\_from(self.R, color=**'r'**)  
 edges = g.edges()  
 colors = [g[u][v][**'color'**] **for** u, v **in** edges]  
 nx.draw\_networkx(g, pos=nx.spring\_layout(g), arrows=**True**, with\_labels=**True**, edges=g.edges(), edge\_color=colors)  
 plt.show()  
  
 **def** crossing():  
 c = list()  
 **for** i **in** self.S:  
 **for** j **in** self.R:  
 **if** i == j:  
 c.append(i)  
 g = nx.DiGraph()  
 g.add\_nodes\_from(list(self.A | self.B))  
 g.add\_edges\_from(c)  
 nx.draw\_networkx(g, pos=nx.spring\_layout(g), arrows=**True**, with\_labels=**True**, edges=g.edges())  
 plt.show()  
  
 **def** difference():  
 c = list(self.R)  
 **for** i **in** self.S:  
 **for** j **in** self.R:  
 **if** i == j:  
 c.remove(j)  
 g = nx.DiGraph()  
 g.add\_nodes\_from(list(self.A | self.B))  
 g.add\_edges\_from(c)  
 nx.draw\_networkx(g, pos=nx.spring\_layout(g), arrows=**True**, with\_labels=**True**, edges=g.edges())  
 plt.show()  
  
 **def** no\_R():  
 c = list()  
 **for** i **in** self.A:  
 **for** j **in** self.B:  
 **if** [i, j] **not in** self.R:  
 c.append([i,j])  
 g = nx.DiGraph()  
 g.add\_nodes\_from(list(self.A | self.B))  
 g.add\_edges\_from(c)  
 nx.draw\_networkx(g, pos=nx.spring\_layout(g), arrows=**True**, with\_labels=**True**, edges=g.edges())  
 plt.show()  
  
 **def** inversion\_S():  
 c = list(self.S)  
 **for** i **in** range(len(c)):  
 c[i][0], c[i][1] = c[i][1], c[i][0]  
  
 g = nx.DiGraph()  
 g.add\_nodes\_from(list(self.A | self.B))  
 g.add\_edges\_from(c)  
 nx.draw\_networkx(g, pos=nx.spring\_layout(g), arrows=**True**, with\_labels=**True**, edges=g.edges())  
 plt.show()  
  
 self.win4 = Toplevel(root)  
 self.win4.title(**"4 вікно"**)  
 union\_b = Button(self.win4, text=**"Відношення R∪S"**, command=union)  
 union\_b.grid(row=0, column=0)  
 crossing\_b = Button(self.win4, text=**"Відношення R∩S"**, command=crossing)  
 crossing\_b.grid(row=1, column=0)  
 difference\_b = Button(self.win4, text=**"Відношення R\S"**, command=difference)  
 difference\_b.grid(row=2, column=0)  
 no\_R\_b = Button(self.win4, text=**"Відношення U\R"**, command=no\_R)  
 no\_R\_b.grid(row=3, column=0)  
 inversion\_S\_b = Button(self.win4, text=**"Відношення S-1"**, command=inversion\_S)  
 inversion\_S\_b.grid(row=4, column=0)  
  
 self.text\_S = Text(self.win4, width=100, height=5, font=(**"Arial"**, 10))  
 self.text\_S.grid(row=0, column=1)  
 update\_S\_b = Button(self.win4, text=**"Оновити S"**, command=self.update\_S)  
 update\_S\_b.grid(row=1, column=1)  
 self.update\_S()  
 self.text\_R = Text(self.win4, width=100, height=5, font=(**"Arial"**, 10))  
 self.text\_R.grid(row=2, column=1)  
 update\_R\_b = Button(self.win4, text=**"Оновити S"**, command=self.update\_R)  
 update\_R\_b.grid(row=3, column=1)  
 self.update\_R()  
  
 **def** calc\_variant(self):  
 N = int(self.e1.get())  
 G = int(self.e2.get())  
 Z = int((N+G % 60) % 30+1)  
 self.t1.delete(1.0, END)  
 self.t1.insert(INSERT, Z)  
  
 **def** relations\_S(self):  
 edges\_list\_S = list([])  
 **for** a **in** sample(self.A, randint(len(self.A)//2, len(self.A))):  
 **for** b **in** sample(self.B, randint(len(self.A)//2, len(self.B))):  
 **if** a **in** self.list\_of\_women\_text **and** b **in** self.list\_of\_men\_text:  
 s = **True  
 for** i **in** edges\_list\_S:  
 **if** a **in** i **or** b **in** i:  
 s = **False  
 if** s:  
 edges\_list\_S.append([a, b])  
 self.S = edges\_list\_S  
 self.g1 = nx.DiGraph()  
 self.g1.add\_nodes\_from(list(self.A | self.B))  
 self.g1.add\_edges\_from(edges\_list\_S)  
 nx.draw\_networkx(self.g1, pos=nx.spring\_layout(self.g1), arrows=**True**, with\_labels=**True**, edges=self.g1.edges(), edge\_color=**"b"**)  
 plt.show()  
  
 **def** relations\_R(self):  
 edges\_list\_S = self.S  
 edges\_list\_R = list()  
 **for** a **in** sample(self.A, randint(len(self.A)//4, len(self.A)//2)):  
 **for** b **in** sample(self.B, randint(len(self.A)//5, len(self.B)//2)):  
 **if** a **in** self.list\_of\_women\_text:  
 **for** i **in** edges\_list\_S:  
 **if** b **in** i **and** [a, b] **not in** edges\_list\_S:  
 edges\_list\_R.append([a, i[1]])  
 edges\_list\_R.append([a, i[0]])  
 self.R = edges\_list\_R  
 self.g2 = nx.DiGraph()  
 self.g2.add\_nodes\_from(list(self.A | self.B))  
 self.g2.add\_edges\_from(edges\_list\_R)  
 nx.draw\_networkx(self.g2, pos=nx.spring\_layout(self.g2), arrows=**True**, with\_labels=**True**, edges=self.g2.edges(), edge\_color=**"r"**)  
 plt.show()  
  
 **def** update\_S(self):  
 self.text\_S.delete(1.0, END)  
 self.text\_S.insert(INSERT, str(self.S))  
  
 **def** update\_R(self):  
 self.text\_R.delete(1.0, END)  
 self.text\_R.insert(INSERT, str(self.R))  
  
  
root = Tk()  
root.title(**"Головне меню"**)  
variant = Main()  
root.mainloop()

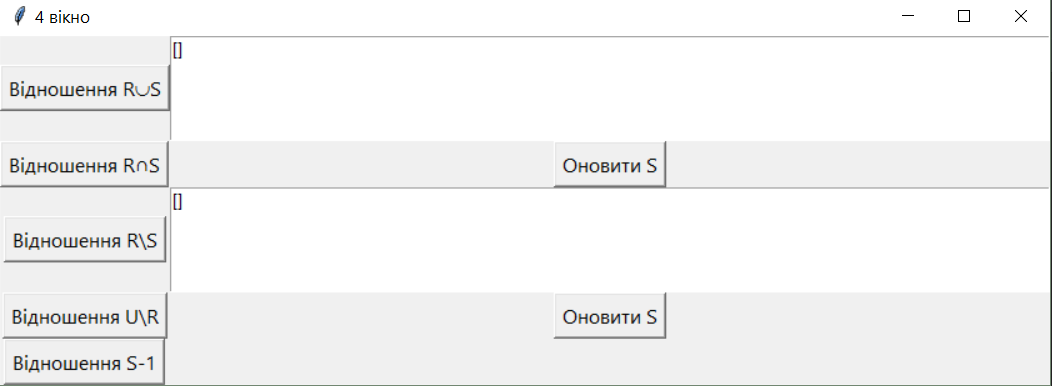
7. Роздруківка результатів виконання програми з контрольним прикладом



Відношення S де а – дружина b. (зображено синім),

Відношення R де а – кума b. (зображено червоним),





8. Аналіз результатів та висновки.

Основною проблемою при виконанні роботи було визначити задання відношень, оскільки я не хотів задавати, якісь відношення наперед. Для вирішення цієї проблеми я застосував модуль random який довільним способом обирав елементи множин і вже на їх основі довільним способом визначав відношення.