Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Дискретна математика

Лабораторна робота №3

«Графи. Способи представлення графів. Остовні дерева. Пошук

найкоротших шляхів»

Виконав: студент групи ІО-64

Андрійчук Д. А.

Залікова книжка № 6401

Перевірив Новотарський М. А.

Київ

2017 р.

**Мета роботи**: Вивчення властивостей графів, способів їх представлення та основних алгоритмів на графах.

**Завдання**:

1. Вивчити способи представлення графів. Написати програму для перетворення графу, заданого списком ребер в матрицю інцидентності та в граф, заданий матрицею суміжності.

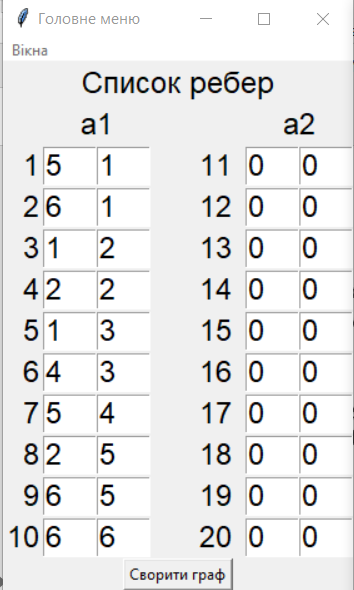
2. За правилом, наданим викладачем, сформувати матрицю інцидентності.

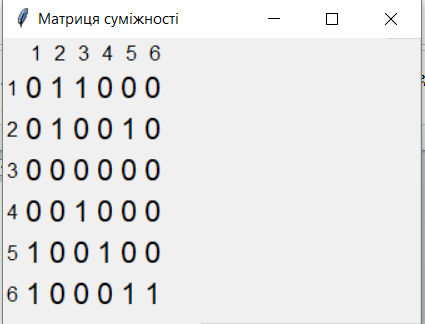
3. Представити початковий граф, заданий матрицею інцидентності, та кінцевий граф, заданий матрицею суміжності, у графічній формі.

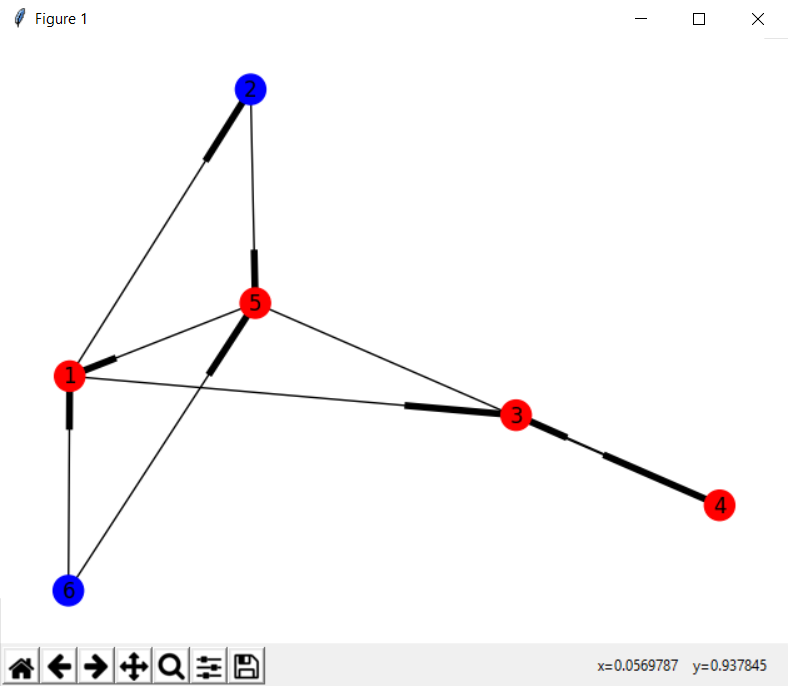
**Код програми**

**from** tkinter **import** \*  
**import** networkx **as** nx  
**import** pylab **as** plt  
**from** copy **import** deepcopy  
  
root = Tk()  
  
  
**class** element\_Matrix\_of\_adjacent:  
 **def** \_\_init\_\_(self, roo, row, column):  
 self.check = Entry(roo, width=3, font=(**"Arial"**, 18))  
 self.check.grid(row=row, column=column)  
 self.check.bind(**"<Button-1>"**, self.limiter)  
 self.check.bind(**"<Button-3>"**, self.limiter)  
 self.check.insert(END, **"0"**)  
  
 **def** limiter(self, event):  
 **if** self.check.get() == **"0"**:  
 self.check.delete(0, END)  
 self.check.insert(END, **"1"**)  
 **else**:  
 self.check.delete(0, END)  
 self.check.insert(END, **"0"**)  
  
 **def** getter(self):  
 **return** self.check.get()  
  
  
**class** element\_Matrix\_of\_incendies:  
 **def** \_\_init\_\_(self, roo, row, column):  
 self.check = Entry(roo, width=3, font=(**"Arial"**, 18))  
 self.check.grid(row=row, column=column)  
 self.check.bind(**"<Button-1>"**, self.limiter)  
 self.check.bind(**"<Button-3>"**, self.limiter)  
 self.check.insert(END, **"0"**)  
  
 **def** limiter(self, event):  
 **if** self.check.get() == **"0"**:  
 self.check.delete(0, END)  
 self.check.insert(END, **"+1"**)  
  
 **elif** self.check.get() == **"+1"**:  
 self.check.delete(0, END)  
 self.check.insert(END, **"-1"**)  
  
 **elif** self.check.get() == **"-1"**:  
 self.check.delete(0, END)  
 self.check.insert(END, **"±1"**)  
  
 **else**:  
 self.check.delete(0, END)  
 self.check.insert(END, **"0"**)  
  
 **def** getter(self):  
 **return** self.check.get()  
  
  
**class** Check:  
 **def** \_\_init\_\_(self, row, column):  
 self.check = Entry(root, width=3, font=(**"Arial"**, 18))  
 self.check.grid(row=row, column=column)  
 self.check.bind(**"<Button-1>"**, self.limiter)  
 self.check.bind(**"<Button-3>"**, self.zero)  
 self.check.insert(END, **"0"**)  
  
 **def** limiter(self, event):  
 **if** self.check.get() == **"10"**:  
 self.check.delete(0, END)  
 self.check.insert(END, **"0"**)  
 **elif** self.check.get():  
 **if** self.check.get().isdigit():  
 n = int(self.check.get())  
 **else**:  
 n = -1  
 self.check.delete(0, END)  
 self.check.insert(END, str(n+1))  
  
 **def** returner(self):  
 **return** self.check.get()  
  
 **def** zero(self, event):  
 self.check.delete(0, END)  
 self.check.insert(END, **"0"**)  
  
  
**class** Main:  
 edges = []  
  
 **def** \_\_init\_\_(self):  
 menubar = Menu(root)  
 filemenu = Menu(menubar, tearoff=0)  
 filemenu.add\_command(label=**"Матрця інцидентності"**, command=self.Matrix\_of\_incendies\_builder)  
 filemenu.add\_command(label=**"Матрця суміжності"**, command=self.Matrix\_of\_adjacent\_builder)  
 filemenu.add\_command(label=**"Варіант"**, command=self.variant)  
 menubar.add\_cascade(label=**"Вікнa"**, menu=filemenu)  
 root.config(menu=menubar)  
  
 l = Label(root, text=**"Список ребер"**, font=(**"Arial"**, 18))  
 l.grid(row=0, column=0, columnspan=6)  
 x = 1  
 **for** i **in** range(0, 4, 3):  
 l = Label(root, text=**"a"**+str(x), font=(**"Arial"**, 18))  
 l.grid(row=1, column=i + 1, columnspan=2)  
 x += 1  
 **for** i **in** range(10):  
 l = Label(root, text=str(i + 1), font=(**"Arial"**, 18))  
 l.grid(row=i + 2, column=0, sticky=E)  
 **for** i **in** range(10):  
 l = Label(root, text=**" "**+str(i + 11), font=(**"Arial"**, 18), width=5)  
 l.grid(row=i + 2, column=3, sticky=E)  
 **for** i **in** range(10):  
 self.edges.append([])  
 **for** j **in** range(2):  
 self.edges[i].append(Check(i + 2, j + 1))  
 **for** i **in** range(10):  
 self.edges.append([])  
 **for** j **in** range(2):  
 self.edges[i+10].append(Check(i + 2, j + 4))  
  
 self.b\_inc = Button(root, text=**"Сворити граф"**, command=self.made\_matrix\_of\_incident\_and\_graph)  
 self.b\_inc.grid(row=13, column=0, columnspan=13)  
  
 **def** made\_matrix\_of\_incident\_and\_graph(self):  
 edges\_list = []  
 **for** i **in** range(len(self.edges)):  
 edges\_list.append([])  
 **for** j **in** self.edges[i]:  
 edges\_list[i].append(j.returner())  
 **for** i **in** range(len(edges\_list)-1, -1, -1):  
 **if "0" in** edges\_list[i] **or** edges\_list.count(edges\_list[i]) > 1:  
 **del** edges\_list[i]  
  
 self.window\_inc = Toplevel(root)  
 self.window\_inc.title(**"Матриця інцидентності"**)  
 **for** i **in** range(len(edges\_list)):  
 l = Label(self.window\_inc, text=**"e"**+str(i + 1), font=(**"Arial"**, 12))  
 l.grid(row=0, column=i+1)  
  
 nodes = set()  
 **for** i **in** edges\_list:  
 **for** j **in** i:  
 nodes.add(j)  
 nodes = list(nodes)  
 nodes.sort()  
 **for** i **in** range(len(nodes)):  
 l = Label(self.window\_inc, text=nodes[i], font=(**"Arial"**, 12))  
 l.grid(row=i+1, column=0, sticky=E)  
  
 **for** row **in** range(len(nodes)):  
 **for** column **in** range(len(edges\_list)):  
 **if** edges\_list[column][0] == nodes[row] **and** edges\_list[column][1] == nodes[row]:  
 l = Label(self.window\_inc, text=**"±1"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
 **elif** edges\_list[column][0] == nodes[row]:  
 l = Label(self.window\_inc, text=**"+1"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
 **elif** edges\_list[column][1] == nodes[row]:  
 l = Label(self.window\_inc, text=**"-1"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
 **else**:  
 l = Label(self.window\_inc, text=**"0"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
  
 window\_sum = Toplevel(root)  
 window\_sum.title(**"Матриця суміжності"**)  
 **for** i **in** range(len(nodes)):  
 l = Label(window\_sum, text=nodes[i], font=(**"Arial"**, 12))  
 l.grid(row=i+1, column=0, sticky=E)  
 l = Label(window\_sum, text=nodes[i], font=(**"Arial"**, 12))  
 l.grid(row=0, column=i+1, sticky=E)  
 **for** row **in** range(len(nodes)):  
 **for** column **in** range(len(nodes)):  
 **if** [nodes[row], nodes[column]] **in** edges\_list:  
 l = Label(window\_sum, text=**"1"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
 **else**:  
 l = Label(window\_sum, text=**"0"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
  
 G = nx.DiGraph()  
 G.add\_edges\_from(edges\_list, color=**'b'**)  
 val\_map = {}  
 **for** i **in** edges\_list:  
 **if** i[0] == i[1]:  
 val\_map.update({i[0]: **"b"**})  
 values = [val\_map.get(node, **"r"**) **for** node **in** G.nodes()]  
 nx.draw(G, node\_color=values, with\_labels=**True**)  
 plt.show()  
  
 **def** Matrix\_of\_incendies\_builder(self):  
 **def** made\_graph():  
 edges\_list = []  
 **for** i **in** range(len(edges)):  
 edges\_list.append([])  
 **for** j **in** range(len(edges[i])):  
 edges\_list[i].append(edges[i][j].getter())  
 tmp = []  
 **for** i **in** range(len(edges\_list)-1, -1, -1):  
 **if** edges\_list[i].count(**"0"**) == 10:  
 edges\_list.pop(i)  
 **elif "±1" in** edges\_list[i] **and** (**"+1" in** edges\_list[i] **or "-1" in** edges\_list[i]):  
 edges\_list.pop(i)  
 **elif "+1" in** edges\_list[i] **and "-1" not in** edges\_list[i] **or "-1" in** edges\_list[i] **and "+1" not in** edges\_list[i]:  
 edges\_list.pop(i)  
 **elif** edges\_list[i].count(**"+1"**) > 1 **or** edges\_list[i].count(**"-1"**) > 1 **or** edges\_list[i].count(**"±1"**) > 1:  
 edges\_list.pop(i)  
 **else**:  
 **if "±1" in** edges\_list[i]:  
 tmp = [edges\_list[i].index(**"±1"**)+1, edges\_list[i].index(**"±1"**)+1]  
 edges\_list[i] = tmp  
 **elif "+1" in** edges\_list[i] **and "-1" in** edges\_list[i]:  
 tmp = [edges\_list[i].index(**"+1"**)+1, edges\_list[i].index(**"-1"**)+1]  
 edges\_list[i] = tmp  
 **else**:  
 edges\_list.pop(i)  
 edges\_list2 = deepcopy(edges\_list)  
 window\_sum2 = Toplevel(root)  
 window\_sum2.title(**"Матриця суміжності"**)  
 nodes = set()  
 **for** i **in** edges\_list2:  
 **for** j **in** i:  
 nodes.add(j)  
 nodes = list(nodes)  
 nodes.sort()  
 **for** i **in** range(len(nodes)):  
 l = Label(window\_sum2, text=nodes[i], font=(**"Arial"**, 12))  
 l.grid(row=i+1, column=0, sticky=E)  
 l = Label(window\_sum2, text=nodes[i], font=(**"Arial"**, 12))  
 l.grid(row=0, column=i+1, sticky=E)  
 **for** row **in** range(len(nodes)):  
 **for** column **in** range(len(nodes)):  
 **if** [nodes[row], nodes[column]] **in** edges\_list:  
 l = Label(window\_sum2, text=**"1"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
 **else**:  
 l = Label(window\_sum2, text=**"0"**, font=(**"Arial"**, 18))  
 l.grid(row=row+1, column=column+1)  
  
 G = nx.DiGraph()  
 **if** len(edges\_list) != 0:  
 G.add\_edges\_from(edges\_list, color=**'b'**)  
 val\_map = {}  
 **for** i **in** edges\_list:  
 **if** i[0] == i[1]:  
 val\_map.update({i[0]: **"b"**})  
 values = [val\_map.get(node, **"r"**) **for** node **in** G.nodes()]  
 nx.draw(G, node\_color=values, with\_labels=**True**)  
 plt.show()  
  
 Matrix\_of\_incendies = Toplevel(root)  
 Matrix\_of\_incendies.title(**"Mатрця інцидентності"**)  
 edges = []  
 **for** i **in** range(20):  
 l = Label(Matrix\_of\_incendies, text=**"e"**+str(i + 1), font=(**"Arial"**, 12))  
 l.grid(row=0, column=i+1)  
 **for** i **in** range(10):  
 l = Label(Matrix\_of\_incendies, text=i+1, font=(**"Arial"**, 12))  
 l.grid(row=i+1, column=0, sticky=E)  
 **for** column **in** range(20):  
 edges.append([])  
 **for** row **in** range(10):  
 edges[column].append(element\_Matrix\_of\_incendies(Matrix\_of\_incendies, row+1, column+1))  
 self.b\_inc = Button(Matrix\_of\_incendies, text=**"Сворити граф"**, command=made\_graph)  
 self.b\_inc.grid(row=13, column=0, columnspan=20)  
  
 **def** Matrix\_of\_adjacent\_builder(self):  
 **def** made\_graph():  
 edges\_list = []  
 **for** i **in** range(len(edges)):  
 edges\_list.append([])  
 **for** j **in** range(len(edges[i])):  
 edges\_list[i].append(edges[i][j].getter())  
 tmp = []  
 **for** i **in** range(len(edges\_list)):  
 **for** j **in** range(len(edges\_list[i])):  
 **if** edges\_list[i][j] == **"1"**:  
 tmp.append([str(j+1), str(i+1)])  
 edges\_list = tmp  
 print(edges\_list)  
 G = nx.DiGraph()  
 G.add\_edges\_from(edges\_list, color=**'b'**)  
 val\_map = {}  
 **for** i **in** edges\_list:  
 **if** i[0] == i[1]:  
 val\_map.update({i[0]: **"b"**})  
 values = [val\_map.get(node, **"r"**) **for** node **in** G.nodes()]  
 nx.draw(G, node\_color=values, with\_labels=**True**)  
 plt.show()  
  
  
 Matrix\_of\_adjacent = Toplevel(root)  
 Matrix\_of\_adjacent.title(**"Матрця суміжності"**)  
 edges = []  
 **for** i **in** range(10):  
 l = Label(Matrix\_of\_adjacent, text=i + 1, font=(**"Arial"**, 12))  
 l.grid(row=0, column=i+1)  
 **for** i **in** range(10):  
 l = Label(Matrix\_of\_adjacent, text=i + 1, font=(**"Arial"**, 12))  
 l.grid(row=i+1, column=0, sticky=E)  
 **for** column **in** range(10):  
 edges.append([])  
 **for** row **in** range(10):  
 edges[column].append(element\_Matrix\_of\_adjacent(Matrix\_of\_adjacent, row+1, column+1))  
 self.b\_inc = Button(Matrix\_of\_adjacent, text=**"Сворити граф"**, command=made\_graph)  
 self.b\_inc.grid(row=13, column=0, columnspan=20)  
  
 **def** variant(self):  
 Variant = Toplevel(root)  
 Variant.title(**"Варіант"**)  
 l = Label(Variant, text=**"Андрійчук Дмитро"**, font=(**"Arial"**, 18))  
 l.grid(row=0, column=0)  
 l = Label(Variant, text=**"Номер розрахункової - 6401"**, font=(**"Arial"**, 18))  
 l.grid(row=1, column=0)  
 l = Label(Variant, text=**"Варіант - 2"**, font=(**"Arial"**, 18))  
 l.grid(row=2, column=0)  
  
root.title(**"Головне меню"**)  
variant = Main()  
root.mainloop()

**На головному екрані можна задати граф списком ребер,**



**Після чого створюється граф, екран на якому зображена матриця суміжності, та екран з матрицею інцидентності.**  



**Якщо відкрити строку меню то можна створити два інших вікна, у першому можна створити граф заданий матрицею суміжності, у іншому матрицею інцидентності після чого знову формується матриця суміжності.** 