Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Дискретна математика

Лабораторна робота №4

«Розфарбовування графа, алгоритми розфарбування».

Виконав: студент групи ІО-64

Андрійчук Д. А.

Залікова книжка № 6401

Перевірив Новотарський М. А.

Київ

2017 р.

**Тема:** «Розфарбовування графа, алгоритми розфарбування».

**Мета роботи:** вивчення способів правильного розфарбовування графа.

**Завдання:** створити програму для правильного розфарбовування графа на основі одного з алгоритмів розфарбування.

Варіант

6 А) Виконати завдання 6 до лабораторної роботи.

Набути теоретичні знання по темі «Розфарбування графів». Створити

програму розфарбування графів, яка реалізує «жадібний» алгоритм

розфарбування.

Б) Програма повинна дозволяти розфарбування довільного графа.

В) Перевірити роботу програми на даному графі *G*.

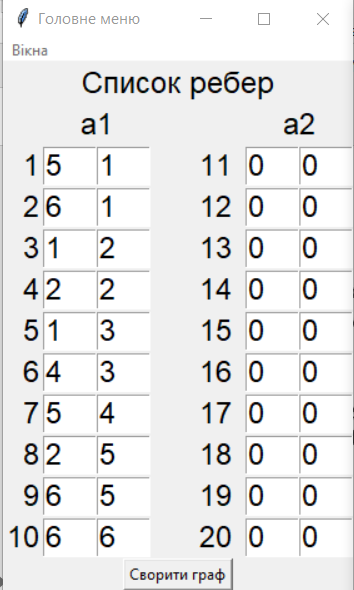
Вивести у графічному режимі розфарбований граф, або включити у

протокол розфарбований вручну граф за результатами роботи програми

**Код програми**

**from** tkinter **import** \*  
**import** networkx **as** nx  
**import** pylab **as** plt  
  
root = Tk()  
  
  
**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):  
  
  
 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]  
  
 nodes = set()  
 **for** i **in** edges\_list:  
 **for** j **in** i:  
 nodes.add(j)  
 nodes = list(nodes)  
 nodes.sort()  
  
 **for** i **in** range(len(edges\_list)-1, -1, -1):  
 **if** edges\_list[i][0] != edges\_list[i][1]:  
 **if** edges\_list[i][::-1] **in** edges\_list:  
 edges\_list.remove(edges\_list[i])  
 **if** edges\_list[i][0] < edges\_list[i][1]:  
 edges\_list[i] = edges\_list[i][::-1]  
 edges\_list.sort()  
  
 G = nx.Graph()  
 colors = [**"red"**, **"blue"**, **"green"**, **"orange"**, **"yellow"**, **"violet"**, **"gray"**, **"brown"**]  
 G.add\_edges\_from(edges\_list, color=**'b'**)  
 val\_map = {}  
 val\_map.update({nodes[0]: **"red"**})  
 **for** node **in** nodes:  
 i = 0  
 **for** edge **in** edges\_list:  
 **if** node **in** edge:  
 i += 1  
 **while** node **not in** val\_map:  
 **for** j **in** range(i):  
 color = 0  
 **for** edge **in** edges\_list:  
 **if** node **in** edge:  
 **if** node == edge[0]:  
 **if** edge[1] **in** val\_map:  
 **while** val\_map.get(edge[1]) == colors[color]:  
 color += 1  
 **else**:  
 **if** edge[0] **in** val\_map:  
 **while** val\_map.get(edge[0]) == colors[color]:  
 color += 1  
 val\_map.update({node: colors[color]})  
 values = [val\_map.get(node) **for** node **in** G.nodes()]  
 nx.draw(G, node\_color=values, with\_labels=**True**)  
 plt.show()  
  
 **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 - 6"**, 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**