ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ

ԿՐԹՈՒԹՅԱՆ, ԳԻՏՈՒԹՅԱՆ, ՄՇԱԿՈՒՅԹԻ և ՍՊՈՐՏԻ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆԻ

ՍԻՆՈՓՍԻՍ ԿՐԹԱԿԱՆ ԴԵՊԱՐՏԱՄԵՆՏԻ

ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ և ՀԱՄԱԿԱՐԳԵՐԻ ՄՇԱԿՄԱՆ ԾՐԱԳՐԱՅԻՆ ՄԻՋՈՑՆԵՐԻ ԱՄԲԻՈՆ

SYNOPSYS®

ԿՈՒՐՍԱՅԻՆ ԱՇԽԱՏԱՆՔ

ԽՈՒՄԲ	SS919-Ս
ԱՌԱՐԿԱ	Դիսկրետ մաթեմատիկա (Գրաֆների տեսություն)
ԹԵՄԱ	Պարզ ցիկլերի որոնում
ԴԱՍԱԽՈՍ	Գարեգին Սարգսյան
ՈՒՍԱՆՈՂ	Գրիգորի Վերդյան
ՊԱՇՏՊԱՆՈՒԹՅՈՒՆԸ	02/06/2022

2022

Բովանդակություն

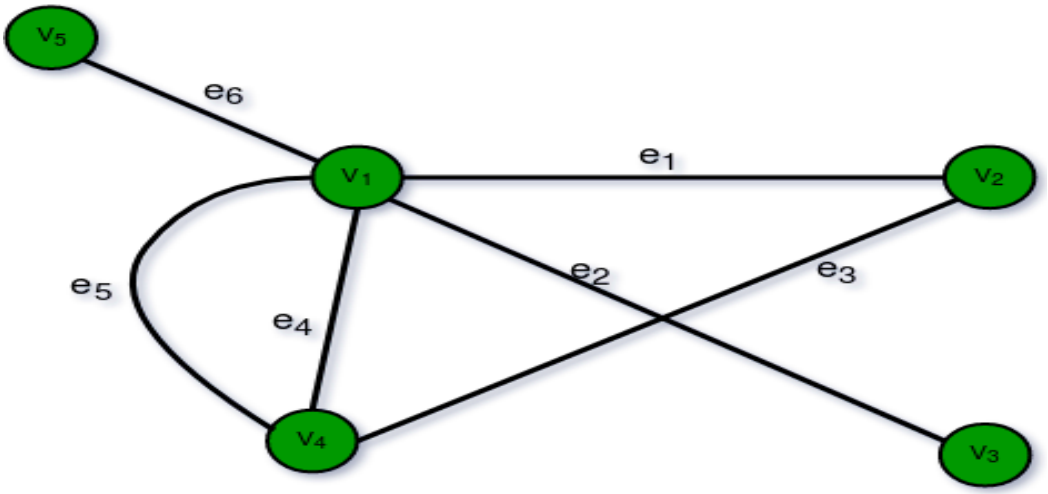
Ներածություն

1. Ինչ է գրաֆը: Ընդհանուր նկարագիր:

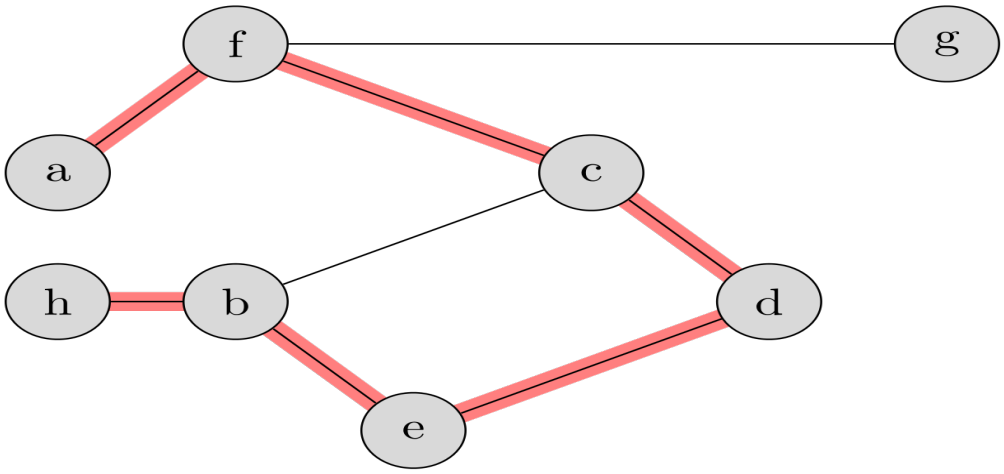
Դիցուք $V = \{v_1, \dots, v_n\}$–ը ցանկացած ոչ դատարկ վերջավոր բազմություն է, և $V^{(2)}$–ն V բազմության տարրերի բոլոր ոչ կարգավոր զույգերի բազմությունն է, այնպես որ $|V^{(2)}| = \binom{n}{2}$:

Ենթադրենք, որ $E \subseteq V^{(2)}$: (V, E) կարգավոր զույգին կանվանենք գրաֆ, և այն կնշանակենք G-ով:

$G = (V, E)$ գրաֆը բաղկացած է գագաթներից(vertices)՝ V, և գագաթներն իրար միացնող կողերից(edges)՝ E:

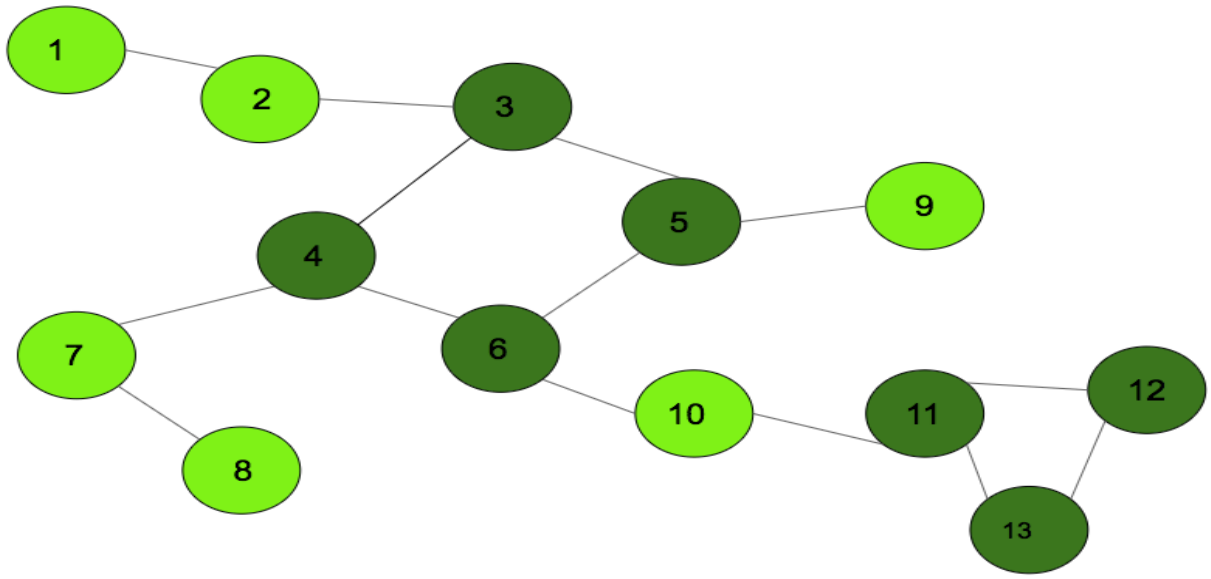


Գրաֆի ճանապարհը դա $V = \{v_1, \dots, v_n\}$ գագաթների հաջորդականություն է, որտեղ $\{v_i, \dots, v_{i+1}\} \in E$ ամեն մի 1 <= I < N: Նման ճանապարհի երկարությունը դա դրանում առկա կողերի քանակն է, որը հավասար է N – 1:



Եթե գրաֆում առկա է այնպիսի կող(v, v), որը գագաթից գնում է դեպի հենց այդ նույն գագաթը, ապա կարող ենք ասել որ գրաֆում առկա է ցիկլ:

Պարզ ցիկլ(Cp, p >= 3) է կոչվում այն ճանապարհը, որում առկա է այնպիսի ցիկլ, որ բոլոր գագաթները տարբերվում են մինյանցից, բացառությամբ առաջին և վերջին գագաթները, որով ձևավորվում է ցիկլը:



2. Գրաֆի ներկայացման եղանակները:

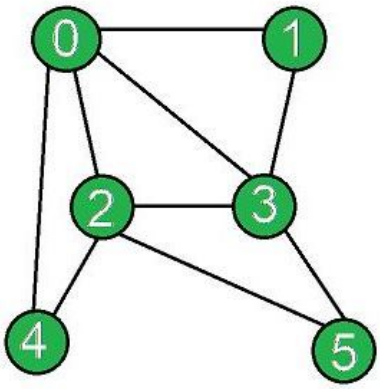
2.1. Գրաֆը կարելի է տալ, նշելով նրա գագաթների և կողերի բազմությունները՝

$G = (V, E)$ գրաֆը, որտեղ $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ և $E = \{v_1v_2, v_2v_3, v_3v_4\}$:

2.2. Գրաֆը կարելի է ներկայացնել նաև կրկնակի զանգվածի՝ մատրիցայի տեսքով, որը կոչվում է հարևանության մատրից(adjacency matrix): Ամեն մի կողի(u, v) համար մատրիցայի համապատասխան տեղում դրվում է true արժեք(կամ 1), և false(կամ 0) հակառակ դեպքում: Այսինքն, եթե $G = (V, E)$ գրաֆում $V = \{v_1, \dots, v_n\}$ և $E = \{e_1, \dots, e_m\}$, ապա այդ գրաֆին համապատասխանեցնենք n x n կարգի $A(G) = (a_{ij})_{nxn}$ մատրիցը հետևյալ կերպ.

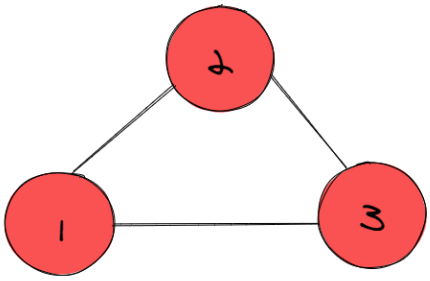
$$a_{ij} = \begin{cases} 1, & \text{եթե } Vi \text{ և } Vj \text{ հարևան են} \\ 0, & \text{հակառակ դեպքում} \end{cases}$$

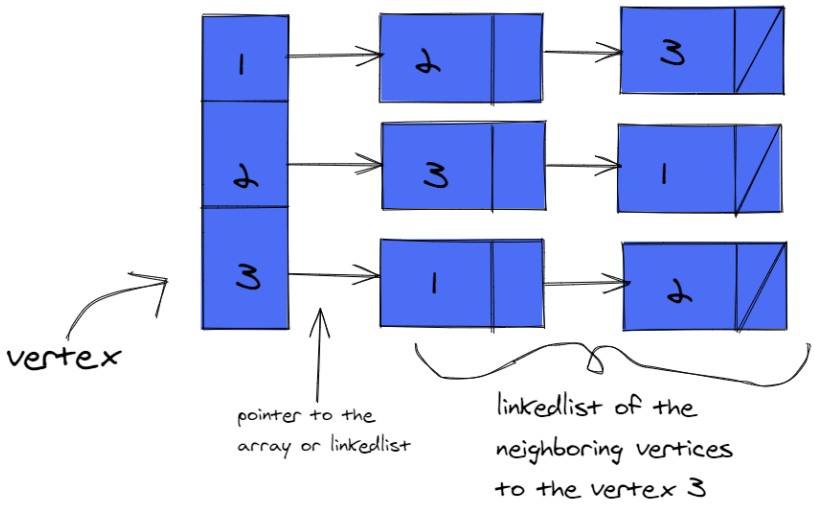
Ցանկացած i-ի համար $(1 \le i \le n)$ $a_{ii} = 0$, և ցանկացած i, j-ի համար $(1 \le i, j \le n)$ $a_{ij} = a_{ji}$:



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

2.3. Գրաֆի մեկ այլ ներկայացման եղանակ է հարևանության զանգվածը(adjacency list): Ամեն մի գագաթի համար պահում ենք հարևան գագաթների զանգված: Այս եղանակն ամենատարածվածն է գրաֆերի ներկայացման խնդիրներում:





Առաջադրված խնդիրը

1. Առաջադրանքը

Տրված G(V, E) վերջավոր գրաֆում Cp(p>=3) ենթագրաֆի որոնման ալգորիթմի մշակում և ծրագրային իրացում:

Խնդիրն իրագործվել է հետևյալ կերպ. տրված p-ի համար պարզ ցիկլերի քանակի հաշվարկ:

2. Աշխատանքը

Github https://github.com/grigverdyan/Graph

Խնդրի իրագործումը

1. Ծրագրի ընդհանուր նկարագիրը

Ծրագիրն աշխատացնելու համար տերմինալում անհրաժեշտ է հավաքել make հրամանը, որն աշխատացնելով մեյքֆայլը, կոմպիլացնում է կոդերը(.cpp, .py), ստեղծում է object file-երը(.o) և ./graph.exe ֆայլը(output-ը):

./graph.exe output ֆայլին պետք է փոխանցվի 2 պարամետր՝

1. ամբողջ թիվ(>=3), որն իրենից ներկայացնում է թե գրաֆում ինչ երկարության պարզ ցիկլերի քանակը պետք է հաշվվի,
2. ֆայլ (.txt ֆորմատով), որտեղից ծրագիրը կարդում և ստանում է գրաֆի հարևանության զանգվածը(adjacency list):

Տերմինալում գրվելիք վերոնշյալ հրամանը կլինի՝

./graph.exe [number] [filename.txt]

Օրինակ՝ ./graph.exe 4 input.txt

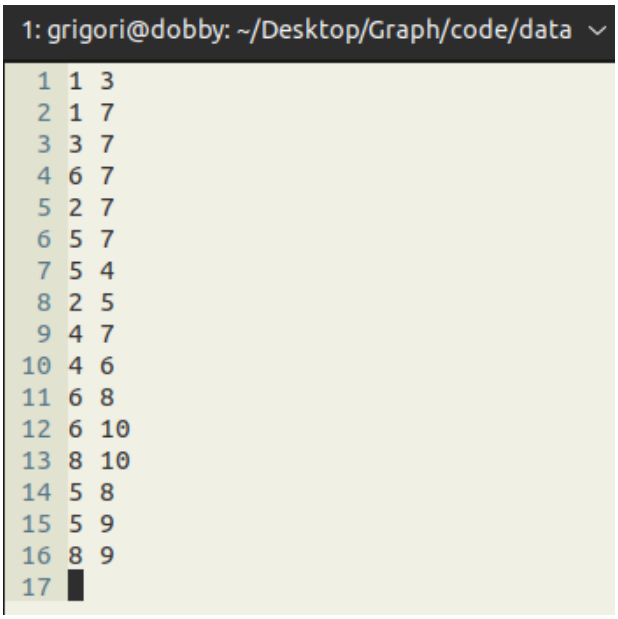
Հրամանի իրագործումից հետո, եթե ինչ որ կարգի error-ներ(file, input etc)չեն հայտնաբերվում, ապա տերմինալում արտածվում է պարզ ցիկլերի քանակը, ապա visual պատկերի տեսքով գրաֆը և նրա հարևանության մատրիցը(adjacency matrix):

2. Ֆայլից գրաֆի տվյալների ստացում

Ֆայլը կարող է լինի ցակացած անունի, բայց պետք է ունենա [.txt] extension-ը, և պետք է անպայման գտնվի data directory-ում:

Սխալ user input-ի դեպքում ծրագիրը Error է նետում: Գրաֆում գագաթների մաքսիմալ արժեքը չպետք է գերազանցի եռանիշի շեմը(թիվը փոփոխվող է՝ ծրագրի իմպլեմենտացիոն դետալ):

Ֆայլում գրաֆը պետք է ներկայացված լինի հետևյալ կերպ՝



Տող առ տող կարդալուն զուգընթաց ստեղծվում է Edge ստրուկտուրա, որի մեջ պահվում է գագաթների կողմերի վերաբերյալ ինֆորմացիան։

3. Գրաֆի ստացում

Ֆայլը կարդալուց հետո ծրագիրը ստեղծում է Graph տիպի օբյեկտ, և հաջորդաբար կատարում հետևյալ գործողությունները.

- ստեղծված Edge ստրուկտուրայի միջոցով ստանում է գրաֆի հարևանության մատրիցի չափը,
- արևանության մատրիցի չափով ստեղծում է մատրիցա և այն default initialize անում 0-ներով,
- ստեղծված 0-ներով մատրիցը՝ Edge ստրուկտուրայում պահպանված գագթների և կողերի վերաբերյալ ինֆորմացիայի միջոցով վերափոխվում է գրաֆի հարևանության մատրիցի,
- եթե գրաֆը պարունակում էր այլ ցիկլեր, քան պարզը, ապա գրաֆը պարզեցվում է՝ այն է, հարևանության մատրիցում մաքսիմալ արժեքը 1 է սարքում,

Այս ամենից հետո գրաֆի հարևանության մատրիցը գրվում է /data/out.txt ֆայլում, վիզուալիզացման հետագա նպատակներով։

4. Պարզ ցիկլերի որոնում: DFS ալգորիթմ

Ստեղծվում է SimpleCycle տիպի օբյեկտ, որին փոխանցվում են ծրագրի կատարման ընթացքում փնտրվող պարզ ցիկլերի երկարության արժեքը և ստեղծման բոլոր փուլերով անցած Graph տիպի օբյեկտը։

4.1. DFS ալգորիթմ

Գրաֆում պարզ ցիկլերի որոնումն իրագործված է «Փնտրում դեպի խորություն», այն է՝ «DFS - Depth for Search» ալգորիթմի միջոցով։

Գրաֆը շրջանցելու մեթոդներից մեկը։ Ալգորիթմի ստրատեգիան, ինչպես հետևում է անվանումից՝ գնալ դեպի խորություն որքան որ հնարավոր է։ Որոնման ալգորիթմը նկարագրվում է ռեկուրսիվ՝ հերթով վերցվում է հանգույցից դուրս եկող բոլոր արմատները։ Եթե կողը տանում է դեպի չբացահայտված հանգույցը, ապա ալգորիթմը վերսկսում ենք այդ հանգույցից, հետո վերադառնում ենք և վերցնում ենք մյուս արմատները։ Վերադարձը կատարվում է այն ժամանակ, երբ դիտարկվող գագաթում չեն մնացել կողեր, որոնք տանում են դեպի չբացահայտված գագաթներ։ Եթե ալգորիթմի ավարտից հետո դեռ մնացել են չբացահայտված հանգույցներ, ապա ալգորիթմը պետք է կիրառել չբացահայտված հանգույցներից որևէ մեկից սկսած։

Ալգորիթմի բարդությունը O(V), է, գագաթների քանակի չափով, իսկ վատագույն դեպքում Θ(V+E).

4.2. Պարզ ցիկլերի քանակը

Ծրագիրն իրականացրել եմ այնպես, որ այցելված գագաթները ներկելու փոխարեն օգտագործվում է տրամաբանական(bool) արժեքների զանգված, որն արտահայտում է գրաֆի այցելված լինել-չլինելու վիճակը։

Ալգորիթմի միջոցով տրված երկարության պարզ ցիկլերի քանակը հաշվելուց հետո, արդյունքն արտածվում է standard output-ում, այն է տերմինալում։

```
There are 4 simple cycles of length 4!
```

5. Վիզուալիզացիա

Կուրսային աշխատանքի վիզուալիզացման փուլում ավելի պատկերավոր պատկեր կստանանք մեր գրաֆի և նրա հարևանության մատրիցի մասին։

Վիզուալիզացումն, իրագործված է Python ծրագրավորման լեզվով, ի տարբերություն կոդի մնացյալ հատվածների, որոնք իրագործել էի C++ լեզվով։

5.1 Ինչու՞ Python վիզուալիզացիայի համար։

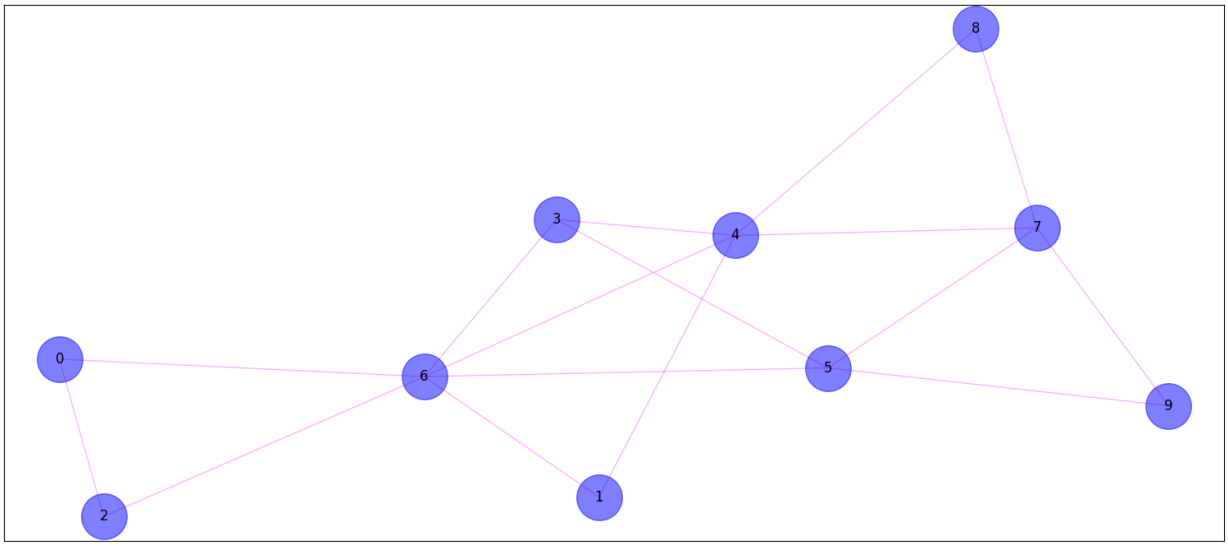
Python-ը համարվելով բարձր մակարդակի ծրագրավորման լեզու, և լինելով Մեքենայական ուսուսման և Արհեստական Բանականության մեջ օգտագործվող ամենատարածված լեզուն, հնարավորություն է տալիս օգտագործել գրադարանների ահռելի ներուժից։

Օգտագործել եմ մաթեմատիկական գործողությունների համար նախատեսված Numpy-ը, բոլոր տեսակի գրաֆների հետ աշխատելու համար նախատեսված Matplotlib-ը և վիզուալիզացման մեծ մեխանիզմներ տրամադրող NetworkX մեխանիզմը։

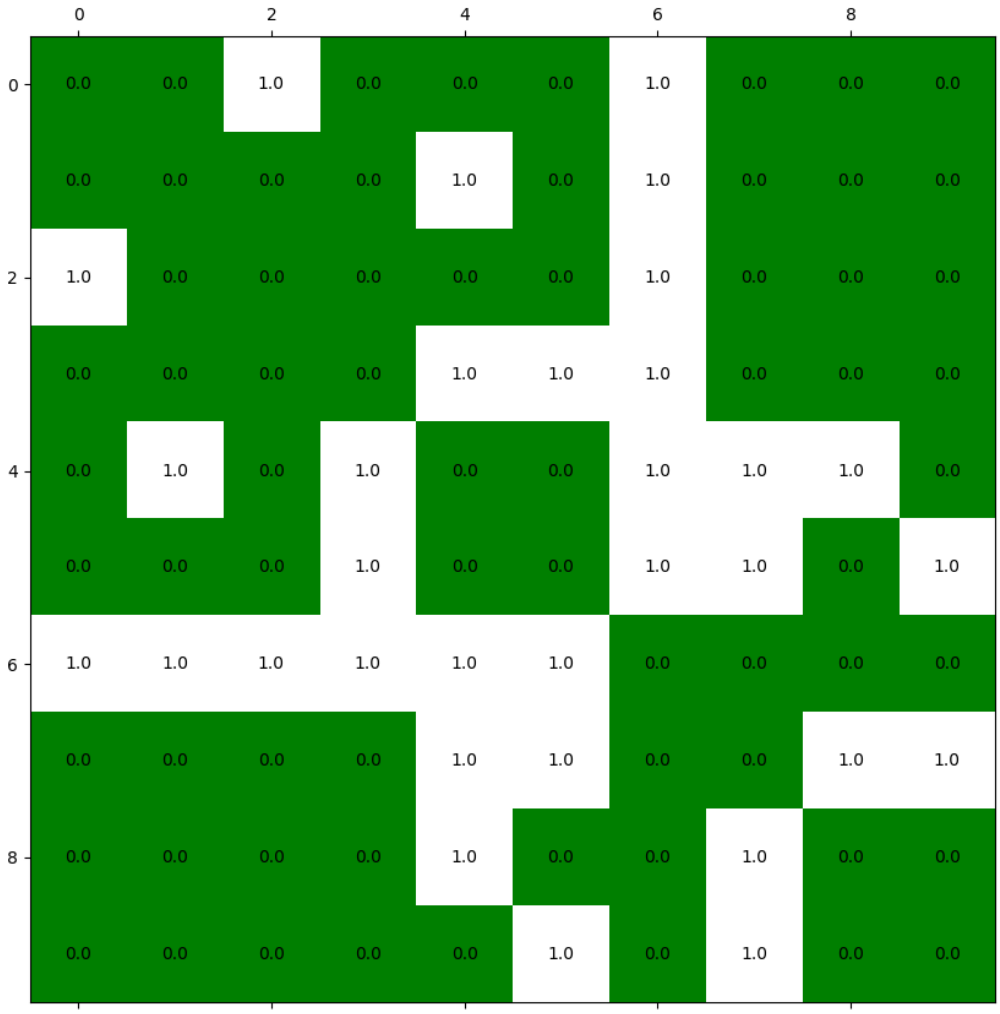
5.2 Ծրագրի արդյունքը

visuzalization.py ֆայլը պատասխանատու է գրաֆի վիզուալիզացման հատվածի համար։ data/out.txt ֆայլից կարդում և ստանում ենք գրաֆի հարևանության մատրիցը։

Տալիս ենք գրաֆի վիզուալիզացման համար էսթետիկ պարամետրեր(տառաչափ, գույն, երկարություն և այլն) և ստանում գրաֆի պատկերը։



Նույն եղենակավ, գրաֆի պատկերն ուսումնասիրելուց հետո ստանում ենք գրաֆի հարևանության մատրիցը։



Գրականության ցանկ

1. Գրքեր

“Introduction to Algorithms” *T. H. Cormen, C. E. Leiserson, R. L. Rivest, C.Stein*

«Գրաֆների տեսություն» *Պ. Ա. Պետրոսյան, Վ. Վ. Մկրտչյան, Ռ.Ռ. Քամալյան*

«Դիսկրետ մաթեմատիկայի դասընթաց» *Ռ. Ն. Տոնոյան*

“Data Structures and Algorithm Analysis in C++” *M. A. Weiss*

2. Հղումներ

Numpy Doc https://numpy.org/doc/

MatplotLib Doc https://matplotlib.org/stable/users/index

NetworkX Doc https://networkx.org/documentation/stable/tutorial.html