# *ლექცია 1* ძირითადი ცნებები გრაფთა თეორიიდან

- შესავალი
- ძირითადი ცნებები გრაფთა თეორიიდან
- გრაფის წარმოდგენა

## შესავალი

კომპიუტერული მეცნიერების სპეციალისტისთვის ალგორითმების შესწავლა მნიშვნელოვანია როგორც პრაქტიკული, ისე თეორიული თვალსაზრისით. პრაქტიკულად მას უნდა ჰქონდეს წარმოდგენა ძირითად ალგორითმებზე, რომლებიც შესაბამის მონაცემთა სტრუქტურის არჩევით, პროგრამული ინსტრუმენტების გამოყენებით, მისცემს ალგორითმის ეფექტურად იმპლემენტაციის საშუალებას ამა თუ იმ ამოცანისთვის. თეორიული თვალსაზრისით, ალგორითმები წარმოადგენენ ბაზისს, რომლის გარეშეც შეუძლებელია პროგრამული კოდის დაწერა.

ალგორითმების შესწავლა არ გულისხმობს რამდენიმე ალგორითმის აღწერას რამდენიმე კონკრეტული ამოცანისთვის, არამედ სტილის და მიდგომების გააზრებას, რომელიც გამოადგება პროგრამისტს ახალი ამოცანის დასმისას, ალგორითმის გამოყენებაში. მეორე მხრივ, პროგრამისტმა უნდა შეძლოს გაარჩიოს ამოხსნადია თუ არა დასმული ამოცანა.

არსებობს ალგორითმების კლასიფიკაციის 2 მიდგომა: ამოცანის ტიპის და პროექტირების მეთოდის მიხედვით.

**ამოცანის ტიპის მიხედვით ალგორითმები შეიძლება დაიყოს შემდეგნაირად:**

- დახარისხება
- ძებნა
- სტრიქონების დამუშავება
- ამოცანები გრაფებზე
- კომბინატორული ამოცანები
- გეომეტრიული ამოცანები
- რიცხვითი ამოცანები

პროექტირების მეთოდის მიხედვით ალგორითმები შეიძლება დაიყოს

- უხეში ძალის“ მეთოდი
- დეკომპოზიციის მეთოდი
- ამოცანის ზომის შემცირების მეთოდი
- გარდაქმნის მეთოდი
- დროის და სივრცის კომპრომისის მეთოდი
- ხარბი მეთოდი
- დინამიკური დაპროგრამების მეთოდი
- დაბრუნებით ძებნის მეთოდი
- შტოების და საზღვრების მეთოდი

ჩვენს მიერ შესასწავლი ალგორითმების უმრავლესობას აქვს პოლინომიალური მუშაობის დრო, რაც ნიშნავს, რომ უარეს შემთხვევაში მუშაობის დრო არის $O(n^k)$. ასეთ ამოცანებს მიაკუთვნებენ P კლასს. უფრო ფორმალურად, P-ში შედის მხოლოდ **გადაწყვეტილების მიღების** ამოცანები (decision problems), ანუ ამოცანები, რომელზეც პასუხია „კი“ ან „არა“. მაგრამ ყველა ამოცანის ამოხსნა არ შეიძლება პოლინომიალურ დროში. არსებობს **გადაწყვეტილების მიღების** ამოცანები, რომელიც არ იხსნება საერთოდ, არც ერთი ალგორითმით. ასეთ ამოცანებს უწოდებენ **ამოუხსნადს** (undecidable). ამ ტიპის ამოცანის ცნობილი მაგალითია ალან ტიურინგის მიერ 1936 წელს დასმული გაჩერების ამოცანა(halting problem): მოცემული კომპიუტერული პროგრამისთვის და შემავალი მონაცემებისთვის, განსაზღვრეთ დაამთავრებს თუ არა პროგრამა მუშაობას, თუ ის იმუშავებს უსასრულოდ.

NP კლასში შედის გადაწყვეტილების მიღების ამოცანები, რომლებიც „ექვემდებარებიან შემოწმებას“ პოლინომიალურ დროში. სხვა სიტყვებით რომ ვთქვათ, თუ გვაქვს ამ ამოცანის რაიმე ამონახსნი, პოლინომიალურ დროში შეიძლება შევამოწმოთ მისი კორექტულობა. P კლასის ნებისმიერი ამოცანა ეკუთვნის NP კლასს: $P \subseteq NP$. მაგრამ ჯერჯერობით (კუკის მიერ, მისი დასმის მომენტიდან, 1971წ.) ღიად რჩება საკითხი: P კლასი NP კლასის მკაცრი ქვესიმრავლეა, თუ ეს ორი კლასი ემთხვევა ერთმანეთს?

ამოცანებისთვის, ამოცანათა კლასიდან, რომელიც ცნობილია NP-სრული კლასის სახელწოდებით, არ არის ნაპოვნი ამოხსნის ეფექტური ალგორითმები, მაგრამ ისიც არ არის დამტკიცებული, რომ ასეთი ალგორითმები არ არსებობს. მეორე მხრივ, NP-სრულ ამოცანებს აქვთ ერთი შესანიშნავი თვისება: თუ ეფექტური ალგორითმი არსებობს ერთი მაინც ამოცანისთვის ამ კლასიდან, მაშინ მისი ფორმულირება შეიძლება ამ კლასის ყველა დანარჩენი ამოცანისთვისაც.

NP-სრული ამოცანების განსაკუთრებული თვისება არის ის, რომ ზოგიერთი მათგანი, ერთი შეხედვით, ანალოგიურია ამოცანებისა, რომელთათვისაც არსებობს ალგორითმები პოლინომიალური მუშაობის დროით. მაგალითად, განვიხილოთ

წყვილები, რომლებშიც ერთი იხსნება პოლინომიალურ დროში, ხოლო მეორე - NP-სრული ამოცანაა.

*1.ეილერის და ჰამილტონის ციკლების:*

**ეილერის ციკლის პოვნის ამოცანა**: ვიპოვოთ $G=(V,E)$ ბმულ ორიენტირებულ გრაფში ციკლი, რომელიც ყველა წიბოს შემოივლის მხოლოდ ერთხელ, თუმცა დასაშვებია წვეროების რამდენჯერმე შემოვლა. ეილერის ციკლის არსებობის დადგენა, აგრეთვე, მისი შემადგენელი წიბოების პოვნა შეიძლება $O(E)$ დროში.

**ჰამილტონის ციკლის პოვნის ამოცანა**: ვიპოვოთ $G=(V,E)$ ორიენტირებულ გრაფში მარტივი ციკლი, რომელიც ყველა წვეროს შემოივლის. ჰამილტონის ციკლის არსებობის დადგენის ამოცანა არის NP-სრული.

*2.გრაფში ორ წვეროს შორის უმოკლესი გზის და ყველაზე გრძელი გზის პოვნის ამოცანები:*

$G=(V,E)$ ორიენტირებულ გრაფში ორ წვეროს შორის **უმოკლესი გზის პოვნის** ამოცანა შეიძლება ამოიხსნას $O(VE)$ დროში. ორ წვეროს შორის ყველაზე **გრძელი გზის პოვნის** ამოცანა რთულია. ამოცანა იმის შესახებ, შეიცავს თუ არა გრაფი მარტივ გზას, რომელშიც წიბოების რაოდენობა მოცემულ რიცხვზე ნაკლები არაა, NP-სრულია.

თუ რაიმე ამოცანისთვის დადგინდა, რომ იგი NP-სრულია, მაშინ პროგრამისტი უფრო ეფექტურად დახარჯავს დროს, თუ შექმნის ამ ამოცანისთვის მიახლოებით ალგორითმს ან ამოხსნის ამ ამოცანის მარტივ ვარიანტს, იმის მაგივრად, რომ ეძებოს სწრაფი ალგორითმი, რომელიც იძლევა ზუსტ ამონახსნს.

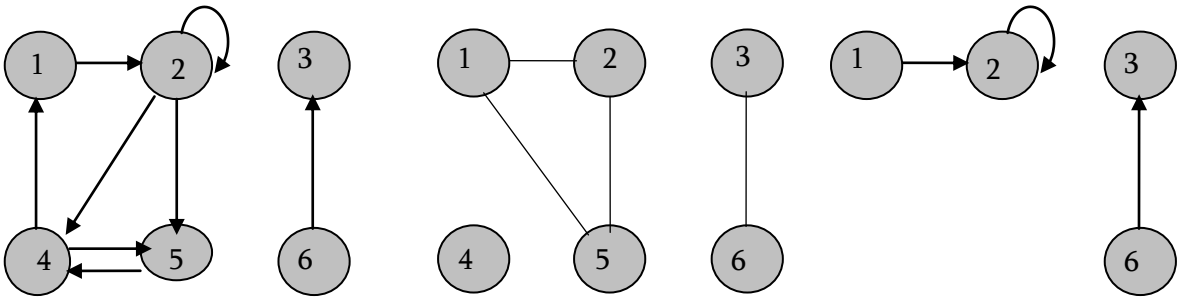
ალგორითმების აგების ამ კურსს ვიწყებთ გრაფებზე ალგორითმების შესწავლით, ამისთვის გავეცნოთ ძირითად ცნებებს და განსაზღვრებებს გრაფების შესახებ.

## ძირითადი ცნებები და განსაზღვრებები გრაფებზე

**ორიენტირებული გრაფი** (directed) $G$ განისაზღვრება როგორც $(V,E)$ წყვილი, სადაც $V$ სასრული სიმრავლეა, ხოლო $E$ წარმოადგენს $V$-ს ელემენტთა ბინარულ დამოკიდებულებას, ანუ $V\times V$ სიმრავლის ქვესიმრავლეა. ორიენტირებულ გრაფს ზოგჯერ **ორგრაფს** (digraph) უწოდებენ. $V$ სიმრავლეს უწოდებენ გრაფის **წვეროთა სიმრავლეს** (vertex set), ხოლო $E$-ს - **წიბოთა სიმრავლეს** (edge set). მათ ელემენტებს

შესაბამისად ეწოდებათ **წვერო** (vertex) და **წიბო** (edge). წიბოს, რომელიც წვეროს საკუთარ თავთან აერთებს, უწოდებენ **მარყუჟს(ციკლურ წიბოს)** .

**არაორიენტირებულ გრაფში** (undirected graph) $G=(V,E)$ წიბოთა $E$ სიმრავლე შედგება წვეროთა დაულაგებელი (unordered) წყვილებისაგან. წიბოს აღსანიშნავად გამოიყენება ჩანაწერი $(u,v)$. არაორიენტირებულ გრაფში $(u,v)$ და $(v,u)$ ერთი და იგივე წიბოს აღნიშნავს, ხოლო მარყუჟი არ შეიძლება არსებობდეს, რადგან წიბო ორი განსხვავებული წვეროსაგან უნდა შედგებოდეს.




ნახ 1.

ნახ.1 ა)-ზე მოცემულია ორიენტირებული გრაფი 6 წვეროთი და 8 წიბოთი

($V=\{1,2,3,4,5,6\}$ და $E=\{(1,2),(2,2),(2,4),(2,5),(4,1),(4,5),(5,4),(6,3)\}$), ნახ.1 ბ)-ზე – არაორიენტირებული გრაფი 6 წვეროთი და 4 წიბოთი ($V=\{1,2,3,4,5,6\}$ და $E=\{(1,2),(1,5),(2,5),(3,6)\}$) . $(u,v)$ წიბოს შესახებ ორიენტირებულ გრაფში იტყვიან, რომ იგი გამოდის $u$ წვეროდან და შედის $v$ წვეროში. ნახ.1 ა)-ზე 2 წვეროდან გამოდის სამი წიბო – (2,2), (2,4), (2,5) და 2 წვეროში შედის ორი წიბო – (1,2), (2,2). არაორიენტირებულ გრაფში $(u,v)$ წიბოს შესახებ იტყვიან, რომ იგი $u$ და $v$ წვეროების **ინციდენტურია** (incident).

თუ $G$ გრაფში არსებობს $(u,v)$ წიბო, იტყვიან, რომ $v$ წვერო $u$ **წვეროს მოსაზღვრეა** (is adjacent to $u$) არაორიენტირებულ გრაფებში მოსაზღვრეობა სიმეტრიული მიმართებაა, ხოლო ორიენტირებულ გრაფებისთვის ეს დებულება არ არის სამართლიანი. თუკი ორიენტირებულ გრაფში $v$ წვერო $u$ წვეროს მოსაზღვრეა, წერენ $u \to v$.

არაორიენტირებულ გრაფში წვეროს **ხარისხს** (degree) უწოდებენ ამ წვეროსადმი ინციდენტური წიბოების რაოდენობას. მაგალითად ნახ.1 ბ)-ზე 2 წვეროს ხარისხია 2. წვეროს, რომლის ხარისხიც არის 0, ეწოდება **იზოლირებული**(isolated) წვერო. წვეროს, რომლის ხარისხი არის 1 ეწოდება **დაკიდული** წვერო. ორიენტირებულ გრაფში განასხვავებენ **შემავალ** (in-degree) და **გამომავალ** (out-degree) ხარისხებს (შესაბამისად წვეროში შემავალი და გამომავალი წიბოების რაოდენობის მიხედვით) და მათ ჯამს უწოდებენ წვეროს ხარისხს. მაგალითად, ნახ. 1 ა)-ზე 2 წვეროს ხარისხია 5 (შემავალი ხარისხი – 2, გამომავალი ხარისხი – 3). წვეროს, რომლის გამომავალი

ხარისხი ნულია, ეწოდება **ჩასადენი**(sink), წვეროს, რომლის შემავალი ხარისხი ნულია, ეწოდება **წყარო**( source)

$k$ **სიგრძის გზა(მარშრუტი)** (path of length k) $u$ წვეროდან $v$ წვეროში განისაზღვრება როგორც წვეროთა $< v_0, v_1, v_2, \ldots, v_k >$ მიმდევრობა, სადაც $v_0 = u,\ v_k = v$ და $(v_{i-1}, v_i) \in E$ ნებისმიერი $i = 1\ ,..2k$ -სათვის. გზის სიგრძე განისაზღვრება მასში შემავალი წიბოების რაოდენობით. გზა **შეიცავს** (contains) $v_0, v_1, v_2, \ldots, v_k$ წვეროებს და $(v_0, v_1), (v_1, v_2)\ ,..\ (v_{k-1}, v_k)$ წიბოებს. ყოველთვის არსებობს ნული სიგძის გზა წვეროდან თავის თავში. $v_0$ წვეროს უწოდებენ გზის **დასაწყისს**, ხოლო $v_k$ წვეროს – გზის **ბოლოს** და ამბობენ, რომ გზა მიდის $v_0$ -დან $v_k$ -საკენ. თუ მოცემული $u$ და $u'$ წვეროებისთვის არსებობს $p$ გზა $u$ -დან $u'$ -ში, მაშინ ამბობენ, რომ $u'$ **მიღწევადია** $u$ -დან $p$ გზით ( $u'$ is reachable from $u$ via $p$ ). გზას ეწოდება **მარტივი** (simple), თუკი ყველა წვერო მასში განსხვავებულია. ნახ.1 ა)-ზე 3 სიგრძის მქონე გზა <1,2,5,4> მარტივია, ხოლო იმავე სიგრძის გზა <2,5,4,5> – არაა მარტივი.

$p = < v_0, v_1, v_2, \ldots, v_k >$ გზის **ქვეგზა** (subpath) ეწოდება ამ გზიდან მიყოლებით აღებულ $< v_i, v_{i+1}, \ldots, v_j >$ წვეროთა მიმდევრობას, რომლისთვისაც $0 \le i \le j \le k$. ორიენტირებულ გრაფში **ციკლი** (cycle) ეწოდება გზას, რომელშიც საწყისი წვერო ემთხვევა ბოლო წვეროს და რომელიც ერთ წიბოს მაინც შეიცავს. ციკლს ეწოდება **მარტივი**, თუკი მასში არ მეორდება არც ერთი წვერო პირველისა და ბოლოს გარდა. **მარყუჟი** წარმოადგენს ციკლს სიგრძით 1. აიგივებენ ციკლებს, რომლებიც განსხვავდებიან მხოლოდ წანაცვლებით ციკლის გასწვრივ. მაგ. ნახ.1 ა)-ზე გზები <1,2,4,1>, <2,4,1,2> და <4,1,2,4> წარმოადგენენ ერთსა და იმავე ციკლს. ამავე ნახაზზე ციკლი<2,2> შექმნილია ერთი წიბოთი. ორიენტირებულ გრაფს უწოდებენ **მარტივს**, თუკი იგი არ შეიცავს მარყუჟებს.

არაორიენტირებულ გრაფში $< v_0, v_1, v_2, \ldots, v_k >$ გზას ეწოდება **მარტივი ციკლი,** თუ $k \ge 3,\ v_0 = v_k$ და ყველა $v_1, v_2, \ldots, v_k$ წვერო განსხვავებულია. ნახ.1 ბ)-ზე მარტივი ციკლია <1,2,5,1>. გრაფს, რომელშიც არაა ციკლები, **აციკლური** (acyclic) ეწოდება.

არაორიენტირებულ გრაფს ეწოდება **ბმული** (connected), თუკი წვეროთა ნებისმიერი წყვილისათვის არსებობს გზა ერთიდან მეორეში. არაორიენტირებულ გრაფში გზის არსებობა ერთი წვეროდან მეორეში წარმოადგენს ეკვივალენტურ მიმართებას წვეროთა სიმრავლეზე. ეკვივალენტობის კლასებს ეწოდებათ გრაფის **ბმული კომპონენტები** (connected components). მაგ. ნახ.1 ბ)-ზე სამი ბმული კომპონენტია: {1,2,5}, {3,6} და {4}. არაორიენტირებული გრაფი ბმულია მაშინ და მხოლოდ მაშინ, როცა ის შედგება ერთადერთი ბმული კომპონენტისაგან.

ორიენტირებულ გრაფს ეწოდება **ძლიერად ბმული** (strongly connected), თუკი მისი ნებისმიერი წვეროდან მიღწევადია (ორიენტირებული გზებით) ნებისმიერი სხვა წვერო. ნებისმიერი ორიენტირებული გრაფი შეიძლება დაიყოს **ძლიერად ბმულ კომპონენტებად** (strongly connected components). ნახ. 1 ა)-ზე არის სამი ასეთი კომპონენტი {1,2,4,5}, {3} და {6}. შევნიშნოთ, რომ 3 და 6 წვეროები ერთად არ ჰქმნიან ძლიერად ბმულ კომპონენტს, რადგან არ არსებობს გზა 6-დან 3-ში.

$G=(V,E)$ და $G'=(V',E')$ გრაფებს ეწოდებათ **იზომორფულები** (isomorphic), თუკი არსებობს ურთიერთცალსახა შესაბამისობა $f:V\to V'$ მათი წვეროების სიმრავლეებს შორის, რომლის დროსაც $(u,v)\in E$ მაშინ და მხოლოდ მაშინ, როცა $(f(u),f(v))\in E'$. შეიძლება ითქვას, რომ იზომორფული გრაფები ეს ერთი და იგივე გრაფია, სადაც წვეროები სხვადასხვაგვარადაა დასახელებული.

$G'=(V',E')$ გრაფს $G=(V,E)$ ეწოდება გრაფის **ქვეგრაფი** (subgraph), თუ $V'\subseteq V$ და $E'\subseteq E$. თუ $G=(V,E)$ გრაფში ავირჩევთ $V'$ წვეროთა ნებისმიერ სიმრავლეს, მაშინ შეგვიძლია განვიხილოთ $G$-ს ქვეგრაფი, რომელიც შედგება ამ წვეროებისა და მათი შემაერთებელი წიბოებისაგან. ამ ქვეგრაფს უწოდებენ $G$ გრაფის **შეზღუდვას** $V'$ წვეროთა სიმრავლეზე. ნახ.1 ა)-ზე მოცემული გრაფის შეზღუდვა {1,2,3,6} წვეროთა სიმრავლეზე ნაჩვენებია ნახ.1გ)-ზე და შეიცავს სამ წიბოს (1,2), (2,2), (6,3).

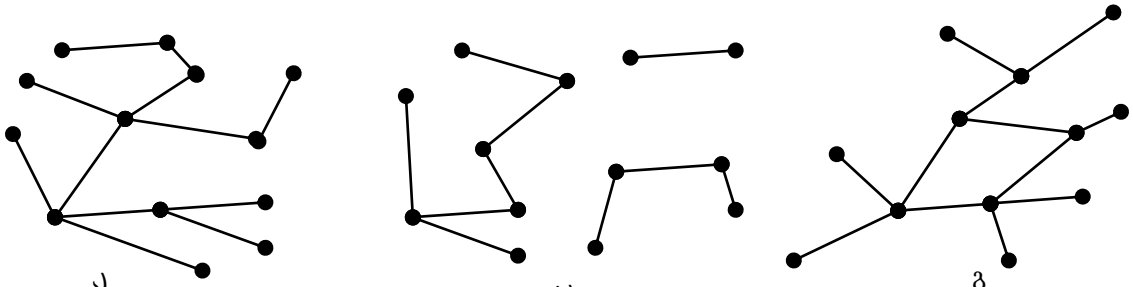
ნებისმიერი არაორიენტირებული გრაფისათვის შეიძლება განვიხილოთ მისი **ორიენტირებული ვარიანტი** (directed version), თუკი ყოველ $(u,v)$ არაორიენტირებულ წიბოს შევცვლით ორიენტირებული წიბოების $(u,v)$ და $(v,u)$ წყვილით, რომლებსაც ექნებათ ურთიერთსაწინააღმდეგო მიმართულებები. მეორე მხრივ, ნებისმიერი ორიენტირებული გრაფისათვის შეიძლება განვიხილოთ მისი **არაორიენტირებული ვარიანტი** (undirected version), თუკი ამოვშლით მარყუჟებს და $(u,v)$ და $(v,u)$ წიბოებს შევცვლით არაორიენტირებული $(u,v)$ წიბოთი. ორიენტირებულ გრაფში წვეროს **მეზობელი** (neighbor) ეწოდება ნებისმიერ წვეროს, რომელიც შეერთებულია მასთან ნებისმიერი მიმართულების წიბოთი, ე.ი. $v$ წვერო არის $u$-ს მეზობელი თუ $v$ არის $u$-ს მოსაზღვრე ან $u$ არის $v$-ს მოსაზღვრე. არაორიენტირებულ გრაფში კი ცნებები "მეზობელი" და "მოსაზღვრე" სინონიმებია.

**სრული** (complete) გრაფი ეწოდება არაორიენტირებულ გრაფს, რომელიც შეიცავს ყველა შესაძლებელ წიბოს წვეროთა მოცემული სიმრავლისათვის, ე.ი. ნებისმიერი წვერო შეერთებულია ყველა დანარჩენთან. არაორიენტირებულ $(V,E)$ გრაფს უწოდებენ **ორწილას** (bepartite), თუ $V$ წვეროთა სიმრავლე შეიძლება გავყოთ ისეთ ორ $V_1$ და $V_2$ ნაწილად, რომ ნებისმიერი წიბოს ბოლოები სხვადასხვა ნაწილში აღმოჩნდეს. $G=(V,E)$ გრაფის **დამატება** ეწოდება გრაფს, რომელსაც წვეროთა იგივე სიმრავლე აქვს, ხოლო ორი წვერო მოსაზღვრეა მაშინ და მხოლოდ მაშინ, როცა ისინი არ არიან მოსაზღვრე წვეროები $G$ გრაფში. აციკლურ არაორიენტირებულ გრაფს უწოდებენ **ტყეს** (forest), ხოლო ბმულ აციკლურ არაორიენტირებულ გრაფს უწოდებენ (**თავისუფალ**) **ხეს** (free tree). **ორიენტირებული აციკლური გრაფის(**direcred acyclic graph) აღსანიშნავად ზოგჯერ იყენებენ მის აბრევიატურას — **dag**.

თუ გრაფის წვეროების ხარისხების საშუალო მნიშვნელობა $2|E|/|V|$ ახლოსაა $|V|$-თან, ან, სხვა სიტყვებით რომ ვთქვათ, $|E|$ იმავე რიგისაა, რაც $|V|^2$, გრაფს ეწოდება **მკვრივი** (dence) გრაფი. **ხალვათი** (sparse) ეწოდება გრაფს, რომელშიც $|E|$ მკვეთრად მცირეა $|V|^2$-თან შედარებით. სხვა განმარტებით, **ხალვათი** ეწოდება გრაფს, რომლის დამატებაც მკვრივია.

# ხეები

როგორც ზემოთ აღვნიშნეთ, ბმულ აციკლურ არაორიენტირებულ გრაფს უწოდებენ **ხეს** ან **თავისუფალ ხეს** (free tree), ხოლო აციკლურ არაორიენტირებულ გრაფს უწოდებენ ტყეს (forest). ტყე შედგება ხეებისაგან, რომლებიც მის ბმულ კომპონენტებს წარმოადგენენ. ხეებისათვის ვარგისი ბევრი ალგორითმი გამოსადეგია ტყეებისთვისაც.




ნახ 2

ნახ.2 ა)-ზე გამოსახულია ხე, ნახ.2ბ)-ზე – ტყე (ის არ წარმოადგენს ხეს იმის გამო, რომ ბმული არაა), ხოლო ნახ.2გ) ნახაზზე მოცემული გრაფი არც ხეა და არც ტყე, რადგან იგი ციკლს შეიცავს.

***თეორემა 1.1 (ხეთა თვისებები).*** ვთქვათ, $G=(V,E)$ არაორიენტირებული გრაფია. მაშინ ტოლფასია შემდეგი თვისებები:

1) $G$ ხეა;

2) $G$-ს ნებისმიერი ორი წვეროსათვის არსებობს მათი შემაერთებელი ერთადერთი მარტივი გზა;

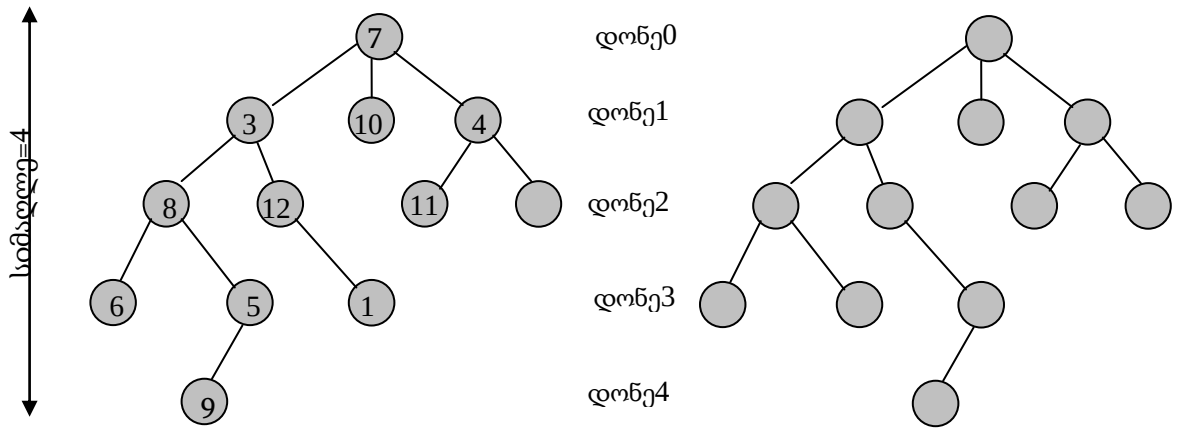
3) $G$ გრაფი ბმულია, მაგრამ კარგავს ბმულობას, თუ ამოვიღებთ ნებისმიერ წიბოს;

4) $G$ გრაფი ბმულია და $|E|=|V|-1$;

5) $G$ გრაფი აციკლურია და $|E|=|V|-1$;

6) $G$ გრაფი აციკლურია, მაგრამ ნებისმიერი წიბოს დამატებით მასში ჩნდება ციკლი.

**ხე ფესვით** (rooted tree) მიიღება მაშინ, როცა ბმულ აციკლურ არაორიენტირებულ გრაფში გამოყოფილია ერთ-ერთი წვერო, რომელსაც **ფესვს** (root) უწოდებენ. ხშირად, ფესვის მქონე ხის წვეროებს **კვანძებს**(nodes) უწოდებენ. ნახ.3-ზე ნაჩვენებია ფესვის მქონე ხე 12 წვეროთი და ფესვით 7.




ნახ. 3.

ვთქვათ $x$ არის $r$ ფესვის მქონე $T$ ხის კვანძი. ნებისმიერ $y$ კვანძს, რომელიც მდებარეობს (ერთადერთ) გზაზე $r$-დან $x$-ში, უწოდებენ $x$ კვანძის **წინაპარს** (ancestor). თუ $y$ არის $x$ -ის წინაპარი, მაშინ $x$-ს უწოდებენ $y$-ის **შთამომავალს**(descendant). ყოველი კვანძი შეიძლება ჩაითვალოს საკუთარ წინაპრად ან შთამომავლად. თუ $y$ არის $x$ -ის წინაპარი და $x \neq y$ ,მაშინ $y$-ს უწოდებენ **საკუთარ წინაპარს**(proper ancestor), ხოლო $x$-ს უწოდებენ **საკუთარ შთამომავალს**(proper descendant).

ყოველი $x$ კვანძისათვის შეიძლება განვიხილოთ ხე, რომელიც $x$-ის ყველა შთამომავლისაგან შედგება და $x$ ითვლება ამ ხის ფესვად. მას უწოდებენ **ქვეხეს** $x$ **ფესვით**. მაგალითად ნახ.3ა)- ზე ქვეხე ფესვით 8 შეიცავს წვეროებს 8, 6, 5 და 9.

თუ $(y, x)$ უკანასკნელი წიბოა $T$ ხეში $r$ ფესვიდან $x$-საკენ მიმავალ გზაზე, მაშინ $y$-ს უწოდებენ $x$-ის **მშობელს** (parent), ხოლო $x$-ს უწოდებენ $y$-ის **შვილს** (child). ფესვი ერთადერთი კვანძია, რომელსაც მშობელი არ ჰყავს. კვანძებს, რომელთაც საერთო მშობელი ჰყავს, უწოდებენ **დედმამიშვილებს** (siblings, რუსულ ლიტერატურაში — ძმები). ფესვის მქონე ხის კვანძს, რომელსაც არა ჰყავს შვილები, უწოდებენ **ფოთოლს** (leaf, external node). წვეროებს, რომელთაც ჰყავთ შვილები, უწოდებენ **შინაგანს** (internal). ფესვის მქონე ხის კვანძის შვილების რაოდენობას უწოდებენ მის **ხარისხს** (degree). გზის სიგრძეს ფესვიდან ნებისმიერ $x$ კვანძამდე უწოდებენ $x$ წვეროს **დონეს** (depth). ხის კვანძის **სიმაღლე** (height) არის წიბოების რაოდენობა ყველაზე გრძელ გზაზე ამ კვანძიდან მის შთამომავალ ფოთლამდე.ხის სიგრძედ ითვლება მისი ფესვის სიგრძე.

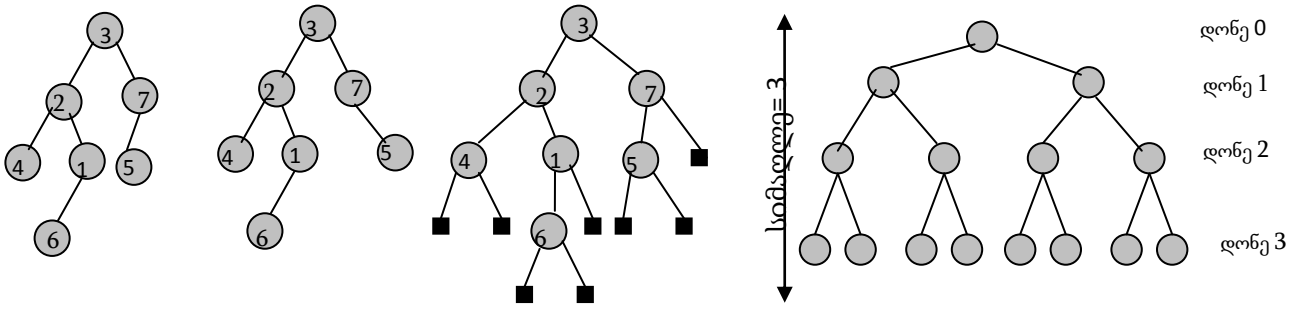
**დალაგებული ხე** (ordered tree) ეწოდება ფესვის მქონე ხეს, რომელსაც დამატებითი სტრუქტურა გააჩნია: ყოველი კვანძისათვის მისი შვილების სიმრავლე დალაგებულია (ცნობილია თუ რომელია წვეროს პირველი შვილი, მეორე შვილი და

ა.შ.). ნახ.3-ზე გამოსახული ორ ხეს ერთი და იგივე ფესვი აქვს და ისინი განსხვავდებიან მხოლოდ შვილების დალაგებით.

**ორობითი ხე** (binary tree) ყველაზე მარტივი სახით განისაზღვრება რეკურსიულად, როგორც კვანძთა სასრული სიმრავლე, რომელიც: ან ცარიელია (არ შეიცავს კვანძებს), ან გაყოფილია სამ არაგადამკვეთ ნაწილად: წვერო რომელსაც ეწოდება **ფესვი** (root), ორობითი ხე, რომელსაც ეწოდება ფესვის **მარცხენა ქვეხე** (left subtree) და ორობითი ხე, რომელსაც ეწოდება ფესვის **მარჯვენა ქვეხე** (right subtree)

ორობით ხეს, რომელიც არ შეიცავს კვანძებს, ეწოდება **ცარიელი** (empty). ზოგჯერ მას აღნიშნავენ NIL-ით. თუ მარცხენა ქვეხე არაცარიელია, მაშინ მის ფესვს უწოდებენ მთლიანი ხის ფესვის **მარცხენა შვილს** (left child), შესაბამისად განისაზღვრება **მარჯვენა შვილიც** (right child). ორობითი ხის მაგალითი მოცემულია ნახ. 4-ზე.

არასწორი იქნებოდა განგვესაზღვრა ორობითი ხე, როგორც დალაგებული ხე, რომელშიც თითოეული კვანძის ხარისხი არ აღემატება 2-ს. ამის მიზეზია ის, რომ ორობით ხეში მნიშვნელობა აქვს როგორია ეკვანძისრთადერთი შვილი – მარცხენა თუ მარჯვენა, ხოლო დალაგებული ხისათვის ასეთი განსხვავება არ არსებობს. ნახ.4 ა)-ზე და ნახ.4 ბ)-ზე ნაჩვენები ორობითი ხეები განსხვავდებიან, რადგან ა)-ზე 5 არის 7-ის მარცხენა შვილი, ხოლო ბ)-ზე – მარჯვენა. როგორც დალაგებული ხეები ისინი ერთნაირები არიან.




ა) ბ) გ) დ)

ნახ. 4.

ხშირად ორობით ხეზე ცარიელ ადგილებს ავსებენ ფიქტიური ფოთლებით, მიიღება **სრულად ორობითი ხე** (full binary tree). ამის შემდეგ ყველა კვანძი ან ფოთოლია, ან აქვს ხარისხი 2, 1-ს ტოლი ხარისხის მქონე კვანძები ასეთ ხეში არ არის. ეს გარდაქმნა ნაჩვენებია ნახ.4 გ) -ზე.

**სრული** $k$**-ობითი ხე** (complete k-ary tree) ეწოდება $k$-ობით ხეს, რომელშიც ყველა ფოთოლს აქვს ერთნაირი დონე და ყველა შინაგან კვანძს აქვს ხარისხი $k$. ასეთ შემთხვევაში ხის სტრუქტურა მთლიანად განისაზღვრება მისი სიმაღლით. ნახ.4 დ)-

ზე გამოსახულია სრული ორობითი ხე სიმაღლით 3. ფესვი ყოველთვის არის 0 დონის ერთადერთი კვანძი, მისი $k$ შვილი არის 1 დონის მქონე კვანძები, რომელთა $k^2$ შვილი წარმოადგენენ 2 დონის კვანძებს და ა.შ. $h$ დონეზე გვექნება $k^h$ ფოთოლი. $h$ სიმაღლის სრული $k$ -ობითი ხის შინაგანი კვანძების რაოდენობის ტოლია

$$1+k+k^2+\ldots+k^{h-1}=\frac{k^h-1}{k-1}$$

კერძოდ, სრული ორობითი ხის შინაგანი კვანძების რაოდენობაა $2^h-1$.

## გრაფთა წარმოდგენა

გრაფის წარმოდგენისთვის გამოიყენება ორი ძირითადი სტრუქტურა:

ა) **მოსაზღვრე წვეროების სია** (adjacency-list representation);
ბ) **მოსაზღვრეობის მატრიცა** (adjacency-matrix representation).

$G=(V,E)$ გრაფის წარმოდგენა მოსაზღვრე წვეროების სიებით იყენებს $|V|$ ცალი წვეროსგან შემდგარ ბმულ სიას. მოსაზღვრე წვეროთა ყველა სიის სიგრძეთა ჯამი ორიენტირებული გრაფისათვის წიბოთა რაოდენობის ტოლია, ხოლო არაორიენტირებული გრაფისათვის — წიბოთა გაორმაგებული რაოდენობის ტოლი, რადგან $(u,v)$ წიბო წარმოშობს ელემენტს როგორც $u$, ასევე $v$ წვეროს მოსაზღვრე წვეროთა სიაში. ორივე შემთხვევაში მეხსიერების საჭირო მოცულობაა $O(V+E)$. ამ წარმოდგენით მოსახერხებელია წონადი გრაფების (weighted graphs) შენახვაც, სადაც ყოველ წიბოს შეესაბამება ნამდვილი რიცხვი – ამ წიბოს **წონა** (weight), ანუ მოცემულია **წონის ფუნქცია** (weight function) $w:E\to R$. ამ შემთხვევაში მოსახერხებელია $(u,v)\in E$ წიბოს $w(u,v)$ წონის შენახვა $u$ წვეროსთან ერთად წვეროს მოსაზღვრე წვეროთა სიაში. თუმცა ამ წარმოდგენას აქვს მნიშვნელოვანი ნაკლი: $u$ -დან $v$ -ში წიბოს არსებობის დასადგენად, საჭიროა გადავამოწმოთ მთლიანი სია, მასში $v$ -ს მოსაძებნად. ძებნას შესაძლოა თავი ავარიდოთ, თუ გამოვიყენებთ მოსაზღვრეობის მატრიცას, თუმცა ამ შემთხვევაში მეტი მანქანური მეხსიერებაა საჭირო.

მოსაზღვრეობის მატრიცის გამოყენებისას უნდა გადავნომროთ $G=(V,E)$ გრაფის წვეროები $1,2,\ldots,|V|$ რიცხვებით და განვიხილოთ $|V|\times|V|$ ზომის $A=(a_{ij})$ მატრიცა, სადაც $a_{ij}=1$, თუ $(i,j)\in E$ და $a_{ij}=0$, წინააღმდეგ შემთხვევაში. მეხსიერების საჭირო მოცულობაა $O(V^2)$. არაორიენტირებული გრაფისათვის მოსაზღვრეობის მატრიცა სიმეტრიულია მთავარი დიაგონალის მიმართ და

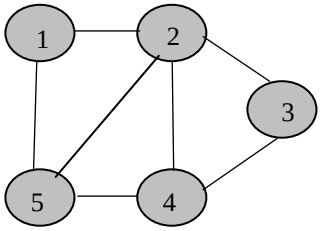
ამიტომ საკმარისია შევინახოთ მხოლოდ მთავარი დიაგონალი და მის ზემოთ მყოფი ელემენტები, რაც თითქმის ორჯერ ამცირებს გამოყენებულ მეხსიერებას. წონადი გრაფების შენახვა პრობლემა არც ამ მეთოდისთვისაა — $(u,v)$ წიბოს $w(u,v)$ წონა მატრიცაში თავსდება $u$ სტრიქონისა და $v$ სვეტის გადაკვეთაზე, ხოლო წიბოს არარსებობის შემთხვევაში მატრიცის შესაბამისი ელემენტი მიიღებს ცარიელ მნიშვნელობას *NIL* (ზოგ ამოცანაში შეიძლება გამოვიყენოთ 0 ან $\infty$ ).

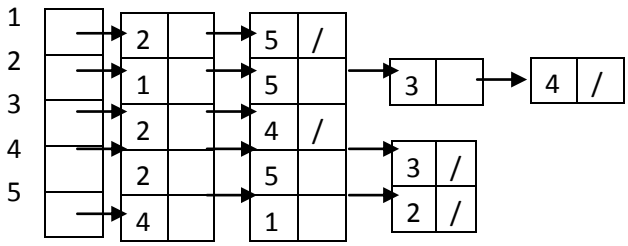
ნახ.5-ზე მოცემულია გრაფთა წარმოდგენის ორივე მეთოდი როგორც არაორიენტირებული, ისევე ორიენტირებული გრაფებისათვის.

თითოეული წარმოდგენის გამოყენების შესახებ გადაწყვეტილების მიღება დამოკიდებულია: გრაფის განზომილებაზე (მოსაზღვრეობის მატრიცით წარმოდგენა მოსახერხებელია პატარა ან მკვრივი გრაფებისათვის), ალგორითმზე (მოსაზღვრეობის მატრიცით წარმოდგენა უმჯობესია ალგორითმებისთვის, რომლებიც პასუხობენ კითხვაზე, მაგ, არის თუ არა $(u,v)$ წიბო $G$ გრაფში?) იმაზე, ხალვათია თუ მკვრივი გრაფი (თუ გრაფი მკვრივია, მატრიცით წარმოდგენა უფრო მოსახერხებელია, რადგან ყველა შემთხვევაში $O(V^2)$ მეხსიერების გამოყენება მოგვიწევს).

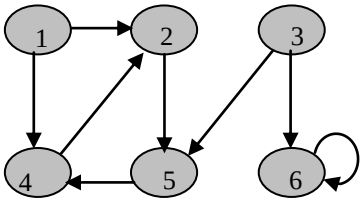
თუკი მანქანური მეხსიერება საკმარისია, სჯობს გრაფი აღვწეროთ მოსაზღვრეობის მატრიცის საშუალებით, რადგან უმეტეს შემთხვევაში მასთან მუშაობა უფრო მოსახერხებელია. ამას გარდა, თუკი გრაფი წონადი არ არის, მოსაზღვრეობის მატრიცის ელემენტები შესაძლოა განიხილოთ როგორც ბიტები, რომლებიც შეგვიძლია გავაერთიანოთ მანქანურ სიტყვებში — ეს იძლევა მეხსიერების მნიშვნელოვან ეკონომიას.

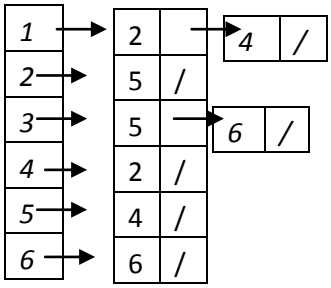
გრაფისთვის ეფექტური ალგორითმის შერჩევის დროს, ერთ-ერთი მთავარი ფაქტორია ინფორმაცია იმის შესახებ ხალვათია თუ მკვრივი გრაფი. მაგ; თუ რაიმე ამოცანის გადასაწყვეტად შეგვიძლია შევიმუშაოთ 2 ალგორითმი: პირველს სჭირდება $|V|^2$ , ხოლო მეორეს - $|E|\lg|E|$ ბიჯი. ეს ფორმულები გვიჩვენებს, რომ პირველი ალგორითმი უფრო გამოდგება მკვრივი გრაფებისთვის, ხოლო მეორე - ხალვათისთვის. მაგ, განვიხილოთ მკვრივი გრაფი $|E|=10^6$ $|V|=10^3$ $|V|^2$ 20-ჯერ მეტია $|E|\lg|E|$ - ზე. მეორე მხრივ, ხალვათი გრაფისთვის, $|E|=10^6$ $|V|'=10^6$' ალგორითმი $|E|\lg|E|$ სირთულით მილიონის რიგით აჯობებს $|V|^2$ სირთულის ალგორითმს.




| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 1 | 1 | 1 |
| 3 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 1 | 0 | 1 | 0 |




| | | | | | | |
|---|---|---|---|---|---|---|
| *1* | *0* | *1* | *0* | *1* | *0* | *0* |
| *2* | *0* | *0* | *0* | *0* | *1* | *0* |
| *3* | *0* | *0* | *0* | *0* | *1* | *1* |
| *4* | *0* | *1* | *0* | *0* | *0* | *0* |
| *5* | *0* | *0* | *0* | *1* | *0* | *0* |
| *6* | *0* | *0* | *0* | *0* | *0* | *1* |

ნახ. 5.

## სავარჯიშოები

1. ოთხი ადამიანი ღამით მოძრაობს გზაზე ერთი მიმართულებით. მათ უნდა გადაიარონ ხიდზე და აქვთ მხოლოდ 1 ფარანი. ხიდზე ერთდროულად შეუძლია გაიაროს არაუმეტეს 2 ადამიანმა (ან ერთმა, ან ორმა) და ერთ-ერთს აუცილებლად უნდა ეჭიროს ფარანი. არ შეიძლება ფარანის ერთი ნაპირიდან მეორეზე გადაგდება, შეიძლება მისი მხოლოდ ხიდით გადატანა. თითოეული ადამიანი ხიდის გადასვლას უნდება: პირველი-1წთ., მეორე-2წთ., მესამე-5წთ.,მეოთხე-10წთ. თუ ხიდზე გადადის 2 ადამიანი,ისინი მოძრაობენ უფრო ნელის სიჩქარით. რა თანმიმდევრობით უნდა გადაიარონ ხიდი, რომ ამისთვის დასჭირდეთ ზუსტად 17 წუთი.

2.მოიყვანეთ შემდეგი გრაფების მაგალითები ან აჩვენეთ, რომ ასეთი გრაფები არ არსებობს:

1) გრაფი, რომელსაც აქვს ჰამილტონის ციკლი, მაგრამ არ აქვს ეილერის ციკლი.
2) გრაფი, რომელსაც აქვს ეილერის ციკლი, მაგრამ არ აქვს ჰამილტონის ციკლი.
3) გრაფი, რომელსაც აქვს როგორც ეილერის, ისე ჰამილტონის ციკლი.
4) გრაფი, რომელსაც აქვს ყველა წვეროს შემცველი ციკლი, მაგრამ არ აქვს არც ჰამილტონის, არც ეილერის ციკლი.

3. დაამტკიცეთ, რომ $V$ წვეროს შემცველი არაორიენტირებული გრაფი შეიცავს არაუმეტეს $|V|(|V|-1)/2$ წიბოს.

4. დაამტკიცეთ, რომ ნებისმიერი $G=(V,E)$ არაორიენტირებული ბმული გრაფისთვის სრულდება $|E|\geq|V|-1$.

5. დაამტკიცეთ, რომ ნებისმიერი $G=(V,E)$ აციკლური ბმული არაორიენტირებული გრაფი შეიცავს $|V|-1$ წიბოს.

6. ა) გრაფის მოსაზღვრეობის მატრიცის რა თვისებები მეტყველებს იმაზე, რომ:

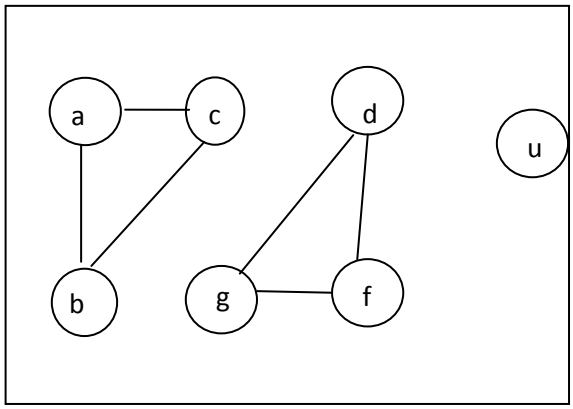
1) გრაფი სრულია.
2) გრაფი შეიცავს მარყუჟს.
3) გრაფი შეიცავს იზოლირებულ წვეროს.

ბ) გაეცით პასუხები ა)-ში დასმულ შეკითხვებზე გრაფის მოსაზღვრე წვეროთა სიით წარმოდგენის შემთხვევისთვის.

7. შეამოწმეთ, რომ მიმართება „მიღწევადია -დან“ არაორიენტირებულ გრაფში არის ეკვივალენტობის მიმართება გრაფის წვეროთა სიმრავლეზე.

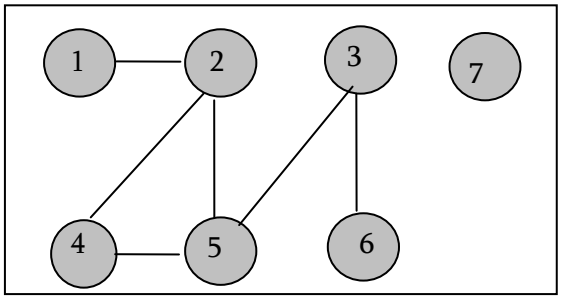
8. ეკვივალენტობის მიმართების 3 თვისებიდან რომელი ირღვევა „მიღწევადია -დან“ მიმართებისთვის ორიენტირებულ გრაფში?

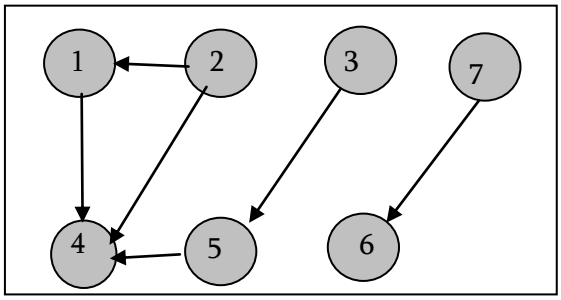
9. იპოვეთ ეკვივალენტობის კლასები შემდეგი გრაფისათვის:




10. რამდენი ბმული კომპონენტი აქვს ხეს?

11. შემდეგი გრაფებისთვის ააგეთ მოსაზღვრეობის მატრიცა და მოსაზღვრე წვეროთა სია.



## ლიტერატურა


1. Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн. АЛГОРИТМЫ ПОСТРОЕНИЕ и АНАЛИЗ,второе издание, Москва Санкт-Петербург Киев, 2010 г.
2. А. Левитин АЛГОРИТМЫ Введение в разработку и анализ. Москва Санкт-Петербург Киев, 2006 г.