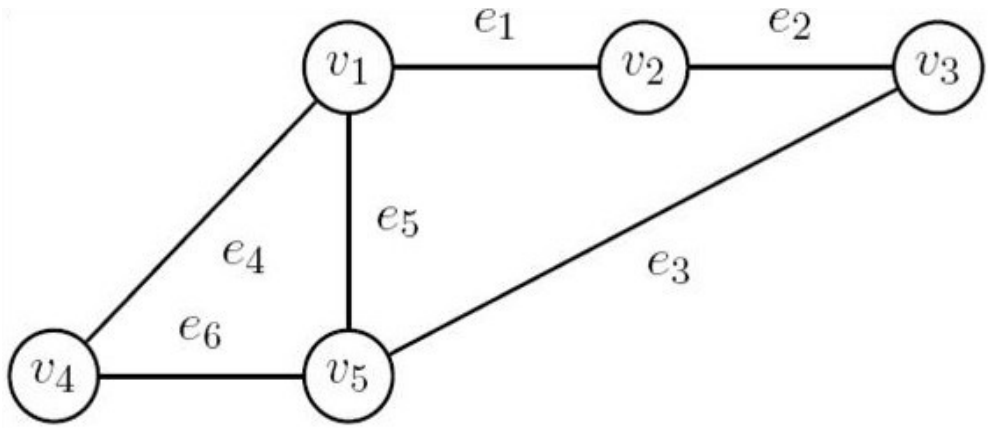
Вовед во теорија на графови

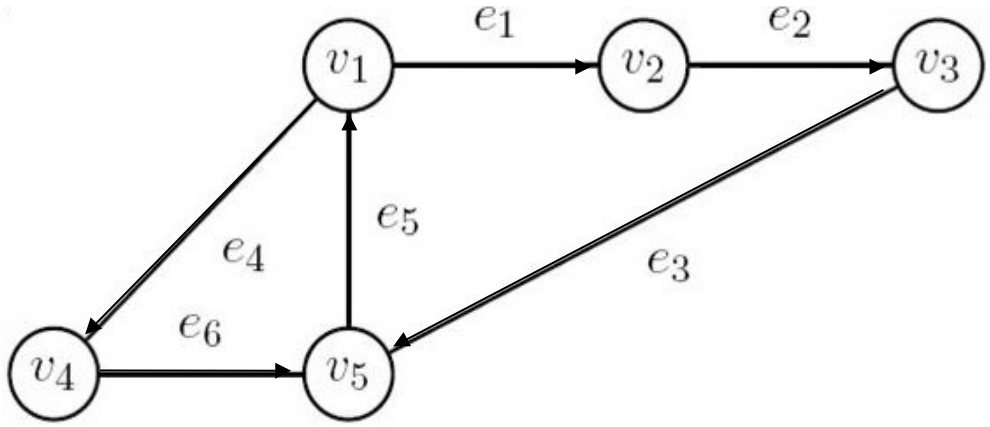
Граф се дефинира како двојка (V,E), каде што V е множество темиња, а E е множество ребра и притоа на секое ребро му е придружен пар темиња преку т.н. функција на соседство E -> V x V. (Секое ребро е определено од точно две темиња, кои нужно не мора да бидат различни). Просто кажано, графовите се во суштина множество од темиња заедно со множество од линии (ребра) кои ги поврзуваат тие темиња.

Ако на ребрата им зададеме и некаква насока, тогаш ребрата ги нарекуваме **ориентирани ребра**, а графот го нарекуваме **ориентиран граф** или **скратено оргграф**. Доколку ребрата немаат никаква насока, тогаш ребрата ги нарекуваме **неориентирани ребра**, а графот со вакви ребра се нарекува **неориентиран граф.**

Пример 1: неориентиран граф



Пример 2: ориентиран граф



Нека $\mathbf{V_1}$ и $\mathbf{V_2}$ се две темиња од ориентиран граф. Ако постои ребро помеѓу V_1 и V_2, и стрелката се движи од V_1 кон V_2, тогаш реброто го запишуваме како $\boldsymbol{e = (V_1 , V_2)}$ при што $\mathbf{V_1}$ се нарекува **почеток на реброто** $\boldsymbol{e}$ (претходник на $\mathbf{V_2}$), а $\mathbf{V_2}$ се нарекува **крај на реброто** $\boldsymbol{e}$ (следбеник на реброто $\mathbf{V_1}$). Ако има случај $\boldsymbol{e = (V_1, V_1)}$, тоа ребро се нарекува **лупа**.

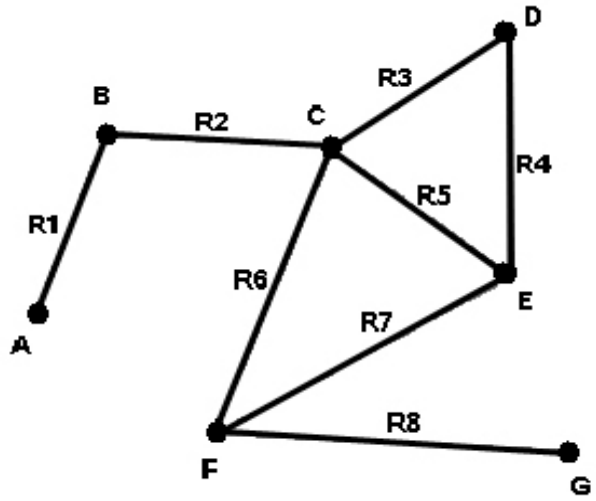
Во графот може да постојат **паралелни ребра** (тоа се повеќе ребра помеѓу две исти темиња).

Графот G=(V,E) се нарекува **прост граф**, ако E не содржи лупи и паралелни ребра.

Бројот на ребра чии што еден крај е темето V_i, се нарекува **степен** на тоа теме V_i. Степен на едно теме $\mathbf{V_i}$ најчесто се обележува со $\mathbf{d\ (V_i)}$.

Во практиката, многу често се сретнуваат графови кај кои на секое ребро му е придружен реален број што може да претставува: растојание, цена, проточност, профит итн. Таквите графови ги нарекуваме **тежински графови**.
Под **маршрута** во граф подразбираме конечна наизменична низа на темиња и ребра што почнува и завршува со теме. Должина на маршрутата се одредува според бројот на поминатите ребра.
Маршрутата во која сите ребра се различни ја нарекуваме **верига**.
Маршрутата во која сите темиња се различни ќе ја нарекуваме **пат или патека**. Обично кај патот се дозволува првото и последното теме да се еднакви и тогаш тој е **затворен пат или циклус.**

Пример 3:



<u>Маршрути и патишта</u>

Примери за маршрути :
A – **R1** – B – **R2** – C – **R3** – D – **R4** – E – **R4** –D
F – **R7** – E – **R5** – C – **R6** – F – **R7** – E – **R5** – C

Пример за верига:
B – **R2** – C – **R5** – E – **R7** – F – **R6** – C – **R3** – D

Пример за пат:
A – **R1** – B – **R2** – C – **R6** – F – **R8** – G

Пример за циклус:
C – **R5** – E – **R7** – F – **R6** – C

Множеството темиња може да се подели на подмножества темиња меѓу кои постои пат. Овие подмножества се нарекуваат **сврзани компоненти на графот G**.
Неориентираниот граф G е сврзан ако постои точно една сврзана компонента т.е. ако било кои две темиња се поврзани со пат.
Сврзаните графови што немаат циклус се нарекуваат **дрва или стебла**.
Планарен се нарекува оној граф што може да биде графички претставен во рамнината така што ребрата претставени со произволни линии немаат пресек.

Претставување на графови

Графовите можат да се претстават на повеќе начини и тоа: може да се даде листа од темиња и да се наредат ребрата, може да се претстават со матрица, или пак со поврзани листи.
Претставувањето со матрици може да биде:

1) **<u>Со матрица на соседство</u>** (тоа ќе биде квадратна матрица Anxn –> n е број на темиња во графот), при што ***A*[*i, j*] = 1 ако постои врска меѓу теме i и теме j** и ***A*[*i, j*] = 0 ако не постои врска меѓу теме i и теме j**. Предноста овде е што веднаш можеме да провериме дали постои врска мегу 2 темиња или не.

Матрица на соседство за графот од Пример 1:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Матрица на соседство за графот од Пример 2:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2) <u>Со матрица Anxm (n е број на темиња, а m е број на ребра), при што:</u>

- **<u>кај неориентирани графови</u>**

$$A[i,j] = \begin{cases} 1, & \text{ако ребро j е поврзано со теме i} \\ 0, & \text{ако ребро j не е поврзано со теме i} \end{cases}$$

- **<u>кај ориентирани графови</u>**

$$A[i,j] = \begin{cases} 1 & \text{, ако темето i е почеток на ребро j} \\ -1 & \text{, ако темето i е крај на ребро j} \\ 0 & \text{, во останати случаеви} \end{cases}$$

Матрица A_{5x6} за неориентиран граф од Пример 1:

$$A = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{matrix}$$

Матрица A_{5x6} за ориентиран граф од Пример 2:

$$A = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & -1 \end{bmatrix} \end{array} \begin{matrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{matrix}$$

Алгоритми за пребарување низ графови

1) Пребарување по ширина BFS (Breadth-First Search)

Кај овој алгоритам се тргнува од едно теме кое се избира како извор (или евентуално е дадено дека е извор-почетно теме) и се бараат сите темиња до кои може да се стигне од тоа теме. При тоа, темињата до кои сме стигнале ги боиме сиво, додека темињата од кои продолжуваме натаму ги боиме црни доколку нема повеќе не посетени соседни темиња на. Кога едно теме е обоено црно, веќе знаеме каде се може да се стигне од тоа теме и веќе не правиме никаква проверка со истото.

Постапките на алгоритмот се:

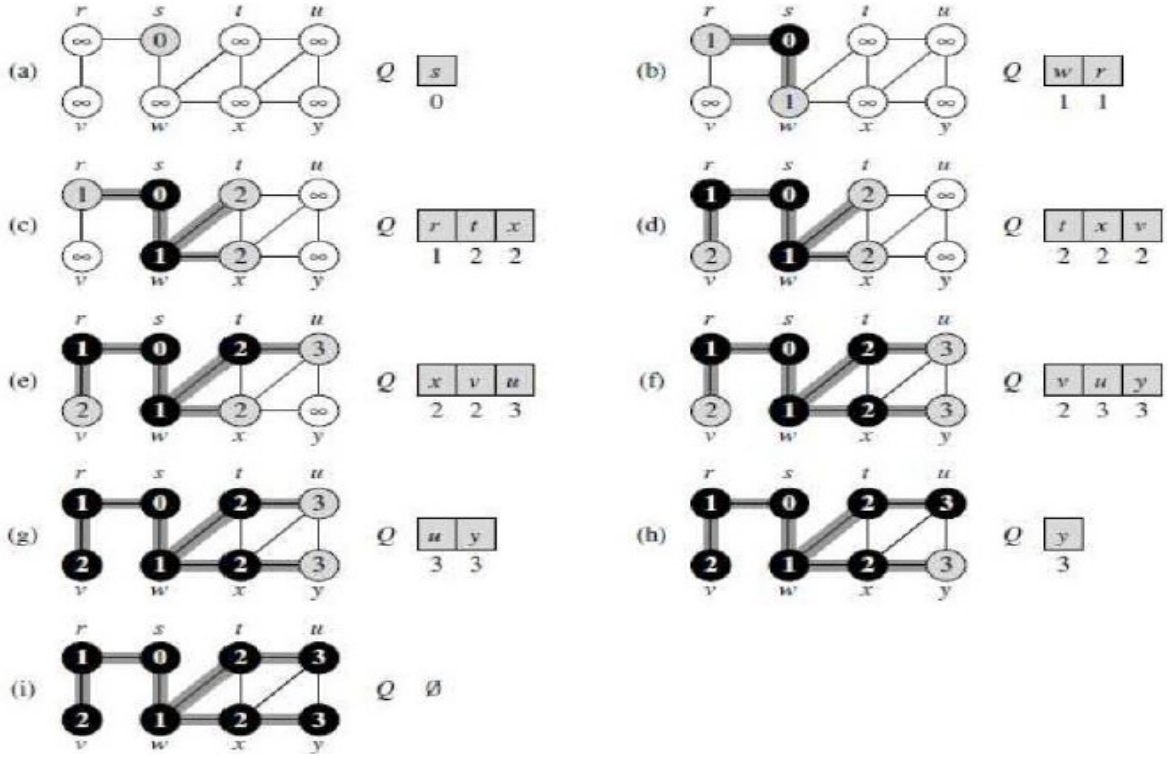
1) сите темиња се поставуваат да бидат бели.
2) потоа изворот го боиме сиво и гледаме каде се може да се стигне со еден чекор од тоа теме. Изворот добива вредност 0, додека сите темиња до кои стигнуваме со еден чекор добиваат вредност 1. Реброто кое при тоа сме го употребиле го додаваме во нова структура, (во нова матрица која првично е поставена целосно на 0, ставаме 1 на местото на тоа ребро)
3) кога ова ќе го направиме со сите темиња до кои можеме да стигнеме со еден чекор од изворот, изворот го боиме црно.
4) понатаму постапката продолжува на следниот начин. Земаме сиво обоено теме, со најмала вредност, на пример темето v, на кое му е придружена вредност i. Гледаме во кои се непосетени темиња можеме да стигнеме од тоа теме во еден чекор. И повторно, сите темиња до кои стигаме со еден чекор од v се бојат во сива боја и добиваат вредност i+1. Реброто кое при тоа сме го употребиле го додаваме во нова

структура, (во новата матрица ставаме 1 на местото на тоа ребро) Кога ова ќе го направиме со сите темиња до кои можеме да стигнеме со еден чекор од темето v, тоа теме v го боиме со црна боја.

Овој алгоритам можеме да го искористиме за да провериме дали постои пат меѓу две темиња и ако постои кој е најкраткиот пат.

Временска сложеност на алгоритмот: Прво сите темиња се поставуваат на 0 - временската сложеност е O(V), а потоа се поминува секое ребро –временската сложеност е O (E) , што значи вкупно O= O(V)+O(E)

Пример за BFS



2) Пребарување на графови по длабочина (Depth-First Search)

Кај овој алгоритам се почнува од некое означено теме како почетното теме(извор). Попрецизно, откако ќе го одбереме почетното теме потоа се посетува теме поврзано со почетното и се продолжува со посетување на сите со него поврзани темиња. Доколку почетното теме е поврзано со повеќе темиња, потоа се посетуваат и тие како и темињата поврзани со нив. Притоа графот може да содржи циклуси, но секое теме мора барем еднаш да биде посетено. Затоа се памтат темињата што биле посетени. Оваа постапка се повторува се додека не бидат посетени сите темиња до кои може да се пристапи од соодветниот извор (почетно теме). Ако останат некои непосетени темиња едно од нив се избира како нов извор и целата постапка се повторува од тој извор.

Целата постапа се повторува се додека сите темиња не се посетени.

За да го знаеме стадиумот на посетеност на темињата тие се бојат: бело кога се непосетени, сиво кога се открени со пребарувањето и црни кога нивната листа на соседство е помината т.е. немаат ни еден сосед кој што не е посетен.

DFS на секое теме му придружува време на пристап (***timestamps***). Секое теме има по два податоци за времето на пристап, т.е. во првото време ***d*[*v*]** се зачувува времето во кое прв пат се пристапува во темето v кога тоа е означено како сиво, а во второто време ***f* [*v*]** се

зачувува податокот за тоа време кога сите соседи на темето *v* се посетени и тоа ќе стане црно.

Овие времиња се цели броеви во рангот од 1 до 2*|V| каде V е множеството на темиња. Бидејќи постои еден број за пристапот до темето а друг за завршувањето со посетување на неговите соседи за секое теме v важи *d*[*v*] < *f* [*v*] . Темето *v* е бело пред моментот *d*[*v*], сиво во периодот од *d*[*v*] до *f* [*v*] и црно после периодот *f* [*v*].

Овој алгоритам може да се користи за пронаоѓање на било какво решение меѓу поврзани елементи (темиња) (каде што не е битен најкраткиот пат) на пример:

- дали графот има циклуси
- при пронаоѓање на сите можни потези на одредена шаховска фигура која се наоѓа на одредено почетно место во шаховската табла
- решавање на загатки со само едно решение, како на пр. лавиринти
- поврзаност на сите градови во некоја мапа (пр. Google map)
- тополошко линеарно подредување на темињата во ориентирани нецикличен граф
- да се најди секоја директна врска меѓу два темиња во ориентиран граф

Временска сложеност на алгоритмот: Прво сите темиња имаат вредности 0 - временска сложеност O(V), а потоа се поминува секое ребро – временска сложеност O (E), или вкупно O= O(V)+O(E)

Пример за DFS