**Grafo** é uma tripla ordenada (N, A, g), onde:

**N** é um conjunto não vazio de vértices (nós ou nodos);

**A** é um conjunto de arestas (arcos);

**g** é uma função que associa cada aresta *a* a um par não ordenado x-y de vértices chamados de extremos de *a*.

Dois vértices são **adjacentes** quando se forem os extremos de uma mesma aresta.

Um **laço** em um grafo são arestas com extremos *n*-*n*. Um grafo pode não conter laços, caso no qual é chamado de **sem laços.**

Duas arestas que possuem o mesmo extremo são chamadas de **arestas paralelas**. Um vértice isolado não é adjacente a qualquer outro vértice.

Um **grafo simples** é aquele que não possui arestas paralelas e nem laços. O **grafo completo**, é aquele no qual todo vértice distinto é adjacente e é denotado por Km, sendo *m* o número de vértices. Já um **grafo regular** é aquele que todos os seus vértices possuem mesmo grau.

Um **subgrafo** de um grafo consiste em um conjunto de vértices e um conjunto de arestas que são subconjuntos dos conjuntos de vértices e arestas originais, respectivamente, nos quais os extremos de qualquer aresta precisam ser os mesmos que no grafo original. Resumidamente, é um grafo obtido apagando-se parte do grafo original e deixando o restante sem alterações.

O **caminho** de um vértice *x* a um vértice *y* é uma sequência de vértices e arestas, um **ciclo** é um caminho de algum vértice *x* até *x* de novo, da forma que nenhum outro vértice ocorra mais de uma vez no caminho. O **comprimento** de um caminho é o número de arestas que ele possui. Um grafo sem ciclos é definido como **acíclico**. Um grafo é dito **conexo** se houver entre quaisquer dois vértices um caminho.

**Grafos Isomórficos** são dois grafos (N1, A1, g1) e (N2, A2, g2) se existirem bijeções *f*1:N1 -> N2 tais que para cada aresta *a E* A1, g1(*a*) = x-y se, e somente se, g2[*f*2(*a*)] = *f*1(*x*)-*f*2(*y*). Para os grafos simples aplica-se o **Teorema sobre Isomorfismo para grafos simples**, o qual afirma que dados dois grafos (N1, A1, g1) e (N2, A2, g2), se houver uma bijeção *f*: N1 -> N2 tal que para qualquer vértice *n*ie *n*j de N1 , *n*ie *n*j são adjacentes se, e somente se, *f(n*i)e *f*(*n*j) são adjacentes também.

Um grafo é **bipartido completo** ou **bipartite completo** se seus vértices podem ser particionados em dois conjuntos não-vazios N1 e N2 tais que dois vértices *x* e *y* sejam adjacentes se, e somente se, *x E* N1 e *y E* N2. Se |N1| = m e |N2| = n, este grafo é denotado por Km,n.

Um **grafo planar** é um grafo que pode ser desenhado em um plano (uma folha de papel, por exemplo) de forma que suas arestas se interceptem apenas em vértices. Deve-se lembrar que a palavra-chave na definição de grafo planar é que ele pode ser desenhado de outra maneira.

O matemático suíço Leonhard Euler descobriu um fato em que um grafo simples, conexo e planar divide o plano em um número de regiões (chamadas de faces nos polígonos), incluindo as regiões totalmente fechadas e uma região infinita exterior. Euler observou uma relação entre os números devértices, arestas e regiões neste tipo de grafo, chamada **fórmula de Euler**: *n* + *a* = *r* + 2.

**Teorema sobre o número de vértices e arestas** afirma que para um grafo conexo, simples e planar com *n* vértices e *a* arestas:

1. Se a representação planar divide o plano em *r* regiões, então: *n* - *a + r* = 2;
2. Se n >= 3, então: *a* <= 3\**n* - 6;
3. Se *n >=* 3 e não existem ciclos de comprimento 3, então: *a* <= 2\**n* - 4.

Os grafos K5 e K3,3 ilustram uma regra básica de todos grafos não-planares. Para enunciar tal regra precisamos da definição de **grafos homeomorfos**, dois grafos são se ambos puderem ser obtidos do mesmo grafo por uma sequência de subdivisões elementares, nas quais uma única aresta *x*-*y* é substituída por duas novas arestas *x*-*v* e *v*-*y* que se conectam a um novo vértice *v*. Obs.: um grafo homeomorfo não pode ser obtido através de outro grafo homeomorfo.

**Teorema de Kuratowski** afirma que um grafo é não-planar se, e somente se, contém um subgrafo homeomorfo a K5 e K3,3.

**Teorema das quatro cores** afirma que 4 cores são suficientes para colorir qualquer mapa no plano, ou melhor, qualquer grafo simples, conexo e planar.

Um **grafo dual** do mapa é a transformação de um mapa qualquer em grafo. Colocando um vértice em cada região do mapa, e uma aresta entre dois vértices que representam países adjacentes.

**Coloração** (dos vértices) de um grafo é a atribuição de uma cor (ou qualquer marcador, números por exemplo) a cada vértice do grafo de tal forma que dois vértices adjacentes não possuem a mesma cor. Já **número cromático** do grafo é o menor números de cores necessárias para se obter uma coloração.

Lema: em um grafo simples, conexo e planar há pelo menos um vértice com grau menor ou igual a 5.

**Teorema das cinco cores** afirma que número cromático de um grafo simples, conexo e planar é no máximo 5.

**Árvore** é um grafo acíclico e conexo com um nó designado como a **raiz**. Normalmente, os cientistas da computação desenham as árvores com a raiz no topo.

Um grafo acíclico e conexo sem a designação de um vértice como raiz é chamado de **árvore não-enraizada**.

Se T1, ..., Tt são árvores disjuntas com raizes R1, ..., Rt, o grafo formado pela ligação de um novo vértice R, por uma única aresta a cada um dos vértices R1, ..., Rt constitui uma árvore de raiz R. Os vértices R1, ..., Rt são **filhos** de R, e R é **pai** de R1, ..., Rt.

Como a árvore é um grafo conexo, então existe um caminho entre a raiz e todos seus vértices e como a mesma é acíclica, logo este caminho é único. A **profundidade de um vértice** é o comprimento da raiz até o vértice (a raiz tem profundidade zero). A **altura (profundidade) da árvore** é a maior profundidade de todos seus vértices, ou seja, o comprimento do maior caminho entre a raiz e um certo vértice.

Um vértice sem filhos é chamado de **folha** e os vértices que não são folhas são chamados de **vértices internos** ou **nós internos**. Uma **floresta** é qualquer grafo acíclico (não necessariamente conexo), portanto uma floresta é uma coleção de árvores disjuntas.

**Árvores binárias** são árvores em que cada nó possui no máximo dois filhos, constituindo um caso de particular interesse. Cada filho é designado como **filho da esquerda** e **filho da direita** deste nó. Uma **árvore completa** é aquela em que todos os nós internos têm dois filhos e todas as folhas têm a mesma profundidade.

O **teorema no número de aresta de uma árvore** diz que se uma dada árvore possui *n* nós, então ela possui *n* - 1 arestas.

Há os **grafos direcionados**, que são uma tripla ordenada (N, A, g), onde:

**N** é um conjunto não vazio de vértices (nós ou nodos);

**A** é um conjunto de arestas (arcos);

**g** é uma função que associa cada aresta a um par ordenado (*x*,*y*) de vértices, onde *x* é o **ponto inicial** e *y* é o **ponto** **final** de *a*.

Um **caminho** do vértice *n*0 até o vértice *n*x é uma sequência onde para cada *i*, *n*i é o ponto inicial e *n*i + 1 é o ponto final de *a*. Caso exista um caminho do vértice *n*0 até o vértice *n*x, então *n*x é **alcançável** a partir de *n*0. A definição de ciclo também se aplica a grafos direcionados.

Quando desejamos que os vértices possuem informações de identificação, teremos então um **grafo rotulado**. Além de tal, podemos ainda ter um **grafo ponderado**, em que cada aresta possui um valor numérico, ou um peso, associado.

Expressões algébricas envolvendo operações binárias podem ser representadas através de árvores binárias rotuladas. As folhas rotuladas são os operandos, enquanto cada nó interno são rotulados como operadores binários, sendo que para qualquer deste a operação binária é realizada sobre as expressões associadas a suas subárvores a esquerda e a direita.

Redes neurais, são instrumentos utilizados pela inteligência artificial para os quais tarefas como reconhecimento de padrões, são representadas por grafos direcionados e ponderados. Possuindo vários níveis constituindo-se em unidades de entrada, unidade de saída e um nível oculto de unidade. Os pesos das arestas do grafo são ajustados à medida que a rede neural aprende como reconhecer certos padrões de julgamento.

Supondo que um grafo tenha *n*1, ..., *n*n. De posse dos vértices ordenados, podemos formar uma matriz *n* X *n*, onde o elemento *i*, *j* é o número de arestas entre os vértices *n*ie *n*j. Esta matriz é chamada então de **matriz de adjacências** **A** do grafo com relação à ordenação. Portanto: *a*ij = *p* onde existem *p* arestas entre *n*ie *n*j.

A matriz de adjacência é simétrica (ou seja, igual a sua matriz é igual a sua transposta), o que ocorrerá para qualquer grafo não-direcionado. Logo, a simetria da matriz indica que os elementos abaixo da diagonal principal são iguais os elementos acima da diagonal principal, possibilitando então o armazenamento de apenas uma das duas secções (tal motivo é para economia de memória, consequentemente eficiência).

Para um grafo direcionado, a matriz de adjacência reflete a direção das arestas. Portanto: *a*ij = *p* onde *p* é o número de arestas do vértice *n*ipara o vértice *n*j. Logo, sua matriz não é necessariamente simétrica.

Em um grafo simples ponderado, os elementos da matriz de adjacências podem indicar o peso da aresta no lugar de indicar a presença da aresta pelo número 1.

Um grafo pode vir a apresentar poucas arestas, fazendo com que o armazenamento em matrizes sejam ineficientes, pois teria no pior dos casos de percorrer *n*2 elementos. Para tanto, há uma outra forma de representação, agora utilizando alocações dinâmicas, através de **listas encadeadas** as quais armazenam apenas os elementos não-nulos. A sua implementação se dá de forma que possuímos um vetor de *n* elementos , sendo *n* o número de vértices, em que cada posição armazena uma lista encadeada que contem os elementos os quais são adjacentes.

A desvantagem da lista de adjacência é que caso desejamos determinar se um vértice *n*j é adjacente ao vértice *n*i, temos de varrer toda a lista encadeada de *n*, enquanto na matriz de adjacência bastaria pesquisar o elemento *i*, *j* diretamente. Todavia, a vantagem é que para saber todos os vértices que são adjacentes a *n*j, apenas teríamos de varrer a lista referente a *n*j, que deve ter menos elementos que os *n* que teríamos de examinar na matriz de adjacência.

Em um grafo não-direcionado, cada aresta é representada duas vezes na lista de adjacências. Caso o grafo for rotulado ou ponderado, outras informações são acrescentadas ao lado do nome do vértice.

Em uma linguagem de programação que não ofereça ponteiros, podemos implementar listas de adjacências através de vetores de diversas colunas (vetor possuindo como elemento outro vetor, ou um vetor de *struct*), onde cada posição contém um vetor com os nome de cada vértice adjacente, considerado um “pseudoponteiro”. A desvantagem deste fato é que a alocação é estática, ou seja, a quantidade de memória é reservada inicialmente, não sendo possível aumentá-la em tempo de execução.

A representação de árvores binárias podem ser feitas como as anteriores, porem diferem-se pois possuem características as quais não desejamos perder, especificamente a identificação dos filhos à esquerda e à direita. A representação através de uma matriz de adjacência se dá pela criação de uma matriz com duas colunas apenas, em que a primeira representa os filhos à esquerda e a segunda os filhos à direita. Já a representação através de uma lista de adjacência, se dá pela criação de *structs* com os campos referentes aos filhos cada um contendo ponteiros para outra *struct* deste tipo.