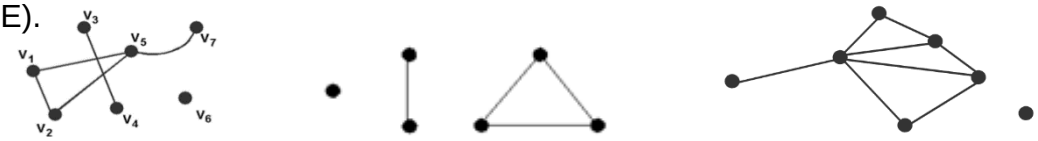
Universidade Federal de Roraima – UFRR
Aluno: Paulo Fábio dos Santos Ramos
Disciplina: Análise de Algoritmos
Professor: Herbert Oliveira Rocha
Data: 26/06/2018

2ª Lista de Exercícios

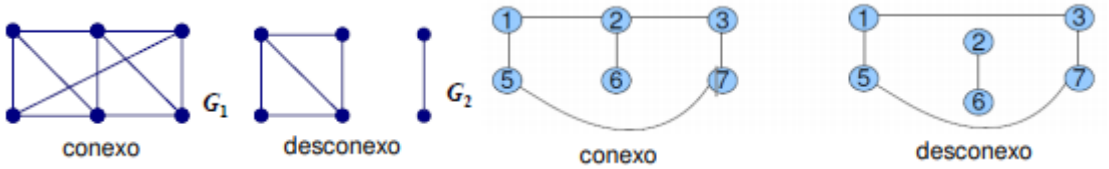
2) a) **Grafo:** É um modelo matemático que representa relações entre objetos, consiste de um conjunto de vértices V, ligados por um conjuntos de arestas ou arcos G = (V, E).

Exemplos.



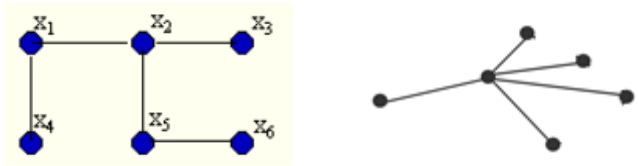
b) **Conexo:** Um grafo é conexo se existir um caminho entre qualquer par de vértices, se existir pelo menos 1 par de vértices que não está ligado a nenhum caminho, ele é desconexo.

Exemplos.



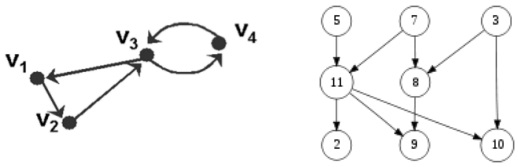
Acíclico: É considerado acíclico quando não possuir ciclos. Uma arvore é um grafo acíclico conexo.

Exemplos.



Direcionado: Cada aresta possui uma única direção de x para y.

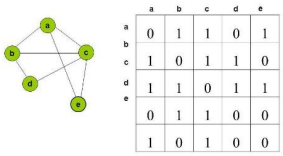
Exemplos:



c) A vizinhança de um vértice V consiste em todos os vértices que são adjacentes a V, e pode ser representado tanto pela matriz de adjacência como pela lista de adjacência.

Matriz de adjacência: Dado um grafo G, a matriz de adjacências M é uma matriz n x n tal que:

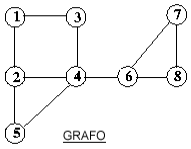
n = número de vértices e M[i,j] = 1, se existir aresta de i a j e M[i,j] = 0, se não existir aresta de i a j.



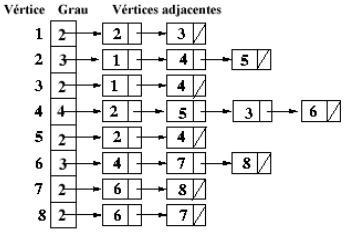
	a	b	c	d	e
a	0	1	1	0	1
b	1	0	1	1	0
c	1	1	0	1	1
d	0	1	1	0	0
e	1	0	1	0	0

Lista de adjacência: Dado um grafo G, a estrutura de adjacência é um vetor de n elementos que são capazes de apontar, cada um, para uma lista linear. O iesimo elemento do vetor aponta para a lista linear das arestas que incidem no vértice i.

Exemplo.



GRAFO



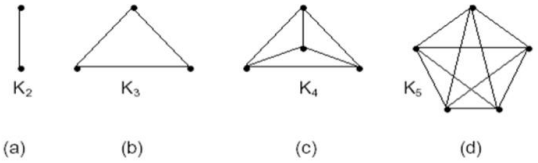
LISTA DE ADJACÊNCIAS

D) **Grafo planar:** É considerado um grafo planar, aqueles que são submetidos no plano de tal forma que suas arestas não se cruzem.

Exemplo.

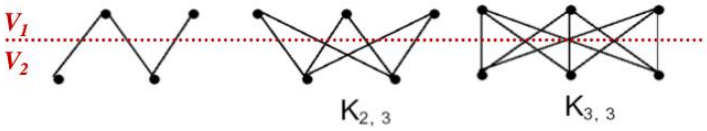
F) **Grafo completo:** Um grafo é completo se todos os seus vértices forem adjacentes.

Exemplos.



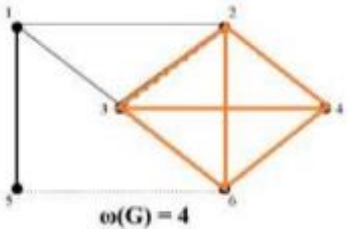
Grafo bipartido: Um grafo G = (V,E) é bipartido quando o seu conjunto de vertices V pode ser dividido em dois subconjuntos V1 e V2 tais quais todas as arestas do cojunto E une um vertice de V1 a outro vertice de V2.

Exemplo.



Clique: Em um grafo não-dirigido, um clique é um conjunto de vértices dois a dois adjacentes, ou seja, se tiver a seguinte propriedade: para todo par (v,w) de vértices distintos em C, existe uma aresta com pontas (v e w).

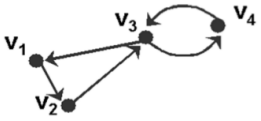
Exemplo.



G)**Grafo simples.** Um grafo simples é um grafo que não possui arestas múltiplas.

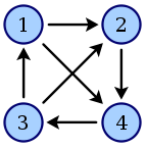
Exemplo.

Multigrafo. Quando possui mais de uma aresta interligando os mesmos dois vertces.



Digrafo: O mesmo que um grafo direcionado, cada aresta possui uma única direção de x para y.

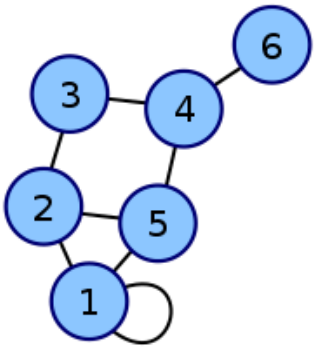
Exemplo.



3) Uma **matriz de incidência** representa computacionalmente um grafo através de uma matriz bidimensional, onde uma das dimensões são vértices e a outra dimensão são arestas. Para representar um grafo sem pesos nas arestas e não direcionado, basta que as entradas da matriz M contenham 1 se a aresta incide no vértice, 2 caso seja um laço (incide duas vezes) e 0 caso a aresta não incida no vértice.

Exemplo.

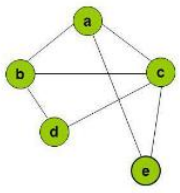
$$\begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



Matriz de adjacência: Dado um grafo G, a matriz de adjacências M é uma matriz n x n tal que:

n = número de vértices e M[i,j] = 1, se existir aresta de i a j e M[i,j] = 0, se não existir aresta de i a j.

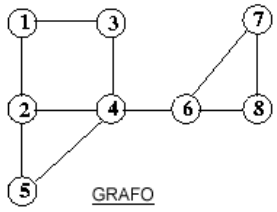
Exemplo.



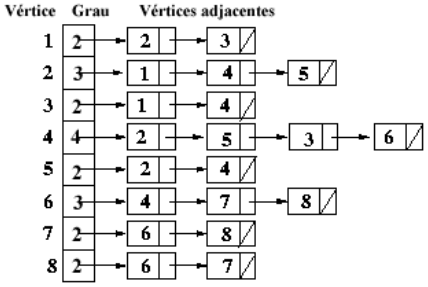
	a	b	c	d	e
a	0	1	1	0	1
b	1	0	1	1	0
c	1	1	0	1	1
d	0	1	1	0	0
e	1	0	1	0	0

Lista de adjacência: Dado um grafo G, a estrutura de adjacência é um vetor de n elementos que são capazes de apontar, cada um, para uma lista linear. O iesimo elemento do vetor aponta para a lista linear das arestas que incidem no vértice i.

Exemplo.



GRAFO



LISTA DE ADJACÊNCIAS

5) b) **Programação dinâmica** é um método para a construção de algoritmos para a resolução de problemas computacionais, em especial os de otimização combinatória. Ela é aplicável a problemas nos quais a solução ótima pode ser computada a partir da solução ótima previamente calculada e memorizada - de forma a evitar recálculo - de outros subproblemas que, sobrepostos, compõem o problema original.

b) **Algoritmo guloso,** toma decisões com base nas informações disponíveis na iteração corrente, sem olhar as consequências que essas decisões terão no futuro. Um algoritmo guloso jamais se arrepende ou volta atrás: as escolhas que faz em cada iteração são definitivas, ou seja, ele sempre busca a melhor solução que aparecer no momento, sem ver onde isso pode dar.

c) **Backtracking** é um algoritmo genérico que busca, por força bruta, soluções possíveis para problemas computacionais. Ele busca por candidatos à soluções e abandona cada candidato parcial C quando C não pode resultar em uma solução válida. Quando sua busca chega a uma extremidade da estrutura de dados, como um nó terminal de uma árvore, o algoritmo realiza um retrocesso tipicamente implementado através de uma recursão.

9) a) **Problema SAT**: é o problema de determinar se existe uma determinada valoração para as variáveis de uma determinada fórmula booleana tal que esta valoração satisfaça esta fórmula em questão. Por exemplo, tomando x1, x2, x3, x4 como as variáveis booleanas e a expressão $(X1 \vee \neg X3 \vee X4) \wedge (\neg X2 \vee X3 \vee \neg X4)$ caso exista uma atribuição de valores de verdade para as variáveis da fórmula

que torne a fórmula avaliada VERDADEIRA, esta fórmula é considera satisfazível, em contrapartida se nenhuma atribuição levou a uma avaliação da fórmula como verdadeira, ela é considerada insatisfazível.

b)A classe **P** é a classe de todos os problemas de busca que são resolvidos em tempo polinomial.(2sat, arvore geradora minima, mochila fracionaria..)

A classe **NP** é a classe de todos os problemas de busca, seja ele resolvido em tempo polinomial ou não.

A classe **NP-Dificil** é um subconjunto de NP, que são informalmente definidos como “Pelo menos tão dificeis quanto os problemas mais dificeis em NP”. Um problema H é NP-difícil se e somente se existe um problema NP-completo L que é Turing-redutível em tempo polinomial para H.

NP-Completo: Um problema de decisão X é completo em NP (ou NP-completo) se X está em NP e qualquer outro problema em NP pode ser polinomialmente reduzido a X.