SCC 503 - ALGORITMOS E ESTRUTURAS DE DADOS II

Grafos - Conceitos

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

*Material baseado em aulas dos professores: Elaine Parros Machado de Souza, Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr., Maria Cristina Oliveira e Cristina Ciferri.

Por que aprender Grafos?

- Modelar problemas que envolvam conjuntos de objetos e relacionamentos entre pares de objetos estabelecidos por conexões.
- Modelar relações e processos em diversos sistemas:
  - Físicos, biológicos, sociais e de informação

Por que aprender Grafos?

- Redes de
  - Comunicação (Facebook)
  - Organização de dados
  - Dispositivos computacionais
  - Fluxo de Computação
- Sistemas de recomendação (Amazon, Netflix, etc.)
- Otimização de caminhos (Google Maps, Uber, etc.)
- Modelar sintaxe de linguagem natural

Por que aprender Grafos?

- Estudo de átomos e moléculas
- Medir prestígio
- Espalhamento de rumor
- Amizades entre pessoas
- Padrões de reprodução de animais
- Espalhamento de **doenças**
- Relação entre genes
- ...

Programa completo Jupiterweb

- Conceitos fundamentais e aplicações computacionais de grafos.
- Estruturas de dados para representação de grafos: lista de arestas, lista de adjacências e matriz de adjacências.
- Percursos em grafos e aplicações: busca em largura e profundidade.

Programa completo Jupiterweb

- Algoritmos clássicos sobre grafos e aplicações, tais como caminhos mínimos, árvores geradoras mínimas e ordenação topológica.

Definições

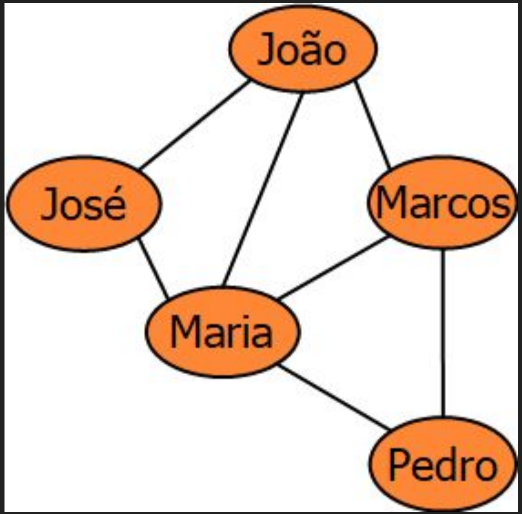
Definições

- Grafos
  - Estruturas abstratas que modelam objetos e a relação (conexão) entre eles.
- Teoria dos Grafos
  - Área de matemática combinatória
  - Resolução de problemas em computação

Definições

- O que é um Grafo?
  - Uma estrutura definida por conjuntos de nós (ou vértices) e as arestas que ligam esses nós
  - Grafo **G** definido como um par (V, A)
    - V: Conjunto de nós, ou vértices
    - A: Conjunto de pares de vértices chamados de arestas ou arcos

Definições



- Exemplo de Grafo
  - Rede social de amizade
    - Cada vértice é uma pessoa
    - Cada aresta representa uma amizade entre pessoas

Representação

Representação

- Para criarmos um Tipo Abstrato de Dado (TAD) de Grafo geralmente usamos uma dessas 2 soluções:
  - Lista de Adjacência
  - Matriz de Adjacência
- A escolha de cada uma depende das características do grafo e dos algoritmos utilizados na resolução do problema
  - Existe impacto no desempenho destes algoritmos

Representação

- É possível criar implementações de operações que independem da implementação específica do TAD

Operações Básicas

Operações Básicas

- Inicializa(G)
  - Cria um grafo G vazio.
- InsereVertice(G, v)
  - Insere um vértice v isolado no grafo G
- RemoveVertice(G,v)
  - Remove do grafo G o vértice v e suas arestas incidentes

Operações Básicas

- InsereAresta(G, v, u, P)
  - Insere a aresta (v,u) no grafo G com peso P.
- ExisteAresta(G, v, u)
  - Verifica se existe a aresta (v,u) no grafo G.
- RetiraAresta(G, v, u, P)
  - Retira a aresta (v, u) do grafo G e retorna seu peso.
- LiberaGrafo(G):
  - Liberar o espaço ocupado pelo grafo G.

Operações Básicas

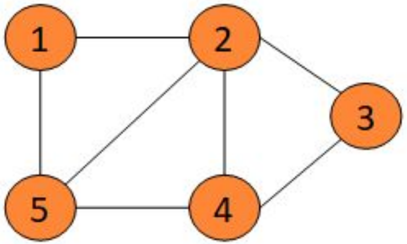
- ExisteAdj(G, v):
  - Retorna verdadeiro se existe algum vértice adjacente a v.
- PrimeiroAdj(G, v):
  - Retorna o endereço do primeiro vértice adjacente a v.
- PróximoAdj(G, v, p):
  - Retorna o endereço do próximo vértice adjacente a v a partir de p.

Matriz de Adjacências

Matriz de Adjacências

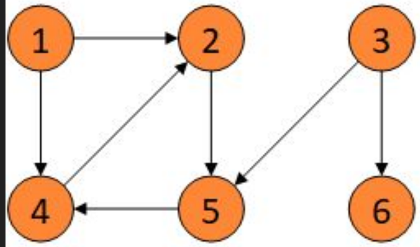
- Dado G = (V, A) um grafo com n vértices
  - n = |V| e n>=1.
- A matriz de adjacências de G é uma matriz M:
  - n x n
  - M[v,u] = 1 se e somente se existe aresta do vértice v para o vértice u.
  - Grafos ponderados: M[v,u] contém o rótulo ou peso da aresta.

grafo não direcionado



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

grafo direcionado



	1	2	3	4	5	6
1		1		1		
2					1	
3					1	1
4		1				
5				1		
6						

Matriz de Adjacências

Vamos Programar

Em C - Matriz

```
#define MAXNUMVERTICES 100

typedef int tpeso;
typedef int tvertice;
typedef int tapontador;

typedef struct {
    tpeso mat[MAXNUMVERTICES][MAXNUMVERTICES];
    int num_vertices;
} tgrafo;
```

Em C - Inicialização

```
void inicializa_grafo(tgrafo *grafo, int num_vertices) {
    int i, j;

    grafo->num_vertices = num_vertices;
    for (i = 0; i < grafo->num_vertices; i++)
        for (j = 0; j < grafo->num_vertices; j++)
            grafo->mat[i][j] = 0;
}
```

Em C - Inserção de aresta e verificação da existência

```
void insere_aresta(tvertice v, tvertice u,
                   tpeso peso, tgrafo *grafo) {
    grafo->mat[v][u] = peso;
}

int existe_aresta(tvertice v, tvertice u,
                  tgrafo *grafo) {
    return grafo->mat[v][u] != 0;
}
```

Em C - Remoção da Aresta

```
void retira_aresta(tvertice v, tvertice u,
                   tgrafo *grafo) {
    if (grafo->mat[v][u] == 0)
        printf("Erro: Aresta inexistente");
    else
        grafo->mat[v][u] = 0;
}
```

Em C - Verificação de Adjacência

```
int existe_adj(tvertice v, tgrafo *grafo) {
    tvertice aux;

    for (aux = 0; aux < grafo->num_vertices; aux++) {
        if (grafo->mat[v][aux] != 0)
            return 1;
    }
    return 0;
}
```

Em C - Buscar endereço do primeiro adjacente

```
tapontador primeiro_adj(tvertice v, tgrafo *grafo) {
    tapontador aux;

    for (aux = 0; aux < grafo->num_vertices; aux++)
        if (grafo->mat[v][aux] != 0)
            return aux;
    return NULO;
}
```

Em C - Buscar endereço do próximo adjacente e recuperar adjacente por endereço

```c
tapontador proximo_adj(tvertice v, tapontador aux,
                       tgrafo *grafo) {

    for (aux++; aux < grafo->num_vertices; aux++)
        if (grafo->mat[v][aux] != 0)
            return aux;
    return NULO;
}

void recupera_adj(tvertice v, tapontador p, tvertice *u,
                  tpeso *peso, tgrafo *grafo) {
    *u = p;
    *peso = grafo->mat[v][p];
}
```

Características

Características - Matriz de Adjacência

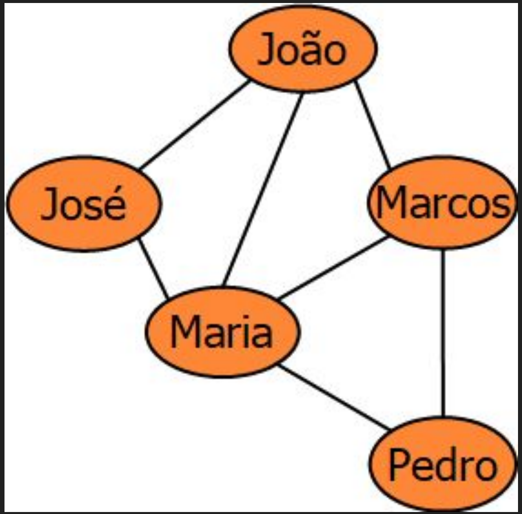
- Para que categoria de grafo usar?
  - Grafos densos, onde $|A|$ é próximo de $|V|^2$.
- Tempo necessário para acessar um elemento é independente de $|V|$ e $|A|$ => $O(1)$
- Muito útil para algoritmos que precisam saber com rapidez se...
  - Existe uma aresta ligando dois vértices.

Características - Matriz de Adjacência

- Desvantagem:
  - Espaço => $O(|V|^2)$
  - Ler ou percorrer a matriz => complexidade de tempo $O(|V|^2)$.
- Matriz é simétrica para grafos não direcionados
  - Cerca de metade do espaço pode ser economizado representando a matriz triangular superior ou inferior.

Mais Definições

Definições



- Exemplo de Grafo
  - Rede social de amizade
    - Cada vértice é uma pessoa
    - Cada aresta representa uma amizade entre pessoas

Definições

- Se eu sou seu amigo, isso significa que você é meu amigo?
  - Se aresta (x,y) sempre implica em (y,x)
    - Grafo não-direcionado
  - Caso contrário
    - Grafo direcionado (dígrafo).
  - Como seria um grafo sobre “ouviu falar de”?

Definições

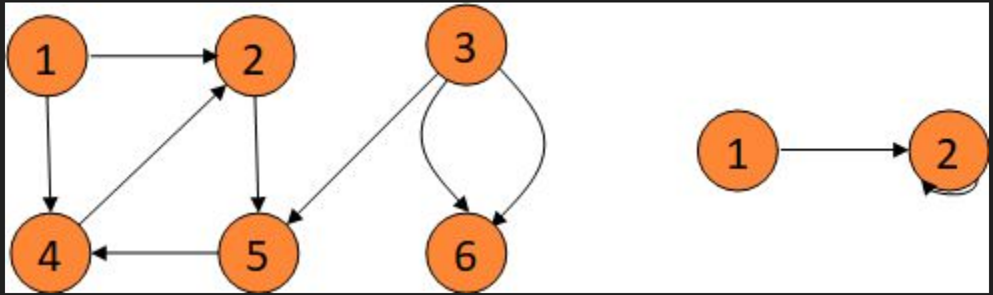
- Eu sou amigo de mim mesmo?
  - Aresta (x,x)
    - Laço ou self-loop.
- Eu posso ser seu amigo diversas vezes?
  - Relação modelada com arestas múltiplas ou paralelas.

Definições

- Um grafo direcionado (grafo orientado ou Dígrafo) G é um par (V,A), em que:
  - V é um conjunto finito de vértices
  - A é uma relação binária ordenada em V
- Uma aresta (u,v) sai do vértice u (origem) e chega no vértice v (destino)
  - O vértice v é adjacente ao vértice u
  - A existência de (u,v) não implica a existência de (v,u)

Definições

- Podem existir arestas de um vértice para ele mesmo
  - Self-loops.
- Arestas múltiplas:
  - Arestas com mesma origem e mesmo destino



Referências

- WIRTH,N. Algorithms and Data Structures, Englewood Cliffs, Prentice-Hall, 1986.
- CORMEN, H.T.; LEISERSON, C.E.; RIVEST, R.L. Introduction to Algorithms, MIT Press, McGraw-Hill, 1999.
- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.
- SZWARCFITER,J.L. Grafos e Algoritmos Computacionais. Editora Campus, 1983.