Universidade do Estado de Santa Catarina
Estudante: Guilherme Panizzon
Disciplina: OTPA0001

GRAFOS

Escopo da Apresentação

- Conceito Geral;
- Tipos de Grafos;
- Representação Computacional;
- Algoritmos de Busca em Grafos;
- Aplicações;

Objetivo da Apresentação

- Realizar uma revisão da disciplina de Teoria dos Grafos abordando a representação de grafos das quatro principais formas;

Relembrando…

- **G=(V,E)**, onde **V** é o conjunto de vértices e **E** o conjunto de arestas formado por pares ordenados ou não de vértices do conjunto V.

Tipos de Grafos

- Direcionados / Não-direcionados;
- Ponderados / Não-ponderados;
- Cíclicos e Acíclicos;
- Esparsos / Densos;

Grafos conexos-não direcionados acíclicos?

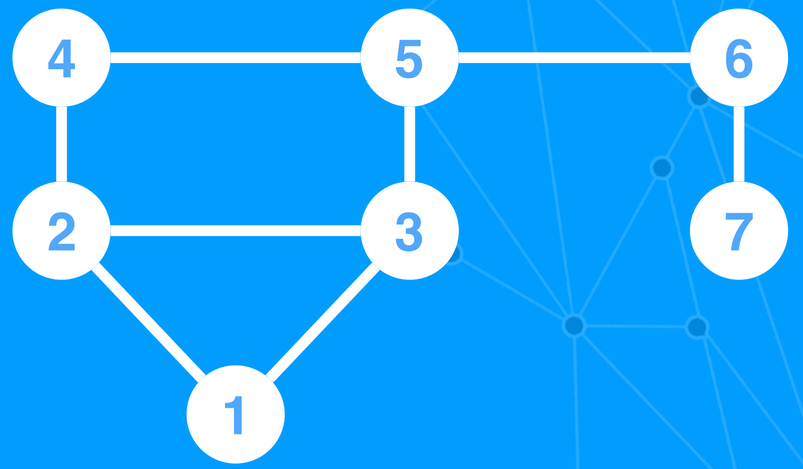
Árvores!

- Um grafo G=(V,E) é chamado **esparso** se |E| for muito menor que $|V|^2$;
- **Denso** se |E| é próximo ao número de $|V|^2$;

Representação Computacional

- Matriz de Adjacência;
- Lista de Adjacência;
- Lista de Arestas;
- Parent-Child Tree;

Matriz de Adjacência:

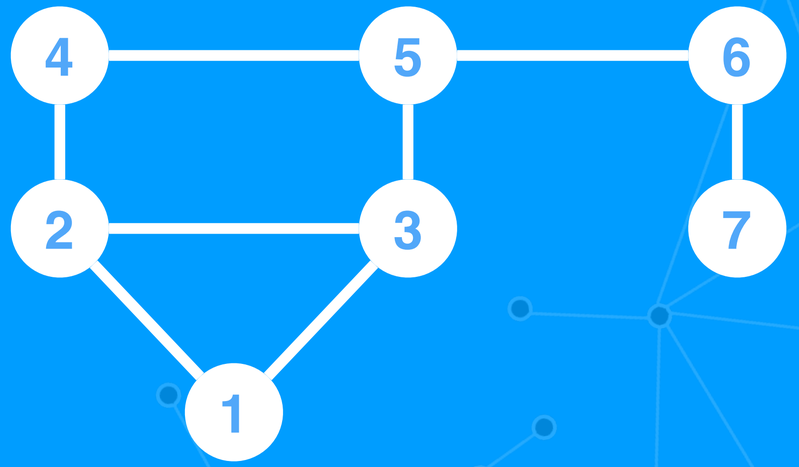


	1	2	3	4	5	6	7
1		1	1				
2	1		1	1			
3	1	1			1		
4		1			1		
5			1	1		1	
6					1		1
7						1	

Matriz de Adjacência:

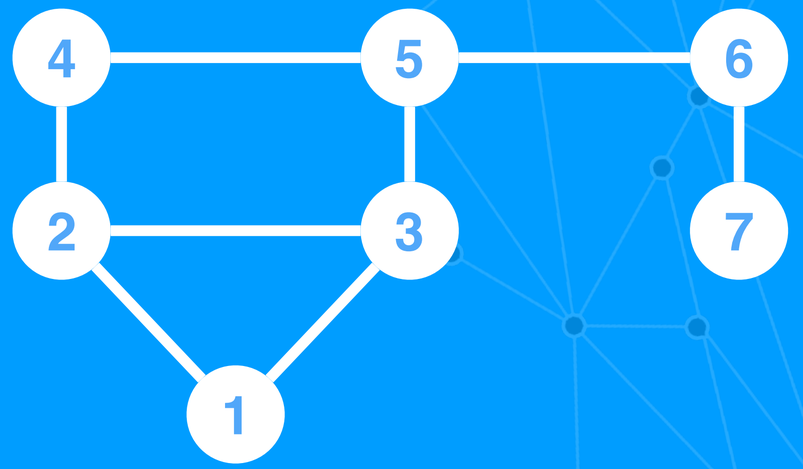
Exemplo

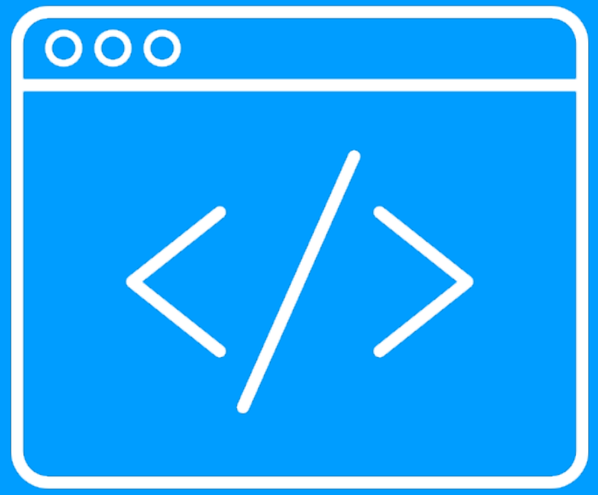
Entrada	*Saída*
7 8	**0110000**
1 2	**1011000**
1 3	**1100100**
3 2	**0100100**
3 5	**0011010**
2 4	**0000101**
4 5	**0000010**
5 6	
6 7	



Matriz de Adjacência:

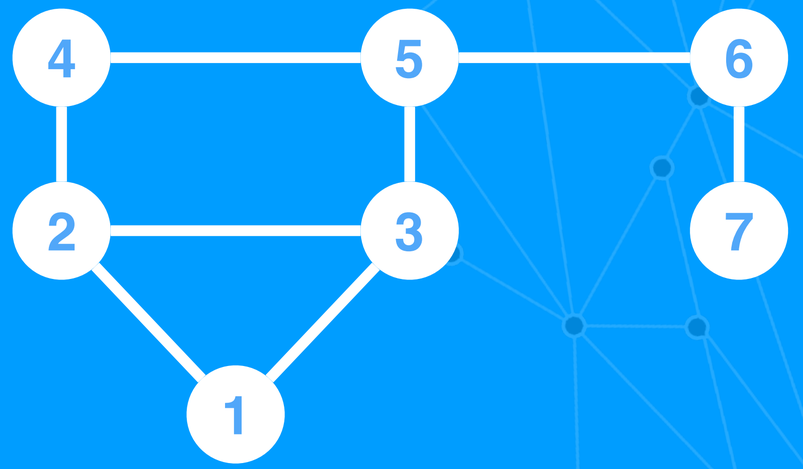
Implementação em C++





Matriz de Adjacência:

Considerações

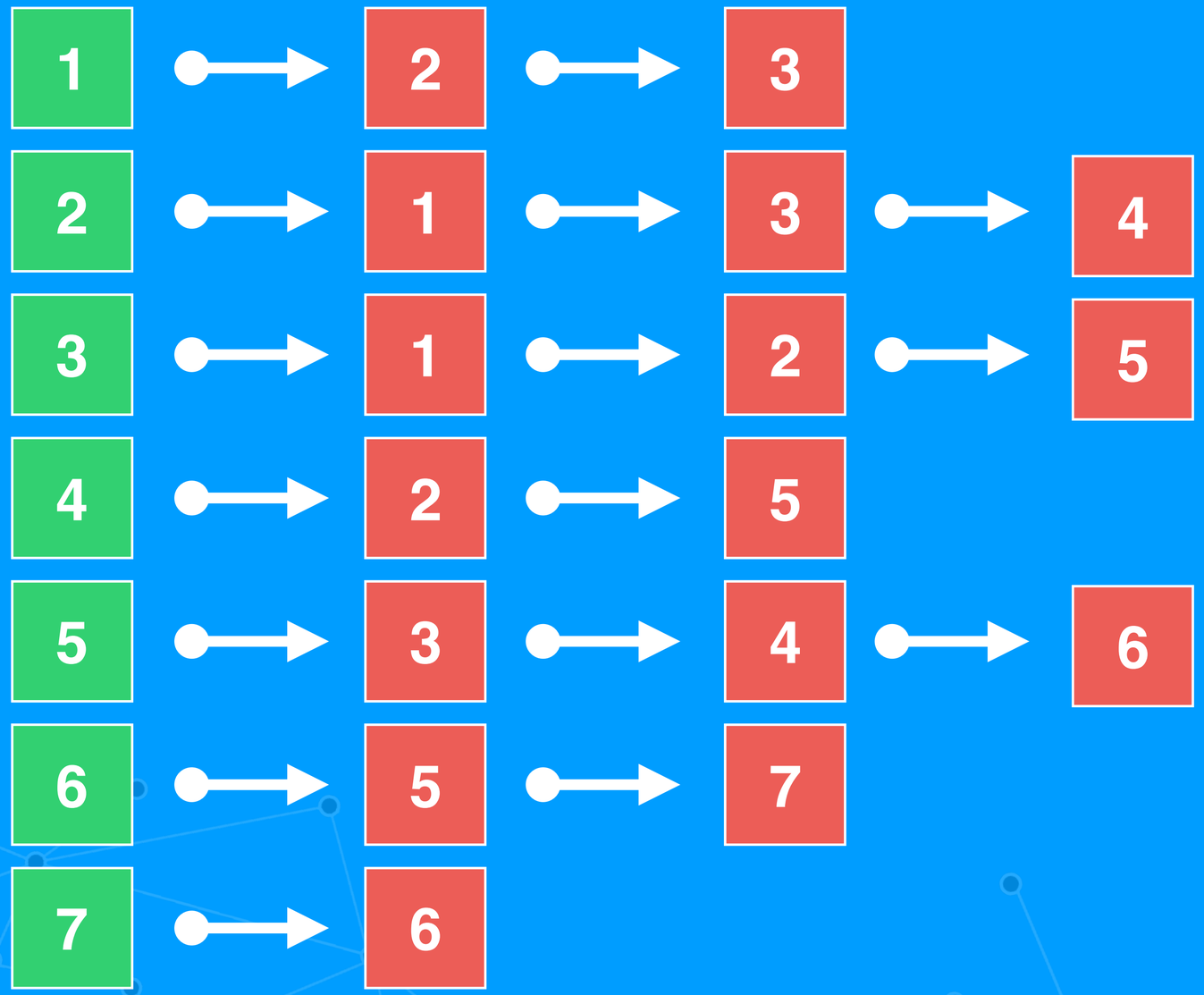


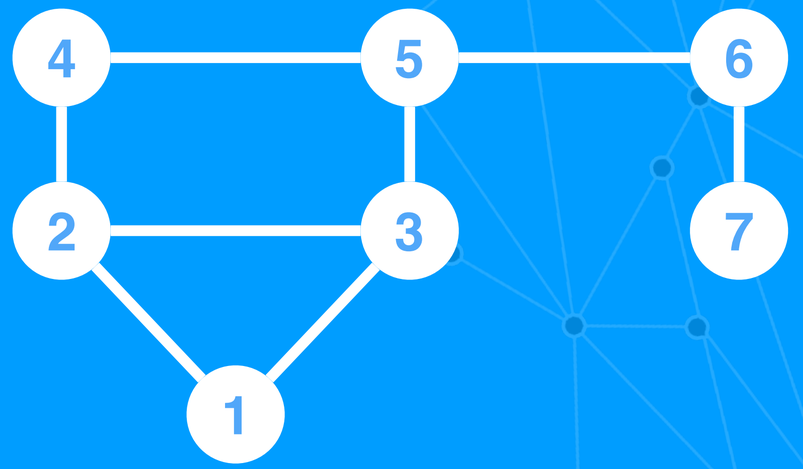
- Para um grafo com peso associado à aresta:

$$\begin{cases} \texttt{MyGraph[i][j] = peso(i,j);} \\ \texttt{0, caso contrário;} \end{cases}$$

- Consideravelmente bom se a conectividade entre dois vértices em um grafo de baixa densidade é frequentemente requisitada;
- Ruim se o grafo tiver uma quantidade grande de zeros (grafo denso);

Lista de Adjacência





Lista de Adjacência

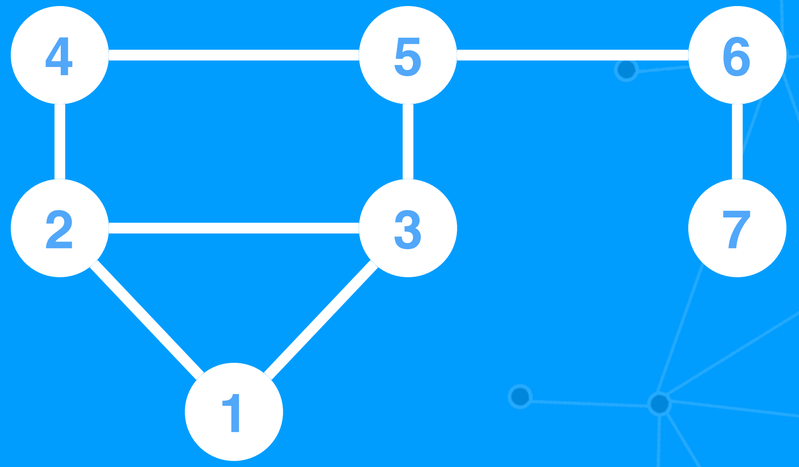
Exemplo

Entrada

```
7 8
1 2
1 3
3 2
3 5
2 4
4 5
5 6
6 7
```

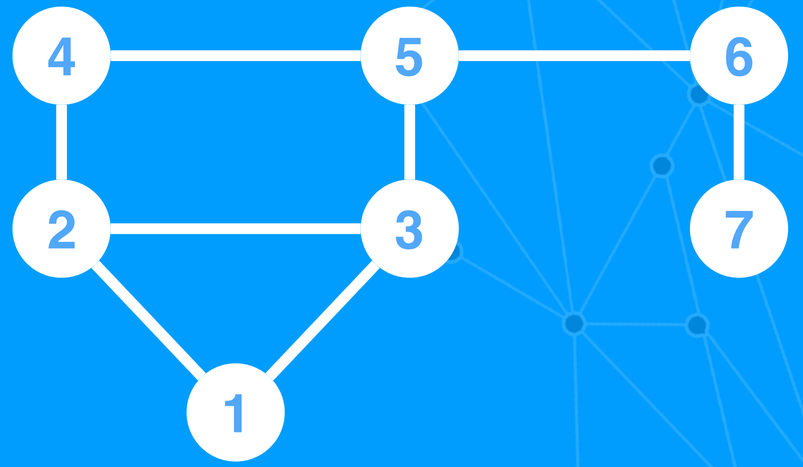
Saída

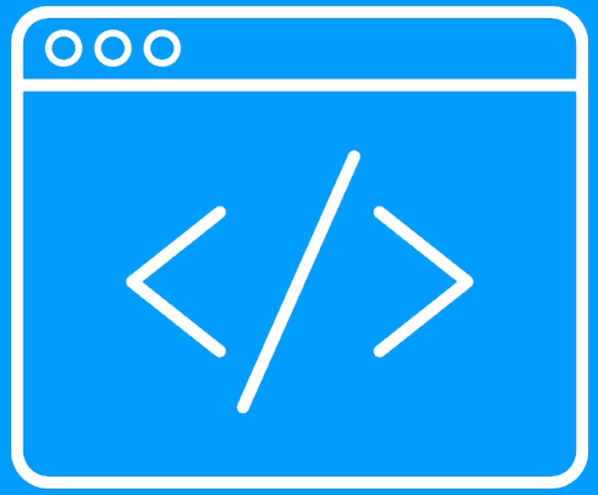
```
1 -> 2 -> 3
2 -> 1–> 3 -> 4
3 -> 1 -> 2 -> 5
4 -> 2 -> 5
5 -> 3 -> 4
6 -> 5 -> 7
7 -> 6
```



Lista de Adjacência

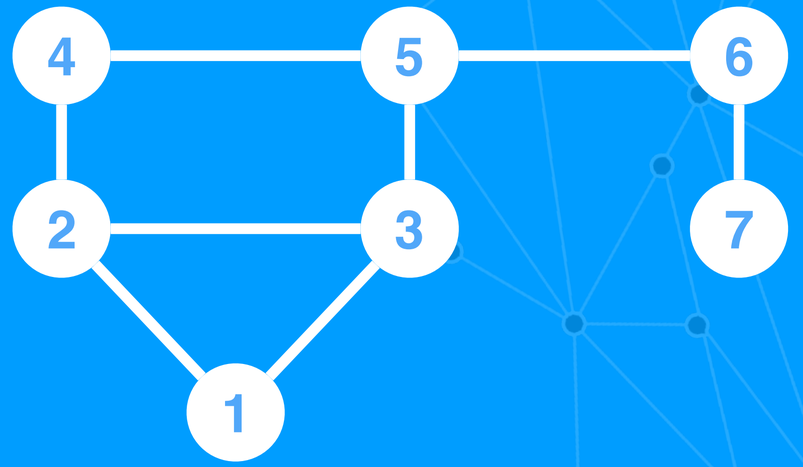
Implementação em C++





Lista de Adjacência

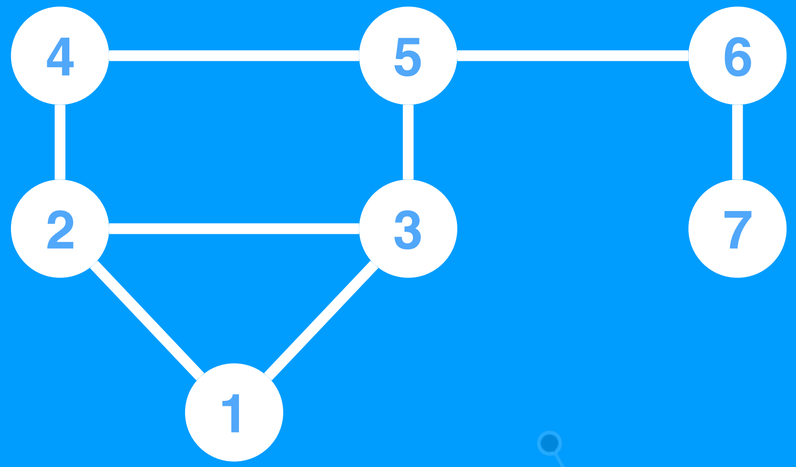
Considerações



- Solução compacta e eficiente de representação;
- Considere como a melhor escolha na maioria dos casos;

Lista de Arestas

1	⟷	2
1	⟷	3
2	⟷	3
2	⟷	4
3	⟷	5
4	⟷	5
5	⟷	6
6	⟷	7



Lista de Arestas

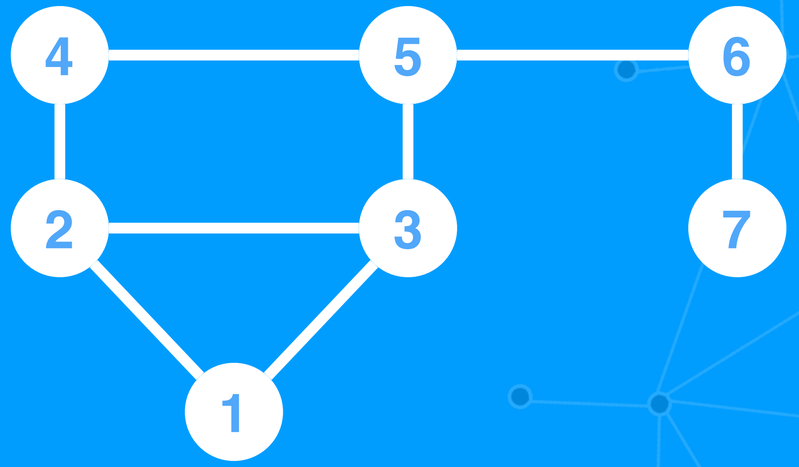
Exemplo

Entrada

```
7 8
1 2
1 3
2 3
2 4
3 5
4 5
5 6
6 7
```

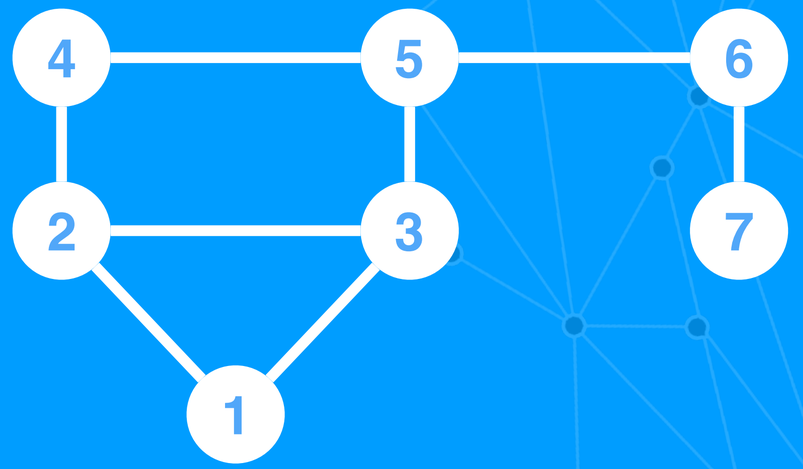
Saída

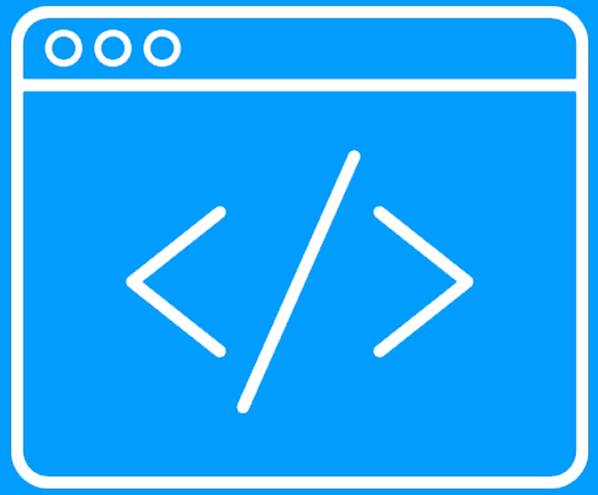
```
6 <-> 7
5 <-> 6
4 <-> 5
3 <-> 5
2 <-> 4
2 <-> 3
1 <-> 3
1 <-> 2
```



Lista de Arestas

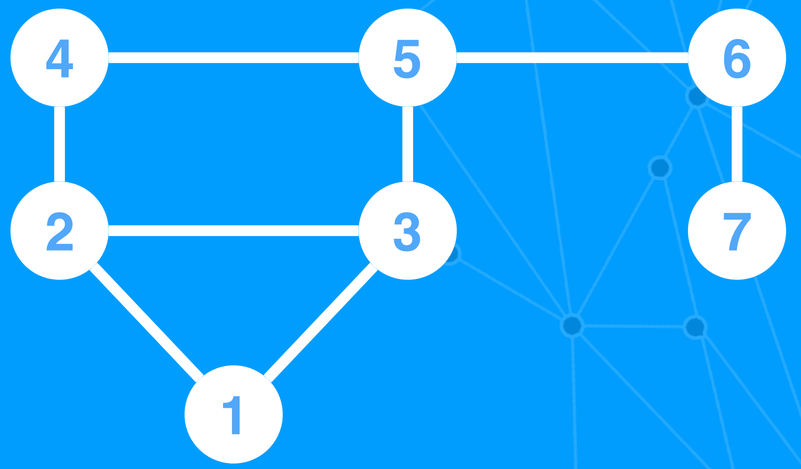
Implementação em C++





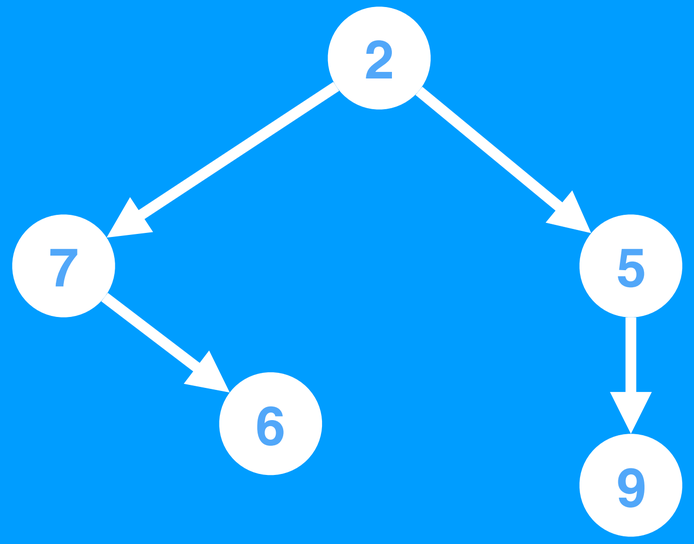
Lista de Arestas

Considerações



- Estrutura eficiente para a implementação do algoritmo de Kruskal;

Parent-Child Tree



- Para cada vértice é armazenado apenas 2 atributos: o pai (NULL quando vértice raiz) e a lista de filhos (NULL para folhas);

p(2) = null
c(2) = {7,5}
p(5) = 2
c(5) = {9}
p(6) = 7
c(6) = null
p(7) = 2
c(7) = {6}
p(9) = 5
c(9) = null

Comparativo

Tarefa	Vencedor
Checar se (u,v) pertence ao grafo	Matrizes de adjacência **O**(1)
Checar o grau de um vértice	Listas de adjacência
Memória em grafo pequenos	Listas de adjacência (n+m) vs n^2
Inserção ou remoção de arestas	Matrizes de adjacência **O**(1) vs **O**(d)
Busca em grafos	Listas de adjacência Θ(m+n) vs Θ(n^2)
Velocidade de implementação	Matrizes de adjacência
Melhor para a maioria dos problemas	Listas de adjacência

Fonte: [TADM], p.152

Algoritmos de Busca em Grafos

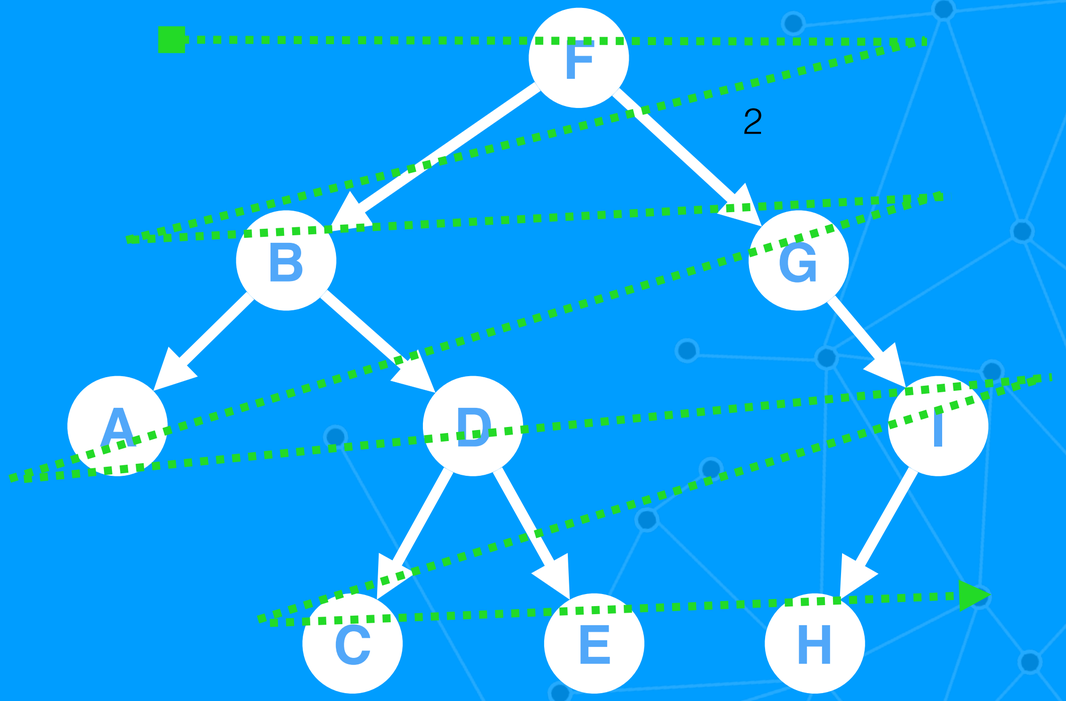
- BFS (Breath First Search) - Busca em Largura
- DFS (Depth First Search) - Busca em Profundidade

2

Algoritmos de Busca em Grafos

- **BFS (Breath First Search) - Busca em Largura**

enquanto a fila não estiver vazia faça
    tire um vértice v da fila
    para cada vizinho w de v que ainda não foi numerado
        numere w e coloque-o na fila



Ordem: F, B, G, A, D, I , C, E, H

Algoritmos de Busca em Grafos

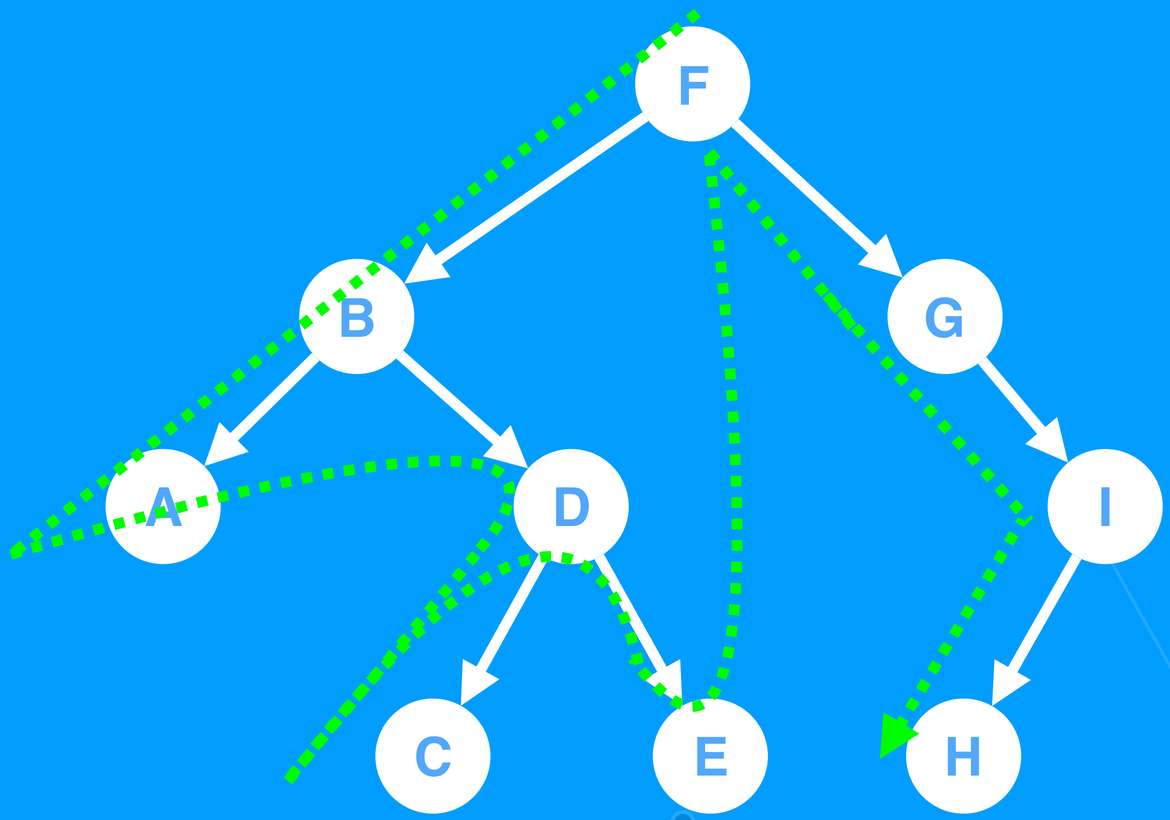
- **BFS - Implementação**

```
void bfs(int origem, int num_vertices) {
    for (int i = 0; i < num_vertices; ++i) {
        visitado[i] = false;
    }
    // Fila que guarda a ordem de visita dos vértices
    list<int> fila;
    // Adiciona a origem na fila
    fila.push_back(origem);
    visitado[origem] = true;

    int atual; // vértice atual que estamos visitando
    while (!fila.empty()) {
        atual = fila.front(); // pega o 1o vértice
        fila.pop_front(); // remove ele da fila
        for (int i = 0; i < num_vertices; ++i) {
            if (grafo[atual][i] && !visitado[i]) {
                lista.push_back(i);
                visited[i] = true;
            }
        }
    }
```

Algoritmos de Busca em Grafos

- **DFS (Depth First Search) - Busca em Profundidade**



2

Ordem: F, B, A, D, C, E, G , I, H

Algoritmos de Busca em Grafos

- **DFS - Implementação**

```
// Declarando
const int MAX_VERTICES = 100;
int grafo[MAX_VERTICES][MAX_VERTICES];
bool visitado[MAX_VERTICES];
int num_vertices; // qtd de vértices do grafo atual

// Recursivamente visita todos os vértices
// alcançáveis a partir de atual
void dfs(int atual) {
    for (int i = 0; i < num_vertices; ++i) {
        if (grafo[atual][i] && !visitado[i]) {
            visitado[i] = true;
            dfs(i);
        }
    }
}
```

Aplicações

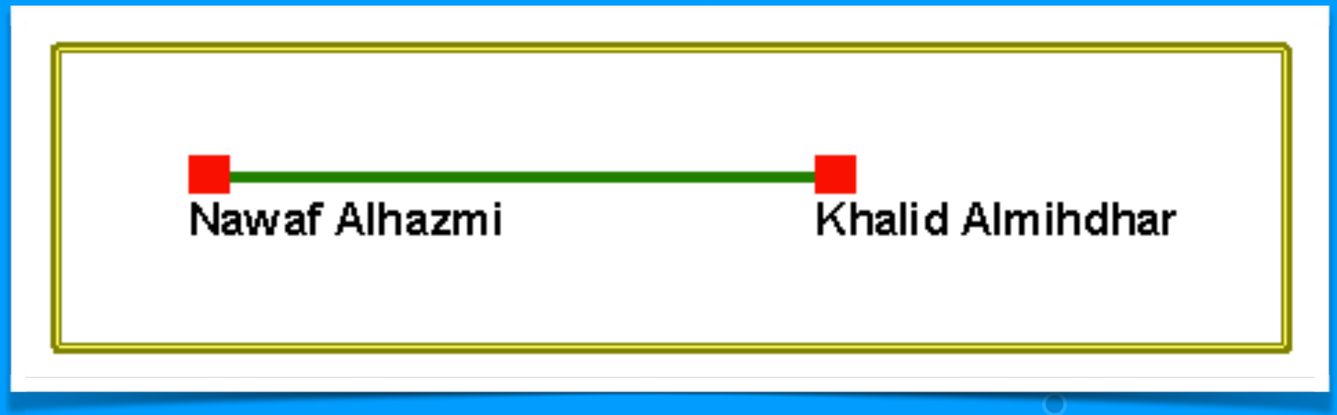
- Social Networking Analysis;
- Verificar se um grafo é bicolorível;
- Detecção de ciclos;
- Ordenação topológica;
- Caminhos mínimos(Grafos não ponderados);
- Conectividade;

Curiosidade - Aplicação

- Social Network Analysis
    - Análise estrutural para investigar a disseminação da informação em comunidades como também analise da interação humana e predições de comportamento;[2]
    - Mapeamento das redes terroristas no mundo;

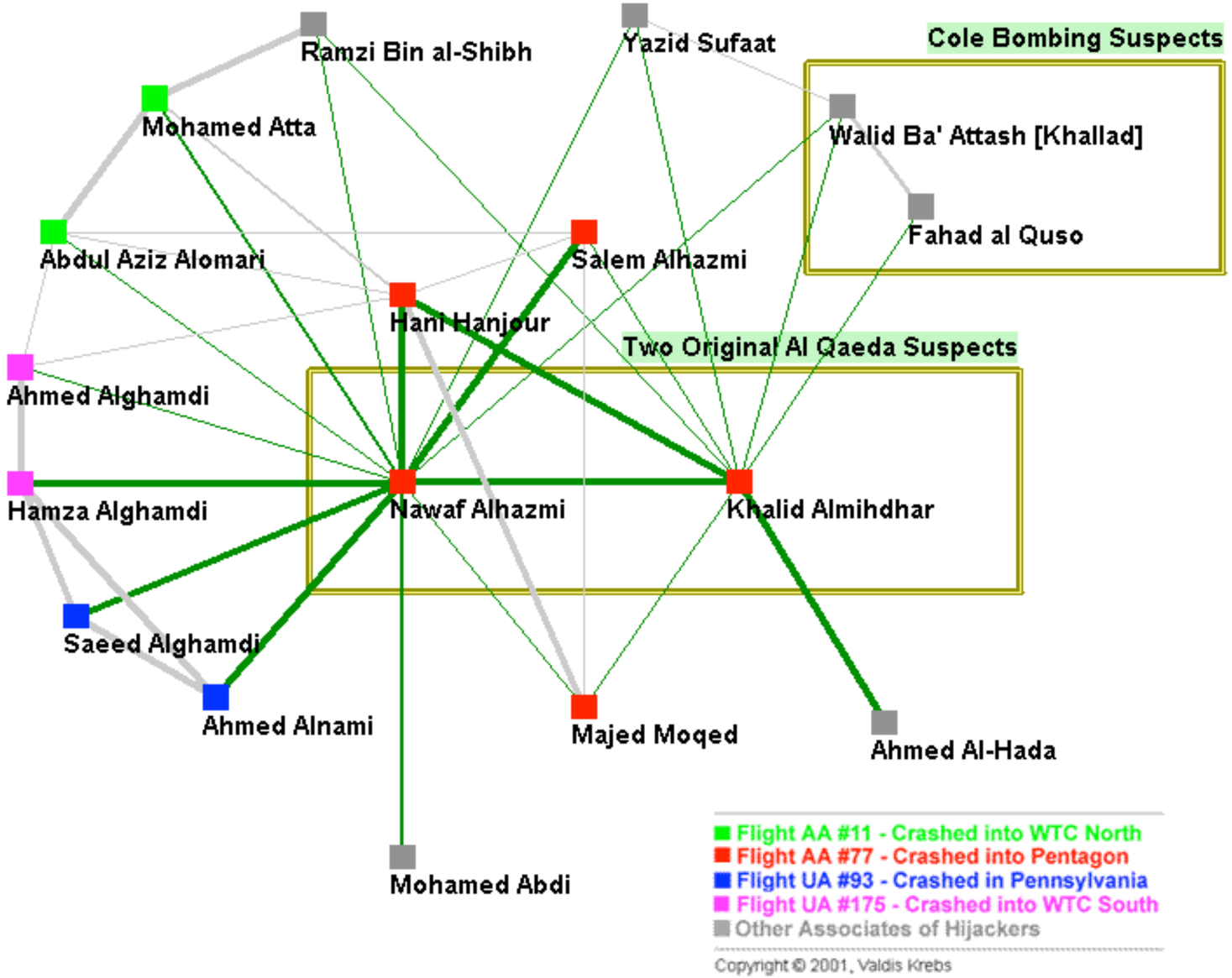
Aplicações

- Redes terroristas



Aplicações

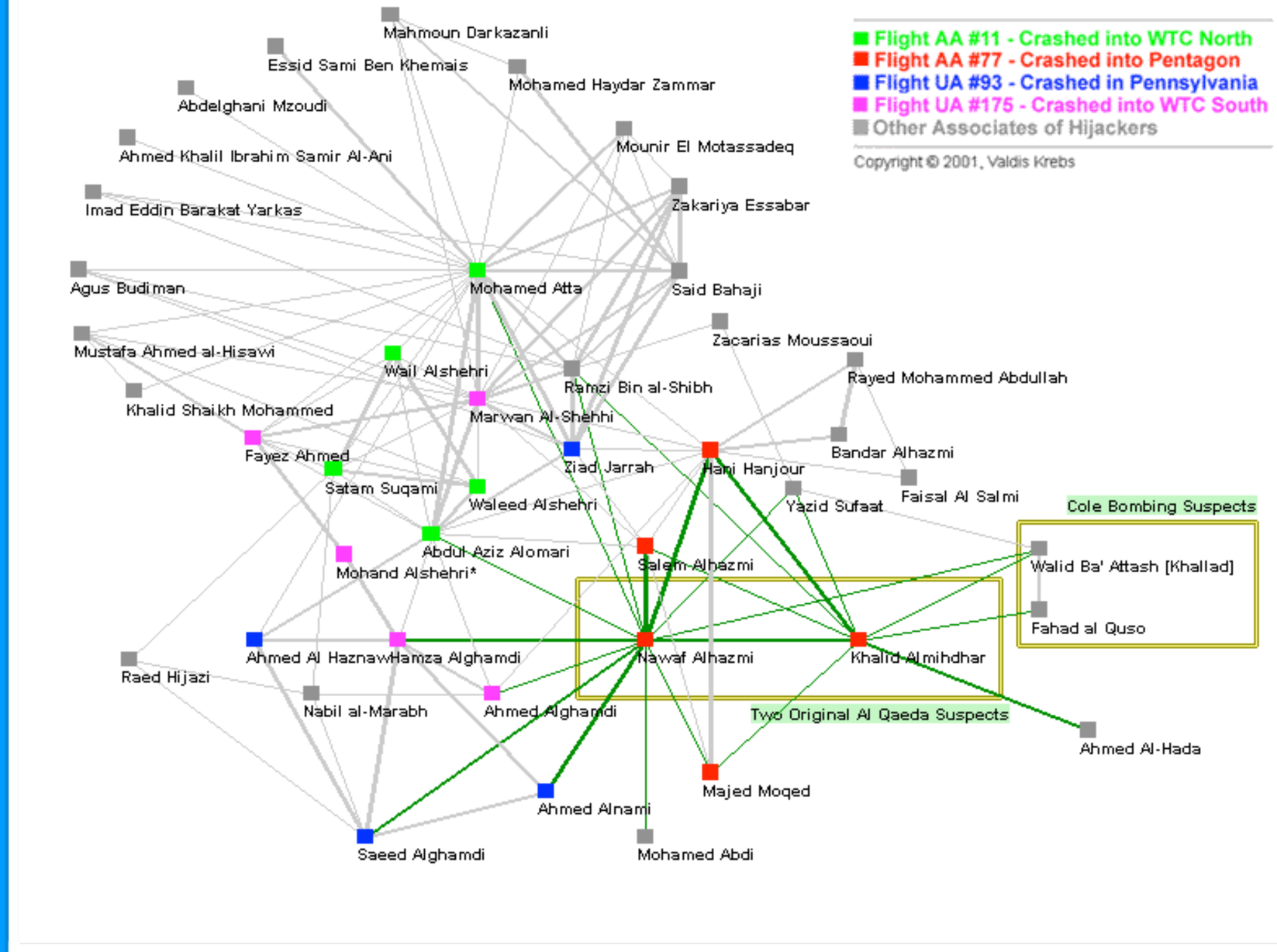
- Redes terroristas



Fonte: http://orgnet.com/tnet.html

Aplicações

- Redes terroristas



Fonte: http://orgnet.com/tnet.html

Aplicações

- **Redes terroristas**
  - Qual o vértice (terrorista) de maior grau?
  - Qual é o menor caminho entre jihadX jihadY?
  - Quem está no centro do grafo?…

Integration		Reach		Connector	
Mohamed Atta	422	Mohamed Atta	28	Mohamed Atta	939
Nawaf Alhazmi	388	Nawaf Alhazmi	26	Ramzi Bin al-Shibh	773
Hani Hanjour	334	Ramzi Bin al-Shibh	24	Nawaf Alhazmi	741
Marwan Al-Shehhi	320	Marwan Al-Shehhi	23	Zacarias Moussaoui	602
Ramzi Bin al-Shibh	310	Hani Hanjour	22	Marwan Al-Shehhi	590

Fonte: http://orgnet.com/tnet.html

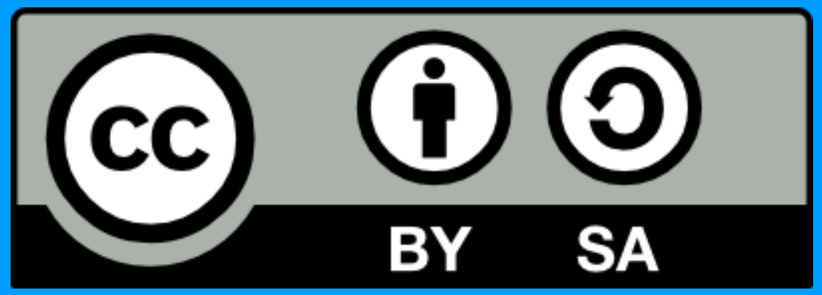
Referências

[TADM] - Steven S. Skiena - The Algorithm Design ManuaCompetitive, 2008.

HALIM, S. Programming 3: The New Lower Bound of Programming Contests, 2009.

IME USP. Busca em largura (BFS). Disponível em: http://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/bfs.html. Acesso em: 16/09/2016.

Este trabalho está licenciado sob uma licença Creative Commons



brigado!