

# Grafos

## Programação de computadores II

Prof. Renan Augusto Starke

Instituto Federal de Santa Catarina – IFSC  
Campus Florianópolis  
`renan.starke@ifsc.edu.br`

18 de novembro de 2016



**INSTITUTO FEDERAL**  
**SANTA CATARINA**

Ministério da Educação  
Secretaria de Educação Profissional e Tecnológica  
**INSTITUTO FEDERAL DE SANTA CATARINA**

# Tópicos da aula

- 1 Introdução
- 2 Definição básica e aplicações
- 3 Algoritmos importantes
- 4 Desenhando grafos
- 5 Implementação
- 6 Exercícios

- 1 Introdução
- 2 Definição básica e aplicações
- 3 Algoritmos importantes
- 4 Desenhando grafos
- 5 Implementação
- 6 Exercícios

- ▶ Entender o conceito e aplicações de grafos
- ▶ Aprender algoritmos importantes
  - busca em largura
  - busca em profundidade
- ▶ Utilizar estruturas de dados conhecidos para representação e manipulação de grafos

- 1 Introdução
- 2 Definição básica e aplicações
- 3 Algoritmos importantes
- 4 Desenhando grafos
- 5 Implementação
- 6 Exercícios

# Definição básica

## Grafo

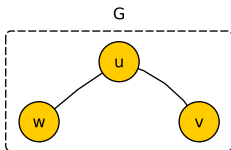
É uma forma simples de representar relações entre conjunto de objetos.

Um grafo  $G = (V, E)$

- ▶ conjunto finito de vértices (nós):  $V$
- ▶ conjunto finito de arestas:  $E$

As arestas são responsáveis por “ligar” um par de nós:

- ▶ Uma aresta  $e \in E$  representa um subconjunto de  $V : e = \{u, v\}$  para um par de vértices  $u, v \in V$ .



$$V = \{u, v, w\}$$

$$E = \{\{u, v\}, \{u, w\}\}$$

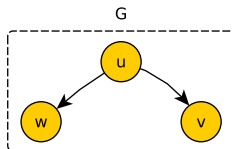
# Definição básica

- ▶ As arestas representam relações *simétricas* entre dois nós
- ▶ Frequentemente precisamos de relações *assimétricas*, criando um grafo **direcionado**

## Grafo direcionado

Um grafo  $G' = (V, E)$  é direcionado quando o par ordenado de arestas  $V : e = \{u, v\}$  não é permutável.

- ▶ a aresta “sai” de  $u$
- ▶ a aresta “chega” em  $v$

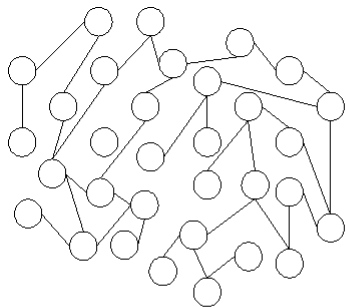


$$V = \{u, v, w\}$$

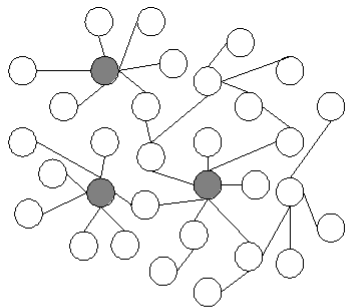
$$E = \{\{u, v\}, \{u, w\}\}$$

<b>Grafo</b>	<b>Vértice</b>	<b>Aresta</b>
Comunicação	Centrais telefônicas, Computadores, Satélites	Cabos, Fibra óptica, Enlaces de microondas
Circuitos	Portas lógicas, registradores, processadores	Filamentos
Hidráulico	Reservatórios, estações de bombeamento	Tubulações
Financeiro	Ações, moeda	Transações
Transporte	Cidades, Aeroportos	Rodovias, Vias aéreas
Escalonamento	Tarefas	Restrições de precedência
Arquitetura funcional de um software	Módulos	Interações entre os módulos
Internet	Páginas Web	Links
Jogos de tabuleiro	Posições no tabuleiro	Movimentos permitidos
Relações sociais	Pessoas, Atores	Amizades, Trabalho conjunto em filmes





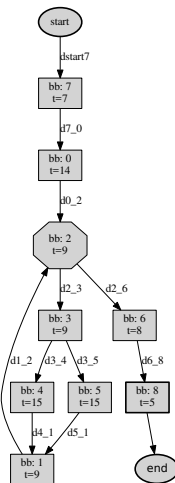
(a) Random network



(b) Scale-free network

# Aplicações – compiladores e análise de código

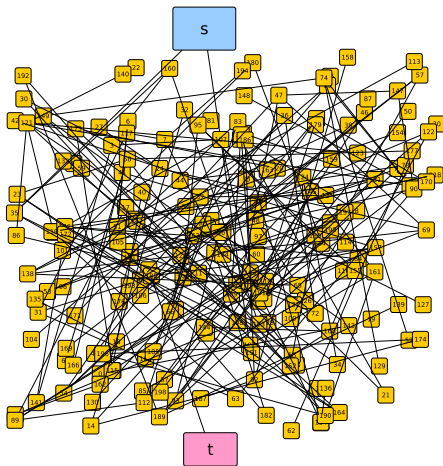
```
1  int main(int argc ,
2      char** argv){
3
4      int i = 1;
5      int j = 0;
6
7      for (j = 0; j < 5; j++){
8          if (i < 6){
9              i++;
10             i+=1;
11             i+=2;
12             i+=3;
13         } else {
14             i--;
15             i-=1;
16             i-=2;
17             i-=3;
18         }
19     }
20 }
```



- 1 Introdução
- 2 Definição básica e aplicações
- 3 Algoritmos importantes**
- 4 Desenhando grafos
- 5 Implementação
- 6 Exercícios

# Conectividade

- ▶ Suponha um grafo  $G = (V, E)$  e dois nós particulares  $s$  e  $t$
- ▶ Como descobrir eficientemente se existe um caminho entre  $s$  e  $t$ ?



- ▶ Busca em largura  
(*Breadth-First Search* – BFS)
- ▶ Busca em profundidade  
(*Depth-First Search* – DFS)

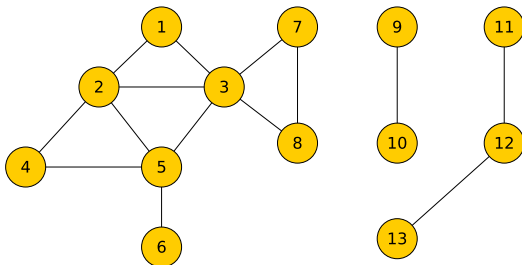
## Busca em Largura

Busca em largura foi inventado no final de 1950 por E. F. Moore para descobrir o caminho de saída de um labirinto. Paralelamente também foi descoberto por C. Y. Lee para roteamento de fios.

- ▶ Inicia-se com um nó  $s$  qualquer
- ▶ Explora-se cada vizinho de  $s$
- ▶ Quando explorar cada vizinho, move para o segundo nível de vizinho

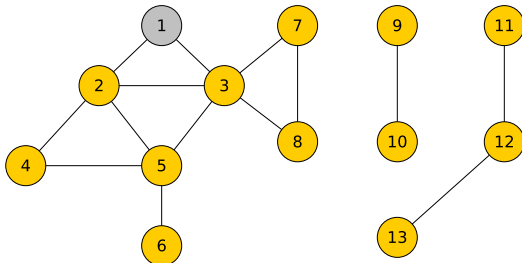
# Busca em Largura

- Exemplo: busca em largura a partir de 1



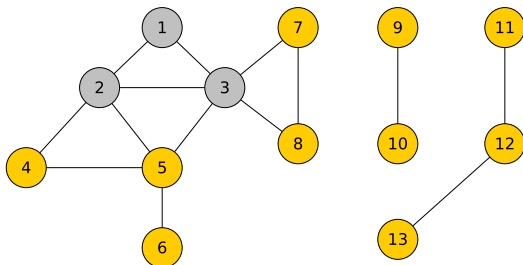
# Busca em Largura

- ▶ Exemplo: busca em largura a partir de 1



# Busca em Largura

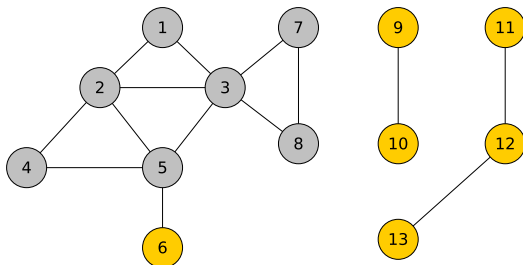
- Exemplo: busca em largura a partir de 1





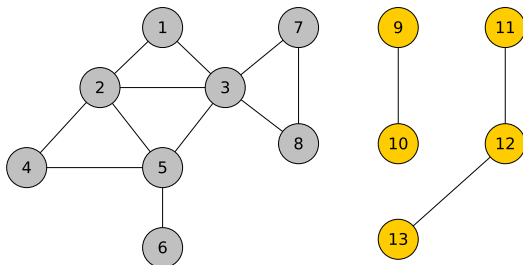
# Busca em Largura

- ▶ Exemplo: busca em largura a partir de 1



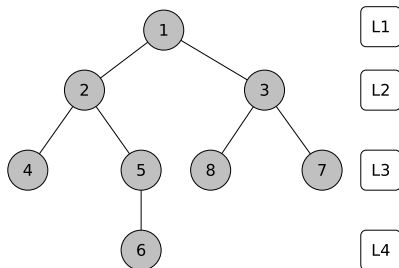
# Busca em Largura

- Exemplo: busca em largura a partir de 1



# Busca em Largura

- ▶ Exemplo: busca em largura a partir de 1



- ▶ Um nó não aparece em uma camada se, e somente se, não houver uma caminho para ele
- ▶ A busca em largura não computa apenas todos os nós alcançáveis a partir de  $s$ , mas também o menor caminho para eles.
- ▶ A busca produz uma árvore, com sua raiz no nó inicial
- ▶ A árvore de saída da busca de largura é um componente conexo
- ▶ Complexidade:
  - $O(|V| + |E|)$

# Algoritmo

```
1 busca_largura(grafo_t *G, no_t s):
2     no = obter_cabeca(G);
3     // inicializa todos os nos
4     while (no) {
5         no.dist = INFINITO;
6         no.pai = NULL;
7         no = obter_proximo(no);
8     }
9     //cria fila
10    fila = cria_fila();
11    s.dist = 0;
12    enqueue(s, fila);
13
14    while(!fila.vazia(fila)) {
15        no = dequeue(fila);
16
17        vizinhos[] = obter_vizinhos(no);
18
19        // marca todos os vizinhos do no
20        for (i=0; i < tamanho(vizinhos); i++){
21            if (vizinhos[i].n.dist == INFINITO)
22                vizinhos[i].n.dist = no.dist + 1    //computa nivel
23                vizinhos[i].pai = no                //marca predecessores
24                enqueue(vizinhos[i], fila);
25        }
26    }
27 }
```

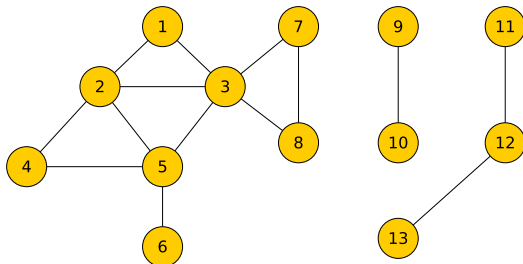
## Busca em Profundidade

Busca em profundidade foi inventado no século 19 por Charles Pierre Trémaux para descobrir o caminho de saída de um labirinto.

- ▶ Inicia-se com uma nó s qualquer
- ▶ Explora todos nós adjacentes o mais “profundo” possível (ao invés da largura) antes de retornar
- ▶ Retorna recursivamente quando chegar em uma folha

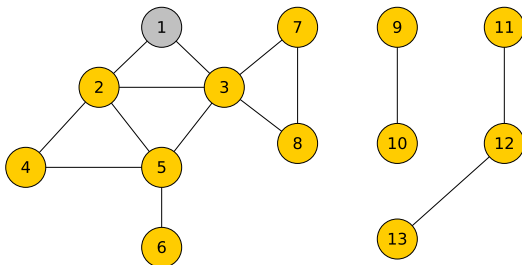
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



# Busca em Profundidade

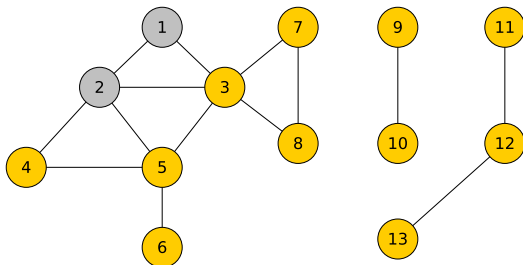
- Exemplo: busca em profundidade a partir de 1





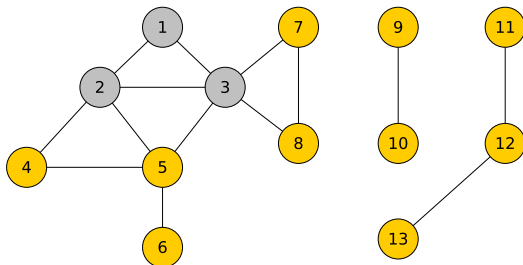
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



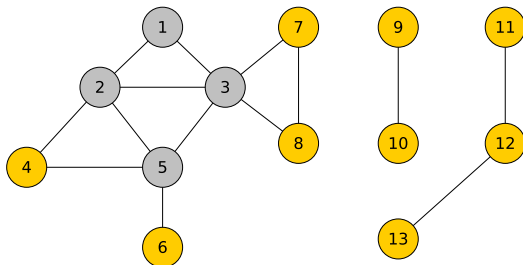
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



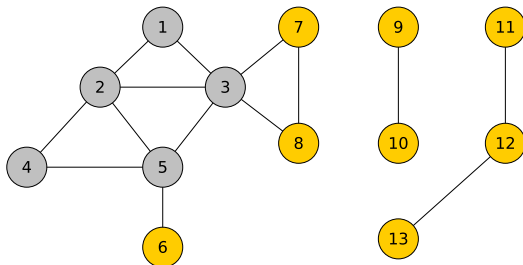
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



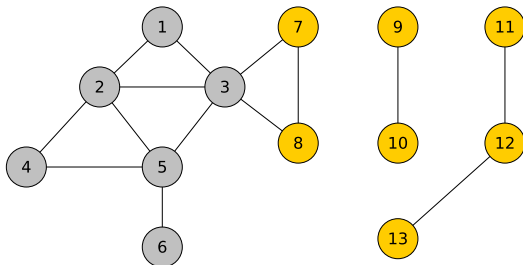
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



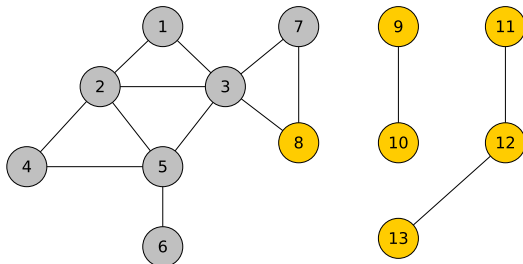
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



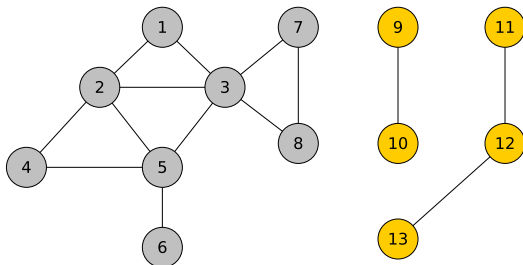
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



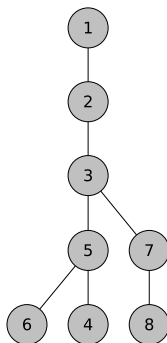
# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



# Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1





- ▶ A busca em profundidade obtém todos os elementos até chegar em um “deadend”
- ▶ É intrinsecamente recursivo
- ▶ É semelhante a largura em termos de conectividade
- ▶ A árvore de saída é também um componente conexo, mas com estrutura diferente da busca em largura
- ▶ Complexidade:
  - $O(|V| + |E|)$

# Algoritmo

```
1 busca_profundidade(grafo_t *G, no_t s):
2     no = obter_cabeca(G);
3
4     // inicializa todos os nos
5     while (no) {
6         no.visit = FALSO;
7         no = obter_proximo(no);
8     }
9
10    //cria pilha
11    pilha = cria_pilha();
12    push(s, pilha);
13
14    while(!pilha_vazia(pilha)) {
15        no = pop(pilha);
16
17        if (no.visit == FALSE) {
18            no.visit = TRUE;
19
20            vizinhos[] = obter_vizinhos(no);
21
22            // adiciona todos os vizinhos do no
23            for (i=0; i < tamanho(vizinhos); i++)
24                push(vizinho[i], pilha);
25        }
26    }
27 }
```

# Exemplo de aplicações das buscas

## ► Busca em largura:

- Descobrir o caminho mínimo entre dois nós
- Testar se um grafo é bi-partido
- Manipulação de árvores binárias

## ► Busca profundidade:

- Descobrir componentes fortemente conexos e ciclos
- Criação de ordem topológica de grafos
- Verificação de dependências (A deve ser feito antes de B e B depois de C)

# Tópico

- 1 Introdução
- 2 Definição básica e aplicações
- 3 Algoritmos importantes
- 4 Desenhando grafos**
- 5 Implementação
- 6 Exercícios

# Desenhando grafos

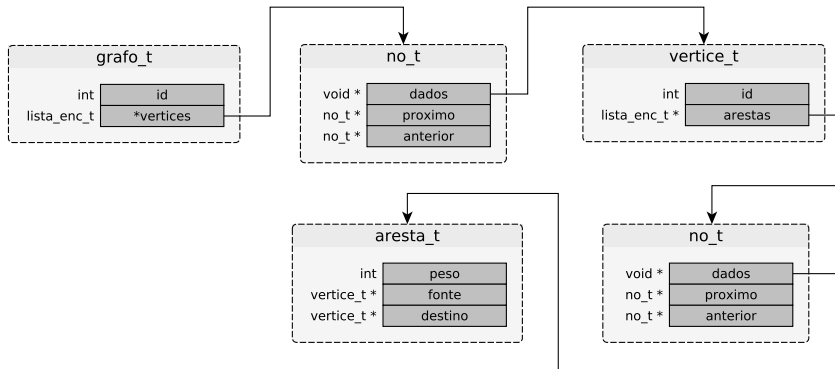
- ▶ Para exportar grafos de seus programas:
  - Linguagem: DOT (graph description language)
  - Programa: Graphviz - Graph Visualization Software
  - <http://www.graphviz.org/>
  - O código abaixo representa o grafo nos exemplos das buscas

```
1 graph {  
2   1 --- 3  
3   1 --- 2  
4   2 --- 4  
5   2 --- 5  
6   2 --- 3  
7   3 --- 5  
8   4 --- 5  
9   5 --- 6  
10  3 --- 7  
11  3 --- 8  
12  7 --- 8  
13  9 --- 10  
14  11 --- 12  
15  12 --- 13  
16 }
```

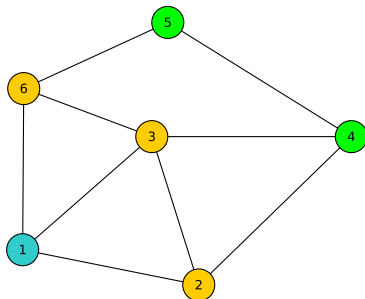
# Tópico

- 1 Introdução
- 2 Definição básica e aplicações
- 3 Algoritmos importantes
- 4 Desenhando grafos
- 5 Implementação**
- 6 Exercícios

# Implementação: listas



# Implementação: matriz de adjacência



	1	2	3	4	5	6
1	0	1	1	0	0	1
2	1	0	1	1	0	0
3	1	2	0	1	0	1
4	0	1	1	0	1	0
5	0	0	0	1	1	0
6	1	0	1	0	1	0



# Tópico

- 1 Introdução
- 2 Definição básica e aplicações
- 3 Algoritmos importantes
- 4 Desenhando grafos
- 5 Implementação
- 6 Exercícios**

- ▶ Utilizando a implementação de grafos com listas encadeadas, implemente:
  - busca em largura
  - busca em profundidade
- ▶ Utilizando a implementação de grafos com matriz de adjacência, implemente:
  - a construção de um grafo adicionando elementos à matriz
  - extração do grafo para DOT (graph description language)
  - busca em largura
  - busca em profundidade