Grafos

Programação de computadores II

Prof. Renan Augusto Starke

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

5 de maio de 2017

INSTITUTO FEDERAL
SANTA CATARINA

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

Tópicos da aula

Tópico

Objetivos

- 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

Tópico

Definição básica

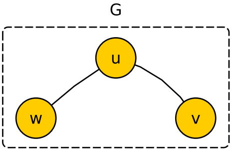
Grafo

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 $u, v \in V$.



$$
\begin{aligned}
V &= \{u, v, w\} \\
E &= \{\{u, v\}, \{u, w\}\}
\end{aligned}
$$

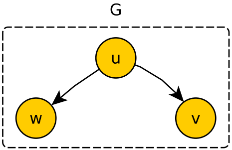
Definição básica

- As arestas representam relações *simétricas* entre dois nós
- Frequentemente necessitamos 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

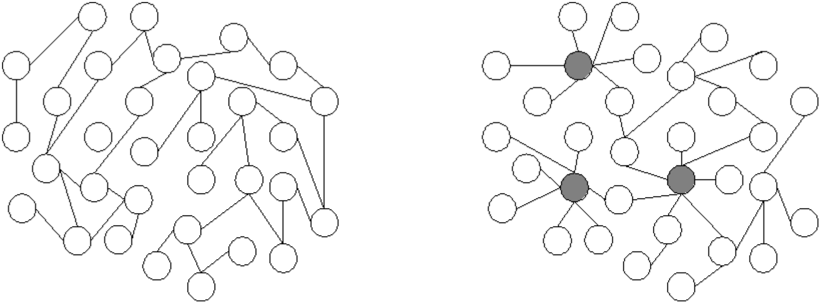


$$\begin{aligned} V &= \{u, v, w\} \\ E &= \{\{u, v\}, \{u, w\}\} \end{aligned}$$

Aplicações

Grafo	Vértice	Aresta
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

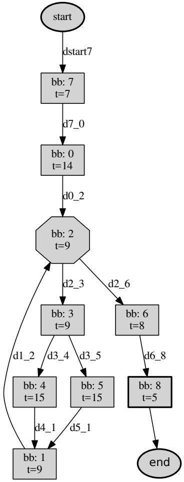
Aplicações – redes



(a) Random network (b) Scale-free network

Aplicações – compiladores e análise de código

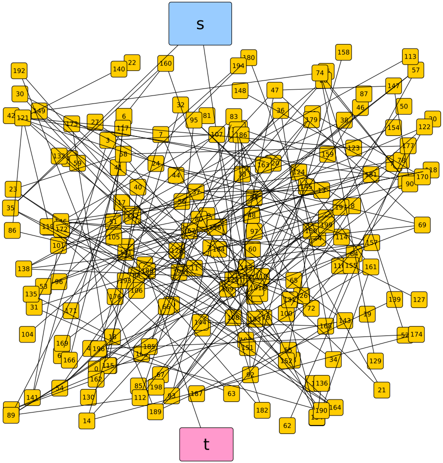
```c
int main(int argc,
    char** argv){

  int i = 1;
  int j = 0;

  for (j = 0; j < 5; j++){
      if(i < 6){
          i++;
          i+=1;
          i+=2;
          i+=3;
      } else {
          i--;
          i-=1;
          i-=2;
          i-=3;
      }
  }
}
```



Tópico

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

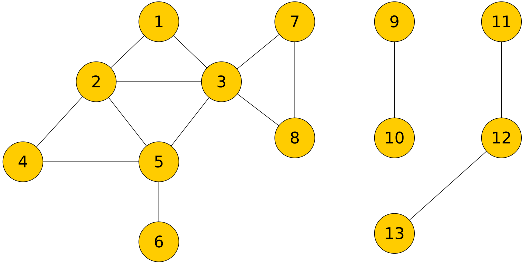
Busca em Largura

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

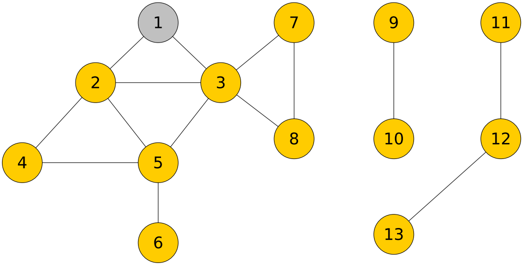
Busca em Largura

- Exemplo: busca em largura a partir de 1



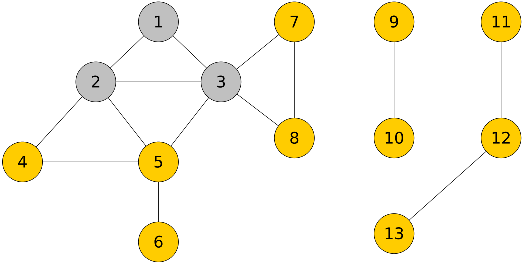
Busca em Largura

- Exemplo: busca em largura a partir de 1



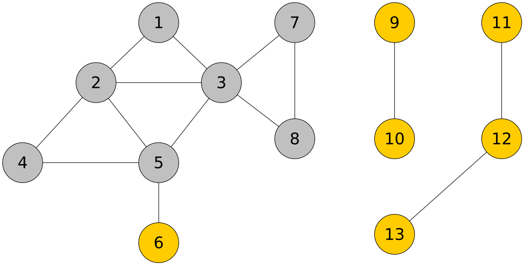
Busca em Largura

- Exemplo: busca em largura a partir de 1



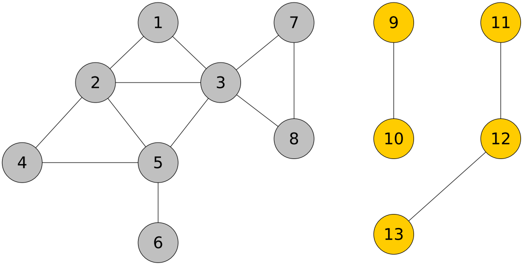
Busca em Largura

- Exemplo: busca em largura a partir de 1



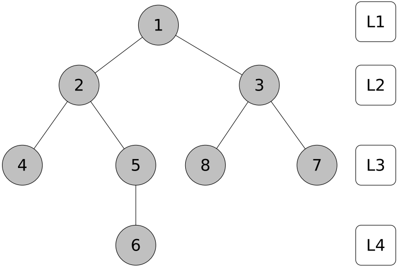
Busca em Largura

- Exemplo: busca em largura a partir de 1



Busca em Largura

- Exemplo: busca em largura a partir de 1



Busca em Largura

- 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

```
function BUSCA_LARGURA(G (grafo), s (vertice_fonte))
    Q ← cria_fila_vazia();

    for all v ∈ G do                                  ▷ Inicializa todos os vértices de G
        v.dist ← ∞ ;                                  ▷ Distância desconhecida da fonte até v
        v.pai ← NULL;                                 ▷ Vértice antecessor vindo da fonte
    end for

    s.dist ← 0;                                       ▷ Distância da fonte até ela mesma é 0
    enqueue(s, Q);

    while Q não for vazia do
        u ← dequeue(Q)                                ▷ Inicia-se com fonte, ela é 0

        for all v ∈ adjacente(u) do
            if v.dist = ∞ then
                v.dist ← u.dist + 1                   ▷ Encontrado novo nível
                v.pai ← u
                enqueue(v, Q);
            end if
        end for
    end while
end function
```

Busca em Profundidade

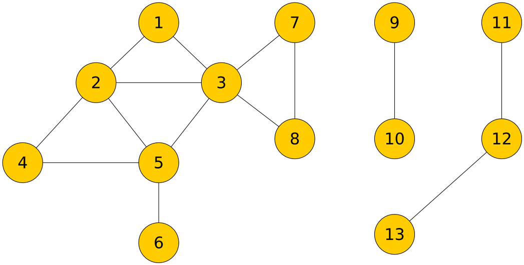
Busca em Profundidade

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

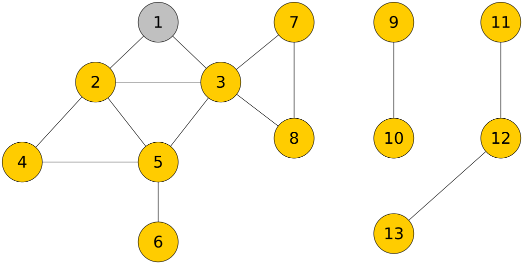
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



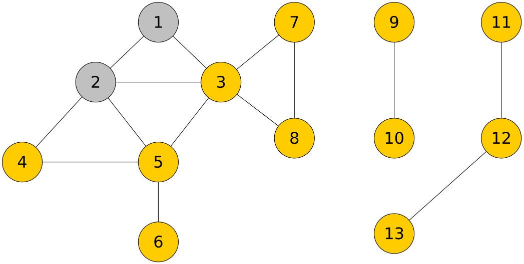
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



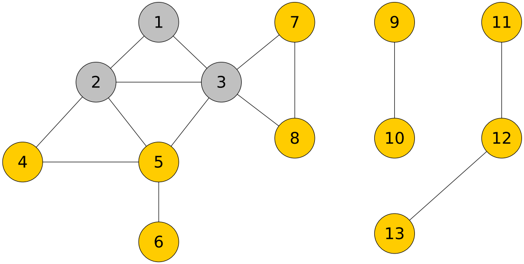
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



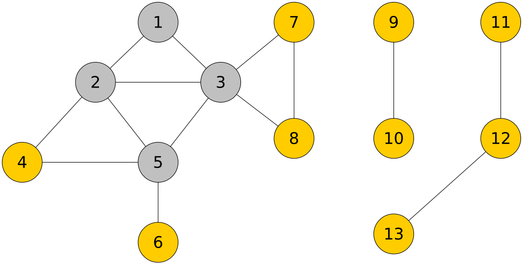
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



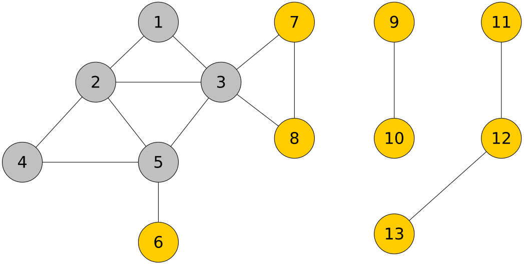
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



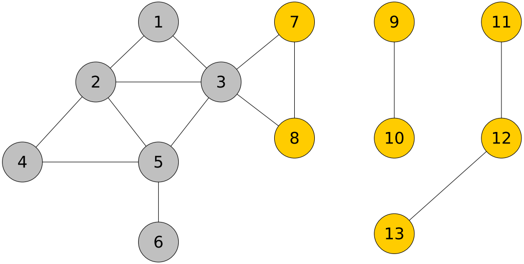
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



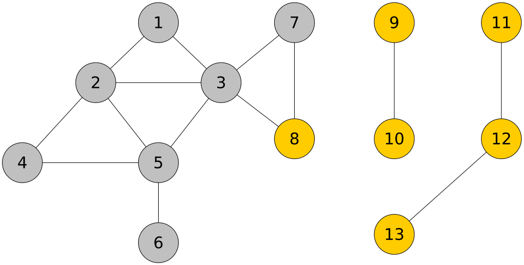
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



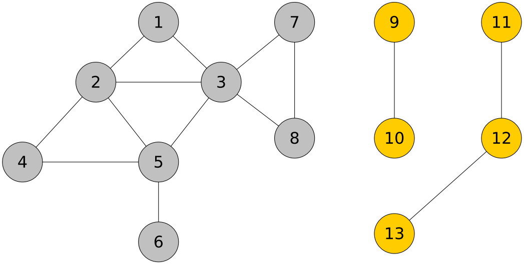
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



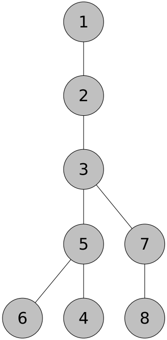
Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



Busca em Profundidade

- Exemplo: busca em profundidade a partir de 1



Busca em Profundidade

- 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

```
function BUSCA_PROFUNDIDADE(G (grafo), s (vertice_fonte))
    S ← cria_pilha_vazia();

    for all v ∈ G do                                  ▷ Inicializa todos os vértices de G
        v.visitado ← FALSO;                           ▷ Vértice não visitado
    end for

    push(s, S);

    while S não for vazia do
        u ← pop(Q)                                    ▷ Inicia-se com fonte

        if u.visitado = FALSO then
            u.visitado ← VERDADEIRO

            for all v ∈ adjacente(u) do
                push(v, S);
            end for
        end if
    end while
end function
```

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

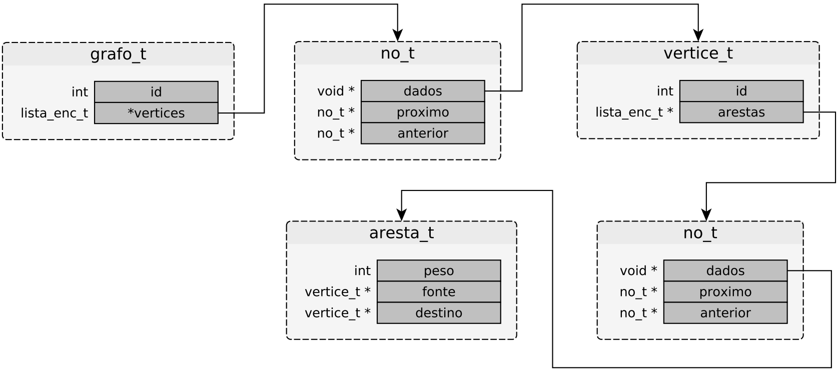
Desenhando grafos

- 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

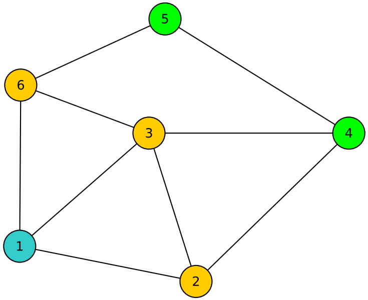
```
graph {
  1 -- 3
  1 -- 2
  2 -- 4
  2 -- 5
  2 -- 3
  3 -- 5
  4 -- 5
  5 -- 6
  3 -- 7
  3 -- 8
  7 -- 8
  9 -- 10
  11 --12
  12 -- 13
}
```

Tópico

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	1	0	1	0	1
4	0	1	1	0	1	0
5	0	0	0	1	0	1
6	1	0	1	0	1	0

Tópico

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