Teoria dos Grafos
Aula 4

Aula passada

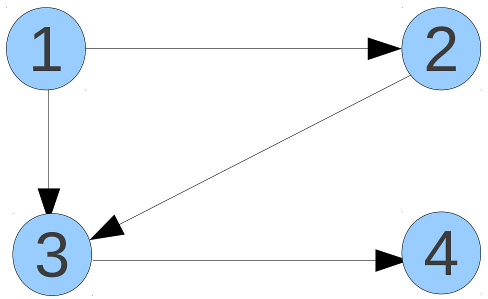
- Explorando grafos
- Mecanismos genéricos
- Busca em Lagura-BFS
- Busca em Profundidade-DFS
- Complexidade
- Conectividade

Aula de hoje

- Grafos direcionados
- Busca em grafos direcionados
- Ordenação topológica

Grafo Direcionado (Dígrafo)

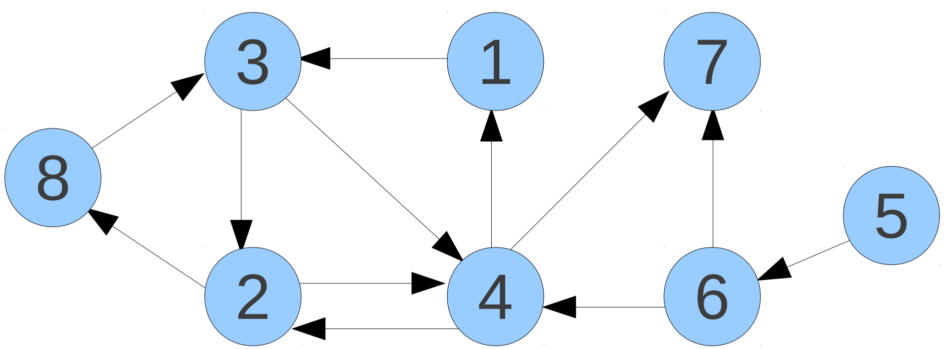
- Relacionamento não é simétrico!
  - A estar relacionado com B, **não** implica B estar relacionado com A
  - Exemplo de relacionamento assimétrico?
- Abstração: arestas têm “direção”
  - par de vértices é ordenado
- Exemplo: G = (V, E)
  - V = {1, 2, 3, 4}
  - E = {(1,2), (1,3), (2,3), (3,4)}



Grau

- Grau de entrada: número de arestas que “entram” em *v*: $|\{(*,v)\}|$
- Grau de saída: número de arestas que “saem” de *v*: $|\{(v,*)\}|$

- Exemplo: G = (V, E)



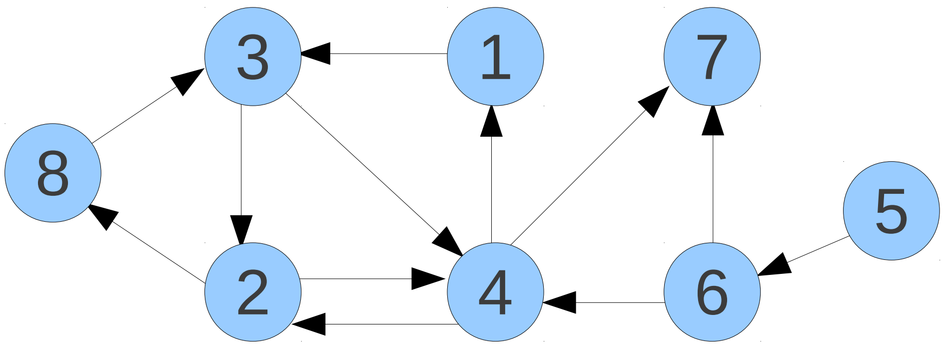
- $g_e(3) = ?$
- $g_s(3) = ?$
- $g_e(4) = ?$
- $g_s(7) = ?$

- Número máximo de arestas em G? ← n(n-1)
- Relação entre $g_e(v)$ e $g_s(v)$? ← Nenhuma

Caminho, Ciclo, Distância

- Mesma definição de antes!
- Respeitando o direcionamento das areas

- Exemplo: G = (V, E)

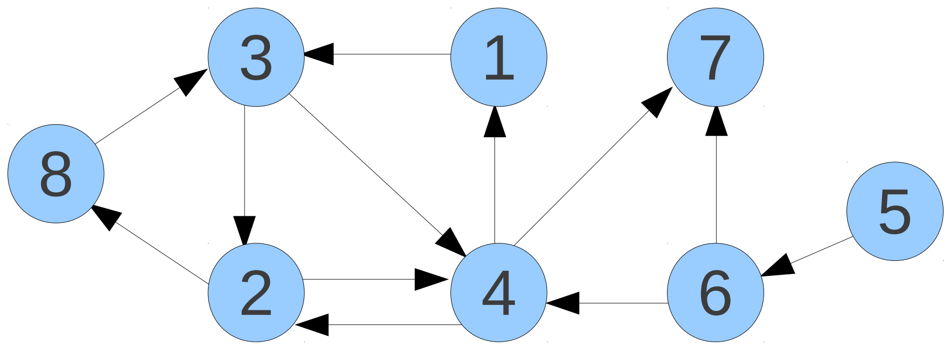


- Existe caminho de 5 para 2?
- Ciclo que contém 1?
- d(2,8) = ?
- d(8,2) = ?
- d(7,1) = ?

Fortemente Conexo

- Análogo a conexo (no caso não direcionado)
- Existe caminho entre qualquer par de vértices
  - mas caminho de *u* a *v*, **não** implica caminho de *v* a *u*

- Exemplo: G = (V, E)

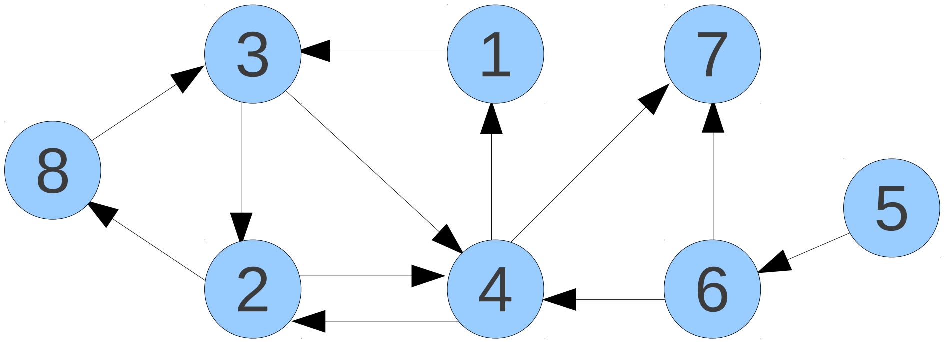


- É fortemente conexo?

Componentes Fortemente Conexos

- Análogo a componentes conexos
- Subgrafos maximais de G que são fortemente conexos

- Exemplo: G = (V, E)



- Componentes fortemente conexos?

Busca em Grafos Direcionados

Mesma idéia!

- Busca precisa respeitar direcionamento das arestas
  - (u, v) não é igual a (v, u)
- BFS e DFS praticamente idênticos
- Mesmos algoritmos, mesma complexidade

Mas há diferenças

Busca em Grafos Direcionados

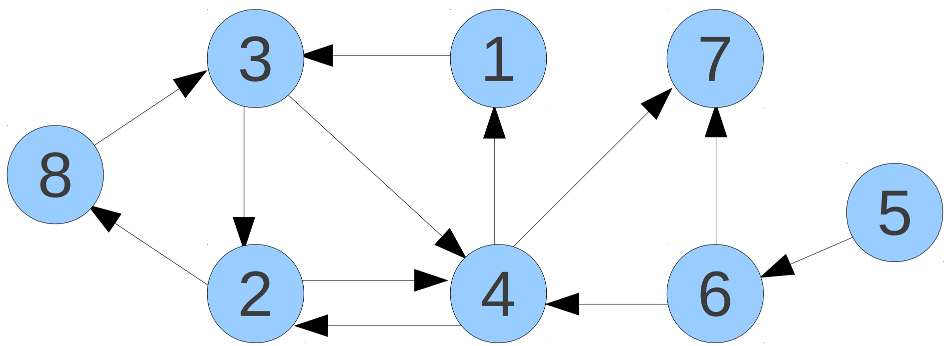
- Escolher vértice s (grafo direcionado)
- Executar BFS à partir de *s*
- Qual significado dos vértices marcados?

Vértices que s alcança!

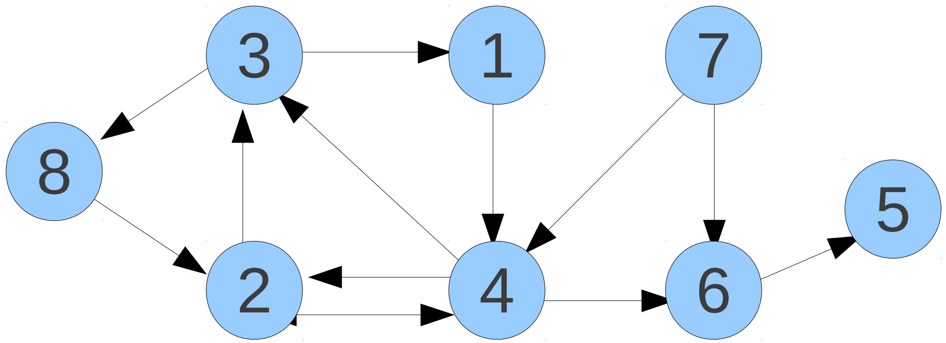
- Tais vértices alcançam *s*?
  - existe caminho de volta a *s*?

Busca em Grafos Direcionados

- Exemplo
  - determinar vértices que chegam em 1?



- Grafo revertido: G_{rev}



- Executar BFS à partir de 1
- Vértices marcados chegam em 1 no grafo original

Fortemente Conexo

- **Problema:** Como determinar se um grafo direcionado é fortemente conexo?

- Como fizemos no caso não-direcionado?
  - Escolher nó inicial, executar BFS, verificar marcação

Idéias?

- Mesmo princípio de antes!
  - Se *s* chega aos vértices *u* e *v*, e os vértices *u* e *v* chegam a *s*
  - Então *u* chega a *v*, via *s*

Fortemente Conexo

- **Problema:** Como determinar se um grafo direcionado é fortemente conexo?
- Escolher vértice *s* qualquer
- Executar BFS à partir de *s*
- Construir G_{rev}
- Executar BFS à partir de *s* em G_{rev}
- Se todos os vértices foram marcados nas duas buscas, então G é fortemente conexo

Complexidade?

Executando Tarefas

- **N** tarefas precisam ser executadas
- Tarefas são dependentes
  - ex. tarefa B só pode ser executada depois de A

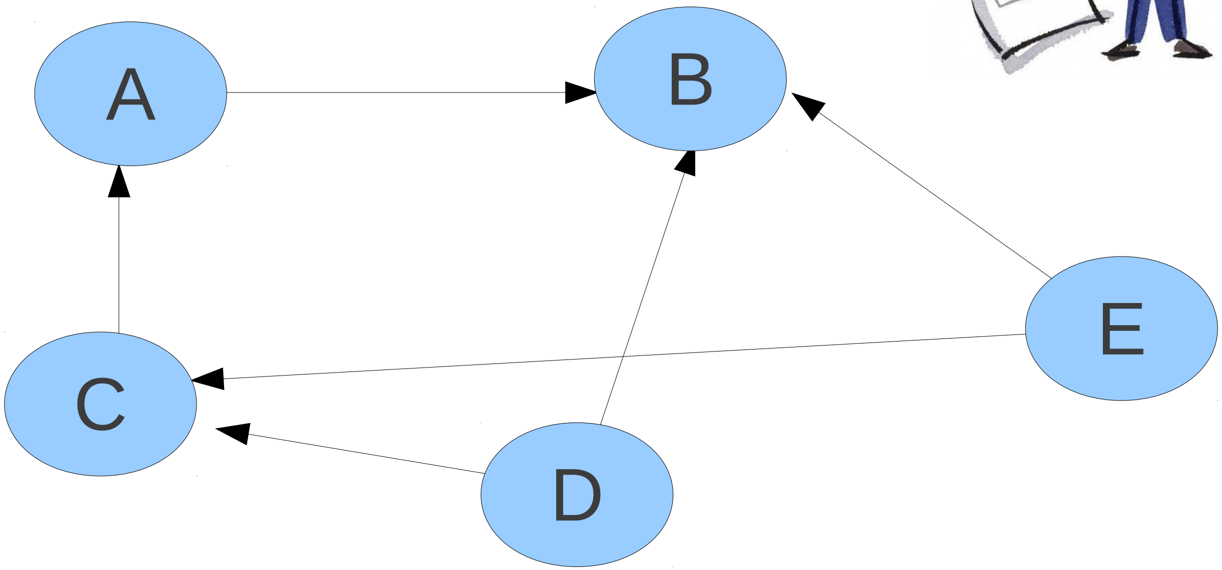
- **Problema:** Qual ordem de execução não viola as dependências?

Modelar problema utilizando grafos direcionados

Executando Tarefas

- Exemplo com 5 tarefas

B depende de A
A depende de C
C depende de D
B depende de E
B depende de D
C depende de E



- Qual é a ordem de execução?

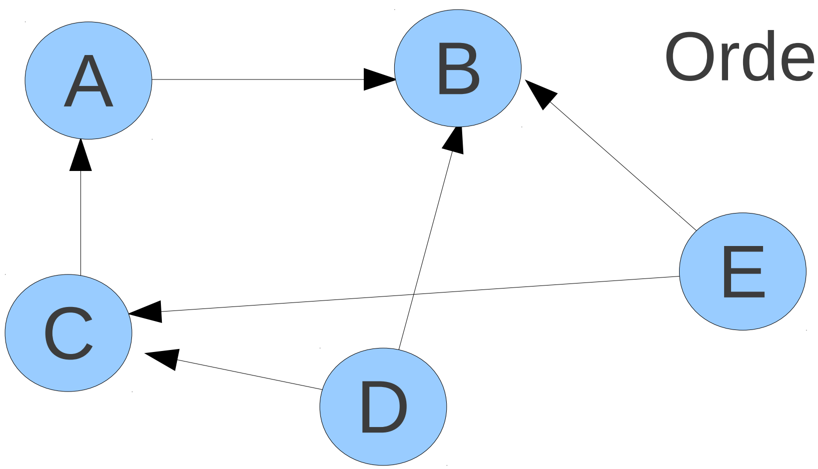
DAG

- Grafo direcionado acíclico (DAG, em inglês)
  - importante estrutura em grafos!
- Tarefas podem ser executadas, somente se grafo de dependência é um DAG
  - não podemos ter ciclos!
- Dado um DAG, como descobrir ordem de execução das tarefas?

Algoritmo (eficiente)!

Ordenação Topológica

- Ordenação dos vértices de forma que arestas "apontam" sempre para frente
- Dado grafo direcionado G
- Ordenação dos vértices de G: $v_1, v_2, \ldots, v_n$, tal que para toda aresta (v_i, v_j), $i < j$



Ordenação topológica?

$v_1 = E$

$v_2 = D$

$v_3 = C$

$v_4 = A$

$v_5 = B$

Ordenação Topológica

- Define uma ordem de execução das tarefas
- **Problema**: Dado um DAG, como determinar uma ordernação topológica?

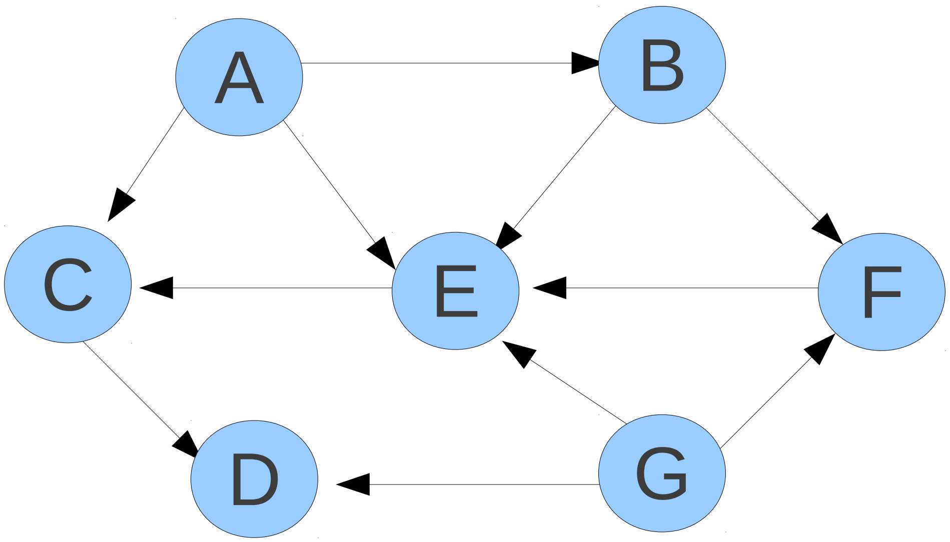
Idéias?

Ordenação Topológica

- Algoritmo

```
1.Ordem = 0
2.Enquanto |V| > 0 faça
3.     Encontrar u com grau de entrada zero
4.     Ordem = Ordem + u
5.     Remover u do grafo G
```

- Exemplo



Ordenação Topológica

- Complexidade?

```
1.Ordem = 0
2.Enquanto |V| > 0 faça
3.     Encontrar u com grau de entrada zero
4.     Ordem = Ordem + u
5.     Remover u do grafo G
```

- Depende do tempo necessário para encontrar u
- Procurar sequencialmente: O(n)
- Complexidade $O(n^2)$

Busca em Grafos Direcionados

- Como descobrir quais vértices alcançam *s*?
- Solução ruim
  - Para cada vértice do grafo, executar BFS e verificar se *s* é marcado

Idéias melhores?

- **Inverter a direção das arestas!**
  - executar BFS no novo grafo à partir de *s*
  - vértices que *s* alcançou, alcançam *s* no grafo original!

Ordenação Topológica

- Como melhorar complexidade?
- Complexidade: O(m + n)