

Teoria dos Grafos

Aula 9

Aula passada

- Grafos direcionados
- Busca em grafos direcionados
- Ordenação topológica

Aula de hoje

- Grafos com pesos
- Dijkstra
- Implementação
- Fila de prioridades e Heap
- Dijkstra (o próprio)

Relacionamentos de Peso

- Relacionamentos entre objetos nem sempre são idênticos (mesma intensidade)

■ Exemplos?

■ Amizade

- mais ou menos amigo (Orktut)

■ Distância física

- perto ou longe

■ Tempo de traslado

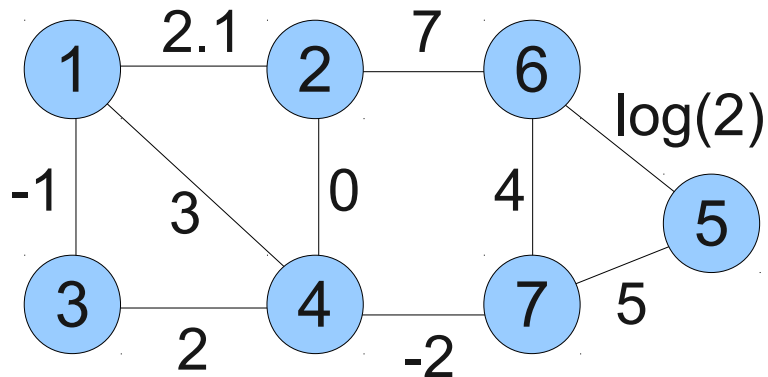
- mais ou menos tempo

**Como
representar tais
relacionamentos?**

Grafos com Pesos

- Anotar arestas do grafo com “intensidade” do relacionamento
 - peso da aresta (*weight*)
 - função $w(e)$ retorna peso da aresta e
 - Ex. $w : E \rightarrow \mathbb{R}$

■ Graficamente



$$\begin{aligned}w((2,6)) &= 7 \\w((5,6)) &= \log(2) \\w((2,4)) &= 0\end{aligned}$$

Distância com Pesos

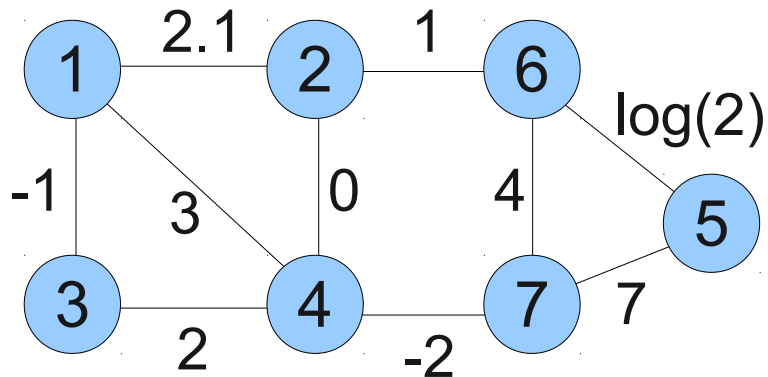
- Comprimento de um caminho

- **soma** dos pesos das arestas que definem o caminho

- Distância

- comprimento do **menor** caminho simples de entre dois vértices

- Graficamente



Comprimento do caminho 1,2,6,7?

Comprimento do caminho 1,3,4,7?

Comprimento do caminho 1,4,7?

Distância entre 1 e 7?

Distância entre 3 e 5?

Grafos Direcionados com Peso

- Relacionamentos assimétricos com pesos (diferentes intensidades)
- Exemplo?
- Mesma idéia: anotar arestas do grafo com “intensidade” do relacionamento
 - peso da aresta (*weight*)
 - função $w(e)$ retorna peso da aresta e
 - aresta direcionada, pesos potencialmente diferentes

Viagem entre Cidades

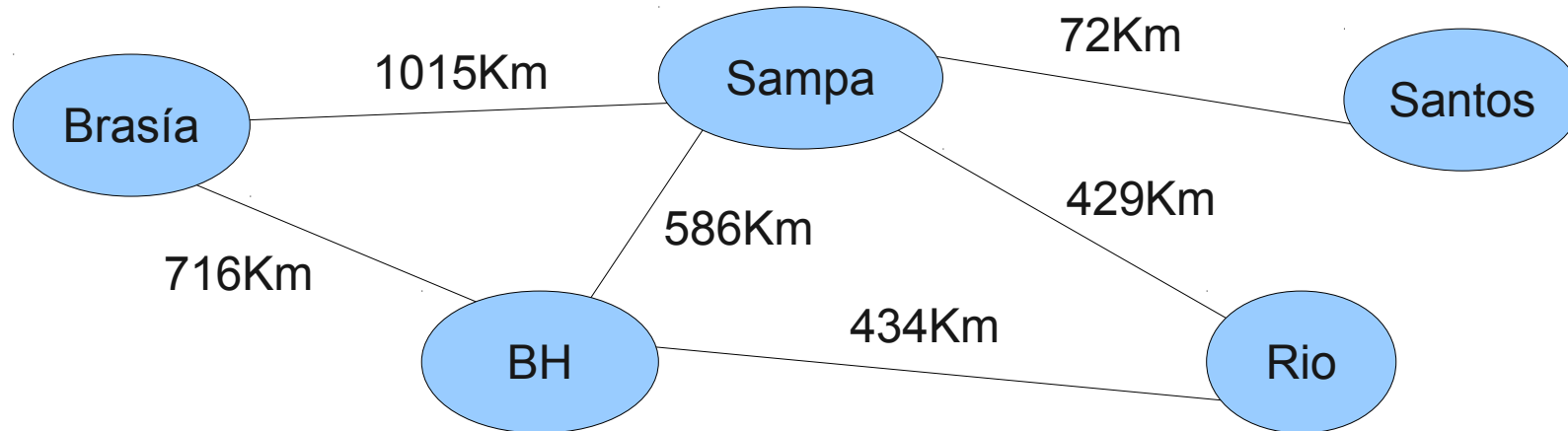


- Cidades brasileiras
- Estradas entre cidades

- **Problema 1:** Como saber se duas cidades estão “conectadas” por estradas?
- **Problema 2:** Qual é o menor (melhor) caminho entre duas cidades?

Viagem entre Cidades

- Abstração via grafos com pesos



- **Problema 1:** Como as cidades estão “conectadas”?

Resolvido!

- **Problema 2:** Qual é o menor (melhor) caminho entre duas cidades?

Distância em Grafos com Peso

- Calcular caminho mais curto entre cidades é calcular a distância em grafos com peso
 - pesos sempre maior que zero
- Dado G , com pesos
- Qual é a distância do vértice s ao d ?

Como resolver este problema?

- Como resolvemos o problema sem pesos?
- Podemos adaptar algumas idéias?

Distância em Grafos com Peso

■ Idéia

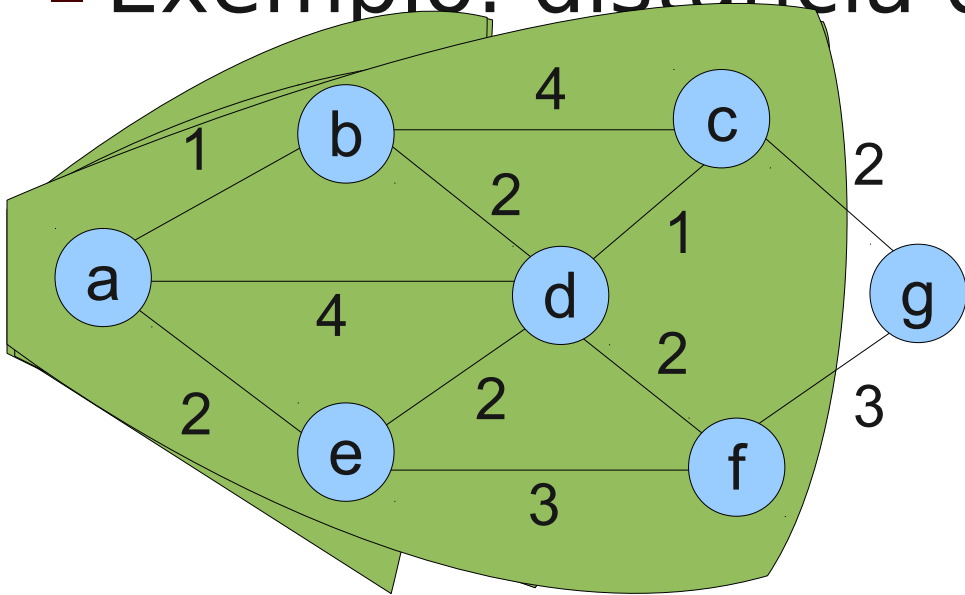
- Partindo de s , expandir os caminhos, incluindo vértices

Mas em que ordem?

- Na ordem em que caminhos mínimos sejam garantidos!
- Expandir caminhos mínimos até chegar em d de maneira gulosa!

Distância em Grafos com Peso

- Exemplo: distância de a à g ?



**Algoritmo
de Dijkstra!**

- Começar em a , expandir
- Qual próximo vértice?
- Qual vértice nos dá caminho mínimo garantido?

Algoritmo de Dijkstra

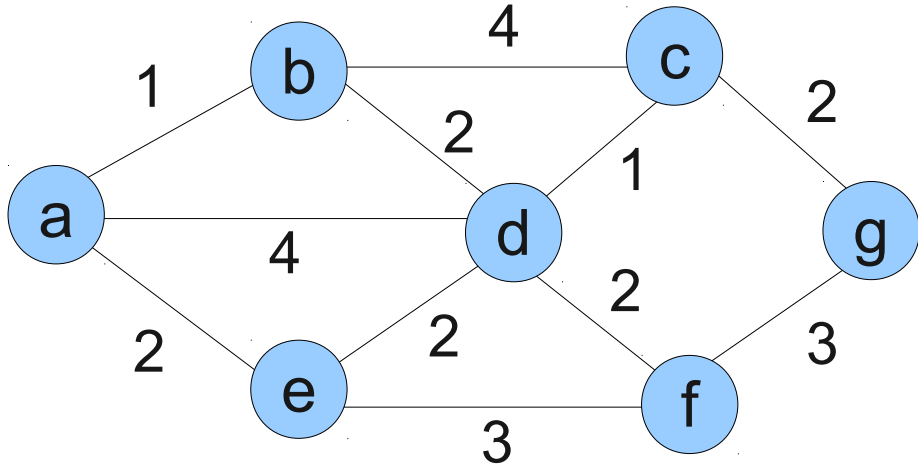
- Como tornar a idéia em algoritmo?
 - adicionar o vértice para o qual temos o menor caminho
- **Idéias:**
 - Manter dois conjuntos de vértices
 - Manter comprimento do menor caminho conhecido até o momento para cada vértice
 - Adicionar o vértice de menor caminho
 - Atualizar distâncias

Algoritmo de Dijkstra

1. Dijkstra(G, s)
2. Para cada vértice v
3. $\text{dist}[v] = \text{infinito}$
4. Define conjunto $S = \emptyset$ // vazio
5. $\text{dist}[s] = 0$
6. Enquanto $S \neq V$
7. Selecione u em $V - S$, tal que $\text{dist}[u]$ é mínima
8. Adicione u em S
9. Para cada vizinho v de u faça
10. Se $\text{dist}[v] > \text{dist}[u] + w((u,v))$ então
11. $\text{dist}[v] = \text{dist}[u] + w((u,v))$

■ Como o algoritmo executa?

Executando o Algoritmo



- Manter tabela com passos e distâncias

| Conjunto S | d(a) | d(b) | d(c) | d(d) | d(e) | d(f) | d(g) |
|-----------------|------|------|------|------|------|------|------|
| {} | 0 | inf | inf | inf | inf | inf | inf |
| {a} | - | 1 | inf | 4 | 2 | inf | inf |
| {a,b} | | - | 5 | 3 | 2 | inf | inf |
| {a,b,e} | | | 5 | 3 | - | 5 | inf |
| {a,b,e,d} | | | 4 | - | | 5 | inf |
| {a,b,e,d,c} | | | - | | | 5 | 6 |
| {a,b,e,d,c,f} | | | | | | - | 6 |
| {a,b,e,d,c,f,g} | | | | | | | - |

Analizando o Algoritmo

- Provar que algoritmo sempre produz resultado correto – caminho mínimo entre dois vértices

Teorema:

- Considere um vértice u pertencente ao conjunto S em qualquer ponto do algoritmo. Temos que “ $\text{dist}[u]$ ” é a distância entre s e u .

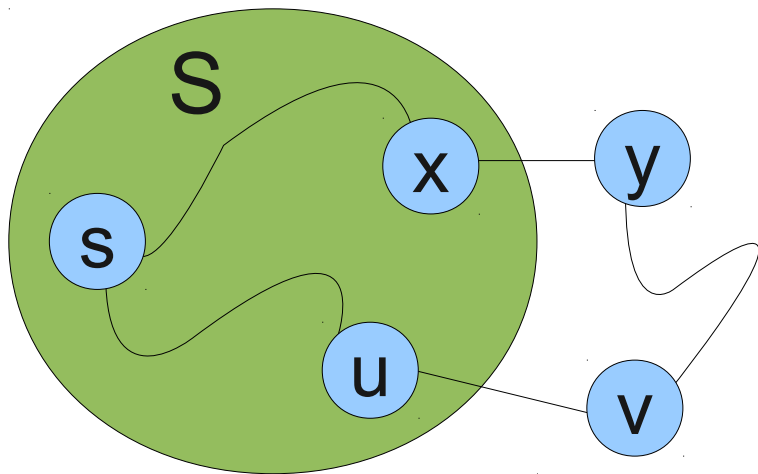
Prova

- Prova por indução no tamanho de S
- Caso base: $|S| = 1$
 - $S = \{s\}$, $\text{dist}[s] = 0$, devido inicialização
- Hipótese: $|S| = k$
 - Para $|S| = k$, assumamos que $\text{dist}[u]$ é igual a distância entre s e u
- Caso geral: $|S| = k + 1$
 - $k+1$ adiciona v à S , dando origem a P_v
 - seja (u,v) última aresta no caminho P_v
 - Pela hipótese, P_u é caminho mínimo $s-u$

Prova

- Considere outro caminho $s-v$, P , que não passa por u
- Precisamos provar que P é maior (ou igual) a P_v
- P passa pela aresta (x,y) , com x em S e y fora de S (para algum x e y qualquer)

■ Situação:



- No passo $k+1$, algoritmo escolhe v para adicionar a S (e não y)
- Então, caminho $s-y$ é maior ou igual a P_v
- Como $\text{dist}(y,v) \geq 0$, caminho P será maior ou igual a P_v
- Logo, P_v é caminho mínimo e $\text{dist}[v]$ será distância até v

Complexidade

■ Qual é a complexidade do algoritmo?

1. Dijkstra(G, s)
2. Para cada vértice v
3. $\text{dist}[v] = \text{infinito}$
4. Define conjunto $S = \emptyset$ // vazio
5. $\text{dist}[s] = 0$
6. Enquanto $S \neq V$
7. Selecione u em $V - S$, tal que $\text{dist}[u]$ é mínima
8. Adicione u em S
9. Para cada vizinho v de u faça
10. Se $\text{dist}[v] > \text{dist}[u] + w((u, v))$ então
11. $\text{dist}[v] = \text{dist}[u] + w((u, v))$

■ Depende do tempo para selecionar u tal que $\text{dist}[u]$ seja mínima!

Complexidade

- Algoritmo simples
 - Percorre vértices e encontra $\text{dist}[u]$ mínimo
 - Complexidade? ← $O(n^2)$

Outra idéia?

- Fila de prioridades – muito usada em grafos!
 - Chave: distâncias
 - Extrair o mínimo (vértice com menor distância)
 - Atualizar chaves (distâncias)

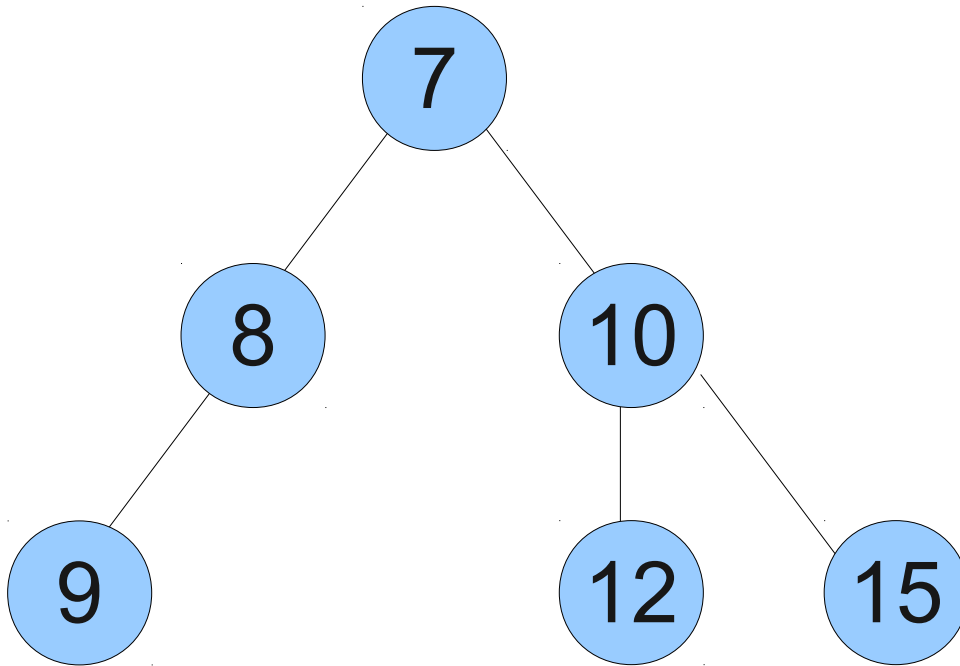
Fila de Prioridade

- Estrutura de dados poderosa (*priority queue*)
- Mantém um conjunto de elementos S
- Cada elemento possui uma chave (número de prioridade)
- Permite inserir, remover e modificar elementos de S através de sua chave
- Permite remover menor chave (maior prioridade)
- Complexidade para inserir ou modificar elemento: $O(\log n)$

Heap

- Como implementar Fila de Prioridade?
- **Heap!**
- O que é um heap?
- Estrutura de dados em árvore
- Armazena conjunto de elementos, todos associados a uma chave
- Chave dos elementos define árvore
- *Heap order*: $\text{chave}(u) \leq \text{chave}(v)$, quando u é pai de v

Heap - Exemplo



- Chaves podem inseridas, removidas ou mudar de valor
- Heap precisa ser reajustado - *heapify*
 - custo $O(\log n)$
- Manter árvore “balanceada”

Complexidade

■ Usando um heap?

1. Dijkstra(G, s)
2. Para cada vértice v
3. $\text{dist}[v] = \text{infinito}$
4. Define conjunto $S = \emptyset$ // vazio
5. $\text{dist}[s] = 0$
6. Enquanto $S \neq V$
7. Selecione u em $V-S$, tal que $\text{dist}[u]$ é mínima
8. Adicione u em S
9. Para cada vizinho v de u faça
10. Se $\text{dist}[v] > \text{dist}[u] + w((u,v))$ então
11. $\text{dist}[v] = \text{dist}[u] + w((u,v))$

■ Cada operação no heap: $O(\log n)$

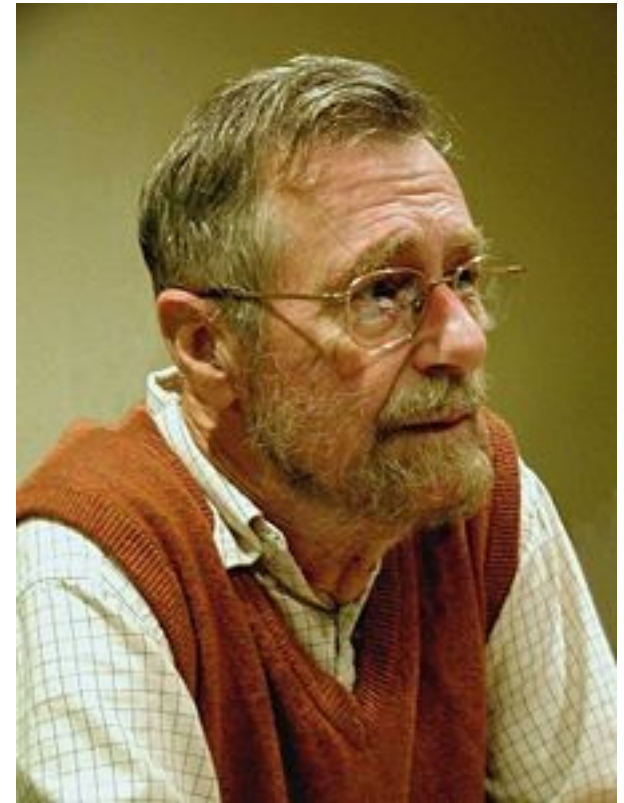
- n operações de remoção

- m operações de atualização

■ Complexidade: $O((n+m)\log n) = O(m \log n)$

Dijkstra, o Próprio

- Edsger Wybe Dijkstra
- Professor e pesquisador na área de Ciência da Computação
- Recebeu Turing Award 1972
 - mais renomado prêmio da Computação
- Contribuições fundamentais em ling. de programação e verificação formal
- Algoritmo de Dijkstra utilizado em vários sistemas (redes, GPS, etc)



11/5/1930 – 6/8/2002