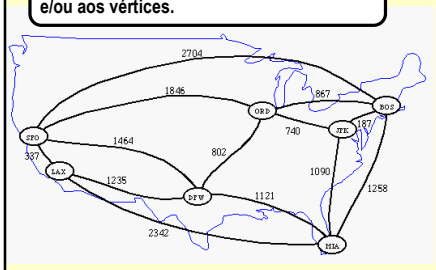
Grafos – aula 3

Relembrando ....

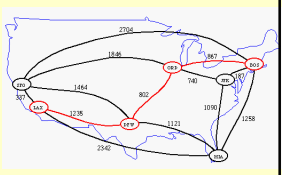
Um grafo é *valorado* (ou *ponderado*) se possuir valores associados às linhas e/ou aos vértices.



- Rota mais curta entre 2 aeroportos
- Caminho mais curto entre 2 máquinas, para transmissão de dados via internet
- Distâncias mais curtas entre cidades

Problema do caminho mínimo

- Dado um grafo ponderado e dois vértices ***v*** e ***u***, queremos encontrar o caminho de custo total mínimo entre ***v*** e ***u***
- O comprimento (ou peso) de um caminho ***P*** é a soma dos pesos das arestas que compõem ***P***
- Exemplo: qual o menor caminho de Boston a Los Angeles?



Problema do caminho mínimo

Propriedade 1:

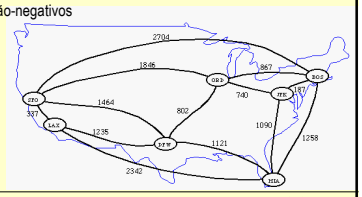
Um sub-caminho de um caminho mínimo é, ele próprio, um caminho mínimo

Propriedade 2:

Existe uma árvore de caminhos mínimos partindo de um vértice inicial até todos os outros vértices

Algoritmo de Dijkstra

- Implementação do método **guloso**
- O algoritmo de Dijkstra encontra o caminho mínimo a partir de um vértice de partida ***v*** até todos os outros vértices
- Funciona em:
  - Grafos dirigidos e não-dirigidos
  - Grafos com pesos de arestas não-negativos



Algoritmo de Dijkstra: funcionamento

- Dado um vértice ***v*** (vértice de partida), o algoritmo calcula, para cada vértice ***u***, a distância mínima até ***v***
- O algoritmo mantém o conjunto de vértices (**nuvem C**) cujas distâncias tenham sido computadas
- Cada vértice ***u*** possui um rótulo ***D[u]*** associado.
  - ***D[u]*** armazena uma aproximação da distância entre ***v*** e ***u***.
  - O algoritmo altera ***D[u]*** cada vez que encontra uma distância menor
- Quando um vértice ***u*** é adicionado à nuvem, seu rótulo ***D[u]*** é igual a distância final (até o momento) entre ***v*** e ***u***
- Condição inicial
  - D[v] = 0 (distância de ***v*** até ***v*** é zero)
  - D[u] = ∞ para u ≠ v

Algoritmo: expandindo a nuvem

- A nuvem ***C*** é inicialmente vazia
- Repita até que todos os vértices tenham sido adicionados à nuvem ***C***:
  - Selecione o vértice ***u*** que não esteja em ***C*** e que tenha o menor rótulo ***D[u]***
    - na primeira iteração, será escolhido o vértice inicial ***v*** – ver condições iniciais!
  - Coloque-o em ***C***
  - Atualize os rótulos dos vértices adjacentes a ***u*** da seguinte forma:

```
para cada vértice z adjacente a u faça
  se z não está na nuvem C, então
      se D[u] + weight(u,z) < D[z] então
          D[z] = D[u] + weight(u,z)
```


Algoritmo: expandindo a nuvem

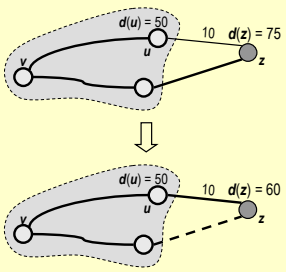
- A nuvem ***C*** é inicialmente vazia
- Repita até que todos os vértices tenham sido adicionados à nuvem ***C***:
  - Selecione o vértice ***u*** que não esteja em ***C*** e que tenha o menor rótulo ***D[u]***
    - na primeira iteração, será escolhido o vértice inicial ***v*** – ver condições iniciais!
  - Coloque-o em ***C***
  - Atualize os rótulos dos vértices adjacentes a ***u*** da seguinte forma:

```
para cada vértice z adjacente a u faça
  se z não está na nuvem C, então
      se D[u] + weight(u,z) < D[z] então
          D[z] = D[u] + weight(u,z)
```

Estruturas de Dados - Grafos

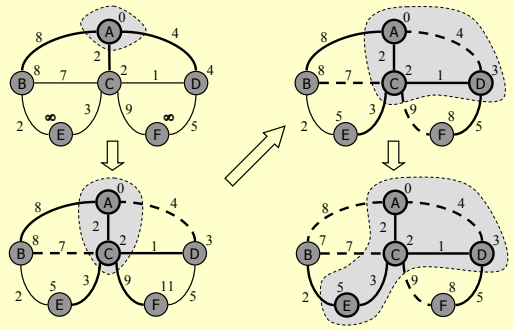
Atualizando a distância d(z)

- Considere uma aresta ***e = (u,z)*** tal que
  - ***u*** é o último vértice adicionado à nuvem
  - ***z*** não está na nuvem
- A atualização da distância ***d(z)*** em função da aresta ***e*** é realizada para encontrar a situação ótima
- O **relaxamento da aresta *e* = (*u*,*z*)** atualiza a distância ***d***(***z***)



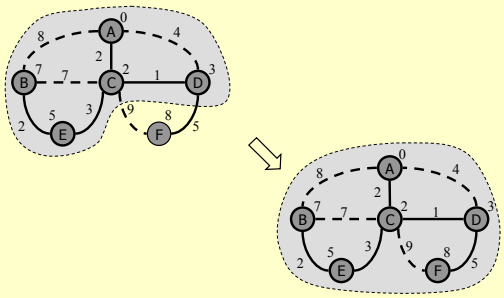
Estruturas de Dados - Grafos

Exemplo #1



Estruturas de Dados - Grafos

Exemplo #1



Estruturas de Dados - Grafos

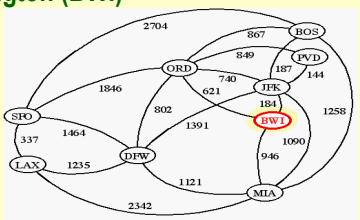
Algoritmo de Dijkstra

Algoritmo CaminhoMínimo(G,s)
Entrada: grafo G e vértice inicial s
Saída: rótulo D[u] para cada vértice u de G

Q ← lista ordenada por distância
para todo *v* ∈ *G*
  se *v = s*
    v.setDistance(0) //a distância para s é inicializada em zero
  senão
    v.setDistance(∞) // a distância de s para todos os outros nodos é inicializada com infinito
    enquanto *Q não for vazia* **faça**
      u ← *Q.removeMin*() // remove de Q o elemento u com a menor distância para v
      para todo vértice *z* **adjacente a** *u* **tal que** *z*
      esteja em *Q* **faça**
        se *D[u] + w(u,z) < D[z]* **então**
        z.setDistance(D[u]+w(u,z)) //relaxa a aresta (u,z)
        reordene lista *Q*
Retorna o rótulo D[u] para cada vértice u de G

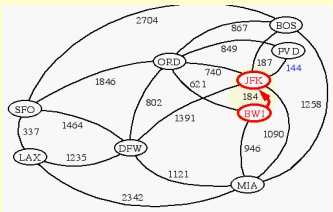
Estruturas de Dados - Grafos

Exemplo #2: caminho mais curto começando por Washington (BWI)



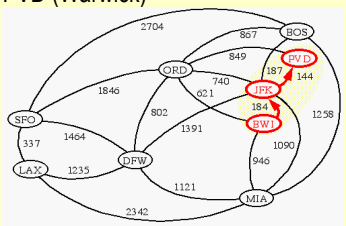
	parent	distance
BOS		∞
BWI		0
DFW		∞
JFK	BWI	184
LAX		∞
MIA	BWI	946
ORD	BWI	621
PVD		∞
SFO		∞

- JFK é o mais próximo...



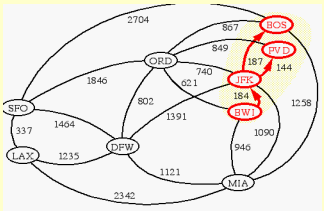
	parent	distance
BOS	JFK	371
BWI		0
DFW	JFK	1575
JFK	BWI	184
LAX		∞
MIA	BWI	946
ORD	BWI	621
PVD	JFK	328
SFO		∞

- Seguido por PVD (Warwick)



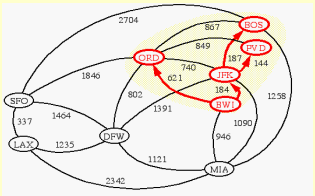
	parent	distance
BOS	JFK	371
BWI		0
DFW	JFK	1575
JFK	BWI	184
LAX		∞
MIA	BWI	946
ORD	BWI	621
PVD	JFK	328
SFO		∞

- Boston (BOS) é o próximo



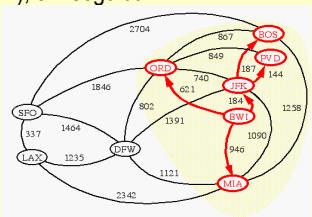
	parent	distance
BOS	JFK	371
BWI		0
DFW	JFK	1575
JFK	BWI	184
LAX		∞
MIA	BWI	946
ORD	BWI	621
PVD	JFK	328
SFO	BOS	3075

- Chicago (ORD) segue



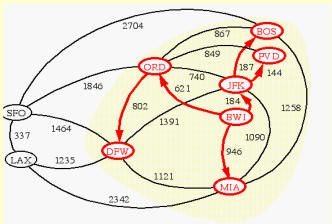
	parent	distance	
BOS	JFK	371	
BWI		0	
DFW	ORD	1423	note that D for DWF was adjusted on this turn
JFK	BWI	184	
LAX		∞	
MIA	BWI	946	
ORD	BWI	621	
PVD	JFK	328	also for SFO
SFO	ORD	2467	

- Miami (MIA), em seguida



	parent	distance
BOS	JFK	371
BWI		0
DFW	JFK	1423
JFK	BWI	184
LAX	MIA	3288
MIA	BWI	946
ORD	BWI	621
PVD	JFK	328
SFO	BOS	2467

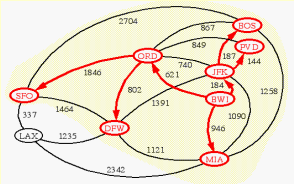
- E depois Dallas (DFW)



	parent	distance
BOS	JFK	371
BWI		0
DFW	JFK	1423
JFK	BWI	184
LAX	DFW	2658
MIA	BWI	946
ORD	BWI	621
PVD	JFK	328
SFO	BOS	2467

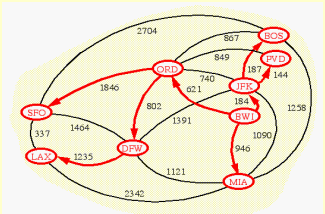
and D for LAX gets updated

- San Francisco (SFO): o próximo



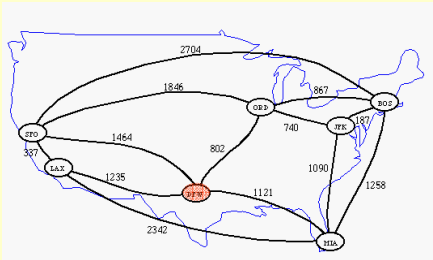
	parent	distance
BOS	JFK	371
BWI		0
DFW	ORD	1423
JFK	BWI	184
LAX	MIA	2658
MIA	BWI	946
ORD	BWI	621
PVD	JFK	328
SFO	BOS	2467

- Los Angeles (LAX) é o último



	parent	distance
BOS	JFK	371
BWI		0
DFW	ORD	1423
JFK	BWI	184
LAX	MIA	2658
MIA	BWI	946
ORD	BWI	621
PVD	JFK	328
SFO	BOS	2467

Exercício



Exercício

- Com base no que foi apresentado em aula a respeito de grafos, escreva a estrutura de dados grafo para grafos não-dirigidos, usando uma matriz de adjacência
  - Demonstre as estruturas de dados necessárias para armazenar um grafo
  - Liste as operações que podem ser realizadas
- Implemente o algoritmo de Dijkstra para essa estrutura.