Distância

Distância

O comprimento de um caminho é o número de arestas, contadas com repetição (se o caminho não for simples).

Distância

O comprimento de um caminho é o número de arestas, contadas com repetição (se o caminho não for simples).

A distância $\mathrm{d}(u, v)$ do vértice u ao vértice v é o menor comprimento de um caminho de u a v. Se v não é acessível a partir de u, definimos $\mathrm{d}(u, v) = \infty$.

1. $\mathrm{d}(u, u) = 0$.

Distância

O comprimento de um caminho é o número de arestas, contadas com repetição (se o caminho não for simples).

A distância $\mathrm{d}(u, v)$ do vértice u ao vértice v é o menor comprimento de um caminho de u a v. Se v não é acessível a partir de u, definimos $\mathrm{d}(u, v) = \infty$.

1. $\mathrm{d}(u, u) = 0$.
2. $\mathrm{d}(u, v) \leq \mathrm{d}(u, w) + \mathrm{d}(w, v)$.

Distância

O comprimento de um caminho é o número de arestas, contadas com repetição (se o caminho não for simples).

A distância $\mathrm{d}(u, v)$ do vértice u ao vértice v é o menor comprimento de um caminho de u a v. Se v não é acessível a partir de u, definimos $\mathrm{d}(u, v) = \infty$.

1. $\mathrm{d}(u, u) = 0$.
2. $\mathrm{d}(u, v) \leq \mathrm{d}(u, w) + \mathrm{d}(w, v)$.

Distância

O comprimento de um caminho é o número de arestas, contadas com repetição (se o caminho não for simples).

A distância $\mathrm{d}(u, v)$ do vértice u ao vértice v é o menor comprimento de um caminho de u a v. Se v não é acessível a partir de u, definimos $\mathrm{d}(u, v) = \infty$.

1. $\mathrm{d}(u, u) = 0$.
2. $\mathrm{d}(u, v) \leq \mathrm{d}(u, w) + \mathrm{d}(w, v)$.

mas, em geral, $\mathrm{d}(u, v) \neq \mathrm{d}(v, u)$.

Distância

O comprimento de um caminho é o número de arestas, contadas com repetição (se o caminho não for simples).

A distância $d(u, v)$ do vértice u ao vértice v é o menor comprimento de um caminho de u a v. Se v não é acessível a partir de u, definimos $d(u, v) = \infty$.

1. $d(u, u) = 0$.
2. $d(u, v) \leq d(u, w) + d(w, v)$.

mas, em geral, $d(u, v) \neq d(v, u)$.

A igualdade vale para digrafos simétricos e grafos.

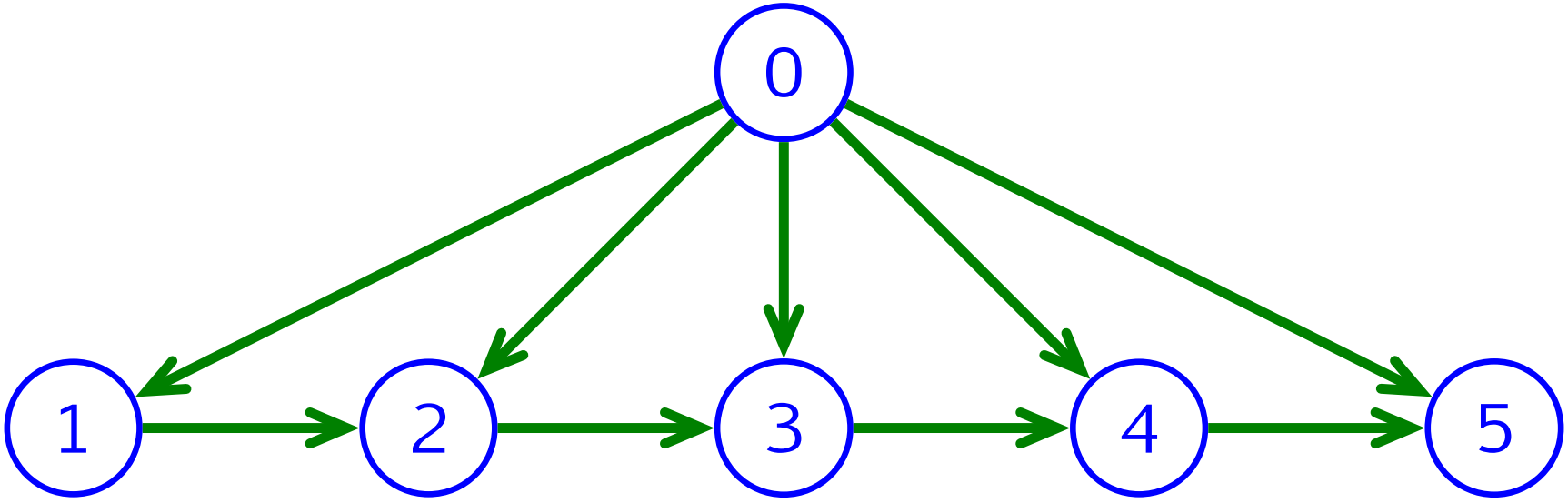
Caminhos compridos

Caminhos compridos

Uma DFS pode encontrar caminhos longos, mesmo que existam outros mais curtos.

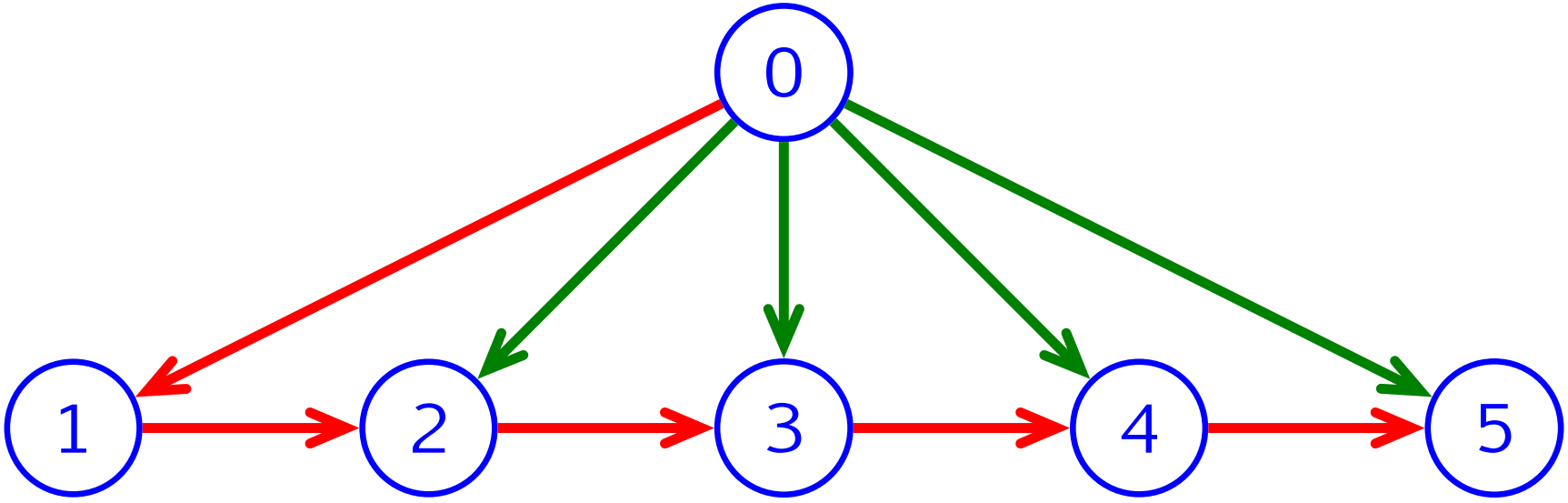
Caminhos compridos

Uma DFS pode encontrar caminhos longos, mesmo que existam outros mais curtos.



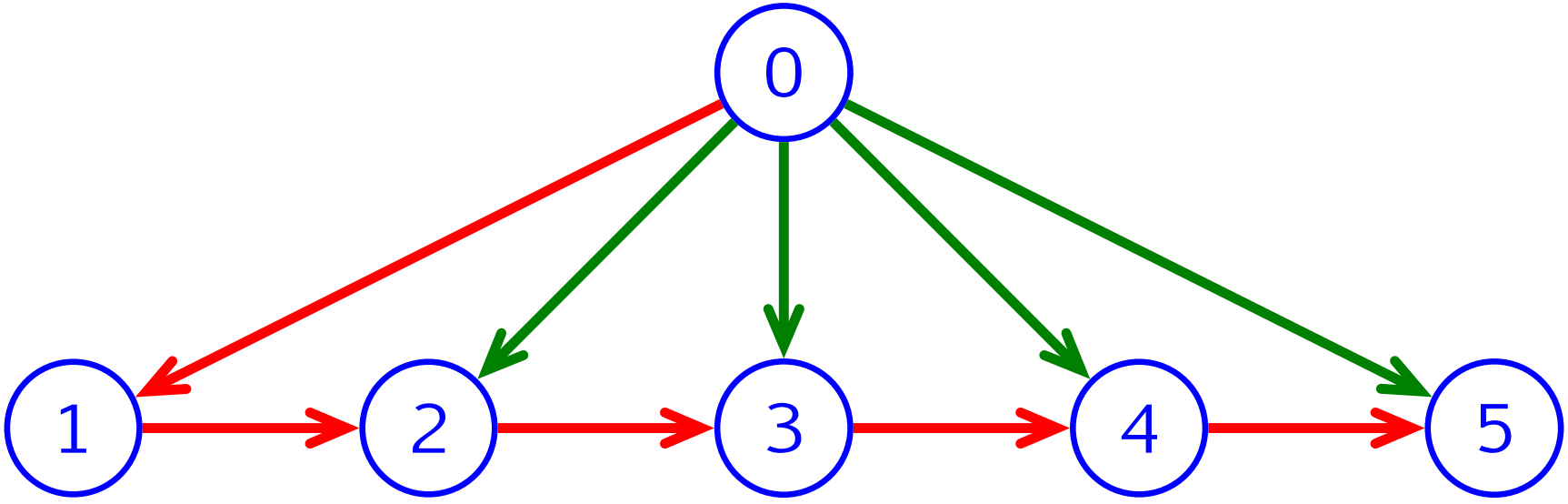
Caminhos compridos

Uma DFS pode encontrar caminhos longos, mesmo que existam outros mais curtos.



Caminhos compridos

Uma DFS pode encontrar caminhos longos, mesmo que existam outros mais curtos.



Não serve para achar distâncias.

BFS

BFS

Busca em largura, ou BFS (breadth first search) é um esquema que, dados um grafo G e um vértice s determina todos os vértices acessíveis a partir de s.

BFS

Busca em largura, ou BFS (breadth first search) é um esquema que, dados um grafo G e um vértice s determina todos os vértices acessíveis a partir de s.

Como na DFS, os vértices acessados ficam organizados numa árvore, só que:

> *Para cada v, a distância de s a v na árvore é a distância em G.*

BFS

Busca em largura, ou BFS (breadth first search) é um esquema que, dados um grafo G e um vértice s determina todos os vértices acessíveis a partir de s.

Como na DFS, os vértices acessados ficam organizados numa árvore, só que:

> *Para cada v, a distância de s a v na árvore é a distância em G.*

BFS

Busca em largura, ou BFS (breadth first search) é um esquema que, dados um grafo G e um vértice s determina todos os vértices acessíveis a partir de s.

Como na DFS, os vértices acessados ficam organizados numa árvore, só que:

> *Para cada v, a distância de s a v na árvore é a distância em G.*

Como na DFS, os vértices podem ter três estados: inicial, descoberto, finalizado.

BFS

Busca em largura, ou BFS (breadth first search) é um esquema que, dados um grafo G e um vértice s determina todos os vértices acessíveis a partir de s.

Como na DFS, os vértices acessados ficam organizados numa árvore, só que:

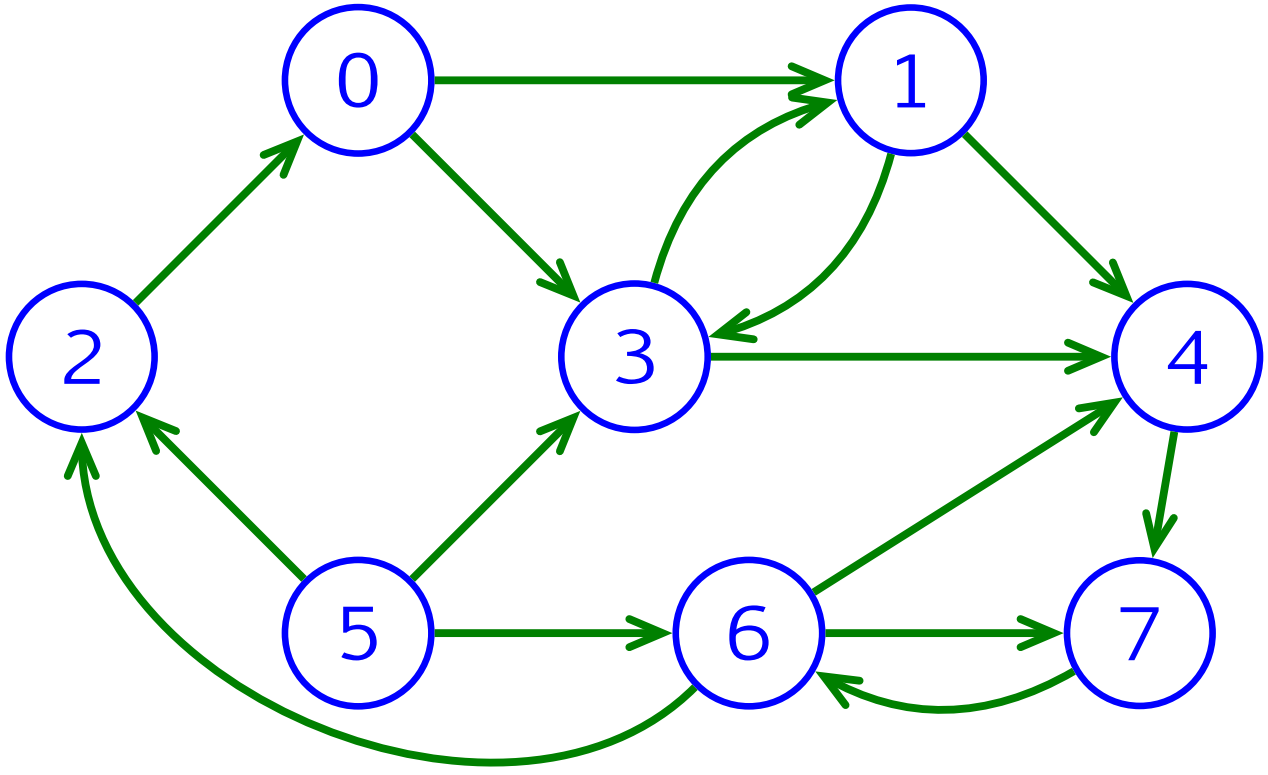
> *Para cada v, a distância de s a v na árvore é a distância em G.*

Como na DFS, os vértices podem ter três estados: inicial, descoberto, finalizado.

Os vértices descobertos são colocados numa fila (no python/sage, use `deque` do módulo `collections`).

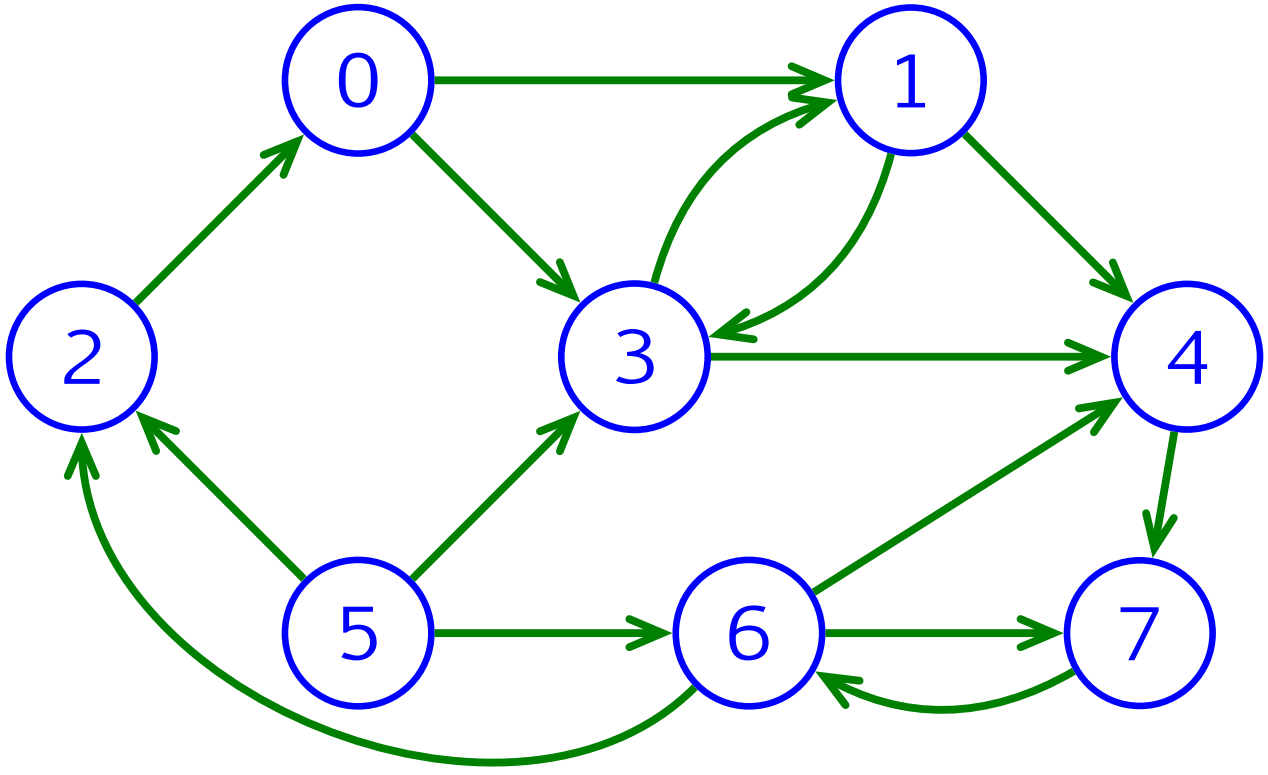
```
BFS(G,s)
    for each vertex u ∈ G.V
        u.state, u.d, u.sob = inicial, ∞, nil
    s.state, s.d = descoberto, 0
    ENFILEIRA(F, s)
    while F ≠ ∅
        u = RETIRA(F)
        for each v ∈ u.out
            if v.state == inicial
                v.state, v.d, v.sob =
                    descoberto, u.d + 1, u
                ENFILEIRA(F, v)
            u.state = finalizado
```

Exemplo



Exemplo

Olhe ali $\longrightarrow$



Alguns fatos

- INVARIANTE: **a)** Se u e v estão na fila, e u está na frente, então $u.d \leq v.d \leq u.d + 1$. **b)** Se u é finalizado antes de v, então $u.d \leq v.d$.

Alguns fatos

- INVARIANTE: **a)** Se u e v estão na fila, e u está na frente, então $u.d \leq v.d \leq u.d + 1$. **b)** Se u é finalizado antes de v, então $u.d \leq v.d$.
- Se um arco $u \to v$ não está na árvore, mas u está, então v está na árvore e $v.d \leq u.d + 1$.

Alguns fatos

- INVARIANTE: **a)** Se u e v estão na fila, e u está na frente, então $u.d \leq v.d \leq u.d + 1$. **b)** Se u é finalizado antes de v, então $u.d \leq v.d$.
- Se um arco $u \to v$ não está na árvore, mas u está, então v está na árvore e $v.d \leq u.d + 1$.
  PROVA: Quando o arco é examinado, se não entra na árvore é porque v já entrou. Assim, $v.sob$ foi finalizado antes de u, logo $v.sob.d \leq u.d$.

Alguns fatos

- INVARIANTE: **a)** Se u e v estão na fila, e u está na frente, então $u.d \leq v.d \leq u.d + 1$. **b)** Se u é finalizado antes de v, então $u.d \leq v.d$.
- Se um arco $u \to v$ não está na árvore, mas u está, então v está na árvore e $v.d \leq u.d + 1$.
  PROVA: Quando o arco é examinado, se não entra na árvore é porque v já entrou. Assim, $v.sob$ foi finalizado antes de u, logo $v.sob.d \leq u.d$.
- Para cada v, a distância de s a v na árvore é a distância em G.

Alguns fatos

- INVARIANTE: **a)** Se u e v estão na fila, e u está na frente, então $u.d \leq v.d \leq u.d + 1$. **b)** Se u é finalizado antes de v, então $u.d \leq v.d$.
- Se um arco $u \to v$ não está na árvore, mas u está, então v está na árvore e $v.d \leq u.d + 1$.
  PROVA: Quando o arco é examinado, se não entra na árvore é porque v já entrou. Assim, $v.sob$ foi finalizado antes de u, logo $v.sob.d \leq u.d$.
- Para cada v, a distância de s a v na árvore é a distância em G.
  PROVA: Indução em d(s, v).

Como provar que um caminho é mínimo?

Como provar que um caminho é mínimo?

Um 1-potencial é um vetor real y indexado pelos vértices, tal que para todo arco $u \to v$ vale:

$$y[v] - y[u] \leq 1.$$

Como provar que um caminho é mínimo?

Um 1-potencial é um vetor real y indexado pelos vértices, tal que para todo arco $u \to v$ vale:

$$y[v] - y[u] \leq 1.$$

Daí segue que se $P : u \rightsquigarrow v$, então $y[v] - y[u] \leq |P|$.

Como provar que um caminho é mínimo?

Um 1-potencial é um vetor real y indexado pelos vértices, tal que para todo arco $u \to v$ vale:

$$y[v] - y[u] \leq 1.$$

Daí segue que se $P : u \rightsquigarrow v$, então $y[v] - y[u] \leq |P|$.

Assim, se vale a igualdade, o comprimento é mínimo.

Como provar que um caminho é mínimo?

Um 1-potencial é um vetor real y indexado pelos vértices, tal que para todo arco $u \longrightarrow v$ vale:

$$y[v] - y[u] \leq 1.$$

Daí segue que se $P : u \rightsquigarrow v$, então $y[v] - y[u] \leq |P|$.

Assim, se vale a igualdade, o comprimento é mínimo.

$y[v] = v.d$ é um potencial, e $y[v] - y[s] = v.d$

Arestas com custos

Arestas com custos

Modelagem com grafos em geral associa valores a vértices e arestas, com finalidades diversas.

Arestas com custos

Modelagem com grafos em geral associa valores a vértices e arestas, com finalidades diversas.

EXEMPLO: Vamos associar um custo real a cada arco, e definir, para cada sequência de arcos, seu custo como sendo a soma dos custos dos arcos.

Problema

Dados vértices s e t, encontrar um caminho de custo mínimo de s a t.

Arestas com custos

Modelagem com grafos em geral associa valores a vértices e arestas, com finalidades diversas.

EXEMPLO: Vamos associar um custo real a cada arco, e definir, para cada sequência de arcos, seu custo como sendo a soma dos custos dos arcos.

Problema

Dados vértices s e t, encontrar um caminho de custo mínimo de s a t.

Arestas com custos

Modelagem com grafos em geral associa valores a vértices e arestas, com finalidades diversas.

EXEMPLO: Vamos associar um custo real a cada arco, e definir, para cada sequência de arcos, seu custo como sendo a soma dos custos dos arcos.

Problema

Dados vértices s e t, encontrar um caminho de custo mínimo de s a t.

Caminho de comprimento mínimo: custos 1.

Arestas com custos

Modelagem com grafos em geral associa valores a vértices e arestas, com finalidades diversas.

EXEMPLO: Vamos associar um custo real a cada arco, e definir, para cada sequência de arcos, seu custo como sendo a soma dos custos dos arcos.

Problema

Dados vértices s e t, encontrar um caminho de custo mínimo de s a t.

Caminho de comprimento mínimo: custos 1.

O custo do arco $e = ij$ será denotado $c(e)$ ou $c(i, j)$ conforme conveniente.

Como provar que um caminho é mínimo?

Como provar que um caminho é mínimo?

Um c-potencial é um vetor real y indexado pelos vértices, tal que para todo arco $u \to v$ vale:

$$y[v] - y[u] \leq c(u, v).$$

Como provar que um caminho é mínimo?

Um c-potencial é um vetor real y indexado pelos vértices, tal que para todo arco $u \to v$ vale:

$$y[v] - y[u] \leq c(u, v).$$

Daí segue que se $P : u \rightsquigarrow v$, então $y[v] - y[u] \leq c(P)$.

Como provar que um caminho é mínimo?

Um c-potencial é um vetor real y indexado pelos vértices, tal que para todo arco $u \to v$ vale:

$$y[v] - y[u] \leq c(u, v).$$

Daí segue que se $P : u \rightsquigarrow v$, então $y[v] - y[u] \leq c(P)$.

Assim, se vale a igualdade, o custo é mínimo.

Uma referência

R, K, Ahuja, T. L. Magnanti, J. B.Orlin, *Network Flows*, Prentice-Hall (1993)

caps 4 e 5

vejam os exercícios