Análise e Síntese de Algoritmos

Caminhos mais curtos.
Dijkstra.

CLRS Cap. 24

Prof. Pedro T. Monteiro

IST - Universidade de Lisboa

2024/2025

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica [CLRS, Cap.15]
  - Algoritmos greedy [CLRS, Cap.16]
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares [CLRS, Cap.22]
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes [CLRS, Cap.23]
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais
  - Complexidade Computacional [CLRS, Cap.34]

Resumo

Caminhos mais curtos: Definições / Relaxação

Algoritmo Dijkstra

Caminhos mais curtos

Tipos de problemas

- Caminhos Mais Curtos com Fonte Única (SSSPs)
  - Identificar o caminho mais curto de um vértice fonte $s \in V$ para qualquer outro vértice $v \in V$
- Caminhos Mais Curtos com Destino Único
  - Identificar o caminho mais curto de qualquer vértice $v \in V$ para um vértice destino $t \in V$
- Caminho Mais Curto entre Par Único
  - Identificar caminho mais curto entre dois vértices u e v
- Caminhos Mais Curtos entre Todos os Pares (APSPs)
  - Identificar um caminho mais curto entre cada par de vértices de V

Caminhos mais curtos

Definições

- Dado um grafo $G = (V, E)$, dirigido, com uma função de pesos $w : E \to IR$, define-se o peso de um caminho $p = \langle v_0, v_1, \ldots, v_n \rangle$, como a soma dos pesos dos arcos que compõem p:

$$w(p) = \sum_{i=1}^{n} w(v_{i-1}, v_i)$$

- O peso do caminho mais curto de u para v é definido por:

$$\delta(u, v) = \begin{cases} \min \{w(p) : u \to_p v\} & \text{se existe caminho de } u \text{ para } v \\ \infty & \text{caso contrário} \end{cases}$$

- Um caminho mais curto de u para v é qualquer caminho p tal que $w(p) = \delta(u, v)$

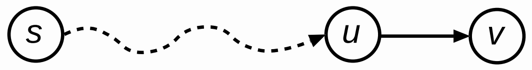
Caminhos mais curtos

Definições

Seja $p = \langle s, \ldots, u, v \rangle$ um caminho mais curto entre s e v, que pode ser decomposto em $p_{su} = \langle s, \ldots, u \rangle$ e (u, v).

O peso do caminho mais curto $\delta(s, v) = \delta(s, u) + w(u, v)$

- p_{su} é (um) caminho mais curto entre s e u
- $\delta(s, v) = w(p) = w(p_{su}) + w(u, v) = \delta(s, u) + w(u, v)$

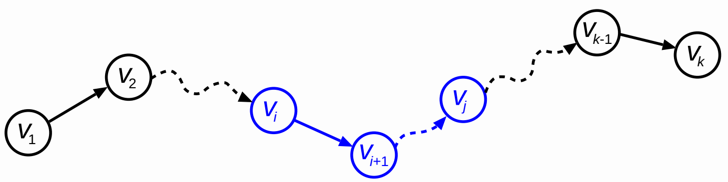


Caminhos mais curtos

Propriedade: Sub-estrutura óptima

Sub-caminhos de caminhos mais curtos são caminhos mais curtos

- Seja $p = \langle v_1, v_2, \ldots, v_k \rangle$ um caminho mais curto entre v_1 e v_k, e seja $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ um sub-caminho de p entre v_i e v_j
- Então p_{ij} é um caminho mais curto entre v_i e v_j



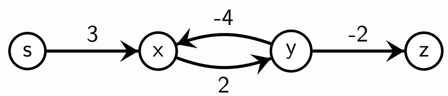
Porquê?

Prova: Se existisse caminho mais curto entre v_i e v_j então seria possível construir caminho entre v_1 e v_k mais curto do que p;
Contradição, dado que p é um caminho mais curto entre v_1 e v_k

Caminhos mais curtos

Arcos com pesos negativo

- Se grafo G sem ciclo negativo, então $\delta(s, v)$ bem definido
- Se ciclo negativo não atingível a partir da fonte s, então $\delta(s, v)$ bem definido
- Se ciclo negativo atingível a partir da fonte s, então os pesos dos caminhos mais curtos não são bem definidos
  - Neste caso, é sempre possível encontrar um caminho mais curto de s para qualquer vértice incluído no ciclo e define-se $\delta(s, v) = -\infty$



$w(< s, x, y, z >) = 3$
$w(< s, x, y, x, y, z >) = 1$
$w(< s, x, y, x, y, x, y, z >) = -1$
$\ldots$

Ciclos

Um caminho mais curto pode conter:

- um ciclo negativo? Não!
- e um ciclo positivo? Também não!

Prova

Dado um caminho $p = \langle v_0, v_1, \ldots, v_k \rangle$ e um ciclo $c = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ positivo ($w(c) > 0$), então o caminho $p' = \langle v_0, v_1, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_k \rangle$ tem peso $w(p') = w(p) - w(c) < w(p)$

- e ciclos de peso 0? Também não!
  Dado que existe um outro caminho de igual peso sem esse ciclo

Representação de caminhos mais curtos

- Para cada vértice $v \in V$ associar predecessor $\pi[v]$
- Após identificação dos caminhos mais curtos, $\pi[v]$ indica qual o vértice anterior a v num caminho mais curto de s para v

Sub-grafo de predecessores $G_\pi = (V_\pi, E_\pi)$:

- $V_\pi = \{v \in V : \pi[v] \neq \text{NIL}\} \cup \{s\}$
- $E_\pi = \{(\pi[v], v) \in E : v \in V_\pi \setminus \{s\}\}$

Representação de caminhos mais curtos

Uma árvore de caminhos mais curtos é um sub-grafo dirigido $G' = (V', E')$, $V' \subseteq V$ e $E' \subseteq E$, tal que:

- V' é o conjunto de vértices atingíveis a partir de s em G
- G' forma uma árvore com raíz s
- Para todo o $v \in V'$, o único caminho de s para v em G' é um caminho mais curto de s para v em G

Observações

- Após identificação dos caminhos mais curtos de G a partir de fonte s, G' é dado por $G_\pi = (V_\pi, E_\pi)$
- Dados o mesmo grafo G e vértice fonte s, G' não é necessariamente único

Relação caminho mais curto com arcos do grafo

Para qualquer arco $(u, v) \in E$ verifica-se $\delta(s, v) \leq \delta(s, u) + w(u, v)$

- Caminho mais curto de s para v não pode ter mais peso do que qualquer outro caminho de s para v

Operação de Relaxação

- Operação básica dos algoritmos para cálculo dos caminhos mais curtos com fonte única
- d[v]: denota a estimativa do caminho mais curto de s para v (limite superior no valor do peso do caminho mais curto)
- Algoritmos aplicam sequência de relaxações dos arcos de G após inicialização para actualizar a estimativa do caminho mais curto

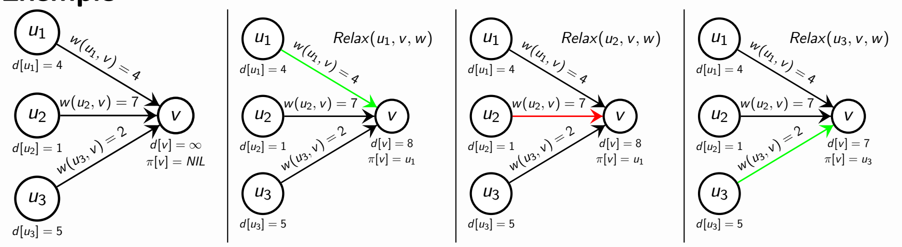
Initialize-Single-Source(G, s)

```
for v ∈ G.V do
  d[v] ← ∞
  π[v] ← NIL
end for
d[s] ← 0
```

Relax(u, v, w)

```
if d[v] > d[u] + w(u,v) then
  d[v] ← d[u] + w(u, v)
  π[v] ← u
end if
```


Exemplo



Triangle inequality
Para qualquer arco $(u, v) \in E$, temos $\delta(s, v) \leq \delta(s, u) + w(u, v)$

Upper-bound property
Para qualquer vértice $v \in V$, temos $\delta(s, v) \leq d[v]$ e uma vez $\delta(s, v) == d[v]$ já não é alterado

No-path property
Se não existir um caminho de s para v, então $\delta(s, v) = d[v] = \infty$

Convergence property
Se $s \rightsquigarrow u \rightarrow v$ é um caminho mais curto em G para algum $u, v \in V$ e se $\delta(s, u) == d[u]$ antes de relaxar o arco (u, v), então $\delta(s, v) == d[v]$ depois de relaxar (u, v)

Path-relaxation property
Se $p = \langle s, v_1, \ldots, v_k \rangle$ é um caminho mais curto de s até v_k, e relaxarmos os arcos de p por ordem $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, então $d[v_k] = \delta(s, v_k)$

Predecessor-subgraph property
Se $\delta(s, v) == d[v]$ para todo $v \in V$, então o grafo de predecessores é uma árvore dos caminhos mais curtos com raíz s

Resumo

Caminhos mais curtos: Definições / Relaxação

Algoritmo Dijkstra

Algoritmo Dijkstra

Intuição

- Algoritmo Greedy
- Todos os arcos com pesos não negativos
- Mantém conjunto de vértices S com pesos dos caminhos mais curtos já calculados (vértices fechados)
- A cada passo selecciona vértice $u \in V - S$ com menor estimativa do peso do caminho mais curto
  - vértice u é inserido em S
  - arcos que saem de u são relaxados
- No final, $\delta(s, v) == d[v]$ para cada $v \in V$

Nota: É uma generalização da BFS (pesos não unitários)

Algoritmo Dijkstra

Dijkstra(G,w,s)

```
Initialize-Single-Source(G,s)
S ← ∅
Q ← G.V
while Q ≠ ∅ do
  u ← Extract-Min(Q)
  for each v ∈ Adj[u] do
    Relax(u,v,w)
  end for
  S ← S ∪ { u }
end while
```

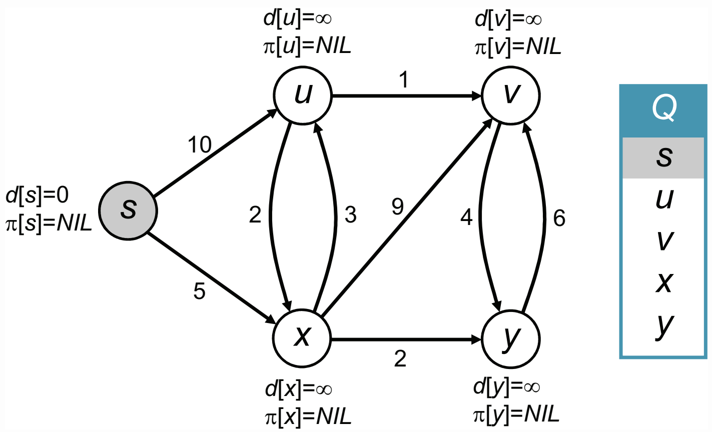
Invariantes

Q = V - S

$\delta(s, u) == d[u]$ quando u é adicionado ao conjunto S

Algoritmo Dijkstra

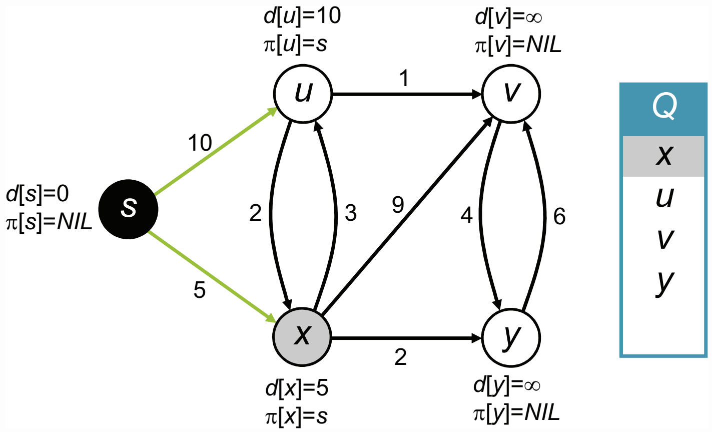
Exemplo



S: { }

Algoritmo Dijkstra

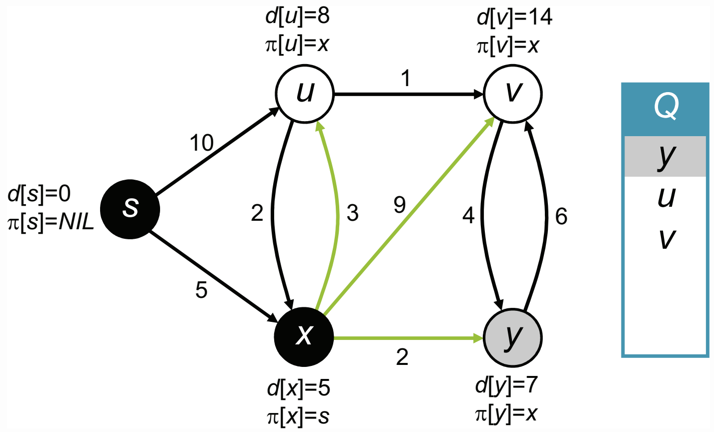
Exemplo



S: { s }

Algoritmo Dijkstra

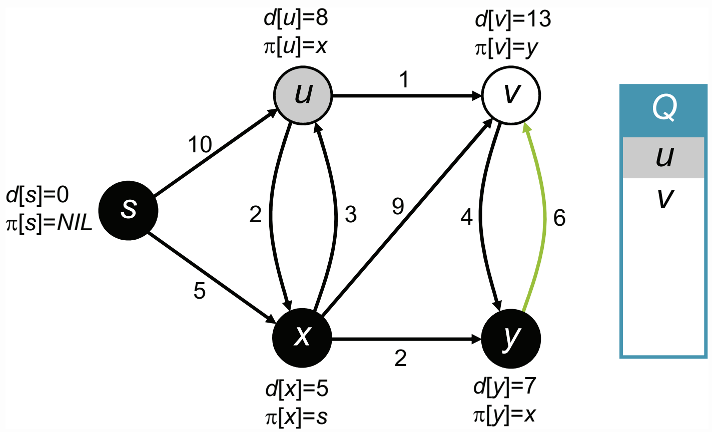
Exemplo



S: { s, x }

Algoritmo Dijkstra

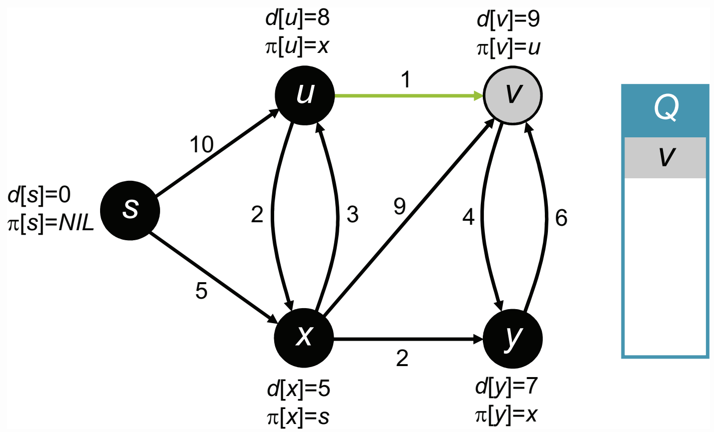
Exemplo



S: { s, x, y }

Algoritmo Dijkstra

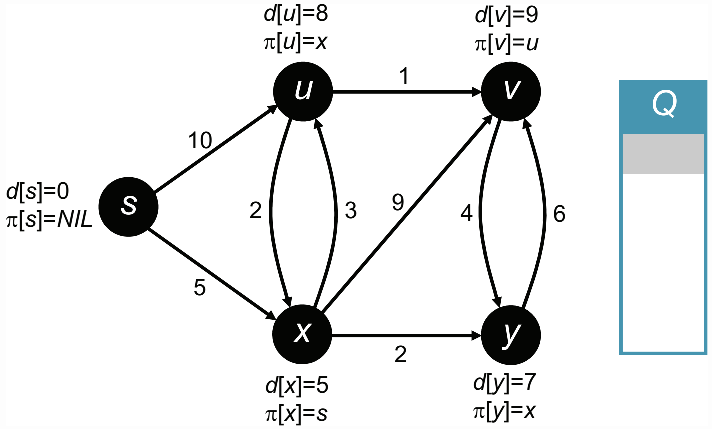
Exemplo



S: { s, x, y, u }

Algoritmo Dijkstra

Exemplo



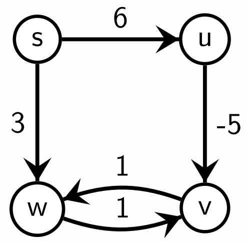
S: { s, x, y, u, v }

Algoritmo Dijkstra

Pesos negativos

Os pesos do grafo têm que ser não-negativos
para garantir a correcção do algoritmo !

Exemplo



- Analisar w
  $d[w] = d[s] + 3 = 3$
- Analisar v
  $d[v] = d[w] + 1 = 4$
- Analisar u
  $d[u] = d[s] + 6 = 6$
- No final temos
  $d[w] = 3 \neq \delta(s, w) = 2$

Algoritmo Dijkstra

Complexidade

- Fila de prioridade Q baseada em amontoados (heap)
- Quando um vértice é extraído da fila Q, implica actualização de Q
  - Cada vértice é extraído apenas 1 vez: $|V|$ vértices
  - Actualização de Q (extract-min): $O(log\ V)$
  - Então, $O(V\ log\ V)$
- Cada operação de relaxação pode implicar uma actualização de Q
  - Cada arco é relaxado apenas 1 vez: $|E|$ arcos
  - Actualização de Q (decrease-key): $O(log\ V)$
  - Então, $O(E\ log\ V)$
- Complexidade algoritmo Dijkstra: $O((V + E)\ log\ V)$

Algoritmo Dijkstra

Correcção do Algoritmo

Provar invariante do algoritmo: $\delta(s, u) == d[u]$ quando u é adicionado ao conjunto S, e que a igualdade é posteriormente mantida

Prova por contradição

- Assume-se que existe um primeiro vértice u tal que $\delta(s, u) \neq d[u]$ quando u é adicionado a S
- Necessariamente temos que:
  - $u \neq s$ porque $\delta(s, s) = d[s] = 0$
  - $S \neq \emptyset$ porque $s \in S$ quando u é adicionado a S
  - existe caminho mais curto de s para u
    caso contrário teríamos $\delta(s, u) = d[u] = \infty$ (no-path property)

Algoritmo Dijkstra

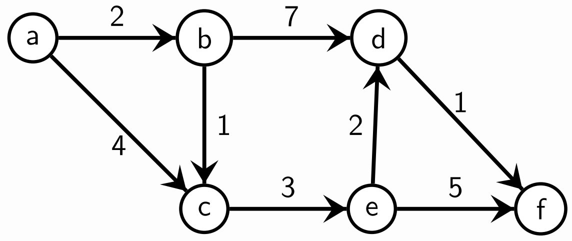
Correcção do Algoritmo (cont.)

Pressuposto: u é o primeiro vértice tal que $\delta(s,u) \neq d[u]$ quando u é adicionado a S

- Seja $p = \langle s, \ldots, x, y, \ldots, u \rangle$ o caminho mais curto de s para u
- Tem que existir pelo menos um vértice do caminho p que ainda não esteja em S, caso contrário, $\delta(s,u) == d[u]$ devido à relaxação dos arcos que compõem o caminho p
- Seja (x,y) um arco de p tal que $x \in S$ e $y \notin S$
  - Temos que $\delta(s,x) = d[x]$ porque $x \in S$ e u é o primeiro vértice em que isso não ocorre
  - Temos também que $\delta(s,y) == d[y]$ porque o arco (x,y) foi relaxado quando x foi adicionado a S (convergence property)
  - Como y é predecessor de u no caminho mais curto até u, então $\delta(s,y) \leq \delta(s,u)$, porque os pesos dos arcos são não-negativos
  - Mas se u é adicionado a S antes de y, temos que $d[u] \leq d[y]$. Logo, $d[u] \leq \delta(s,y) \leq \delta(s,u)$
- Contradição: do pressuposto $d[u] \neq \delta(s,u)$

Algoritmo Dijkstra

Exercício (fazer em casa/quadro: `Dijkstra(a)`)



Questões?

Dúvidas?