Algoritmo de Dijkstra

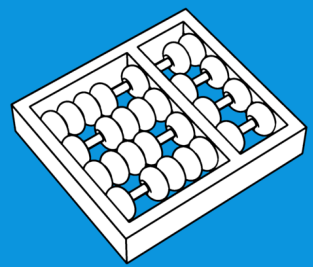
MC558 - Projeto e Análise de Algoritmos II

09/24

12

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

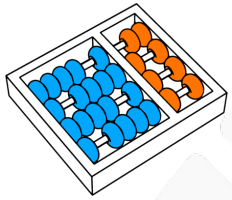
UNICAMP



"Ciência da computação tem tanto a ver com o computador como a astronomia com o telescópio, a biologia com o microscópio, ou a química com os tubos de ensaio. A Ciência não estuda ferramentas, mas o que fazemos e o que descobrimos com elas."

Atribuída a Edsger W. Dijkstra

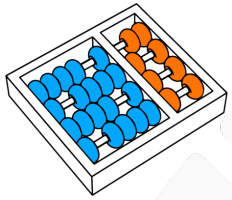
Aula anterior



Representando caminhos mínimos

A saída é similar à da busca em largura a partir de **s**:

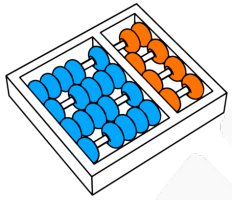
- Para cada **v** $\in$ **V[G]**, associamos um **PREDECESSOR** π[**v**].
- O vetor π induz uma **ÁRVORE DE CAMINHOS MÍNIMOS** com raiz em **s**.
- Um caminho de **s** a **v** na árvore é um caminho mínimo de **s** a **v** no grafo G.



Inicialização

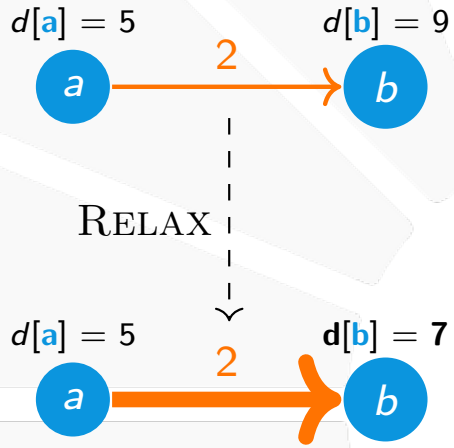
Algoritmo: INITIALIZE-SINGLE-SOURCE(G, s)

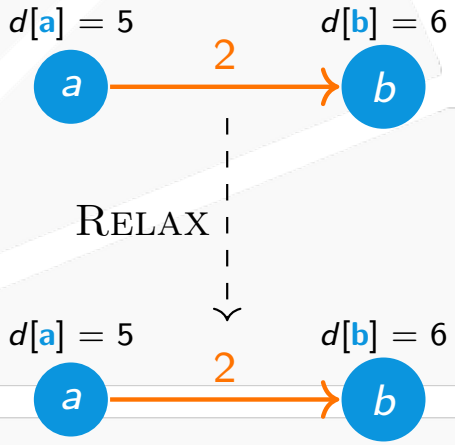
```
1 para cada v ∈ V[G]
2     d[v] ← ∞
3     π[v] ← NIL
4 d[s] ← 0
```



Relaxação

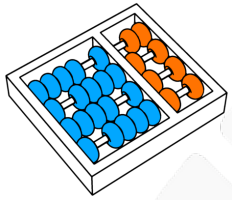
Tenta melhorar a estimativa $d[\mathbf{v}]$ examinando **(u,v)**.





Algoritmo: $\textsc{Relax}(u, v, w)$

```
1 se d[v] > d[u] + w(u,v)
2     d[v] ← d[u] + w(u,v)
3     π[v] ← u
```



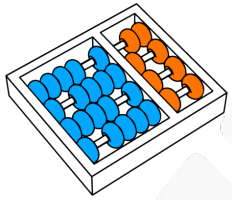
Algoritmos baseados em relaxação

Características:

1. Inicializam d e π com uma sub-rotina INITIALIZE-SINGLE-SOURCE.
2. Alteram d e π apenas com uma sub-rotina RELAX.

Esses algoritmos mantêm algumas invariantes:

- Existe um caminho de **s** a **v** com peso d[**v**].
- Esse caminho pode ser recuperado por meio π.
- Assim, d[**v**] é sempre **MAIOR OU IGUAL** a dist(**s**,**v**).
- Queremos que no final valha d[**v**] = dist(**s**,**v**).

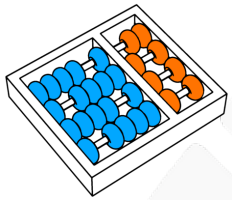


Propriedade de algoritmos baseados em relaxação

- **Limite superior:** Vale $d[v] \geq \text{dist}(s,v)$ e, tão logo $d[v]$ alcança $\text{dist}(s,v)$, nunca mais muda.
- **Inexistência de caminho:** Se não existe caminho de s a v, então $d[v] = \infty$.
- **Subgrafo de predecessores:** Se $d[v] < \infty$, então o subgrafo dos predecessores induzido por π é um caminho de peso $d[v]$.
- **Convergência:** Se p é um caminho mínimo de s até v terminando com a aresta (u,v) e $d[u] = \text{dist}(s,u)$, então ao relaxar (u,v), $d[v] = \text{dist}(s,v)$, que nunca mais muda.
- **Relaxamento de caminho:** Se $p = (v_0, v_1, \ldots, v_k)$ é um caminho mínimo de $s = v_0$ a v_k e relaxamos as arestas de p na ordem $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, então $d[v_k] = \text{dist}(s, v_k)$.

  A propriedade vale mesmo se tivermos realizado quaisquer outras relaxações durante a execução.

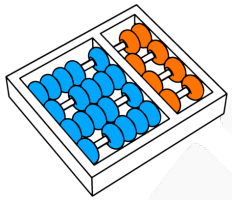
Algoritmo de Dijkstra



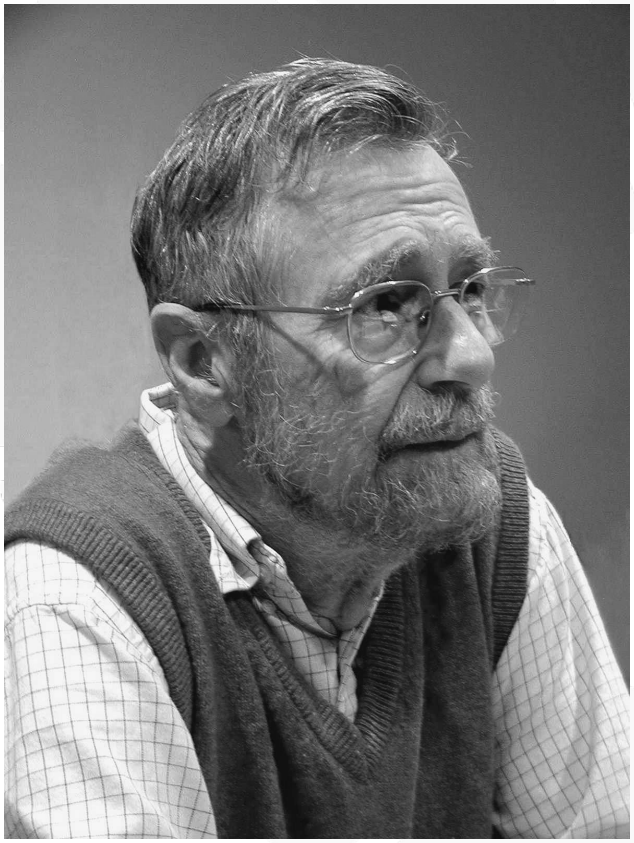
Sobre o algoritmo

Veremos agora um algoritmo para caminhos mínimos em grafos que podem conter ciclos, mas **SEM ARESTAS DE PESOS NEGATIVO**.

O algoritmo também é **BASEADO EM RELAXAÇÃO**.

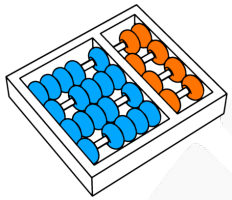


Edsger Wybe Dijkstra (11/05/1930 – 06/08/2002)



Principais áreas de atuação:

- Algoritmos em grafos.
- Programação concorrente e distribuída.
- Sistemas operacionais.
- Compiladores e linguagens de programação.

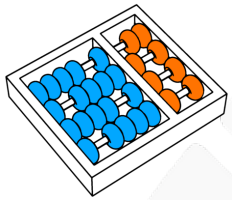


Publicação

1957 *M. Leyzorek, R.S. Gray, A.A. Johnson, W.C. Ladew, S.R. Meaker, R.M. Petry* e *R.N. Seitz. Investigation of model techniques - First annual report - 6 June 1956 - 1 July 1957 - A study of model techniques for communication systems, Case Institute of Technology, Cleveland, Ohio*.

1958 *G.B. Dantzig. On the shortest route through a network, The RAND Corporation, Santa Monica, California*.

1959 *E.W. Dijkstra. A note on two problems in connexion with graphs, Numerische Mathematik*.

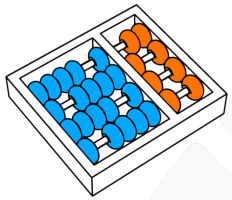


Problema

Problema

Entrada: *Um grafo direcionado* $G = (V, E)$, *uma função de peso* w *nas arestas (sem arestas de peso negativo) e um vértice origem* s.

Saída: *Um vetor* d *com* $d[v] = \text{dist}(s,v)$ *para* $v \in V$ *e um vetor* π *definindo uma* ***ÁRVORE DE CAMINHOS MÍNIMOS***.

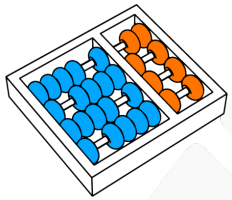


O algoritmo

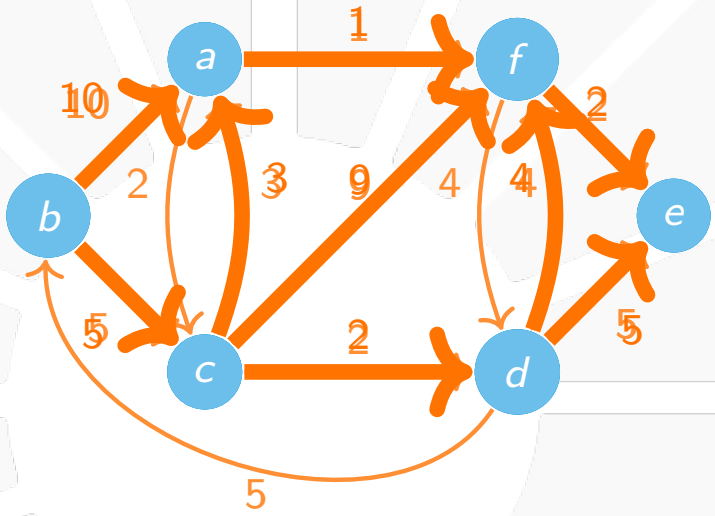
Algoritmo: DIJKSTRA(G, w, s)

```
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← ∅
3 Q ← V[G] enquanto Q ≠ ∅
4     u ← EXTRACT-MIN(Q)
5     S ← S ∪ {u}
6     para cada v ∈ Adj[u]
7         RELAX(u, v, w)
8 devolva d, π
```

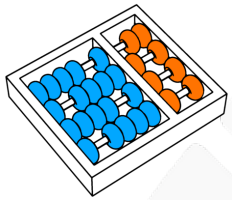
- O conjunto Q é implementado como uma fila de prioridade com chave d.
- O conjunto S não é realmente necessário, mas simplifica a análise do algoritmo.



Exemplo



	a	b	c	d	e	f
d	~~∞~~ 10	0	~~5~~ 5	~~7~~ 7	~~12~~ 12	~~14~~ 14



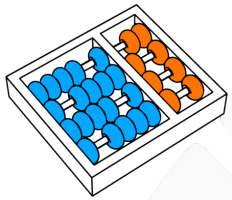
Correção do algoritmo

Precisamos provar que algoritmo está correto, temos que mostrar que, quando o algoritmo termina:

1. $d[\mathbf{v}] = \text{dist}(\mathbf{s},\mathbf{v})$ para todo $\mathbf{v} \in \mathbf{V[G]}$ e
2. π induz uma **ÁRVORE DE CAMINHOS MÍNIMOS**.

Observe que:

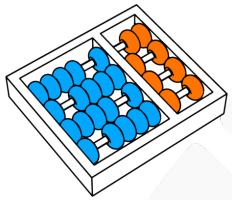
- DIJKSTRA é baseado em relaxação.
- Pela propriedade do **Subgrafo de predecessores**, sabemos que o vetor π induz uma árvore que testemunha d.
- Assim, basta mostrar que de fato $d[\mathbf{v}] = \text{dist}(\mathbf{s},\mathbf{v})$.



Invariante

Iremos mostrar que no no início de cada iteração da linha 4 no algoritmo DIJKSTRA, vale $d[x] = \text{dist}(s, x)$ **para cada** $x \in S$.

- **Inicialização.** No início, **S**=$\emptyset$, então a invariante vale trivialmente.
- **Manutenção.**
  - Suponha que a invariante vale para S no início da iteração.
  - Nessa iteração, DIJKSTRA escolhe um vértice **u** com menor d[**u**] em **Q** e o adiciona a **S**.
  - Queremos mostrar que a invariante vale para **S** $\cup \{$**u**$\}$.
  - Basta verificar que neste instante d[**u**] = dist(**s**,**u**).



Demonstração

Sejam:

- P um caminho mínimo de **s** a **u** (i.e., com peso dist(**s**,**u**)).
- **y** o primeiro vértice de P que não pertence a **S**
- **x** o vértice em P que precede **y**.

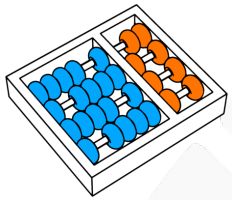
Suponha que $d[\mathbf{u}] > \text{dist}(\mathbf{s},\mathbf{u})$:

- Pela hipótese de indução, $d[\mathbf{x}] = \text{dist}(\mathbf{s},\mathbf{x})$ pois $\mathbf{x} \in \mathbf{S}$. Logo,

$$\begin{aligned} d[\mathbf{y}] &\leq d[\mathbf{x}] + w(\mathbf{x},\mathbf{y}) \quad \text{(pois relaxamos } (\mathbf{x},\mathbf{y})) \\ &= \text{dist}(\mathbf{s},\mathbf{x}) + w(\mathbf{x},\mathbf{y}) \\ &\leq w(P) = \text{dist}(\mathbf{s},\mathbf{u}) < d[\mathbf{u}]. \end{aligned}$$

- Mas, $d[\mathbf{y}] < d[\mathbf{u}]$ contraria a escolha de **u**.
- Portanto, $d[\mathbf{u}] \leq \text{dist}(\mathbf{s},\mathbf{u})$.

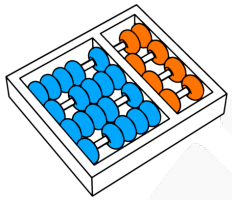
Concluindo que $d[\mathbf{u}] = \text{dist}(\mathbf{s},\mathbf{u})$ (propriedade de **Limite superior**).



Demonstração

Para terminar a demonstração, observe que:

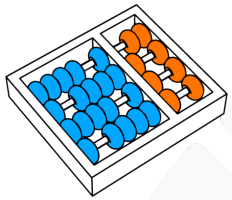
- Após a última iteração, **S** é o conjunto dos vértices atingíveis por **s**.
- Por **Inexistência de caminho**, se um vértice **v** não é atingível, então $d[\mathbf{v}] = \infty$.
- Portanto, para todo $\mathbf{v} \in \mathbf{V}$, vale $d[\mathbf{v}] = \text{dist}(\mathbf{s},\mathbf{v})$.



Dijkstra precisa de arestas com peso não negativo

Na demonstração, supomos que **NÃO HÁ ARESTAS NEGATIVAS**

- Se a hipótese não valer, o algoritmo de Dijkstra pode falhar.
- Encontre um grafo com arestas negativas para o qual o algoritmo de Dijkstra **NÃO** funciona (exercício).
- Existe um exemplo com 4 vértices, com apenas uma aresta negativa e sem ciclos de peso negativo.



Complexidade de tempo

Depende de como a fila de prioridade Q é implementada:

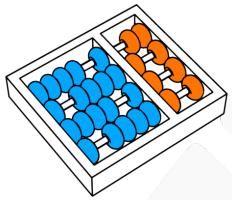
- Operações INSERT, EXTRACT-MIN, DECREASE-KEY.

Observe que:

- Os passos INITIALIZE-SINGLE-SOURCE e $Q \leftarrow V[G]$ escondem chamadas a INSERT.
- RELAX esconde chamada a DECREASE-KEY.

Linha(s)	Tempo total
1–3	$\|V\|$ chamadas a INSERT
5	$\|V\|$ chamadas a EXTRACT-MIN
8	$\|E\|$ chamadas a DECREASE-KEY

Complexidade de Dijkstra:
$O(|V| \times \text{INSERT} + |V| \times \text{EXTRACT-MIN} + |E| \times \text{DECREASE-KEY})$.



Complexidade de tempo

Total: $O(|V| \times \textsc{Insert} + |V| \times \textsc{Extract-Min} + |E| \times \textsc{Decrease-Key})$

Tipo de fila	INSERT	EXTRACT-MIN	DECREASE-KEY	TOTAL
Vetor	$O(1)$	$O(V)$	$O(1)$	$O(V^2)$
Min-Heap	$O(\log V)$	$O(\log V)$	$O(\log V)$	$O((V+E)\log V)$
Fibonacci	$O(1)$	$O(\log V)$	$O(1)$	$O(V \log V + E)$

Algoritmo de Dijkstra

MC558 - Projeto e Análise de Algoritmos II

09/24

12

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP