Algoritmos

Pedro Hokama

Fontes

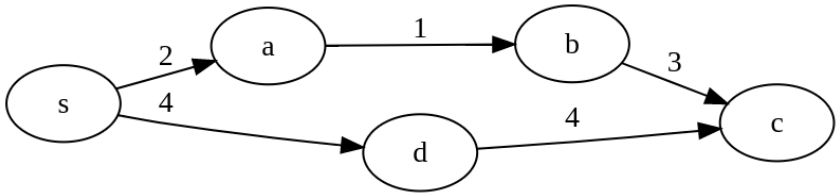
- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

Caminhos mais Curtos de Única Fonte

Problema do Caminho Mínimo de Única fonte

Dado um grafo direcionado $G = (V, A)$ com m arcos e n vértices, em que cada arco $a \in A$ tem um custo c_a, e um vértice fonte $s \in V$. Desejamos calcular para cada vértice $v \in V$ o valor $D(v)$ do $s - v$ caminho mais curto.



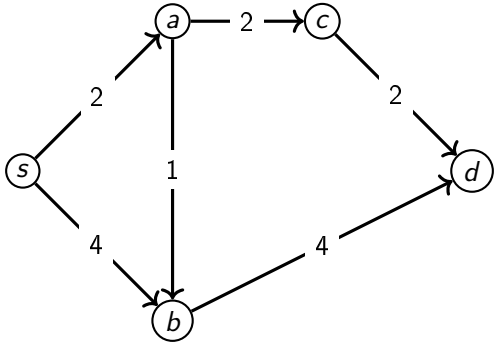
- O Algoritmo de Dijkstra executa em tempo $O(m \log n)$.
- O Dijkstra só funciona corretamente se os comprimentos forem **positivos**, ou seja, $c_a \geq 0, \forall a \in A$.
- Quando os custos podem ser negativos precisamos de outro algoritmo.

Encontrando uma subestrutura ótima

- Note que como não existem ciclos negativos, qualquer caminho mais curto entre a fonte e qualquer outro vértice passa por no máximo $n-1$ arcos.

- Qual o caminho mínimo de $s - v$ com no máximo k arcos?
- Seja $A[i, v]$ o custo do caminho mínimo se s até v com no máximo k arcos:
- $A[0, s] = 0$, $A[0, v] = +\infty$ se $v \neq s$:

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min\limits_{w:(w,v)\in A} \{A[i-1, w] + c_{wv}\} \end{cases}$$



Algoritmo 1: Bellman-Ford(G, s)

Entrada: Um conjunto Grafo, e um vértice fonte s
Saída: O valor do menor caminho de s para todos os outros vértices

1 $A[0, s] = 0$; $A[0, v] = +\infty$ para todo $v \neq s$;
2 **para** $i = 1, 2, \ldots, n-1$ **faça**
3 **para** *todo* $v \in V$ **faça**
4 $A[i, v] = \min\{A[i-1, v]; \min\limits_{w:(w,v)\in A} \{A[i-1, w] + c_{wv}\}$;
5 devolva $A[n-1, *]$;

Algoritmo Bellman-Ford

- O tempo de execução do Bellman-Ford é $O(mn)$
- Você pode parar o algoritmo se em uma iteração a distância para nenhum vértice mudar.
- Podemos detectar ciclos negativos rodando uma iteração a mais $i = n$, se encontrarmos algum caminho mais curto significa que existe um ciclo de custo negativo.

Caminhos Mínimos

- Vimos dois algoritmos para encontrar o caminho mais curto entre um determinado vértice fonte s e todos os outros vértices.
- Dijkstra e Bellman-Ford.
- Mas com frequência estaremos interessados em encontrar o caminho mínimo entre quaisquer dois vértices.

Caminhos Mínimos

Caminho Mínimo entre Todos os Pares de Vértices

Dado um grafo direcionado $G = (V, A)$, com n vértices e m arcos, em que cada arco $a \in A$ tem um custo c_a (pode ser negativo). Desejamos encontrar o custo mínimo para ir de u a v para todos os pares $u, v \in V$ ou detectar se houver um ciclo de custo negativo.

Caminho Mínimo entre Todos os Pares de Vértices

- Podemos resolver esse problema, chamando o algoritmo de Caminho Mínimo de Única fonte algumas vezes. Quantas execuções seriam necessárias?
  a. 1
  b. $n - 1$
  c. n
  d. n^2

Caminho Mínimo entre Todos os Pares de Vértices

- Se o grafo não tem tem custos negativos, poderíamos usar o Algoritmo de Dijkstra, e obter uma complexidade de

$$O(nm \log n) \begin{cases} O(n^2 \log n) \text{ em grafos esparços} \\ O(n^3 \log n) \text{ em grafos densos} \end{cases}$$

Caminho Mínimo entre Todos os Pares de Vértices

- De fato não sabemos se algum algoritmo pode ser mais rápido que $O(n^3)$, uma vez que precisa ser calculado para todos os pares, e podemos imaginar que um trabalho linear por par seria necessário. (Mas lembre do algoritmo de Strassen para multiplicação de matrizes que era subcúbico).

Caminho Mínimo entre Todos os Pares de Vértices

- Mas se o grafo tem tem custos negativos, poderíamos usar o Bellman-Ford (executado n vezes):

$$O(n^2 m) \begin{cases} O(n^3) \text{ em grafos esparços} \\ O(n^4) \text{ em grafos densos} \end{cases}$$

- Parece ser uma complexidade muito grande. De fato o algoritmo de Floyd-Warshall que veremos executa em $O(n^3)$ independente do número de arcos, e funciona mesmo com os custos negativos.
- Portante é pelo menos tão bom quanto n execuções do Bellman-Ford. Muito melhor para grafos densos.
- Para grafos sem custos negativos, executar o Dijkstra n vezes ainda é melhor, mas para grafos densos ambos são competitivos.

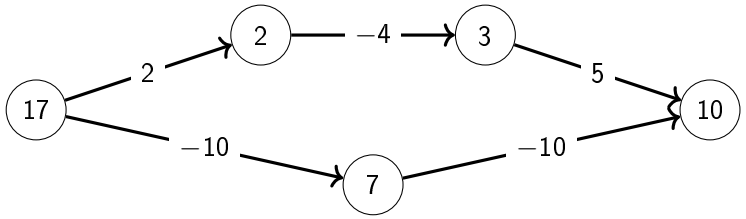
Caminho Mínimo entre Todos os Pares de Vértices

- Esse problema pode aparecer por exemplo se queremos saber se uma relação tem um fechamento transitivo (se existe um caminho orientado entre qualquer par de vértices).
- Assim como o Bellman-Ford, adaptações do Floyd-Warshall podem resolver vários problemas práticos.

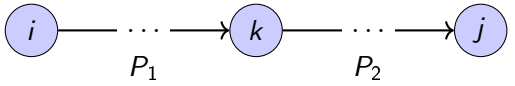
Ideias - Subestrutura Ótima

- Como vimos no Bellman-Ford, encontrar qual subproblema pode ser resolvido não é tão simples. Acabamos por perceber que o subproblema estava no número de arcos que poderíamos utilizar.
- **Idéia:** Vamos limitar o número de vértices que vamos usar. E também limitar quais vértices podemos usar.
- Considere uma ordenação arbitrária de $V = 1, 2, 3, \ldots, n$, e seja o prefixo $V^{(k)} = \{1, \ldots, k\}$.

- Suponha que G não tem ciclos negativos. Fixado uma origem $i \in V$, e um destino $j \in V$ e $k \in \{1, 2, \ldots, n\}$. Seja P o caminho de custo mínimo (portanto sem ciclos) entre i e j com todos os nós internos em $V^{(k)}$.
- Por exemplo, seja $i = 17$, $j = 10$ e $k = 5$.

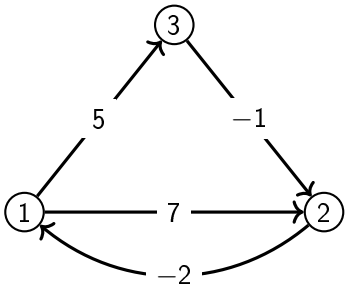


- Considere que $k = n$.
- Caso 1: Se k não está em P, então P é um caminho de custo mínimo com todos os vértices internos em $V^{(k-1)}$.
- Caso 2: Se k está em P então podemos dividir P em dois pedaços.



- P_1 é o caminho de custo mínimo entre i e k com todos os nós internos em $V^{(k-1)}$ e P_2 é o caminho de custo mínimo entre k e j com todos os nós internos em $V^{(k-1)}$

- Seja S um vetor tridimensional indexado em i, j e k.
- $S[i,j,k]$ vai representar o custo do caminho mínimo entre i e j em que todos os nós internos estão em $\{1,2,\ldots,k\}$ (ou $+\infty$ se tal caminho não existir)
- Qual deve ser $S[i,j,0]$ se (I) $i=j$, (II) $(i,j)\in A$, (III) $i\neq j$ e $(i,j)\notin A$.
  a. 0, 0, e $+\infty$
  b. 0, c_{ij} e c_{ij}
  c. 0, c_{ij} e $+\infty$
  d. $+\infty$, c_{ij} e $+\infty$



$k=0$ destino

origem	1	2	3
1			
2			
3			

$k=0$ destino

origem	1	2	3
1	0	7	5
2	-2	0	∞
3	∞	-1	0

$k=1$ destino

origem	1	2	3
1			
2			
3			

$k=1$ destino

origem	1	2	3
1	0	7	5
2	-2	0	3
3	∞	-1	0

$k=2$ destino

origem	1	2	3
1			
2			
3			

$k = 2$

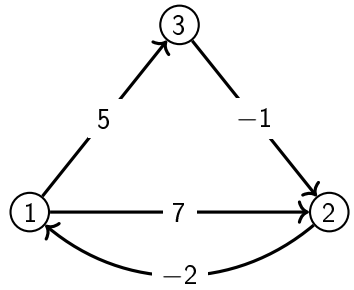
origem \ destino	1	2	3
1	0	7	5
2	-2	0	3
3	-3	-1	0

$k = 3$

origem \ destino	1	2	3
1			
2			
3			

$k = 3$

origem \ destino	1	2	3
1	0	4	5
2	-2	0	3
3	-3	-1	0



Algoritmo 2: Floyd-Warshall(G)

Entrada: Um grafo caminho G
Saída: Os custos dos caminhos mínimos entre todos os pares de vértices

para *todo* $i \in V$ **faça**
  para *todo* $j \in V$ **faça**
    $$S[i,j,0] = \begin{cases} 0 \text{ se } i = j \\ c_{ij} \text{ se } (i,j) \in A \\ +\infty \text{ se } i \neq j \text{ e } (i,j) \notin A \end{cases}$$
para $k \in \{1, \ldots, n\}$ **faça**
  para *todo* $i \in V$ **faça**
    para *todo* $j \in V$ **faça**
      $$S[i,j,k] = \min \begin{cases} S[i,j,k-1] \\ S[i,k,k-1] + S[k,j,k-1] \end{cases}$$

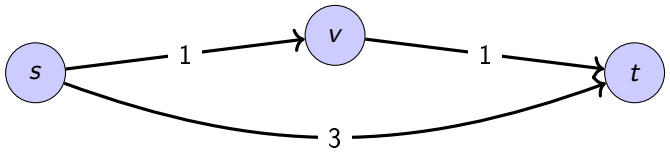
- Corretude: Pela subestrutura ótima e indução
- Complexidade: O(1) por subproblema, total de $O(n^3)$.
- E se o grafo tiver um ciclo de custo negativo?
- Nesse caso, no final do algoritmo você terá um $S[i,i,n] < 0$ para pelo menos um $i \in V$
- E se precisamos reconstruir o caminho entre i e j?
- Nesse caso podemos guardar um vetor bidimensional auxiar $T[i,j]$ que vai guardar o vértice interno com o índice mais alto no caminho de i até j.

	Caminho mínimo única fonte	Caminho mínimo entre todos os pares
Dijkstra	$O(m \log n)$	$O(nm \log n)$
Bellmond-Ford	$O(mn)$	$O(mn^2)$
Floyd-Warshall		$O(n^3)$

Algoritmo de Johnson

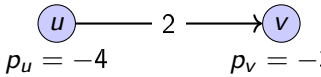
- Relembrando que podemos resolver o problema com n chamadas ao algoritmo de caminhos mínimos de única fonte:
  - Custos não negativos: $O(mn \log n)$ com o Dijkstra
  - Caso geral: $O(mn^2)$ com o Bellman-Ford
- Mas eu ainda quero usar o Dijkstra mesmo com pesos negativos. Será possível?
- O algoritmo de Johnson:
  - Faz 1 chamada ao Bellman-Ford
  - Faz n chamadas ao Dijkstra
- Portanto $O(mn) + O(nm \log n) = O(nm \log n)$

- Podemos só somar um valor em todas as arestas para tornar todos os custos positivos? Não! Suponha no grafo abaixo que **somamos 2** em todas os arcos, perceba que o menor caminha muda. Só funcionaria se todos os caminhos tivessem o mesmo número de arestas.

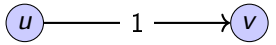


- Nem tudo está perdido, podemos sim usar um rebalanceamento para se livrar dos pesos negativos.

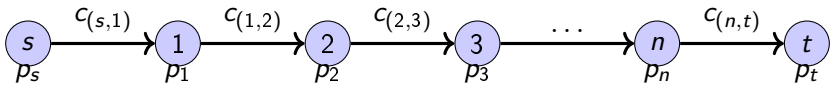
- Seja $G = (V, A)$ um grafo direcionado com custos c_a nos arcos, podendo ser negativos. Fixe um número real p_v para cada vértice $v \in V$.
- Para cada arco $a = (u, v)$ de A, faça $c'_a = c_a + p_u - pv$



$$c'_{(u,v)} = 2 + (-4) - (-3) = 2 - 4 + 3 = 1$$



Considere um $s - t$ caminho qualquer (renomeando os nós internos como $1, 2, \ldots, n$



Custo do caminho:

$$c_{(s,1)} + c_{(1,2)} + c_{(2,3)} + \ldots + c_{(n,t)}$$

Novo custo do caminho:

$$c_{(s,1)} + p_s - p_1 + c_{(1,2)} + p_1 - p_2 + c_{(2,3)} + p_2 - p_3 + \ldots + c_{(n,t)} + p_n - p_t$$

Custo do caminho:

$$c_{(s,1)} + c_{(1,2)} + c_{(2,3)} + \ldots + c_{(n,t)}$$

Novo custo do caminho:

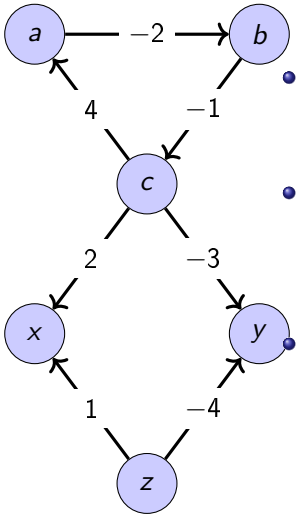
$$\begin{array}{lll} c_{(s,1)} + p_s - p_1 + & c_{(s,1)} + p_s - \cancel{p_1} + & c_{(s,1)} + p_s + \\ c_{(1,2)} + p_1 - p_2 + & c_{(1,2)} + \cancel{p_1} - \cancel{p_2} + & c_{(1,2)} + \\ c_{(2,3)} + p_2 - p_3 + & c_{(2,3)} + \cancel{p_2} - \cancel{p_3} + & c_{(2,3)} + \\ \ldots + & \ldots + & \ldots + \\ c_{(n,t)} + p_n - p_t & c_{(n,t)} + \cancel{p_n} - p_t & c_{(n,t)} - p_t \end{array}$$

- Se o $s - t$ caminho P tem custo L com os custos originais, qual é o custo L' de P com os custos modificados?

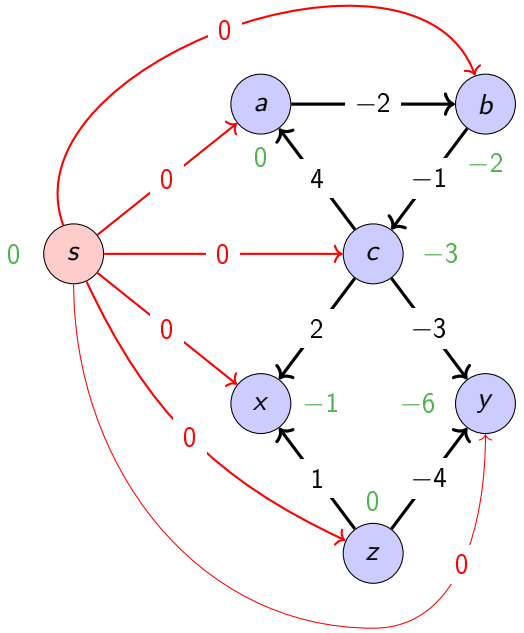
a. L
b. $L + p_s + P_t$
c. $L + p_s - P_t$
d. $L - p_s + P_t$

$$\begin{aligned} L' &= \sum_{a \in P} c'_a \\ &= \sum_{a=(u,v) \in P} c_a + p_u - p_v \\ &= \left(\sum_{a \in P} c_a \right) + p_s - p_t = L + p_s - p_t \end{aligned}$$

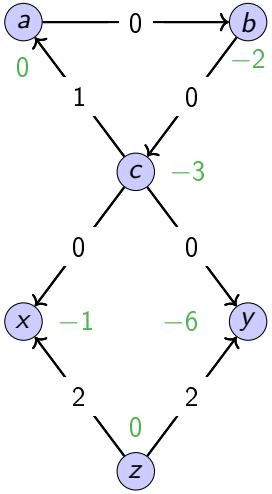
- Note que dessa forma qualquer caminho foi alterado exatamente pelo mesmo custo.
- Dessa forma os caminhos mínimos foram preservados (obviamente não o seu custo, mas é fácil de obtê-lo)
- E se encontramos valores p_v de forma que todos os arcos fiquem com custos não-negativos?
- Como encontrar esses valores? Alias, será que tais valores existem? 🤔
- Se o grafo não tiver ciclos negativos, SIM, esses valores existem!



- A ideia consiste em executar um algoritmo para o problema do caminho mínimo de única fonte.
- Um problema: É preciso alcançar todos os vértices, no grafo ao lado qualquer vértice que você escolher para ser sua fonte não vai alcançar todos os outros.
- Truque, inserir um novo vértice fictício que se liga a todos os outros por um arco de custo 0.



- Note que adicionar s e os novos arcos não cria nenhum novo caminho para qualquer $u, v \in V$.
- como existem arcos com custo negativo, nos resta executar o Bellman-Ford (o que também detecta ciclos negativos)
- Esse são exatamente os valores que procuramos. p_v é o custo do $s - t$ caminho mínimo.



- Podemos então para cada arco $a = (u, v)$ calcular o novo custo $c'_a = c_a + p_u - p_v$
- Agora todos os custos ficaram não negativos 😎 (pelo menos para esse exemplo)
- Agora não precisamos mais usar o Bellman-Ford e podemos usar o Dijkstra!

Algoritmo 3: Johnson(G)

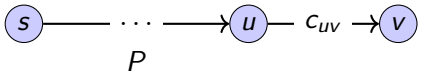
Entrada: Um grafo caminho G
Saída: Os custos dos caminhos mínimos entre todos os pares de vértices

1 Formar G', adicionando à G um vértice s extra e um novo arco (s, v) para todo $v \in V$ com custo 0;
2 Executar o Bellman-Ford em G' com fonte s (se detectar ciclo negativo, terminar);
3 **para** $v \in V$ **faça** $p_v =$ custo do $s - t$ caminho mínimo em G';
4 **para** $(u, v) \in A$ **faça** $c'_{uv} = c_{uv} + p_u - p_v$;
5 **para** $v \in V$ **faça** Executar o Dijkstra em G com fonte em v e os custos $\{c'_a\}$, obtendo os custos $d'(u, v)$;
6 **para** *cada par* $u, v \in V$ **faça** $d(u, v) = d'(u, v) - p_u + p_v$;
7 devolva d;

Tempo de execução:
$O(n) + O(mn) + O(m) + O(nm \log n) + O(n^2) = O(mn \log n)$

- Corretude: Como os caminhos de custo mínimo são preservados, se não houver custos negativos as execuções do Dijkstra os encontram corretamente, logo o algoritmo de Jonhson funciona.
- Resta demonstrar que de fato os custos $\{c'_a\}$ são não negativos.

- Considere um arco (u, v), por construção p_u é custo do caminho mínimo P de s a u, e p_v do caminho de s a v.



- Como $P + (u, v)$ é um possível caminho para v, certamente o caminho de custo mínimo tem um valor menor ou igual a esse. Ou seja

$$p_v \leq p_u + c_{uv}$$
$$0 \leq p_u + c_{uv} - p_v$$
$$c'_{uv} = p_u + c_{uv} - p_v \geq 0$$