# PAA - Aula ~~20~~ 24

## Caminhos Mínimos, Algoritmo de Bellman-Ford

No problema dos Caminhos Mínimos de um para todos recebemos: $|V| = n$ $|E| = m$

- um grafo orientado (ou dirigido) $G = (V, E)$
- um vértice origem $s \in V$
- e uma função de custos $c : E \to \mathbb{R}$
    - Note que o custo $c(e)$ pode ser negativo.

Queremos encontrar o valor do caminho mínimo de $s$ até cada vértice $v \in V$

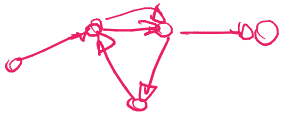
- Também gostaríamos que esses caminhos fossem devolvidos.

Lembre que o algoritmo guloso de Dijkstra leva tempo $O(m \lg n)$

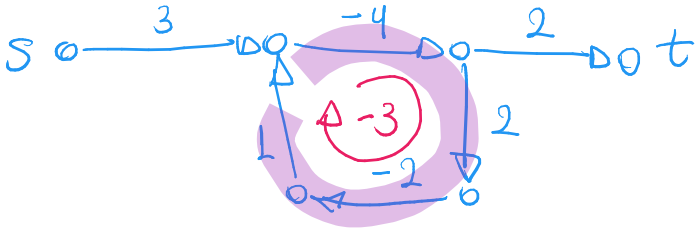
- mas não garante a resposta correta na presença de custos negativos.

- Note que, embora arestas de custo negativo não pareçam fazer sentido
    - quando representamos grandezas físicas como distância ou tempo,
- o problema de caminhos mínimos modela cenários mais gerais,
    - i.e., a sequência de decisões de menor custo até um objetivo,
- onde as decisões correspondem a ganhos ou perdas de algum recurso,
    - como em operações financeiras, manutenção de energia, etc.

Temos que definir mais precisamente o que é um caminho mínimo
- quando existem circuitos de custo negativo.



- Isso porque, se circuitos puderem fazer parte do caminho,
  - o caminho mínimo até um vértice pode ser arbitrariamente "pequeno".
- Basta percorrer o circuito inúmeras vezes antes de ir ao destino.




Por outro lado, se não permitirmos que circuitos façam parte do caminho,
- o que é bem razoável, o problema se torna NP-Difícil.
- Isso significa que ele não é tratável, i.e.,
  - não existe algoritmo polinomial para ele a menos que $P = NP$
    - Veremos a relevância dessa questão no fim da disciplina.
- Curiosidade: para demonstrar que tal problema é NP-Difícil,
  - podemos reduzir o problema do Caminho Hamiltoniano,
    - que é NP-Completo, para ele.

Nossa definição temporária para o problema de caminhos mínimos

- irá permitir que circuitos façam parte de caminhos,
  - pois assim o problema continua tratável,
- mas vamos supor que o grafo de entrada não possui circuitos negativos,
  - pois nesse caso um circuito nunca fará parte de um caminho mínimo.

Para perceber isso, suponha que

  - um caminho mínimo contém um circuito não negativo.


- Note que, removendo este circuito
  - o caminho continua conectando a origem ao destino
    - e seu custo não aumentou.

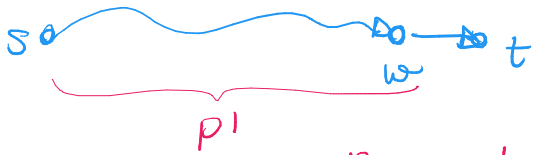
No final, a suposição de que não existem circuitos negativos não será necessária,

- pois nosso algoritmo será capaz de identificar a presença destes,
  - caso o grafo de entrada os contenha.

**Subestrutura ótima**

Começamos imaginando uma solução ótima. Neste problema queremos encontrar
- o caminho mínimo do vértice $s$ para todos os demais vértices em $V$
- mas vamos começar imaginando um caminho mínimo
  - de $s$ até um vértice $t$ específico.

A sequência dos vértices do caminho nos dá uma ordem implícita para seguir.
- Assim, olhemos para o último vértice do caminho
  - e foquemos na última aresta do mesmo.
- O último vértice é $t$ e digamos que a última aresta é $(w,t)$




$$c(p) = c(p') + c(w,t)$$

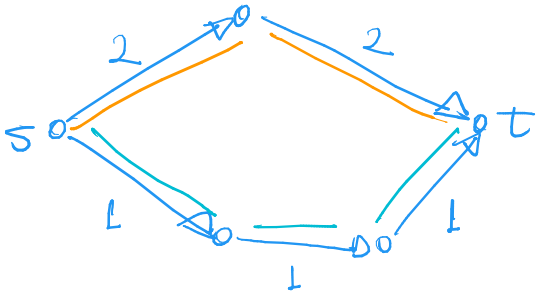
Assim, temos que nosso caminho $p$ de $s$ a $t$ é composto por:
- um caminho $p'$ de $s$ a $w$ e uma aresta $(w,t)$
- Portanto, a relação de custo é $c(p) = c(p') + c(w,t)$

Normalmente, seguiríamos provando que $p'$ é uma solução ótima do subproblema,
- mas vamos parar antes porque esta primeira tentativa tem uma falha.

Na subestrutura ótima, queremos definir/construir a solução ótima
    - em função de subproblemas menores.
- No entanto, não temos qualquer medida para indicar
    - que o problema do caminho mínimo de $s$ a $w$
        - é menor que o problema do caminho mínimo de $s$ a $t$

De fato, essa é uma dificuldade recorrente ao usar programação dinâmica
- para problemas em grafos, já que grafos são definidos por conjuntos,
    - eles não têm ordens bem definidas
        - que tornem óbvio qual problema é maior e qual é menor.

Uma solução para esse problema, usada no algoritmo de Bellman-Ford, é associar
- o tamanho do subproblema ao número de arestas permitidas no caminho.




$L_{1,t} = +\infty$

$L_{2,t} = 4$

$L_{3,t} = 3$

- Sendo $L_{i,t}$ o custo do caminho mínimo de $s$ a $t$ com no máximo $i$ arestas.

Usando esta ideia, temos uma sugestão de subestrutura ótima do problema.

imaginando a sol. ótima

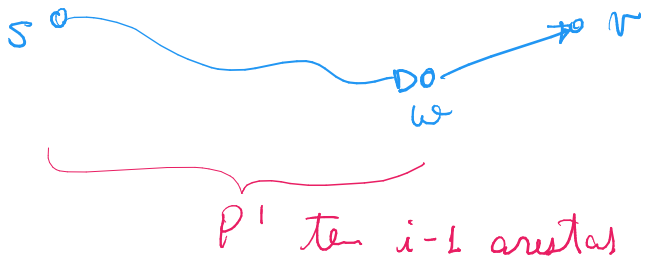
- Para cada vértice $v \in V$, e inteiro $i \in \{0, 1, 2, \ldots\}$ seja
  - $P$ o caminho mínimo de $s$ a $v$ com no máximo $i$ arestas.
- Temos dois casos.

Caso 1) Se $P$ tem menos que $i$ arestas, então

- $P$ é uma solução ótima do subproblema do caminho mínimo
  - de $s$ a $v$ com no máximo $i-1$ arestas.

Caso 2) Se $P$ tem $i$ arestas, seja $(w, v)$ a última aresta de $P$

- Neste caso, seja $P'$ a parte de $P$ que vai de $s$ até $w$




- Então, $P'$ é uma solução ótima do subproblema do caminho mínimo
  - de $s$ a $w$ com no máximo $i-1$ arestas.

Demonstração da otimalidade da subestrutura ótima:

Caso 1) $P$ tem menos que $i$ arestas.

- Queremos mostrar que P é ótimo
  - para o subproblema com $i-1$ arestas e destino
- Supondo, por contradição, que $P$ não é ótimo
  - para o subproblema com no máximo $i-1$ arestas,
  - então existe $P^*$ que é solução para este problema e $c(P^*) < c(P)$
- Como $P^*$ também é solução para o problema do caminho mínimo
  - com no máximo $i$ arestas, temos uma contradição
    - com a hipótese de que $P$ era ótimo.

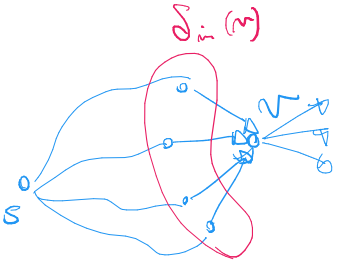
Caso 2) P tem $i$ arestas.

- Queremos mostrar que $P'$ é ótimo
  - para o subproblema com $i-1$ arestas e destino $w$
- Suponha, por contradição, que $P'$ não é ótimo para o subproblema
  - do caminho mínimo de $s$ a $w$ com no máximo $i-1$ arestas.
- Seja $P^*$ um caminho de $s$ a $w$ com no máximo $i-1$ arestas e $c(P^*) < c(P')$
  - Concatenando $P^*$ com $(w,v)$ temos um caminho com
    - no máximo $i$ arestas
      - e custo $c(P^*) + c(w,v) < c(P') + c(w,v) = c(P)$
  - o que contraria a hipótese de que $P$ era ótimo.

## Recorrência

Pela subestrutura ótima, nossa recorrência terá dois parâmetros:

- o número $i$ indicando o máximo de arestas no caminho mínimo,
- e o vértice $v$ que é o destino do caminho.




Na recorrência será escolhido o mínimo entre:

1. o custo do caminho mínimo até o próprio vértice destino $v$
    - mas com número máximo de arestas $i-1$
2. o custo do caminho mínimo com no máximo $i-1$ arestas
    - até um vizinho de $v$ que tenha aresta incidindo em $v$

Note que, não sabemos de qual vizinho virá o melhor caminho.

- Por isso temos que verificar todos.

- Sendo $\delta_{in}(v)$ o conjunto de arestas incidindo (entrando) no vértice $v$
    - o caso 2 na verdade são $|\delta_{in}(v)|$ casos,
        - um para cada uma dessas arestas.

Desse modo, para todo $v \in V$ e $i \in \{0, 1, 2, \ldots\}$ nossa recorrência será:

$$A[i, v] = \min \{ A[i-1, v], \min_{(w,v) \in E} \{ A[i-1, w] + c(w, v) \} \}$$

- com o primeiro termo do mínimo externo correspondendo ao Caso 1,
- e o segundo termo (o mínimo interno) correspondendo ao Caso 2.

A princípio não definimos para quantos valores de $i$

- vamos ter de calcular a recorrência.


- No entanto, note que se não temos circuitos negativos
  - sempre temos caminhos mínimos com no máximo $n-1$ arestas.

Isso porque qualquer caminho com $n$ ou mais arestas

- tem que repetir vértices e, portanto, forma pelo menos um circuito.

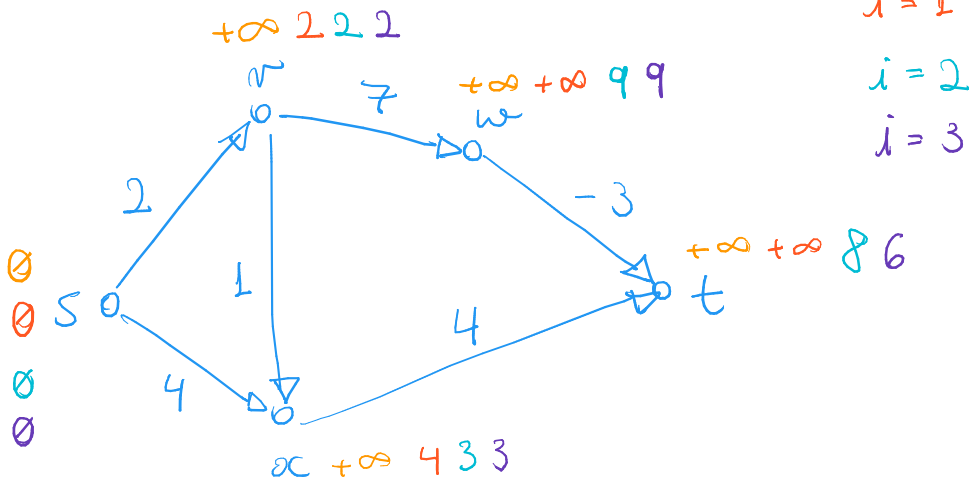
- Como os circuitos são não negativos,
  - o mesmo pode ser removido do caminho e o custo desse não piora.
- Assim, basta calcularmos nossa recorrência para $i = 0, \ldots, n-1$
  - para encontrar todos o caminhos mínimos.

Quiz1: Desenhe a matriz que será preenchida pelo algoritmo usando a recorrência.

- Destaque as células que tem os casos base
  - e aquela que guardará o valor da solução.

# Exemplo do algoritmo de Bellman-Ford

$$A[i,v] = \min\{\underbrace{A[i-1,v]}_{1^\circ \text{ termo}}, \overbrace{\min_{(w,v)\in E}\{A[i-1,w] + c(w,v)\}}^{2^\circ \text{ termo}}\}$$



## Pseudocódigo do algoritmo de Bellmand-Ford

algBellman-Ford(G=(V,E), c, s):

custos nas arestas
vértice origem

para $v \in V$: $A[0,v] = +\infty$

$A[0,s] = 0$

$\Theta(n)$ para $i = 1$ até $n-1$:

$\Theta(m)$ para $v \in V$:

$A[i,v] = \min\{A[i-1,v], \min_{[w,v)\in E]}\{A[i-1,w] + c(w,v)\}\}$

$\Theta(|\delta_{in}(v)|)$

$[w \in \delta_{in}(v)]$

devolva valores em $A[n-1, v]$ p/ todo $v$

Eficiência: Note que são $\Theta(n^2)$ subproblemas por resolver,

- mas o custo para resolver cada subproblema não é constante.

- Esse custo é $\Theta(|\delta_{in}(v)|)$ para o vértice $v$
  - Isso porque é necessário consultar $|\delta(v)| + 1$ posições da matriz.
- Focando em uma iteração do laço mais externo, temos que
  - o número de operações na execução do laço interno será da ordem

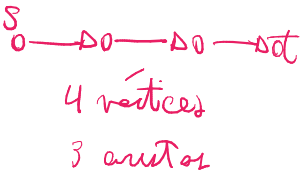
$$\sum_{v \in V} (|\delta_{in}(v)| + 1) = \Theta(m)$$

- Como o laço mais interno é executado $\Theta(n)$ vezes, pois o laço externo realiza $\Theta(n)$ iterações
  - o tempo total do algoritmo é da ordem $\Theta(n \cdot m)$

## Detectando circuito negativo

Para que o algoritmo de Bellman-Ford detecte se o grafo de entrada
    - possui circuitos negativos, basta fazer as seguintes modificações:
- Executar o laço mais externo do algoritmo até $i = n$ e
    - verificar se o valor $A[i, v]$ muda para algum $v$ na $n$-ésima iteração.
- Se ocorrer alguma mudança é porque existem circuitos negativos.
    - Mais precisamente, circuitos negativos alcançáveis a partir de $s$

A intuição do porque isso funciona é que na iteração $n-1$
    - todos os caminhos mínimos com até $n-1$ arestas já estão calculados.
- Assim, se algum caminho reduz o custo na iteração $n$
    - é por usar $n$ arestas.
- Sabemos que esse caminho repete vértices, pois um caminho
    - sempre usa um vértice a mais que seu número de arestas.
- Se ele repete vértices, forma algum circuito.
    - Como o custo diminuiu, esse circuito tem custo negativo.
- De fato, para encontrar tal circuito basta seguir o caminho de trás pra frente
    - a partir do vértice cujo custo foi reduzido
        - usando a ideia de reconstruir a solução a partir da matriz $A$




Para mostrar formalmente que o resultado vale, vamos provar o seguinte lema:

- $G$ não tem circuitos negativos se, e somente se, $\Leftrightarrow$
  - no algoritmo de Bellman-Ford $A[n-1,v] = A[n,v]$ para todo $v \in V$

alcançável a partir de $s$

Prova: ($\rightarrow$) A ida segue da corretude do algoritmo de Bellman-Ford

- e do fato que, na ausência de circuitos negativos
  - todo caminho mínimo tem no máximo $n-1$ arestas.
- Assim, $A[n-1,v]$ possui o valor do caminho mínimo de $s$ a $v$
  - e permitir que se use uma aresta a mais de nada vai ajudar.

$A[i,v]$ custo caminho mínimo de $s$ a $v$ c/ $\leq i$ arestas

($\leftarrow$) Na volta, considere um circuito qualquer $C$ e

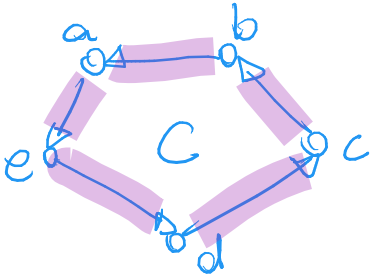
- vamos mostrar que ele não tem custo negativo.
- Note que, para cada vértice $v$ em $C$, temos

$$A[n-1,v] = A[n,v] = \min\{A[n-1,v], \min_{(w,v)\in E}\{A[n-1,w] + c(w,v)\}\}$$
$$\leq A[n-1,w] + c(w,v)$$


- onde a primeira igualdade vem da hipótese da volta,
  - a segunda igualdade vem da recorrência do algoritmo,
  - e a desigualdade vale pela definição do mínimo
    - para qualquer $w$ tal que $(w,v) \in E$

Sendo $w$ o vértice que precede $v$ em $C$, temos

$$A[n-1,v] \leq A[n-1,w] + c(w,v) \Rightarrow c(w,v) \geq A[n-1,v] - A[n-1,w]$$




Logo

$$\begin{cases} c(b,a) \geq A[n-1,a] - A[n-1,b] \\ c(c,b) \geq A[n-1,b] - A[n-1,c] \\ c(d,c) \geq A[n-1,c] - A[n-1,d] \\ c(e,d) \geq A[n-1,d] - A[n-1,e] \\ c(a,e) \geq A[n-1,e] - A[n-1,a] \end{cases}$$

Finalmente, somando o custo de todas as arestas em $C$ temos

$$\sum_{(w,v) \in C} c(w,v) \geq \sum_{(w,v) \in C} [A[n-1,v] - A[n-1,w]] = 0$$

- sendo que a desigualdade segue do cancelamento dos termos $A[n-1,v]$
  - já que cada um aparece duas vezes no somatório,
    - uma vez com o sinal positivo e outra com o sinal negativo.

Assim, concluímos que o custo de qualquer circuito $C$ é não negativo

- quando os valores não mudam entre as iterações $n-1$ e $n$

**Duas melhorias interessantes no algoritmo de Bellman-Ford**

1) Parar assim que os valores $A[i,v]$ não mudarem de uma iteração para outra,
   - i.e., se duas linhas consecutivas forem idênticas.

2) Utilizar vetores $A[v]$ e $B[v]$, para guardar o valor do caminho mínimo até $v$
   - e o predecessor de $v$ nesse caminho, respectivamente.

algMelhoradoBellman-Ford(G=(V,E), c, s):

para $v \in V$: $A[v] = +\infty$

$A[s] = 0$

para $i = 1$ até $n$:

para $v \in V$:

$A[v] = \min\{A[v], \min_{(w,v)\in E}\{A[w] + c(w,v)\}\}$

$B[v] = w$, sendo $w$ o vértice que minimizou a recorrência

se não houve mudança em $A$, devolva $A$ e $B$

- devolva existe ciclo negativo

Quiz2: Por que posso usar só o vetor $A[v]$ ao invés da matriz $A[i,v]$?
   - Dica: observe de quais linhas da matriz a recorrência precisa.