

* Materia codunq

06/08

$\text{Adj}(v)$ = Vértices adjacentes a v

$d(v,v)$ = menor caminho da v a v em G

$g(v) = |\text{Adj}(v)| \leftarrow$ Não direcionado

08/08

Digraph \rightarrow $\text{Adj}^-(v)$ e $\text{Adj}^+(v)$

$g(v) = g^-(v) + g^+(v)$

\Rightarrow resto inserion do problema

13/08

Algoritmo: Grau dos vértices $\Theta(m+n)$

Entrada: Grafo $G = (V, E)$

Saída: vetor g onde $g(v)$ é o grau de v

1. Para todos $v \in V(G)$
2. $g(v) \leftarrow \emptyset$
3. Para Todo $v \in V(G)$
4. Para Todo $w \in \text{Adj}(v)$
5. $g(v) \leftarrow g(v) + 1$

15/08

Pilha/Fila \rightarrow E.D. com tempo $\Theta(|E|) \rightarrow$ Estrutura M

Vértices:

- Explorados
- visitados/marcados (raiz da Busca)
- Não visitados/marcados

Todos vértices
em M vai
ser marcados

- block 2
- gray 1
- white 0

Adj sempre em ordem lexicográfica

$C(v)$ = cor de v

r = raiz da busca

Visite $(v,w) \rightarrow vw$ vai para saída e apontador vai pra próxima aresta de $\text{Adj}(v)$

Algoritmo: Busca广度 em Grafos

Entrada: Grafo $G = (V, E)$ conexo e $r \in V(G)$

Saída: Lista dos arcos de G

1. Para Todo $v \in V(G) \setminus \{r\}$
2. | $c(v) = 0$
3. | $c(r) = 1$
4. | $M \leftarrow \{r\}$
5. | Enquanto $M \neq \emptyset$
6. | | Seja $v \in M$
7. | | Se existe $w \in \text{Adj}(v)$ com $c(vw) = 0$
8. | | | Se $c(w) = 0$ então $c(w) = 1$
9. | | | $M = M \cup \{w\}$
10. | | | Visite (vw)
11. | | | Se não Visite (vw)
12. | | | Se não
13. | | | | $c(v) = 2$
14. | | | | $M = M \setminus \{v\}$

Alg Graf 20/08/19

Algoritmo: Busca广度 em Grafos

Linha no Tamanho da Entrada

* Verificar linha a linha e
logos

• complexidade

• Espaço (estrutura)

Entrada $O(n+m)$

Saída $O(m)$

$O(mn)$ onde $n=1$ número de repetições
Cada nó é Adj(G) é listado uma vez

Armazena vértices (armazenados uma única vez)

M é $O(n)$

Vetor indexado por $G(v)$ e $|v(G)| = n$

C é $O(n)$

v, w são $O(1)$

Total $\rightarrow O(n+m) + O(m) + O(n) + O(n) + O(1) = O(n+m)$

• Tempo (Algoritmo) Passos

Linhos 2, 3, 4, 5, 9, 14 São $O(1)$ Acesso ao vértice

Linha 6 é $O(1)$ encontro da M com um elemento

Linhos 5, 6, 10, 15 São $O(1)$ pois M é uma E.D. com inclusão, retirada e escolha de elemento em $O(1)$

Linhos 11, 12 São VISITÉ arista. Atualizar pending e imprimir uma aresta. Logo São $O(1)$

Linha 13 é $O(1)$ pois Adj(G) tem um pending para cada vértice para a última aresta visitada

Logo $L_3 - L_2 = O(n)$

Logo $L_5 - L_{15} = O(n+m)$ pois cada vértice v é considerado na L6, adicionando g(v)+1 vez a ponte de tamanho

$$\sum_{v \in V(G)} (g(v)+1) = m + \sum_{v \in V(G)} g(v) = n + 2m = O(n+m)$$

* Seja $v \in M \rightarrow$ tempo garantido para retirada da E.D.

Linha 6

$\rightarrow \pi(w) = v$ se v é vértice que marca w na busca
do vetor indexado por $v(\theta)$

Linha 11: $v(\theta)$

Linha 12: $v(\theta)$

* Após $M = \{r\}$: $\pi(r) \leftarrow r$ raiz

* Após $M = M \cup \{w\}$: $\pi(w) \leftarrow v$

Fato: a estrutura π é uma árvore geradora de G

Como G é conexo, todos os vértices são marcados
durante o algoritmo, portanto, todos tem um valor em
 π , isto é, eles aparecem na lista de arestas de π ,
logo π é gerador de G

* Prova: n-1 arestas
Como $\pi(r) = r$, como $\pi(v) \neq v \forall v \in V(G) \setminus \{r\}$
temos exatamente $n-1$ arestas.

* Prova

Como a cada vez que une aresta u incluída em π , ela
liga um vértice marcado a um não marcado que imediatamente
possa ser marcado. π é uma estrutura acíclica, logo π é aciclico

Obs: Se G é disconexo, π é uma floresta geradora

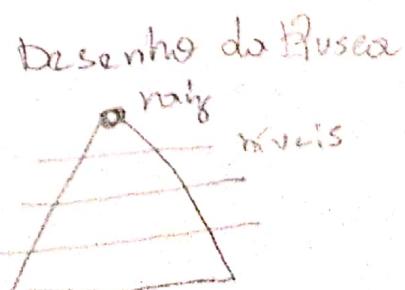
visitado - vértice da árvore

visitado - vértice da árvore

M Busca em π

i pilha Profundidade árvore de profundidade

i fila Largura árvore de largura



a x | b
c

ra a
má p i ma

Prezib da
paternidade
a
b
c
d

D

filha
Auxílio

[ab] ba 2^a visita

[bc] ca 1^a visita

eb

[bd]

db

ae

e |
| a | b | c
| b | c | d | a
| c | d | b |
| d | b | c |

08/119 Alg. Graf.

$\pi(v) \rightarrow$ Pai de v na árvore de Busca $H(v) = n$

nível(v) \rightarrow nº de amigos do caminho da raiz até v na árvore de busca

• Aresta ab

$$e(b)=1 \quad M \leftarrow M \cup \{b\}$$

$$\pi(b)=a \quad L(b)=i \quad e \quad i++$$

$$\text{nível}(b) = \text{nível}(a) + 1$$

visit₂(ab)

• Hé fila

$$L(v)=i$$

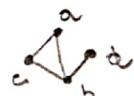
i-ésimo a entrar
ou sair

• Mé pilha

$$PE(x)=i \quad i-\text{ésimo a entrar}$$

$$PS(x)=i \quad i-\text{ésimo a sair}$$

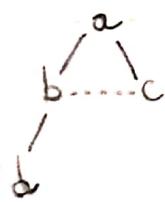
* Busca em Largura:



$r=a$

nível	L	c	
0	1	2	$a \rightarrow b \& c$
1	2	2	$b \rightarrow d \& c$
2	3	2	$c \rightarrow \emptyset$
3	4	2	$d \rightarrow \emptyset$

árvore de Buscar



Saída

ab - árvore

ac - árvore

ba - 2ª visita

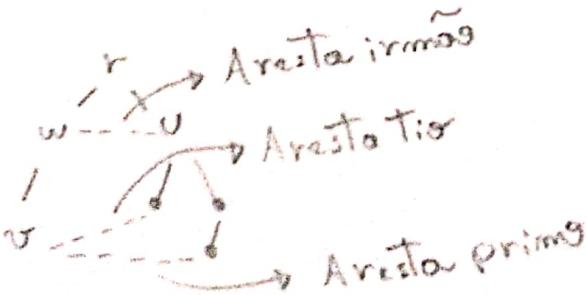
bc - árvore

bd - árvore

ca - 2ª visita

cb - 2ª visita

da - 2ª visita



Se não possuir aresta primo num
irmão na árvore de busca, o graf
é bipartido

03/09/19 Alg Gráf

Algoritmo ótimo \rightarrow cota inferior do problema

Teorema: Se $u, v \in E(G)$, G não direcionado, então em qualquer ABL de G

$$|nível(u) - nível(v)| \leq 1$$

Busca em Largura em Grafos direcionados: Não cai

Em uma ABL $d(u, v) \leq nível(v)$, pois existe caminho com \dagger de u a v com $nível(v)$ arestas

Teorema: Em toda ABL $d(u, v) = nível(v) + v \in V(G)$

$$nível(u) = nível(v)$$

$$\pi(u) = v$$

uv é aresta irmão

Ciclo ímpar

$$\pi(u) \neq v$$

uv é aresta primo

Ciclo ímpar

$$nível(u) < nível(v)$$

$$\pi(v) = u$$

uv é aresta pai (árvore)

$$\pi(v) \neq u$$

uv é aresta filha

* Achar um triângulo é um problema em aberto: fomos
obrigados achar uma aresta primo ou par

Tem ciclo ímpar pois cada aresta faz um triângulo com os caminhos
de seus extremos até o seu ancestral comum em \dagger

Teorema: G é bipartido $\Leftrightarrow G$ não tem ciclos ímpares

Algoritmo:

1. BL em G classificando os frondes

2. Se tiver aresta primo ou írmão, G não é bipartido
Sendo:

$$X = \{v \mid nível(v) \text{ ímpar}\}$$

$$Y = \{v \mid nível(v) \text{ é par}\}$$

Graf 05/09/19

* Busca em Profundidade (BP) em grafos (não direcionados)

ABP → árvore de busca em profundidade

$PE(v)$ = ordem em que v entra em M (Pilha)

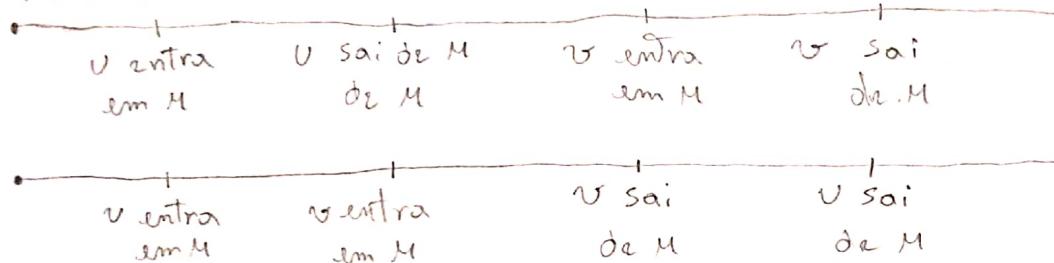
$PS(v)$ = ordem em que v sai de M (Pilha)

$r \rightarrow$ raiz da ABP

$PE(r) = 1$

$PS(r) = n$

$PE(u) < PE(v)$

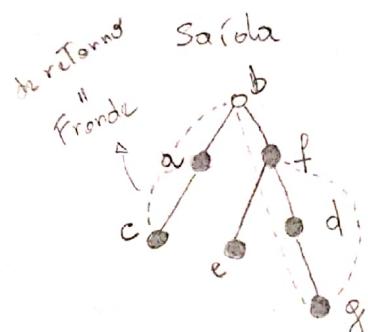
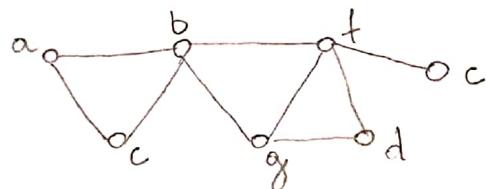


$\rightarrow uv \notin E(G)$

$\rightarrow u$ é ancestral de v na ABP.

Exemple

$$\begin{aligned} n &= 7 \\ r &= b \end{aligned}$$



PS	PE	lowpt	
2	2	$a \rightarrow \cancel{b} \cancel{c}$	b
7	1	$b \rightarrow \cancel{d} \cancel{e} \cancel{f} \cancel{g}$	b
1	3	$c \rightarrow \cancel{d} \cancel{b}$	b
4	5	$d \rightarrow \cancel{f} \cancel{g}$	b
5	7	$e \rightarrow \cancel{f}$	e
6	4	$f \rightarrow \cancel{g} \cancel{b} \cancel{c}$	b
3	6	$g \rightarrow \cancel{b} \cancel{c} \cancel{f}$	b

→ Pilha

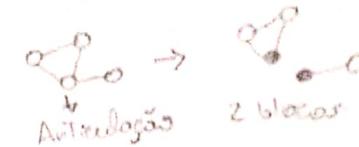
* Verificar se é fronde

* Correr marca de 1 até 2n links de tempo único

→ Um vértice v é explorado em uma BP somente após todos os seus descendentes estarem explorados

* Aplicações

- Determinação de articulações e blocos
- Articulações = vértice de corte



* Se raiz da ABP tem mais de um filho \leftrightarrow raiz é articulação

Se tem um filho não pode ser articulação

* Se G é órvore menor, nenhuma aresta da ABP é fronde

* Grafo conexo \rightarrow busca de todos vértices explorados

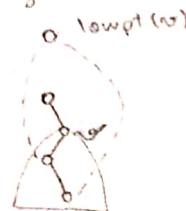
* Grafo conexo

\rightarrow Algoritmo: $O(n \cdot \underbrace{O(n+m)}_{\substack{\text{Para} \\ \text{cada} \\ \text{vértice}}}) = O(n(n+m)) = O(n^2 + mn) = O(n \cdot m)$
 $m \approx n-1$

\rightarrow Algoritmo: Achar articulações com uma única BP (tarjan)

v não é raiz

$lowpt(v)$ = vértice mais perto da raiz (menor nível) que pode ser alcançado a partir de v por um caminho descendente formado por arestas de árvore seguido por no máximo uma fronde.



* Descendente: Possivelmente 1
fronde a partir da própria v

v é articulação $\leftrightarrow v$ tem filho w com $lowpt(w) = w$ ou v
 $\hookrightarrow w$ é demarcador de v

8 Graf 30/09/19

→ Determinar lowpt(v) + v

→ Definição: Se $G - v$ é disc连xoxo (i.e se $G - v$ tem menos componentes conexos do que G), então v é anticelegível.

→ v é anticelegível \Leftrightarrow Se v é raiz e v tem mais de um filho
• Se v não é raiz de t , onde t é ABP
e v tem filho w com $\text{lowpt}(w) = v$

→ Se t não tem outros descendentes além de w $V(t) \cup \{v\}$ é um componente biconexo de G

Para cada $v \in V(G)$

$\text{lowpt}(v) + v$

→ No inicio $O(|V(G)|)$

Em visita

• se v é retorno

✓ Se $\pi(v) \neq w$

tudo

$O(1)$

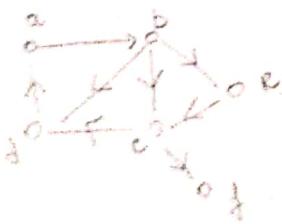
Se nível($\text{lowpt}(v)$) < nível(w)

$\text{lowpt}(v) + w$

• se v é explorado

• para cada $w \in \text{Adj}(v)$
Se $\pi(w) = v$
Se nível($\text{lowpt}(w)$) < nível($\text{lowpt}(v)$)
 $\text{lowpt}(v) \leftarrow \text{lowpt}(w)$

* BP em Grafo Direcionado (não possui segunda visita)



PS	PE	
6	1	$a \rightarrow b$
5	2	$b \rightarrow c, d, e$
3	3	$c \rightarrow d, e$
1	4	$d \rightarrow a$
4	6	$e \rightarrow c$
2	5	$f \rightarrow \text{null}$

pilha

e
f
d
c
b
a



$u \in E(G)$

Aresta de retorno

$$PE(u) > PE(v) \\ PS(u) < PS(v)$$

Aresta de avanço

$$PE(u) < PE(v) \\ PS(u) > PS(v) \\ \pi(v) + u$$

Aresta de cruzamento

$$PE(u) > PE(v) \\ PS(u) > PS(v)$$

→ Definição: G é fortemente conexo se existem caminhos de u para v entre todos os pares $u, v \in V(G)$

→ Fato: os componentes fortemente conexos (efe) de G formam uma partição de $V(G)$.

Bicamente partitiona os vértices

Cada vértice participa de exatamente 1 efe

→ O grafo induzido de um digrafo $R(G)$ é obtido pela aglutinação de todos os vértices de um mesmo efe em um único vértice.

E as arestas de $R(G)$ ligam efe's de G da mesma forma que em G

Conclusão $\rightarrow R(G)$ é acíclico

→ Definição: Um digrafo é frequentemente conexo se o seu grafo subjacente for conexo

Obtido tirando
orientações das arestas

→ Definição: Um digrafo é unilateralmente conexo se existe $v \in V(G)$ tal que existem caminhos de v para todo vértice de $V(G)$

Neste caso, v é chamado de raiz do digrafo

18 Graf 16/09/19

* Elementos Fortemente Cenexes (efc)

B.P. (G digraph)

* Lema: Se T é uma floresta de profundidade de G e $H = (S, E_S)$
é um componente fortemente conexo de G , então H em T é uma
subárvore de uma árvore de T

→ Não podem haver arestas de cruzamento entre os arcos.
→ Arco e sentidos (sempre direita pra esquerda)

Podem ser de 2 sentidos (simples) ou de 1 sentido (simples).
Pode haver 2 sentidos (simples) e 2 sentidos (simples).

$t_r \partial x^+$

$\vdash r \in T$
Supondo por absurdo que $r \in T$ não está em S , mas é o pai de r no T .
 \vdash não existir caminhos de r para v



Devem existir caminhos de radares
neste sistema de radar
em 5 portais


Arista no exercitando com caminho de var forma um caminho de var, de modo que var esteja na mesma cfl.
Podemos ter vários cfl's em uma mesma árvore

Algorithm: Kosaraju - Sharir

Entrada: Signo G

Solutions: Cyclic's or G

1. BP (G) determining PS

2. Determinar β^T (trocar direção das arestas)

3. BP(G^+), pegando como raiz a cada vez, o vértice ainda não explorado com maior PS

↳ Vai me dar as órveras das eje's

→ Grupo nitrogenado: $\text{a} \cdot \text{b} \rightarrow \text{c} \cdot \text{d}$

Digitalizado com CamScanner

* Complexidade de Kosaraju-Sharir

Esp^{ço}

Entrada $\Theta(n+m)$

Saída $\Theta(n)$

G^+ $\Theta(n+m)$

PS $\Theta(n)$

$\Theta(n+m)$

tempo

Linha 1 - BP(G) $\rightarrow \Theta(m+n)$

Linha 2 - $G^+ \rightarrow \Theta(m+n)$

Linha 3 - BP(G^+) $\rightarrow \Theta(n)$

• ordena $V(G)$ por PS crescente - $\Theta(n)$

pois $i \in PS(v) \leq n$

→ Pode visitar no

• percorrer PS - $\Theta(n)$ explorar

• sempre ocorre em $\Theta(n+m)$

Próprio índice da árvore

Percorre todos vértices para visitar vértice $c(v)$ cor

Fato: G e G^+ tem os mesmos efe's

* Pela parte inicial do teorema 1 é o Fato de cada efe de G está em uma árvore de floresta de $BP(G^+)$

* Mostrar que em uma mesma Tr_1 de $BP(G^+)$ não tem mais de um efe

Suponha que $xy \in E(G^+)$, xy seja orientada de Tr_1 com r_1 e x no mesmo efe e y em outro



Em G , temos yx é em $BP(G)$, yx foi classificada como:

- orientação
x não oriente
já que xy oriente
em G^+ , portanto
na mesma efe

- árvore de avanço
 $PS(y) > PS(x)$
não oriente
 $PS(r_1) < PS(x)$
pois r_1 foi escolhido
para ser raiz de Tr
domínio não oriente
 $PS(r_1) < PS(y)$

- Crigamento

 $PS(y) > PS(x) > PS(r_1)$
não ocorre para r_1
é raiz de Tr

1. Algoritmos Estrutural

holly exemplo
exponencial/poisson

2. Método Guloso

3. Programação Dinâmica

Algoritmos estrutural

- Modificações na entrada (estrutural)
- Resolver problema na entrada modificada e converter para entrada original

Modificações estruturais em Grafos

- Dado $G = (V, E)$
- $G - \{v\}$ ou grupo obtido pelo retíngulo de v e seus arestas incidentes
- $G \pm S \rightarrow$ inclusão/exclusão dos vértices/arestas de S no grafo
- Em geral incluir vértice e suas arestas incidentes
- Aglomeração de vértices não adjacentes

$$\begin{array}{c} a-b \\ | \quad | \\ c-d \end{array} \rightarrow \begin{array}{c} ad-c \\ | \quad | \\ b \end{array}$$

- Decomposição por conta de vértices

$\text{SCV}(G)$ com $G-S$ com mais componentes conexas que G

$$\begin{array}{c} a-c \sim e \\ | \quad | \\ b-d \end{array} \xrightarrow{\{b,c\}} \begin{array}{ccc} G_1 & G_2 & G_3 \\ b \backslash c & b \backslash c & b \backslash c \\ a & d & e \end{array}$$

$$S = \{c\} \text{ é articulação } |S| = 1$$

$$\begin{array}{c} a-c \\ | \quad | \\ b-d \end{array} \quad \begin{array}{c} c \\ | \\ e \end{array} \quad \text{e componentes biconexas}$$

Problema de coloração de vértices

Dado grafo $G = (V, E)$ não direcionado e simples

Dado grafo $G = (V, E)$ não direcionado e simples
 Encontrar o número cromático de G , que é o menor número
 cores de alguma coloração de G

de cores de alguma coloração de G
 Uma coloração de G é uma distribuição de cores em $\{1, 2, \dots, k\}$
 para vértices de forma que vizinhos adjacentes tenham cores diferentes.
 Exemplo primitivo

$\chi(G) \rightarrow$ número de vértices
 Exemplo: $\chi(K_3) = \leq 3$

Exemplo: $X^{(3)}$,
que é possível elaborar com menos de 3 vértices
mutuamente adjacentes

metivamente ad $\{l, 2, 3, \dots, K\}$

Se é um clonístico se tem uma K- coloração

$\text{G} \in K$ -eulerável se e somente se K -conectado.

$$\chi(G) \leq k$$

S. $\chi(G) = k$, G é k-connexão

$$\text{Ex } \chi(G) = k, G \cong K_n \rightarrow n \text{ vertices } G = N_n$$

$$\neg \exists - \chi(G) = 1 \iff E(G) = \emptyset \rightarrow \text{if } \exists - E(G) \neq \emptyset$$

$\Theta(n) - \chi(G) = 1 \iff$ $E(G) \neq \emptyset$
 $\Theta(m+n) - \chi(G) = 2 \iff$ G é bipartido
 ↳ Busca em largura

$\chi(G) = 3 \rightarrow ?$ Problema NP-completo
 mesmo p/ $\Delta \leq 3$ e planares \rightarrow teorema dos 4 cores
 Lo Maior grau de G

Algoritmo: Zykov

Entrada: Grafo G
Saída: $X(G)$

\leftarrow Entrada: G → Saída: $X(G)$ → comp.

Senso

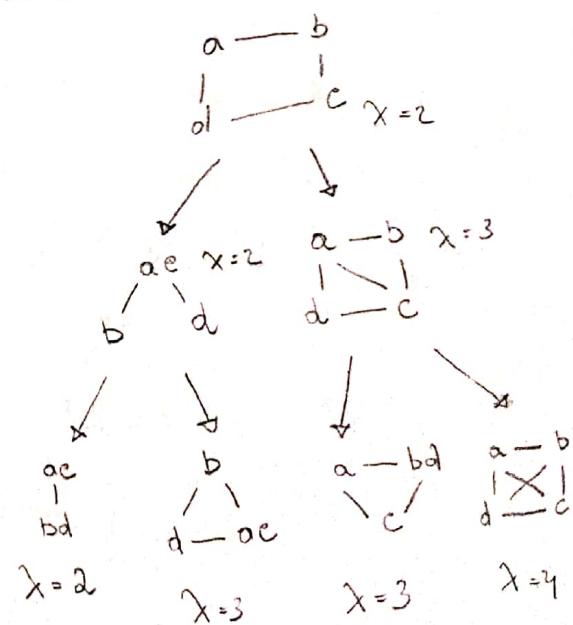
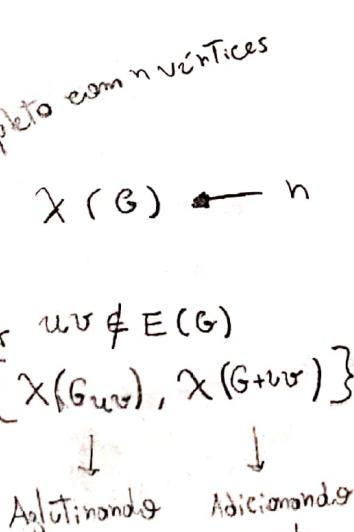
Sejam v, w tais que $vw \notin E(G)$

$$x(G) \leftarrow \min\{x(G_{uv}), x(G+uv)\}$$

$\Theta(2^n)$

$$\chi(G) \leq \min c$$

$\chi(6) \geq \min$



~~Alg-Graf~~ 26/09/19.

* Problemas de Optimização Combinatória
Dominio da função é um conjunto discreto

Exemplo: Coloração de vértices

Entrada: Grafo $G = (V, E)$

$\chi(G)$ = número cromático de G

$\min \{k \mid G \text{ é } k\text{-colorível}\}$

Cada ponto do domínio é uma coloração de G e valor da função no ponto é o nº de cores usadas nessa coloração.

Problema decidível: Possui algoritmo exponencial que o resolve, a princípio, testando todos os casos possíveis.

* Método guloso

escolhe-se um ponto do domínio, um elemento por vez, que é o melhor em termos de escelta. Essa iteração um elemento que é o melhor em termos de escelta. Tal que os elementos já escolhidos de um conjunto guloso d e Tal que os elementos já escolhidos juntos com este novo elemento continue a ser parte de um ponto do domínio (satisfaçõe a propriedade P)

Garantia: Um ponto do domínio é sempre encontrado

→ Algoritmo guloso nunca se arrepende. Após escolher ponto não pode voltar atrás.

Método Guloso para colorações de vértices

1) Considerar vértices de G em uma ordem v_1, v_2, \dots, v_n

2) Para i de 1 até n → critério d

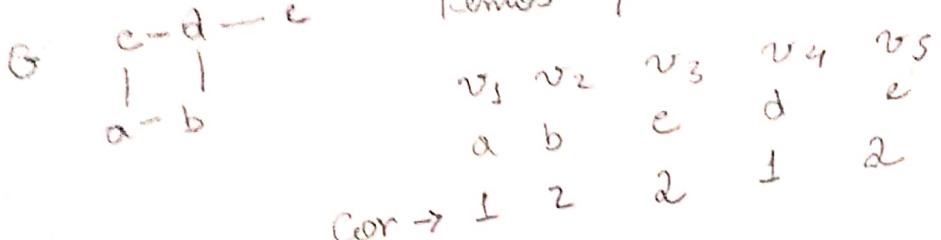
com $c(v_i) \rightarrow$ a menor cor em $\{1, 2, \dots, n\}$ que pode ser

atribuída a v_i , que seja compatível com a restrição $\rightarrow P$

de $G \{v_1, v_2, \dots, v_{i-1}\}$

$$\begin{array}{c} v_i \\ v_j \end{array} \quad \begin{array}{l} \text{se } i > j \\ \text{então } i = j + 1 \end{array}$$

Para: $\phi = \{1, \dots, 5\}$ cores Valor = 2



Cor \rightarrow

Com outra ordem de vértices, como $a \ e \ b \ e \ d$ temos Valor = 3
Não sempre é possível encontrar uma coloração ótima

• Provar pelo Teorema da corerdade

* Algoritmo genérico pra metade gulosa

1. Ordene elementos de E segundo d
2. $S \leftarrow \emptyset$
3. Para $i \leftarrow 1$ até $|E|$, se e_i satisfaçõe P_i , então $S \leftarrow S \cup \{e_i\}$
4. Se $S \cup \{e_i\}$ satisfizer P_i , então $S \leftarrow S \cup \{e_i\}$

Próxima Aula: Árvore Geradora Mínima

Dado: Grafo não direcionado com pesos nos arestas $p: E \rightarrow \mathbb{Q}$

Dado: Grafo não direcionado com pesos nos arestas $p: E \rightarrow \mathbb{Q}$
Vamos falar sobre Árvore Geradora Mínima (T) que é um subgrafo de G que satisfaça $V(T) = V(G)$ tal que

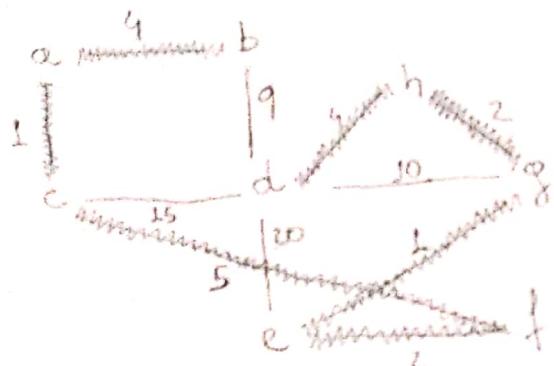
$$\sum_{e \in E(T)} p(e) \leq \text{mínimo}$$

8 Graf 01/30/19

Árvore Geradora Mínima (AGM)

Dado Grafo $G = (V, E)$ \rightarrow conexo
sonsente AGM
 $p: E(G) \rightarrow \mathbb{Q}^+$ Peso não negativo

- Achar 1 AGM de G e seu peso



$$\begin{aligned}ac &= 1 \\cb &= 1 \\cd &= 2 \\ch &= 2 \\cb &= 4 \\dh &= 4 \\cf &= 5 \\fd &= 9 \\df &= 10 \\ce &= 5 \\ef &= 10 \end{aligned}$$

- Entrada Gulosa
 - a considerar a aresta de menor peso primeiro

- Propriedade
 - Mantendo grafo em construção acíclico (se conexo, eliminando arestas de maior peso.)

Algoritmo: A.G. Min - Kruskal

Entrada: Grafo $G = (V, E)$ conexo e $p: E(G) \rightarrow \mathbb{Q}^+$ peso da AGM

Saída: $A \subset E(G)$ t.q. $t = (V, A)$ é A.G. Min de G e P peso da AGM

1. Ordenar $E(G)$ em ordem não decrescente. isto é: $e_1 < e_2 \Leftrightarrow p(e_1) \leq p(e_2)$

2. $A \leftarrow \emptyset$ $P \leftarrow 0$

3. Para $i \leftarrow 1$ até $m = |E(G)|$ faça

4. Seja $e = uv$

5. Se $\#A \cup \{e_i\}$ é acíclico: $A \leftarrow A \cup \{e_i\}$

Se na linha 5 usássemos uma busca com raiz u entre se v estiver no mesmo componente conexo que v, teríamos complexidade $\Theta(n+m)$ para cada aresta. Verificada dando a exemplaridade de L3-L5 de $\Theta(m(\sqrt{nm})) = \Theta(m^2)$

$$\text{L3-L5 de } \Theta(m(\sqrt{nm})) = \Theta(m^2)$$

Como $m > n$, pois G é conexo
e sólido de vértices

* Utilizando Union - Find (Criado para o problema da AGMin) por Torjan

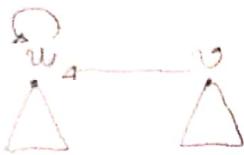
Set(u) - cria conjunho unitário $\{u\}$

$\Theta(1)$

\bullet_u

Union(x, y) = $r_x \rightarrow$ raiz da árvore obtida pelo novo conjunto

$\Theta(1)$



Monta árvore mais equilibrada possível

Find(x) = r_x

$\Theta(\log n)$

↓

Com malhação

na Union (compressão de caminhos)

$\Theta(\lg^* n)$

Quase constante

→ Vertice raiz da árvore que representa o conjunto que contém x

Linha 3.5 Para $i \leftarrow 1$ ate n

set(v_i)

* Nova Linha 5

→ Se $\text{Find}(u) \neq \text{Find}(v)$: $A \leftarrow A \cup \{e_i\}$
 $\text{Union}(\text{Find}(u), \text{Find}(v))$
 $P \leftarrow P + p(uv)$
 (→ Atualizando Piso da AGM)

* Complexidade (usando union find)

L5 $\rightarrow \Theta(\log n)$ para cada Find

L6 $\rightarrow \Theta(1)$ para cada Union

L3-L6 $\rightarrow \Theta(m \log n)$

L2 $\rightarrow \Theta(1)$

L1.5 $\rightarrow \Theta(n)$

L1 $\rightarrow \Theta(m \log m) = \Theta(m \log n)$

Saída $\rightarrow \Theta(n)$

Entrada $\rightarrow \Theta(n+m)$

Total $\rightarrow \Theta(m \log n)$

ornitudo

$t = (V, A)$ é A.G. por construção

Supondo que não seja uma A.G. mostraremos a árvore com peso mínimo

Suponha por absurdo que $t = (V, A)$ não é uma A.G. mínima

Suponha por absurdo que $t = (V, A)$ não é uma A.G. mínima

Sejam e_1, e_2, \dots, e_{n-1} os arestas escolhidas pelo algoritmo AGMin-Kruskal

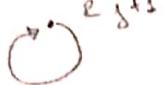
$$p(e_1) \leq p(e_2) \leq \dots \leq p(e_{n-1})$$

e seja $+^*$ uma AGM de G entre todos os AGM's de G com $p(+^*) < p(+)$ escolhida
j maior possível

$$e_1, e_2, \dots, e_{j+1}, e_{j+2}, \dots, e_{n-1} \quad j \neq n-1$$

Seja $+^* + e_{j+1}$

e_{j+2}



$$f \in E(+^*) \setminus E(+)$$

Seja $+^* + e_{j+1} - f$
é árvore

$$p(f) > p(e_{j+1})$$



$$p(+^*) \leq p(+)$$

$$p(+^*) < p(+^*) \rightarrow \text{não ocorre}$$

$$p(+^*) = p(+^*) \rightarrow \text{não ocorre}$$

Graf 03/30/19

- Problema A.B.Min. - (Prim, 1957)

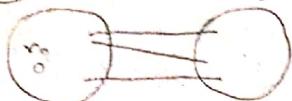
$\alpha \rightarrow$ aresta de menor peso primâneo

$P \rightarrow$ manter conexo

Entre as que ligam
a parte conexa a os
outros vértices do
grafo

$T^* (S, A')$ é conexo a cada iteração

$S \subset r \quad V \setminus S$



Para resolver a escolha de aresta de menor peso por iteração e manter a estrutura necessária para fazer isto eficientemente usamos

(2) - fila de prioridades (de mínimo) com prioridade $q(v)$ = peso da aresta de menor peso que liga $v \in V \setminus S$ a algum vértice de S

No inicio

$$q(r) \leftarrow 0$$

$$q(v) \leftarrow +\infty \quad \forall v \in V \setminus \{r\}$$

* vértice com $q(v)$ mínimo

$\forall u \in \text{Adj}(v) \cap S$

$$q(u) = \min \{ q(u), p(uv) \}$$

- Exercício - Escrever o algoritmo de Prim

Complexidade - $\Theta(m \log n)$

Problema da Seleção de Atividade

Dado: lista de n atividades, cada uma com seu tempo de início e término.

Objetivo: Encontrar o maior conjunto de atividades mutuamente compatíveis.

$a_1 \rightarrow$ escolher os atividades de menor duração primânea

$P \rightarrow$ manter o conjunto das atividades escolhidas como compatível

$a_2 \rightarrow$ Escolher os atividades que terminam primâneamente. Esse funciona

$L \rightarrow$ Provar que está correto

Algoritmo: Seleção de Atividades. Gúleser

Entrada: n pares (s_i, t_i) com $s_i < t_i \quad 1 \leq i \leq n$

Saída: $A \rightarrow$ conjunto de atividades compatíveis máxima e $K = |A|$

1. Ordenar os pares t_i , $t_1 \leq t_2 \dots \leq t_n \rightarrow$ não decrescente

2. $A \leftarrow \{1\}$ (poderia começar aqui com t_1) pois sempre é escolhida)

3. $K \leftarrow 1$

4. $j \leftarrow 1$ (última atividade escolhida)

5. Para $i \leftarrow 2$ até n faça

6. Se $s_i > t_j$ então $A \leftarrow A \cup \{i\}$ // é compatível

7. $K \leftarrow K+1$

8. $j \leftarrow i$

Complexidade

• tempo

ordenação $\Theta(n \log n)$

L5. $\Theta(n)$

L6-8 $\Theta(1)$

L5-8 $\Theta(n)$

//
total $\Theta(n \log n)$

Graf 08/10/19

Problema dos Caminhos Mínimos

Dado $G = (V, E)$ direcionado ou não com $p: E(G) \rightarrow \mathbb{Q}$

$\rightarrow O(x)?$

(A) Dados $u, v \in V(G)$
Tamanho do menor
caminho de u a v
(e tal caminho)

Saída $O(1)$

↓
Algoritmo
em aberto

→ $O(n \cdot x)$
(B) Dado $u \in V(G)$
Tamanho do menor
caminho de u a v
Para todo $v \in V(G)$
(e Tais caminhos)

Saída $O(n)$

↓
Algoritmo
de Dijkstra

$\rightarrow O(n^2 \times)$

(c)

Tamanho dos menores
caminhos de u a v para
Todo u e para Todo
 $v \in V(G)$
(e Tais caminhos)

Saída $O(n^2)$

↓
Floyd (Warshall)

* Distância → Tamanho do menor caminho

• Idéia de Dijkstra (mesma idéia de Prim)

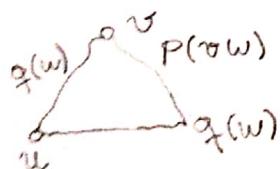
Fila de prioridades

$q_v(v) = \text{distância de } u \text{ a } v$

$v \leftarrow \min \mathbb{Q}$

Para todo $w \in \text{Adj}(v)$ com $w \in \mathbb{Q}$

$$q_v(w) \leftarrow \min \{q_v(w), q_v(v) + p(vw)\}$$



Complexidade $O(m \log n)$

Conectar → método guloso de A.G. Min.

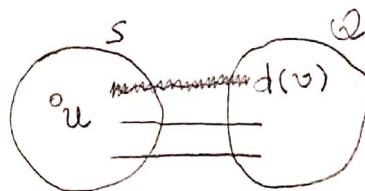
Dijkstra para All-to-All é $O(nm \log n)$

com m grande fixa maior que $O(n^3)$

No inicio

$$q_u(u) = 0$$

$$q_v(v) = +\infty \quad \forall v \in V(G) \setminus u$$



Na atualização quando $v \in \mathbb{Q}$
elemento mínimo da \mathbb{Q}

Ao invés de pegar somente a
menor, é pago Todo o caminho

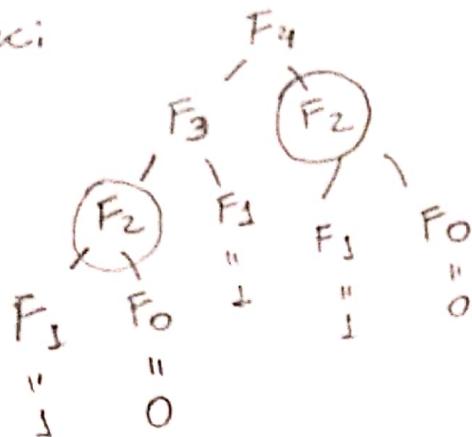
Programação Dinâmica

Tabelas armazenando informações do problema

- Soluções via relocação de recorrência

Exemplo: Fibonacci

$$F_n = F_{n-1} + F_{n-2}$$



Guardar valores já calculados para computar valores somente uma vez na relocação de recorrência

0	1	2	3	4	5
0	1	1	2	3	5

→ Bottom - UP

- Salvar na Tabela essas meninas para maiores
- Complexidade do algoritmo encontrada vai ser:

tamanhos
da Tabela

x exemplo:
de encontrar
um valor da Tabela

Graf 10/10/19

Tanto PD como MG só podem ser aplicados a problemas que satisfaçõem a propriedade da subestrutura o Time

Problema Caminho Mínimo

$$P = P^I \cdot P^{II}$$

onde

P^I é a parte de P de a até c
 P^{II} é a parte de P de c até b

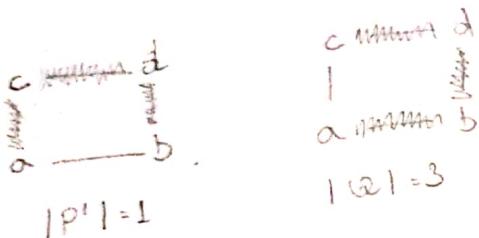
P tem tamanho $x+y$ e é mínimo



(2) da a até c com tamanho menor que x ?

Nesse caso $|P^I| < x+y$
Supõe por absurdo que sim para provar que não.

Problema Caminho Máximo



(2) P^I não é um caminho

Problema das Caminhos Mínimos Todos para Todos

Adjac - Floyd

①

②

P_{ij}^k = Tamanho do menor caminho de vértice i ao vértice j , permutando sempre os vértices em $\{1, 2, 3, \dots, k\}$
entre vértices intermediários no caminho

Queremos $P_{ij}^n + i, j$

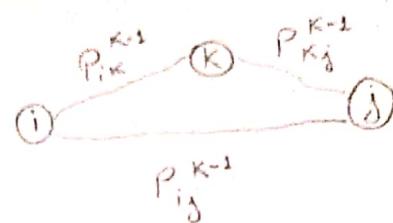
* essa base (nº inicio)

$$P_{ij}^0 = \begin{cases} +\infty, & ij \notin E(G) \text{ ou } \\ 0, & ij \in E(G) \end{cases}$$

Piso da
aresta ij

Regras de Recorrência

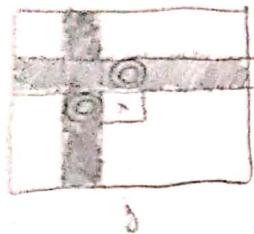
$$P_{ij}^K = \min \{ P_{ij}^{K-1}, P_{ik}^{K-1} + P_{kj}^{K-1} \}$$



Análise da corretude

$$K=0 \quad (n \times n)$$

$$K=1 \quad (n \times n)$$



Tamanho da tabela é $O(n^3)$ e o cálculo de cada elemento é $O(1)$, logo o algoritmo é $O(n^3)$

Algoritmo: Floyd

Entrada: Grafo $G = (V, E)$ e $\rho: E(G) \rightarrow \mathbb{Q}^+$

Saída: matriz P^n quadrada de ordem n , onde P_{ij}^n tem o valor do caminho mínimo de i a j

1. Construir P^n

2. Para $K \leftarrow 1$ até n

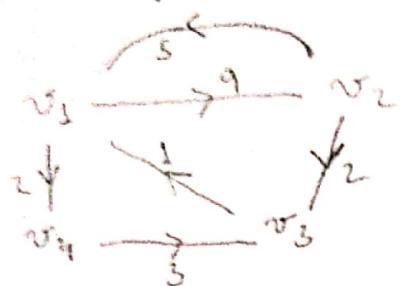
3. Para $i \leftarrow 1$ até n

4. Para $j \leftarrow 1$ até n

5. $P_{ij}^K \leftarrow \min \{ P_{ij}^{K-1}, P_{ik}^{K-1} + P_{kj}^{K-1} \}$

Graf JS/30/19

* Complexidade de Espaço - Algoritmo de Floyd



$$P_{ij}^K = \min \{ P_{ij}^{K-1}, P_{ik}^{K-1} + P_{kj}^{K-1} \}$$

P^0	1	2	3	4
1	0	9	$+\infty$	2
2	5	0	2	$+\infty$
3	1	$+\infty$	0	$+\infty$
4	$+\infty$	$+\infty$	3	0

P^1	1	2	3	4
1	0	9	$+\infty$	2
2	5	0	2	4
3	1	10	0	3
4	$+\infty$	$+\infty$	3	0

P^2	1	2	3	4
1	0	9	11	2
2	5	0	2	4
3	1	10	0	3
4	$+\infty$	$+\infty$	3	0

P^3	1	2	3	4
1	0	9	11	2
2	3	0	2	5
3	1	10	0	3
4	4	13	3	0

P^4	1	2	3	4
1	0	9	5	2
2	3	0	2	5
3	1	10	0	3
4	4	13	3	0

π^3	1	2	3	4
1	1	1	2	1
2	3	2	2	1
3	1	3	1	1
4	1	4	4	1

- Utilizando apenas uma matriz temos $O(n^2)$
- Guardando as matrizes anteriores temos $O(n^3)$

- Guardar informações para obter caminho

$v_i - \pi_{ij} - v_j$

π_{ij}^k = vértice imediatamente anterior a v_i no menor caminho
corrente de v_i a v_j

→ No inicio

$$\pi_{ij}^0 = \begin{cases} i & \text{se } v_i v_j \in E(G) \\ j & \text{se } i=j \\ \text{null} & \text{se } v_i v_j \notin E(G), i \neq j \end{cases}$$

→ Se $p_{ij}^{k-1} > p_{ik}^{k-1} + p_{kj}^{k-1}$:

$$p_{ij}^{k-1} \leftarrow p_{ik}^{k-1} + p_{kj}^{k-1}$$

$$\pi_{ij}^{k-1} \leftarrow \pi_{kj}^{k-1}$$

* Problema da Maior Subsequência Comum

$x \rightarrow$ Sequência de Tamanho m

$y \rightarrow$ Sequência de Tamanho n no mesmo alfabeto

$$x = x_1 x_2 \dots x_m$$

$$y = y_1 y_2 \dots y_n$$

Queremos z a maior subsequência em comum entre x e y
e seu tamanho

Exemplo:

$$\begin{array}{ccccccccc} x & = & a & a & b & c & b & a & c & d \\ & & \diagdown & & \diagup & & \diagdown & & \mid \\ y & = & a & d & b & c & a & & \end{array}$$

$$z = a b c a$$

Algoritmo ingênuo $\rightarrow \sum_{k=1}^n \binom{n}{k} = O(2^n)$

Graf 1440/19

* M.S.C. (P.D.)

Dados $x_0, x_1, x_2, \dots, x_m$
 $y_0, y_1, y_2, \dots, y_n$

Queremos $z_0, z_1, z_2, \dots, z_k$,
a subsequência comum de maior tamanho

$M_{ij} = \text{tamanho da maior subsequência entre } x_i \text{ e } y_j$

$$M_{10} = M_{0j} = 0$$

$$M_{ii} = \begin{cases} 1, & \text{se } x_i = y_i \\ 0, & \text{se } x_i \neq y_i \end{cases}$$

- Determinar M_{mn} para completar a matriz, utilizando recorrência

M	0 1 2 ... n
0	8
1	
m	9

Varrer linhas até o elemento a ser calculado

- Não foi provado em todos os casos estou errado

$$M_{ij} = \begin{cases} M_{i-1, j-1} + 1 & \text{se } x_i = y_j \\ \max\{M_{i-1, j}, M_{i, j-1}\} & \text{se } x_i \neq y_j \end{cases}$$

Algoritmo: M.S.C. - P.D.

Entrada: Duas sequências x e y

Saída: $K \in \mathbb{N}$ tal que $K = |z|$, onde z é a M.S.C. de x e y

1. Para $i \leftarrow 1$ até m , $M_{i0} \leftarrow 0$ // Zerando 1ª linha e coluna
2. Para $j \leftarrow 1$ até n , $M_{0j} \leftarrow 0$
3. Para $i \leftarrow 1$ até n
4. Para $j \leftarrow 1$ até m
5. Se $x_i = y_j$, $M_{ij} \leftarrow M_{i-1, j-1} + 1$
6. Senão se $M_{i-1, j} > M_{ij-1}$, $M_{ij} \leftarrow M_{i-1, j}$
7. Senão $M_{ij} \leftarrow M_{ij-1}$

Exemplo

	0	1	2	3	4	5	6	7	8
x\y	s	o	l	v	e	s	t	e	
0	0	0	0	0	0	0	0	0	0
1	s	0	1	1	1	1	1	1	1
2	o	0	1	2	2	2	2	2	2
3	l	0	1	2	2	2	2	2	2
4	v	0	1	2	3	3	3	3	3
5	e	0	1	2	3	4	4	4	4
6	s	0	1	2	3	4	4	5	5
7	t	0	1	2	3	4	4	5	6

K=6
M.S.C = Sovete

Complexidade

Espeso

$$\mathcal{O}(mn)$$

Entrada

$$\mathcal{O}(n+m)$$

Tempo

$$\underbrace{\mathcal{O}(mn)}_{\text{tamanho da tabela}} \cdot \underbrace{\mathcal{O}(1)}_{\text{cálculo de cada posição}} = \mathcal{O}(mn)$$

Graf 29/10/19

* Fluxo em redes

Rede:

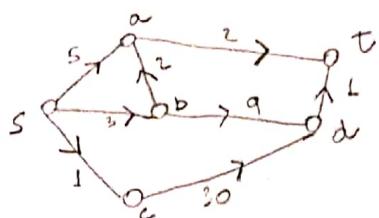
Digraph com $s, t \in V(G)$

t é fonte

s é sumidouro

$\forall v \in V(G) \setminus \{s, t\}$ existe
um caminho de s a v , e um
de v a t

$c: E(G) \rightarrow \mathbb{N}^+$
 \hookrightarrow capacidade de cada arista



$f_1(e) = 0 \quad \forall e \in E(G)$ é fluxo

$$f_1(sa) \cdot f_1(at) = 2$$

$$\text{val}(f_1) = 0$$

$$\text{val}(f_2) = 3$$

$$\text{val}(f_3) = 3$$

$$\text{val}(f_4) = 2$$

* Problema do Fluxo máximo

Dado rede G com capacidades e queremos encontrar o valor de
um fluxo máximo em G e um fluxo desse valor

$e \in E(G)$ é saturada se $f(e) = c(e)$

f é maximal se todo caminho de s a t tem alguma arista
saturada

- Todo fluxo maximal é máximo? não, entra - exemplo
- Todo fluxo máximo é maximal? sim, pois se existe $\exists v_1, v_2, \dots, v_t, t$
um caminho em G sem arestas saturadas podemos determinar $m = \min\{c(e) - f(e) : e \in E(P)\}$ ($m > 0$ pois $c(e) + f(e) \neq c(e)$ $\forall e \in E(G)$) e construir f' tal que

$$f'(e) = \begin{cases} f(e), & \text{se } e \notin E(P) \\ f(e) + m & \text{se } e \in E(P) \end{cases}$$

f' é fluxo?

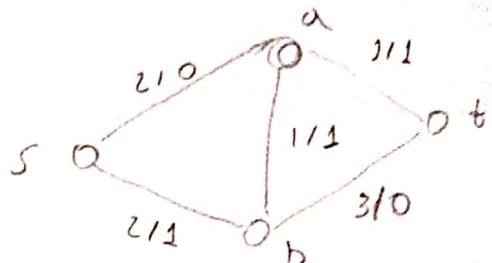
$$0 \leq f'(e) \leq c(e)$$

$$\downarrow \quad \downarrow$$

$$\begin{aligned} f(e) > 0 & \quad \{e\} \in C(e) \\ & \quad \& m = \min \{c(e) - f(e)\} \end{aligned}$$

$$\& \text{val}(f') = \text{val}(f) + m$$

f não é máximo



$$f = 0$$

$$m = 1$$

$$? = (s, b, a, +)$$

* Conte em uma rede

Dado $S \subseteq V(G)$ com $s \in S$ e $t \notin S$, temos:

$$(S, \bar{S})^+ = \{uv \in E(G), u \in S \text{ e } v \in \bar{S}\}$$

$$(S, \bar{S})^- = \{uv \in E(G), v \in S \text{ e } u \in \bar{S}\}$$

$$\text{Corte} = (S, \bar{S}) = (S, \bar{S})^+ \cup (S, \bar{S})^-$$

• Capacidade de um corte:

$$c(S, \bar{S}) = \sum_{e \in (S, \bar{S})^+} c(e)$$

• fluxo em um corte:

$$f(S, \bar{S}) = \sum_{e \in (S, \bar{S})^+} f(e) - \sum_{e \in (S, \bar{S})^-} f(e)$$

• Fato:

$$f(S, \bar{S}) \rightarrow f(S, \bar{S}) \leq c(S, \bar{S})$$

Graf 31/10/19

* Revisão rápida da aula passada

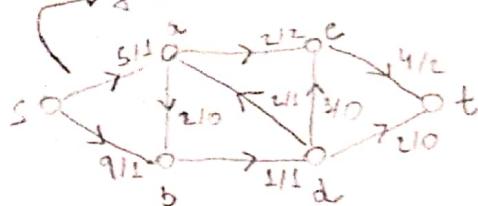
$(S, \bar{S})^+$ → Conjunto dos arcos que vão

$(S, \bar{S})^-$ → Conjunto dos arcos que voltam

$$(S, \bar{S}) = (S, \bar{S})^+ \cup (S, \bar{S})^-$$

$$C: E(G) \rightarrow IN^+$$

→ f é maximal com $val(f) = 2$



$n=6$ vértices

2^6 cortes possíveis

$$\frac{1}{2} n(n-1)$$

$$S_1 = \{S, a\}, z + 2 + a = 18$$

$$(S_1, \bar{S}_1) = \{ac, ab, bc\} \cup \{da\}$$

$$f(S_1, \bar{S}_1) = f(ac) + f(ab) + f(bc) - f(da) = 2$$

$$S_2 = \{S, b\}, z + q = 14$$

$$(S_2, \bar{S}_2) = \{sa, sb\} \cup \{dt\}$$

$$S_3 = \{S, c\}, z + q + r = 18$$

$$(S_3, \bar{S}_3) = \{sa, sb, ct\} \cup \{ac, dc\}$$

• \forall fluxo $f \in (S, \bar{S}) \rightarrow f(S, \bar{S}) \subseteq C(S, \bar{S})$

* Busca do corte de capacidade mínima

Fato → Se f é um fluxo em $G \in (S, \bar{S})$ é um corte em G , então

$$val(f) = f(S, \bar{S})$$

Prova → Por indução em $|S|$

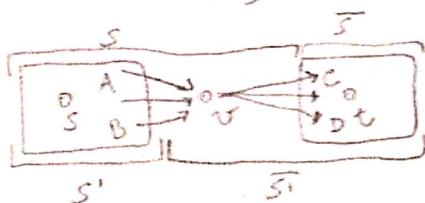
$$\text{Base } \rightarrow |S|=1 \\ S = \{s\}$$

$$f(S, \bar{S}) \stackrel{\text{def}}{=} \sum_{sv \in (S, \bar{S})^+} val(f)$$

→ Seja S , $|S| > 2$

$$v \in S \setminus \{s\}$$

$$S' = S \setminus \{v\}$$

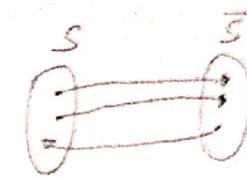


$$f(S, \bar{S}) = f(S', \bar{S}) - [A-B] + [C-D]$$

$$\begin{aligned} &= " + \underbrace{[B+C]}_{= \text{zero pelo teorema conservativo}} - \underbrace{[A+D]}_{\text{cancelado}} \\ &= f(S', \bar{S}') = val(f) \end{aligned}$$

$val(f') = 3 \Rightarrow f'$ é máximo pois existe um corte (S, \bar{S}) em

$$\text{capacidade} = 3 \quad S = \{a, s, b\}$$



* Edsger Ford & Fulkerson ± 1956

Dados $D = (G, s, t, c: E(G) \rightarrow x^+)$, f fluxo em D

(Rede residual) de $D, f \rightarrow D'(f)$ é dada por

$$V(D'(f)) = V(G)$$

$c': V(G) \rightarrow x^+$ → capacidade residual

Se $uv \in E(G)$ e $f(uv) > 0$, então:

contrária $\forall u \in E(D'(f))$ e $c'(vu) = f(vu)$
contrária

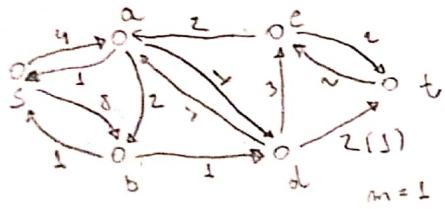
ii) $c(uv) \neq f(uv)$ então:

não saturada

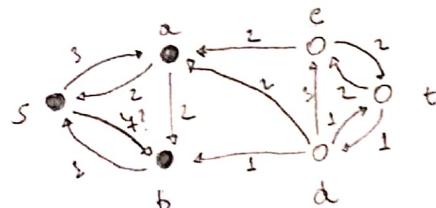
$uv \in E(D'(f))$ e $c(uv) = c(uv) - f(uv)$
dirita

• Aplicando no grafo exemplo

$D'(f_1)$



$D'(f_2)$



$S = \{s, a, b\}$
conjunto dos vértices
visitados definem um corte
(S, T)

→ caminho de s a t em $D'(f) \rightarrow$ Aumentante

- construir mdc
- achar caminho (senão existir parar)

Fato: $c(e) > 0 \wedge e \rightarrow m > 0$

Incrementando o valor do fluxo a cada iteração

Rede Residual

Dado G, s, t $c: E(G) \rightarrow X^+$
 $f: E(G) \rightarrow X^+$

$G(f) \rightarrow G'(f)$ construção da rede



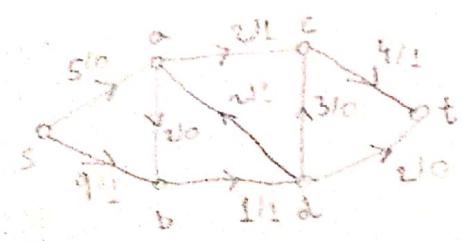
Def. Um caminho aumentante é um caminho de s a t em $G'(f)$

Lema. Se existe um caminho aumentante em $G'(f)$, então f não é máxima

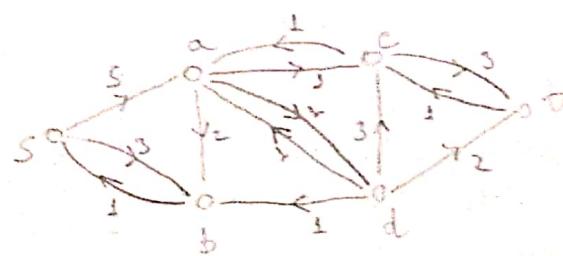
Prova. Seja $P = s v_1 v_2 \dots v_k t$ um caminho em $G'(f)$.

Seja $m = \min \{c'(v_i, v_{i+1}) \mid 0 \leq i \leq k-1\}$ menor valor das capacidades

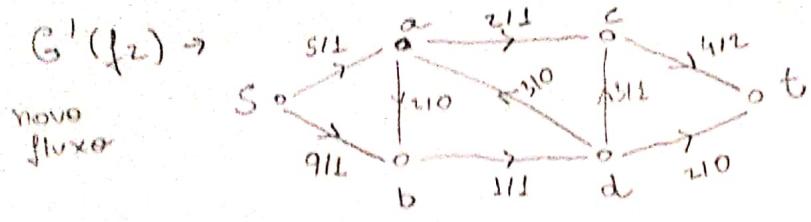
$$f'(uv) = \begin{cases} f(uv), & \text{se } uv \notin P \\ f(uv) + m, & \text{se } uv \in P \text{ e } \leftarrow \text{ direta} \\ f(uv) - m, & \text{se } uv \in P \text{ e } \leftarrow \text{ contrária} \end{cases}$$



$G(f) \rightarrow$



$P = s, a, d, c, t \quad m=1$



$$\text{val}(f_2) = \text{val}(f_1) + m$$

$$f(uv) \leq c(uv) \quad \forall uv \in E(G)$$

E a lei da conservação?

$$\text{N} \neq P \text{ pois } f_2(uv) = \begin{cases} c_{uv} & \text{if } uv \in \text{Adj}^-(v) \\ 0 & \text{if } uv \in \text{Adj}^+(v) \end{cases}$$

inspirar aqui orações da prova?

Iogo f_{j+1} é fluxo em G e, como $\text{val}(f_{j+1}) > \text{val}(f_j)$, f_j não é fluxo máximo

Algoritmo: Ford e Fulkerson (Fluxo máximo)

Entrada: Rede $G, s, t, c: E(G) \rightarrow \mathbb{N}$

Saída: $f: E(G) \rightarrow \mathbb{N}$ e $F = \text{val}(f)$, fluxo de valor máximo em G, c

1. $f(uv) \leftarrow 0 \quad \forall uv \in E(G)$

2. $F \leftarrow 0$

3. Constrói $G'(f)$, $e': F \rightarrow \mathbb{N}$

4. Enquanto existir P um caminho aumentante em $G'(f)$ faça:

$$m = \min \{e'(uv) : uv \in P\}$$

5. Para cada aresta $uv \in P$, se uv é direta $f(uv) = f(uv) + m$
senão $f(uv) = f(uv) - m$

6.

7. constrói $G'(f)$

$$F = F + m$$

8. Seja S o conjunto dos vértices explorados

9. $e(S, \bar{S}) \leftarrow F$ [(S, \bar{S}) é contém mínimo]

Lema da Integralidade

Além disso, Se F^*

é número máximo

Se $X = \mathbb{N}$ e $f = 0$

de iterações, encontrou

Como todos os operações do

$$\text{val}(f) = F^*$$

algoritmo mantém em \mathbb{N} os

mínimos valor de

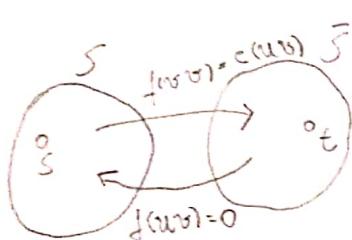
valores de f , $F \in \mathbb{N}$

$$\text{fluxo} = 1$$

Círculo de Ford Fulkerson

Se não existe um caminho aumentante em $G(f)$ então f é o fluxo máximo.

Prova. Seja S o conjunto dos vértices explorados na busca da Linha 4. Como não existe caminhos de s a t , $s \in S$, $t \notin S$. Considerar (s, \bar{s}) em G



$$\text{Val}(f) = f(s, \bar{s}) = \sum_{e \in (s, \bar{s})^+} f(e) - \sum_{e \in (s, \bar{s})^-} f(e)$$

$$= \sum_{e \in (s, \bar{s})^+} c(e) = c(s, \bar{s}) \quad \text{e } f \text{ é máximo}$$

Análise do Algoritmo

L1-3: $O(m)$

complexidade

L4: $O(m)$

$$O(m \cdot \underset{\text{iterações}}{\underset{\text{nº de}}{}}) = O(m \cdot F^*)$$

L5-6: $O(n)$

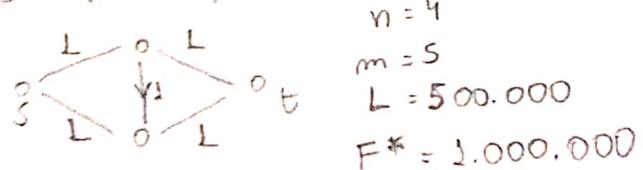
Exponencial

L7: $O(m)$

L8: $O(f)$

L9-10: $O(1)$

Exemplo Exponencial:



$m=1$ vai de 1 em 1
realiza F^* iterações

* Próxima Aula → Ford e Fulkerson Polinomial

Graf 04/33/39

- Teoria de Edmond-Karp 1942

Para encontrar um caminho aumentante em $G^1(f)$, use sempre Busca em Largura (caminhos de s até t com o menor número de arestas)

- Teoria de Dinie 1940

A partir de $G^1(f)$ definir $G^*(f)$ a rede em excessos de D e f , ou seja, encontre um fluxo maximal f^* somando ou subtraindo o valor de f^* de arestas.

A rede em excessos $G^*(f)$ é $G^1(f)$ retirando-se os aristas uv tal que nível(v) \leq nível(u) e todos os vértices que são fonte e não são s , e todos os vértices que são sumidouro e não são t , iterativamente.

Data: 12/11/19

* Complexidade dos Algoritmos

- Edmonds-Karp
- Dinic

	Edmonds-Karp	Dinic
nº de iterações	$O(nm)$	$O(n)$
A cada iteração	Busca menor caminho aumentante Determina m Atualiza fluxo constrói $D^*(f)$ $O(m)$	Encontra f^* , fluxo máximo em $D^*(f)$ Atualiza fluxo $O(m)$ constrói $D^*(f)$ $O(m)$
Total	$O(nm^2)$	$O(n^2m)$

* Teorema Edmond-Karp

Seja P_1, P_2, \dots, P_s os caminhos aumentantes em cada iteração

$$|P_1| \leq |P_2| \leq \dots \leq |P_s|$$

$$s \in nm$$

P_L é caminho de comprimento mínimo em $G^*(f)$

$G^*(f)$

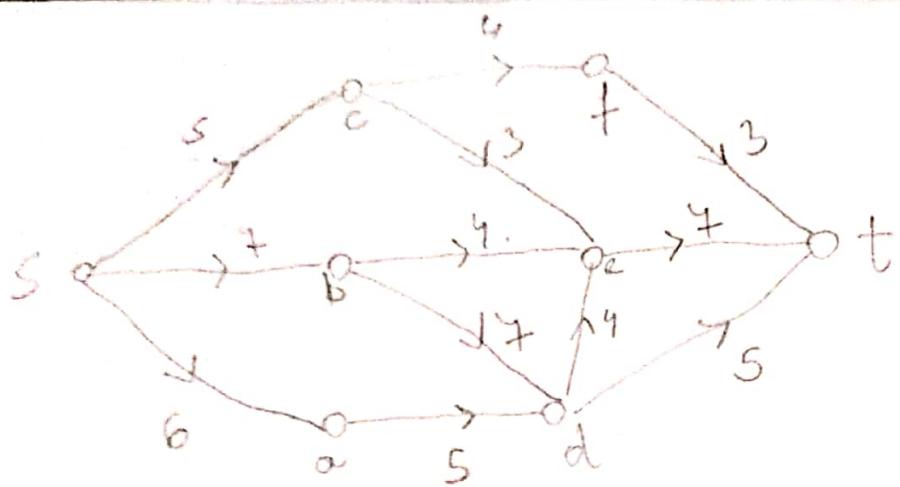
$$S \xrightarrow{v_1} v_2 \xrightarrow{v_3} \dots \xrightarrow{v_j}$$

$$m = \min \{ c^*(v_i, v_{j+1}) \}$$

direta $\xrightarrow{v_0}$

é fechada

possui fluxo



$$f_1 = 0$$

$$\text{Vol}(f_1) = 0$$

$$G^*(f_1) \cong G$$

$G^*(f_1) \rightarrow$ BL em G
Encontrado o fluxo máximo

$$k=3$$

extensão dos caminhos e m's

algoritmo dos índices $O(n^3)$ → não vamos estudar

Emparelhamento máximo em grafos bipartidos

Dado um grafo $G = (X, Y, E)$ um grafo bipartido

construímos uma rede

