

## PROBLEMA TIPO 2:

Determinar todas as pontes  
do grafo dado, versão ite-  
rativa.

COMPONENTES DO GRUPO:

NOME COMPLETO:

VINÍCIUS LIMA MEDEIROS

DRE: 117037920

NOME COMPLETO:

LUCAS RUFINO MARTELLOTTE

DRE: 118046647

NOME COMPLETO:

GABRIEL MARTINS MACHADO CHRISTO

CRC 117217732

NOME COMPLETO:

ALLAN PIMENTEL QUEIROZ

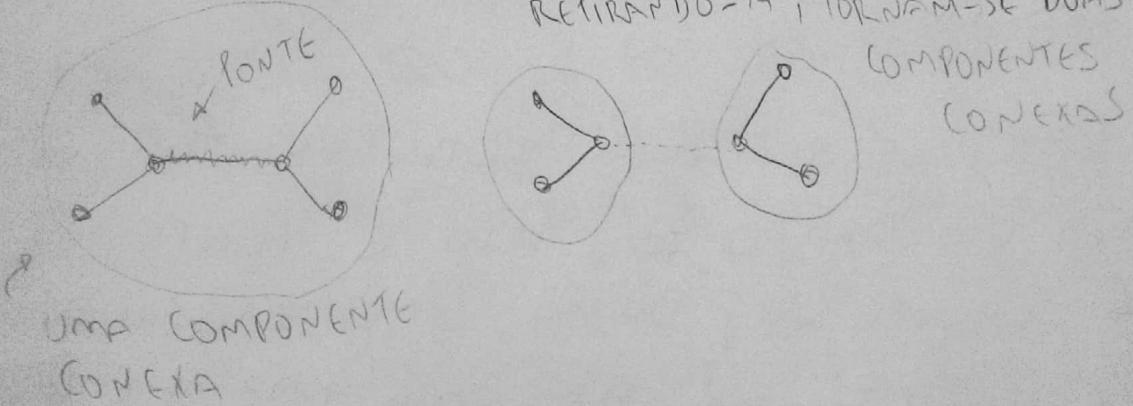
CRC 113028838

## 1) ABORDAGEM TEÓRICA DO PROBLEMA EM QUESTÃO.

O PROBLEMA ABORDADO CONSISTE EM DESENVOLVER UM ALGORITMO PARA ENCONTRAR AS PONTES DE UM DADO GRAFO G ARBITRÁRIO. PRIMEIRO, VAMOS COMEÇAR DEFININDO O QUE É UMA PONTE.

SEJA  $C(x)$  O NÚMERO DE COMPONENTES CONEXAS DE  $G$ , TAL QUE,  $C(G) < C(G - v)$

Uma visão intuitiva de uma ponte é basicamente uma aresta que, quando removida, separa o grafo em partes desconexas retirando-a, tornam-se duas



ENTÃO, A PRIMEIRA PERGUNTA É, COMO PODEMOS IDENTIFICAR UMA PONTE ALGORITMICAMENTE? HÁ MANEIRAS ÓBVIAS, COMO TESTAR PARA TODAS AS ARESTAS A CONECTIVIDADE DE CADA COMPONENTE CONEXA APÓS A REMOÇÃO DELA. NO ENTANTO, UMA SOLUÇÃO ASSIM NÃO SERIA ÓTIMA.

POR ISSO, PARA CONSEGUIR MELHORAR A EFICIÊNCIA DO ALGORITMO PRECISAMOS DE OUTRA IDEIA. O CONCEITO DE LOW-LEVEL

$\text{Low}(v)$  É UMA FUNÇÃO  $V(G) \rightarrow \mathbb{N}$ , TAL QUE  
 $V(G)$  É O CONJUNTO DE VÉRTICES DE UM  
 GRAFO  $G$ . DADA UMA BUSCA EM PRO-  
 FUNDIDADE BP COM RAIZ  $v \in V(G)$ ,  
 $\text{Low}(v)$  EM BP É O MENOR PE( $w$ )  $\forall w$ ,  
 TAL QUE EXISTE UM CAMINHO DE  $v$  PARA  $w$   
 NA ABL (ÁRVORE DE BUSCA EM PROFUNDIDADE)  
 CONSISTINDO APENAS EM ARESTAS DE  
 ÁRVORE, DE MODO QUE: SEJA 'ab' TAL  
 QUE  $\text{Pai}(b) = a$  E NO MÁXIMO UMA  
 FRONDE.

TEOREMA 1: PARA TODA ARESTA  $vw$   
 EM UM GRAFO  $G$ ,  $vw$  É PONTE, SE  
 E SOMENTE SE, DADA UMA BUSCA EM  
 PROFUNDIDADE BP COM RAIZ  $v \in V(G)$ ,  
 $\text{PE}(v) < \text{Low}(w)$ .

COMO O TEOREMA É UMA RELAÇÃO  
 DE EQUIVALÊNCIA, TEMOS QUE PROVAR  
 AS IMPLICAÇÕES NAS DUAS DIRE-  
 COES:

$$A \Leftrightarrow B$$

$$\downarrow \quad \downarrow$$

$$A \Leftarrow B \quad A \Rightarrow B$$

VAMOS COMEÇAR PROVANDO QUE:

SE  $v \in PONTE \Rightarrow PE(v) < LOW(w)$

VAMOS ASSUMIR POR CONTRADIÇÃO QUE EXISTE UM CAMINHO  $C_{vz}$  NA ABP DE  $w$  PARA ALGUM VÉRTICE  $z$  PASSANDO APENAS POR ARESTAS DESCENDENTES E NO MÁXIMO UMA FRONTE TAL QUE  $P(z) \leq PE(v)$ .  
(OBS: OBSERVE QUE SE  $PE(z) = PE(v)$ , OBVIAMENTE  $z = v$ )

NO ENTANTO, COMO  $PE(z) \leq PE(v)$ , ISSO QUER DIZER QUE  $z$  FOI VISITADO ANTES NA ÁRVORE DE BUSCA. E COMO A BUSCA EM PROFUNDIDADE EM UM GRAFO NÃO DIRECIONADO NÃO GERAM ARESTAS QUE LIGAM DUAS SUBÁRVORES DIFERENTES, ISSO SIGNIFICA QUE  $v$  É DESCENDENTE DE  $z$ . OU SEJA, EXISTE UM CAMINHO  $C_{zw}$  NA ABP INDO DE  $z$  PARA  $w$  PASSANDO POR ARESTAS DESCENDENTES.

SE SOMARMOS O CAMINHO  $C_{vz}$  COM  $C_{zw}$ , TEMOS UM CAMINHO NOVO  $C_{vw}$  QUE LIGA OS VÉRTICES  $v$  E  $w$  E NÃO PASSA PELA ARESTA  $vw$ . LOGO, TEMOS DOIS CAMINHOS DISTINTOS LIGANDO ESSES

DOIS VÉRTICES, O QUE CARACTERIZA UM CICLO, E, OBVIAMENTE, POR CAUSA DISSO, QUANDO RETIRAMOS  $vw$ , O GRAFO SERÁ CONEXO.

CONTRADIÇÃO! LOGO, A NOSSA SUPosiÇÃO INICIAL DE QUE EXISTE UMA PONTE  $vw$  TAL QUE  $PE(v) \geq Low(w)$  ESTÁ ERRADA. ASSIM, COMPLETAMOS A PROVA DE IDA.

AGORA, TEMOS QUE PROVAR A VOLTA:  
OU SEJA, SE  $PE(v) < Low(w)$  E EXISTE ARESTA  $vw$ , ELA É PONTE:

COMO  $PE(v) < Low(w)$ , PELO MESMO RACIOCÍNIO DA PROVA ANTERIOR, PODEMOS CONCLUIR QUE O VÉRTICE  $z$  DE MENOR  $PE$  TAL QUE EXISTE CAMINHO DE  $w$  PARA  $z$  NA ABP PASSANDO APENAS POR ARESTAS DESCENDENTES E NO MÁXIMO UMA FRONDE, É DESCENDENTE DE  $v$ . COMO EXISTE UM CAMINHO PASSANDO APENAS POR ARESTAS DESCENDENTES DE  $w$  PARA QUALQUER DESCENDENTE DE  $w$ , SEGUE TAMBÉM PARA TODOS OS SEUS DESCENDENTES O MESMO RAÇIOCÍNIO USADO EM  $w$ .

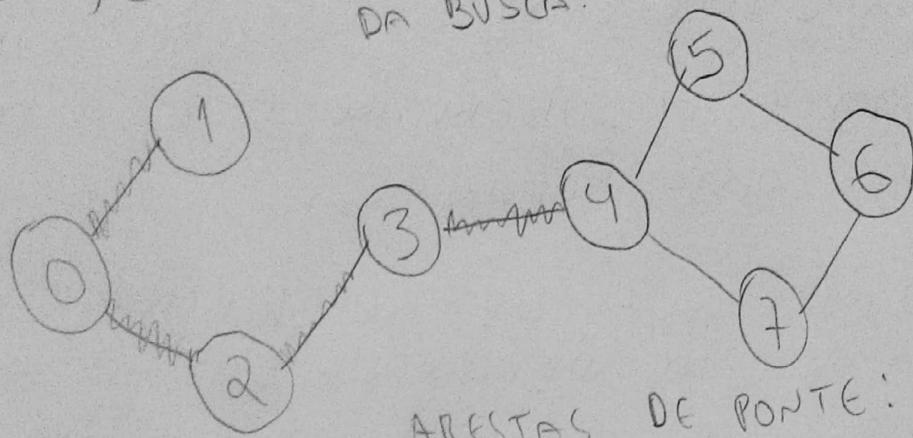
COMO  $w$  É O PRIMEIRO DESCENDENTE DE  $v$ , ISSO QUER DIZER QUE TODOS OS VÉRTICES NA SUBÁRVORE DE RAÍZ  $w$  SÓ PODEM CHEGAR A NO MÁXIMO UMA FRONDE. OU SEJA, NÃO PODE EXISTIR NENHUMA ARESTA LIGANDO ESTA SUBÁRVORE A UM VÉRTICE FORA DESTA SUBÁRVORE.

CONCLUINDO QUE, SE REMOVERMOS A ARESTA  $vw$ , A SUBÁRVORE DE RAÍZ  $w$  NÃO POSSUI MAIS NENHUMA ARESTA CONECTANDO COM O RESTO DA ABP. LOGO, ESTA ARESTA  $vw$ , AO SER REMOVIDA, AUMENTOU O NÚMERO DE COMPONENTES CONEXAS DO GRAFO.

COM ESTE FATO PROVADO, APENAS PRECISAMOS DE UM ALGORITMO LINEAR PARA COMPUTAR O  $LOW(w)$  PARA TODO VÉRTICE  $w \in G$ . E ISSO FAREMOS NA SEÇÃO "O ALGORITMO", ASSIM COMO PROVARREMOS O PORQUÊ DE ELE SEMPRE COMPUTAR CORRETAMENTE ESSA FUNÇÃO.

## 2) RESOLUÇÃO DO PROBLEMA EM 2 GRAFOS

GRAFO 1: SENDO O VÉRTICE 0 A RAIZ  
DA BUSCA.



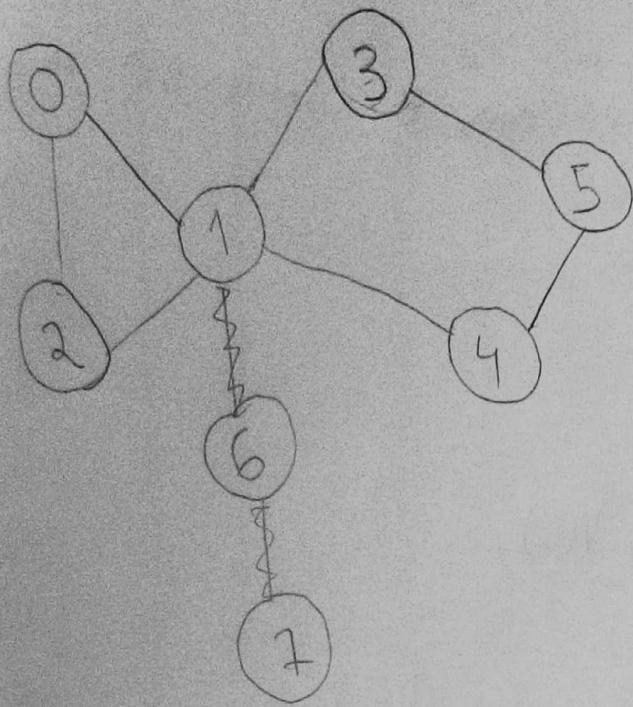
ARESTAS DE PONTE:

v	PE(v)	LOW(v)	
0	1	1	
1	2	2	
2	3	3	$2,3 \rightarrow PE(2) < LOW(3)$
3	4	4	$3 < 4$
4	5	5	
5	6	5	$0,1 \rightarrow PE(0) < LOW(1)$
6	7	5	$1 < 2$
7	8	5	

$0,1 \rightarrow PE(0) < LOW(1)$

$1 < 3$

## GRAFO 2



$$v \quad PE(v) = LOW(v)$$

v	PE(v)	LOW(v)
0	1	1
1	2	1
2	3	1
3	4	2
4	5	2
5	6	2
6	7	7
7	8	8

ARESTAS DE PONTE:

$$6,7 \rightarrow PE(6) < LOW(7)$$

$$7 < 8$$

$$1,6 \rightarrow PE(1) < LOW(6)$$

$$1 < 7$$

### 3) O ALGORITMO

NOME: DET PONTES

ENTRADA: GRAFO G

Saída: LISTA DE ARESTAS P CONTENDO AS PONTES DO GRAFO

1. COUNT  $\leftarrow 0$
2. M  $\leftarrow \{\}$
3. P  $\leftarrow \{\}$
4. R  $\leftarrow \{\}$
5. PARA TODO  $v \in V(G)$ :  
6.     | C( $v$ )  $\leftarrow 0$
7.     | R  $\leftarrow R \cup v$
8. ENQUANTO  $R \neq \emptyset$   
9.     | PARA UM  $n \in R$   
10.     | SE C( $n$ )  $\neq 0$   
11.         | R  $\leftarrow R \setminus \{n\}$   
12.         | SENÃO  
13.         | R  $\leftarrow R \setminus \{n\}$   
14.         | C( $n$ )  $\leftarrow 1$   
15.         | PAI( $n$ )  $\leftarrow n$
16.     M  $\leftarrow M \cup n$
17.     PE( $n$ )  $\leftarrow$  COUNT
18.     LOW( $n$ )  $\leftarrow$  PE( $n$ )
19.     COUNT  $\leftarrow$  COUNT + 1

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.

36.

37.

38.

39.

ENQUANTO  $M \neq \emptyset$

PARA  $v \in M$ :

SE EXISTE  $w \in ADJ(v)$ , TAL QUE  
 $w$  NÃO FOI VISITADA:

SE  $C(w) = 0$ :

$C(w) \leftarrow 1$

$PAI(w) \leftarrow v$

$M \leftarrow M \cup w$

$PE(w) \leftarrow COUNT$

$LOW(w) \leftarrow PE(w)$

$COUNT \leftarrow COUNT + 1$

$VISITE1(vw)$

SENÃO:

$VISITE2(vw)$

SE  $C(w) = 1$  E

$PE(w) < PE(v)$  E

$PAI(w) \neq w$ :

SE  $LOW(w) <$

$LOW(v)$ :

$LOW(v) \leftarrow$

$LOW(w)$

SENÃO

$(vw) \leftarrow 2$

$M \leftarrow M \setminus \{v\}$

40.		PARA TODO $w \in \text{Adj}(v)$ :
41.		SE $\text{PAI}(w) = v$ :
42.		SE $\text{PE}(v) < \text{LOW}(w)$
43.		$\text{IP} \leftarrow \text{P} \cup v w$
44.		SE $\text{LOW}(w) < \text{LOW}(v)$ :
45.		$\text{LOW}(v) \leftarrow \text{LOW}(w)$

VISITE 1 E VISITE 2:

LISTAR  $v w$  NA SAÍDA E AVANÇAR

O PONTEIRO DE  $\text{ADJ}(v)$

- CORRETUDE:

COMO JÁ PROVAMOS O "TEOREMA 1" NA PRIMEIRA PARTE DO TRABALHO, TUDO QUE PRECISAMOS FAZER NA CORRETUDE É PROVAR QUE O ALGORITMO CALCULA CORRETAMENTE O VALOR DE  $\text{LOW}(v)$  PARA TODO  $v \in V(G)$ .

O ALGORITMO É UMA MODIFICAÇÃO DA BUSCA EM PROFUNDIDADE, IÉ, TRIVIAL O FATO DE QUE TODOS OS VÉRTICES E ARESTAS SÃO VISITADOS.

VAMOS SUPOR POR CONTRADIÇÃO QUE O ALGORITMO COMPUTA O LOW( $v$ ) ERRADO PARA ALGUM  $v \in V(G)$ . HÁ QUATRO OCASÕES ONDE O ALGORITMO MODIFICA O LOW DE UM VÉRTICE:

PRIMEIRO CASO: LINHA 18. NESTE CASO, É ATRIBUIDO AO LOW DO VÉRTICE O VALOR DE PE DO VÉRTICE. MAS COMO O VÉRTICE NA LINHA 18 SERÁ A NOVA RAIZ DA BUSCA, ELA TERÁ O MENOR PE DE TODOS OS VÉRTICES NA SUA COMPONENTE CONEXA, LOGO O SEU LOW SÓ PODE SER O PRÓPRIO PE.

SEGUNDO CASO: LINHA 29. NESTE CASO, ATRIBUÍMOS AO LOW DE UM VÉRTICE  $v$  O SEU PE. NOTE QUE, SE NENHUM VÉRTICE DESCENDENTE DE  $v$  VISITAR UM VÉRTICE  $w$  DE PE MENOR QUE  $v$  & SE  $v$  NÃO TIVER FRONDES SAINDO DELE, ENTÃO O SEU LOW JÁ ESTÁ CERTO. LOGO, TEMOS DOIS CASOS PARA NOS PREOCUPAR, QUE SERÃO TRATADOS NOS CASOS SEGUINTES.

TERCEIRO CASO: NA LINHA 36, NESTE CASO, CUIDAMOS DO PRIMEIRO MOMEMTO CRÍTICO DO SEGUNDO CASO, OU SEJA, QUANDO UM VÉRTICE  $v$  POSSUI UMA FRONDE. SE  $v$  POSSUI FRONDE, ELE ESTÁ VISITANDO UM VÉRTICE  $w$  QUE JÁ FOI VISITADO. LOGO, SEU LOW SERÁ IGUAL A MIN (LOW( $v$ )) LOW( $w$ )

QUARTO CASO: LINHA 45. COMO HAVIA MENCIONADO, O ÚLTIMO PROBLEMA QUE TEMOS QUE LIDAR É SE UM DESCENDENTE DE UM VÉRTICE  $v$  VISITA UM VÉRTICE  $w$  DE PE MENOR QUE  $v$ .

ASSIM, NA LINHA 45, NÓS ATUALIZAMOS O LOW DE UM VÉRTICE TODA VEZ QUE ELE É EXPLORADO, POIS, PELO FATO DE SER UMA BUSCA EM PROFUNDIDADE, UM VÉRTICE  $x$  É EXPLORADO SOMENTE APÓS TODOS OS SEUS FILHOS FOREM EXPLORADOS. NO ALGORITMO, COMPAREMOS O LOW( $v$ ) COM O LOW DE TODOS OS SEUS FILHOS, DE MODO QUE PEGAMOS O MÍNIMO DE

TODOS E ATRIBUIMOS A  $v$ . A PARTIR DAÍ, UM SIMPLES ARGUMENTO DE INDUÇÃO É SUFICIENTE PARA PROVAR QUE O LOW DO VÉRTICE  $v$  É CORRETO, POIS OS VÉRTICES QUE SÃO FOLHAS DA ABF JÁ POSSUEM O LOW CORRETO PELO TERCEIRO CASO APENAS, VISTO QUE ESTES VÉRTICES NÃO POSSUEM FILHOS. A PARTIR DISTO OS SEUS PAIS POSSUIRAM OS LOW'S CORRETOS, POIS ESTÃO COMPARANDO SEUS LOW'S COM OS LOW'S DE FOLHAS QUE SÓ ESTÃO COM O VALOR CORRETO.

ASSIM TODOS OS CASOS SÃO COBERTOS E ESTÁ PROVADO QUE O  $\text{Low}(v)$  É COMPUTADO CORRETAMENTE PARA  $v \in G$ .

A PÓS ISSO, SÓ ENUNCIAR O TEOREMA A, JÁ PROVADO ANTERIORMENTE.

## ANÁLISE DA COMPLEXIDADE

LINHA 1 a 4 : INICIALIZANDO VARIÁVEIS  $O(n)$ .

LINHA 5 a 7 : LAÇO PASSANDO POR  $V(G)$

LINHA 8 : TEMOS UM LAÇO QUE DURA ENQUANTO  $R$  NÃO ESTIVER VAZIO.

$R$  CONSEGUE COM TODOS OS VÉRTICES DENTRO DELE E HÁ UM IF DENTRO DO LAÇO , SEM IMPORTAR O VALOR DO IF ,  $R$  SEMPRE PERDE UM ELEMENTO . OU SEJA , DENTRO DO LAÇO TODOS OS VÉRTICES SÃO PERCORRIDOS APENAS UMA VEZ  $O(n)$

LINHA 20 : AQUI TEMOS UM LAÇO ATÉ  $M$  FICAR VAZIA .  $M$  ARMAZENA TODOS OS VÉRTICES VISITADOS DO GRAFO . COMO ESSE ALGORITMO É UMA ADAPTAÇÃO DA BUSCA EM PROFUNDIDADE , É TRIVIAL O FATO DE QUE TODOS OS VÉRTICES SÃO VISITADOS , LOGO A COMPLEXIDADE DESSE LAÇO É  $O(V)$

LINHA 22 : DENTRO DO LAÇO QUE ENGLOBA TODOS OS VÉRTICES , ITERAMOS POR TODAS AS ARESTAS PERTENCENTES A LISTA DE ADJACÊNCIAS DE  $v$  . OU SEJA ,  $O(|E|) = O(|V|)$  , POIS

$\partial E$  É A SOMA DE TODAS AS ARESTAS DE TODOS OS VÉRTICES

LINHA 40: QUANDO O VÉRTICE SE TORNA EXPLORADO, ITERAMOS MAIS UMA VEZ EM TODOS OS VÉRTICES PERTENCENTES A SUA LISTA DE ADJACÊNCIAS, LOGO  $O(\partial E)$

ASSIM, A COMPLEXIDADE TOTAL É:

$$O(n) + O(v) + O(v) + O(\partial E) + O(\partial E) = O(v+E)$$

#### 4) REFERÊNCIA(S) BIBLIOGRÁFICAS

GRAFOS E ALGORITMOS COMPUTACIONAIS - 2ª EDIÇÃO

JAYME LUIZ SZWARCFLICKER