Algoritmos para conexidade em grafos dinâmicos

Arthur Henrique Dias Rodrigues
sob orientação de
Cristina Gomes Fernandes

Instituto de Matemática e Estatística
USP

22 de novembro de 2024

Sumário

1 Conexidade em florestas dinâmicas
- Definição
- Euler Tour Trees

2 Conexidade em grafos dinâmicos
- Definição

3 Floresta maximal de peso mínimo em grafos planos ponderados dinâmicos
- Definição do problema
- Resolvendo MSF com ADPs

4 O limitante inferior de $\Omega(\lg n)$

5 Bibliografia

Conexidade em florestas dinâmicas

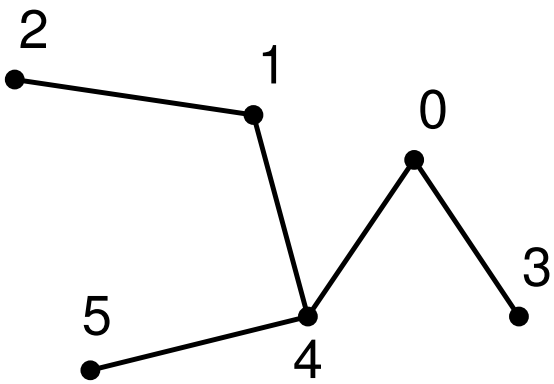
Conexidade em florestas dinâmicas

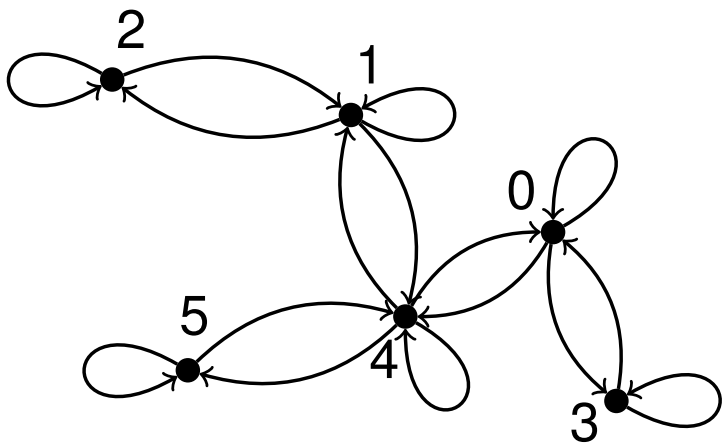
Problema de conexidade em florestas dinâmicas

- NOVAFD(n): retorna uma floresta dinâmica com n vértices isolados;
- LIGUEFD(F,u,v): adiciona uv a F;
- REMOVAFD(F,u,v): remove uv de F; e
- CONECTADOFD(F,u,v): retorna verdadeiro se u e v estão na mesma componente conexa de F e falso, caso contrário.

Para solucionar esse problema, vamos apresentar a estrutura de dados proposta por Henzinger e King.

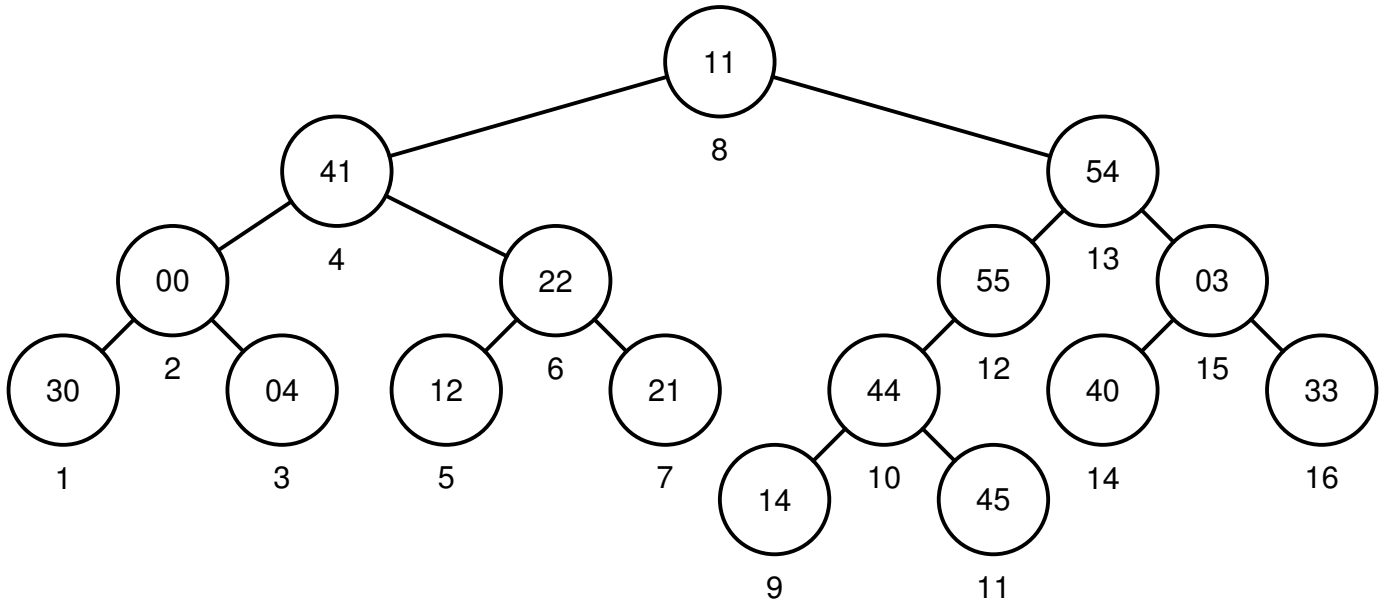
Sequência Euleriana





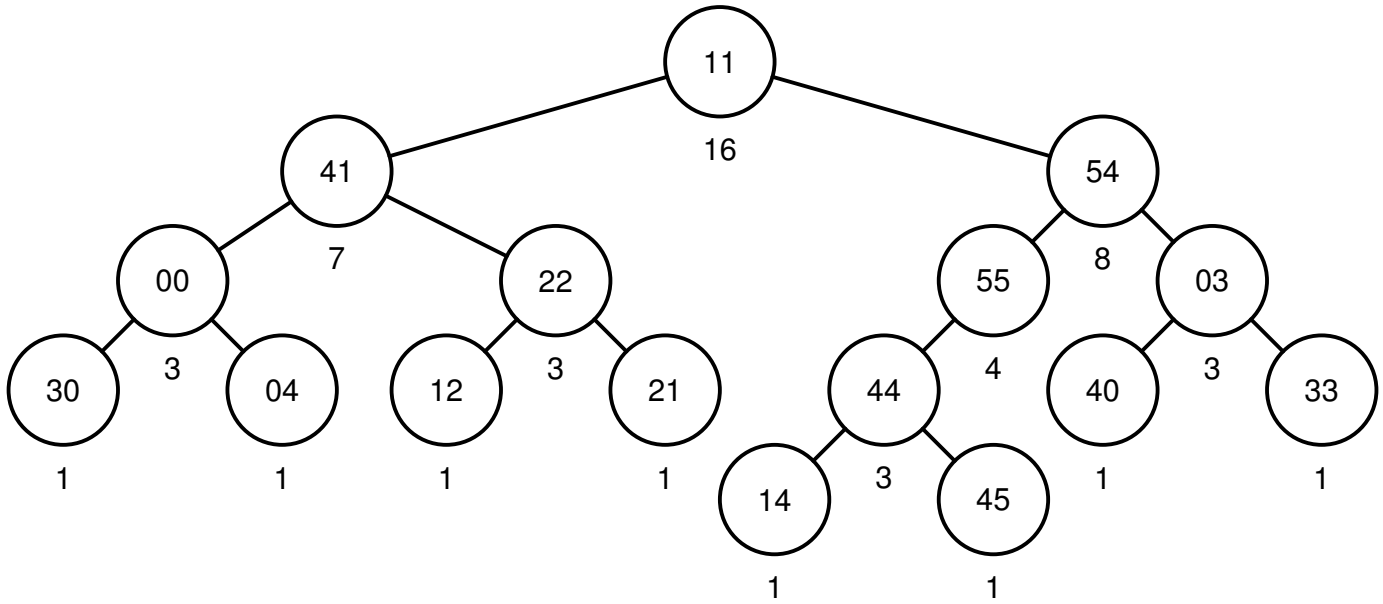
30 00 04 41 12 22 21 11 14 44 45 55 54 40 03 33

Euler Tour Trees



arco	30	00	04	41	12	22	21	11	14	44	45	55	54	40	03	33
índice	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Chaves implícitas



arco	30	00	04	41	12	22	21	11	14	44	45	55	54	40	03	33
índice	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Biblioteca de Euler Tour Trees

Biblioteca de Euler Tour Trees

- NOVONÓ(u, v): retorna uma ABB com um único nó com valor uv;
- JUNTA(T, R): junta as ABBs T e R concatenando as sequências Eulerianas armazenada nelas e retorna a raiz da árvore resultante.
- CORTA(*nó*): corta a ABB que contém um nó *nó* em três ABBs. A primeira ABB contém todos os nós com chave estritamente menor do que a chave de *nó*, a segunda contém somente *nó* e a última contém todos os nós com chave estritamente maior do que a chave de *nó*. Essa rotina retorna as raízes dessas três ABBs; e
- RAIZ(*nó*): retorna a raiz da ABB que contém *nó*;

NOVONÓ: $O(1)$.
As demais operações : $O(\lg n)$.

Tabela de símbolos

Associa $(u, v) \to uv$.

Biblioteca de tabela de símbolos

- $F \leftarrow$ NOVODICIO(n): cria e retorna um dicionário F para uma floresta dinâmica com n vértices;
- $F[u, v] \leftarrow uv$: insere o nó que contém uv, com chave (u, v) e valor associado uv na tabela F. Se o par (u, v) já estiver presente no dicionário, então seu valor associado é substituído por uv;
- $F[u, v] \leftarrow$ NIL: remove o nó associado a (u, v) e seu valor associado do dicionário F;
- $var \leftarrow F[u, v]$: atribui o valor associado à chave (u, v) à variável var; Caso a chave (u, v) não esteja presente em F, atribui NIL a var.

Consumo esperado O(1) por rotina.

Implementação da interface de floresta dinâmica

Algorithm NOVAFD(n)

1: $F \leftarrow$ NOVODICIO(n)
2: **para** $v \leftarrow 1$ até n **faça**
3: $\quad F[v, v] \leftarrow$ NOVONÓ(v, v)
4: **retorne** F

Algorithm CONECTADOFD(F, u, v)

1: $uu \leftarrow F[u, u]$
2: $vv \leftarrow F[v, v]$
3: **retorne** RAIZ(uu) = RAIZ(vv)

NOVAFD : $O(n)$
CONECTADOFD : $O(\lg n)$

Implementação da interface de floresta dinâmica

Algorithm MOVAINÍCIO(F,u)

1: $uu \leftarrow F[u, u]$
2: $A, uu, B \leftarrow$ CORTA(uu)
3: **retorne** JUNTA(uu, B, A)

Algorithm LIGUEFD(F,u,v)

1: $U \leftarrow$ MOVAINÍCIO(F, u)
2: $V \leftarrow$ MOVAINÍCIO(F, v)
3: $uv \leftarrow$ NOVONÓ(u, v)
4: $vu \leftarrow$ NOVONÓ(v, u)
5: $F[u, v] \leftarrow uv$
6: $F[v, u] \leftarrow vu$
7: JUNTA(U, uv, V, vu)

LIGUEFD : $O(\lg n)$
MOVAINÍCIO : $O(\lg n)$

Implementação da interface de floresta dinâmica

Algorithm REMOVAFD(F, u, v)

1: $uv \leftarrow F[u, v]$
2: $vu \leftarrow F[v, u]$
3: $A, uv, B \leftarrow$ CORTA(uv)
4: JUNTA(B, A)
5: CORTA(vu)
6: $F[u, v] \leftarrow$ NIL
7: $F[v, u] \leftarrow$ NIL

REMOVAFD : $O(\lg n)$

Conexidade em grafos dinâmicos

Conexidade em grafos dinâmicos

Conexidade em grafos dinâmicos

- NOVOGD(n): cria um grafo dinâmico com n vértices isolados;
- LIGUEGD(G, u, v): adiciona a aresta uv ao grafo dinâmico G;
- REMOVAGD(G, u, v): remove a aresta uv de G; e
- CONECTADOGD(G, u, v): retorna verdadeiro se u e v estão na mesma componente conexa de G e falso, caso contrário.

Para solucionar esse problema, vamos apresentar a estrutura de dados proposta por Holm, de Lichtenberg e Thorup.

Ideia inicial

Manteremos

- floresta maximal dinâmica F de G; e
- um grafo $R = G - F$

Lista de adjacências

- NOVOGRAFO(n): devolve a representação por listas de adjacências de um grafo com n vértices isolados.
- LIGUEGLA(G, u, v): adiciona u na lista de adjacências de v em G e vice-versa.
- REMOVAGLA(G, u, v): remove u da lista de adjacências de v em G e vice-versa.

NOVOGRAFO : $O(n)$
LIGUEGLA : $O(1)$
REMOVAGLA : $O(1)$

Ideia inicial

Algorithm CONECTADOGD(G, u, v)

1: **retorne** CONECTADOFD($G.F$, u, v)

Algorithm LIGUEGD(G, u, v)

1: **se** CONECTADOFD($G.F$, u, v) **então**
2: LIGUEGLA($G.R$, u, v)
3: **senão**
4: LIGUEFD($G.F$, u, v)

CONECTADOGD : $O(\lg n)$
LIGUEGD : $O(\lg^2 n)$ amortizado!!

Remoção de arestas

Busca por substituição de uma aresta

- Cada aresta possui um **nível**, que é um inteiro entre 1 e $\lceil \log n \rceil$;
- Arestas serão inseridas no nível $\lceil \log n \rceil$;
- O nível de uma aresta pode diminuir, mas nunca aumentar.

Estrutura

$G_{\leqslant i}$: grafo com arestas de nível $\leqslant i$. Para cada camada i, manteremos:

- $F_{\leqslant i}$: floresta maximal de $G_{\leqslant i}$; e
- R_i: arestas de nível $i \notin F_{\leqslant i}$.

Invariantes

- $F_{\leqslant i} \subseteq F_{\leqslant i+1}$, para cada $1 \leqslant i \leqslant \lceil \log n \rceil - 1$;
- $F_{\leqslant i}$ é uma floresta maximal de $G_{\leqslant i}$; e
- Cada componente de $F_{\leqslant i}$ possui menos do que 2^i arestas.

Implementações

Adaptações

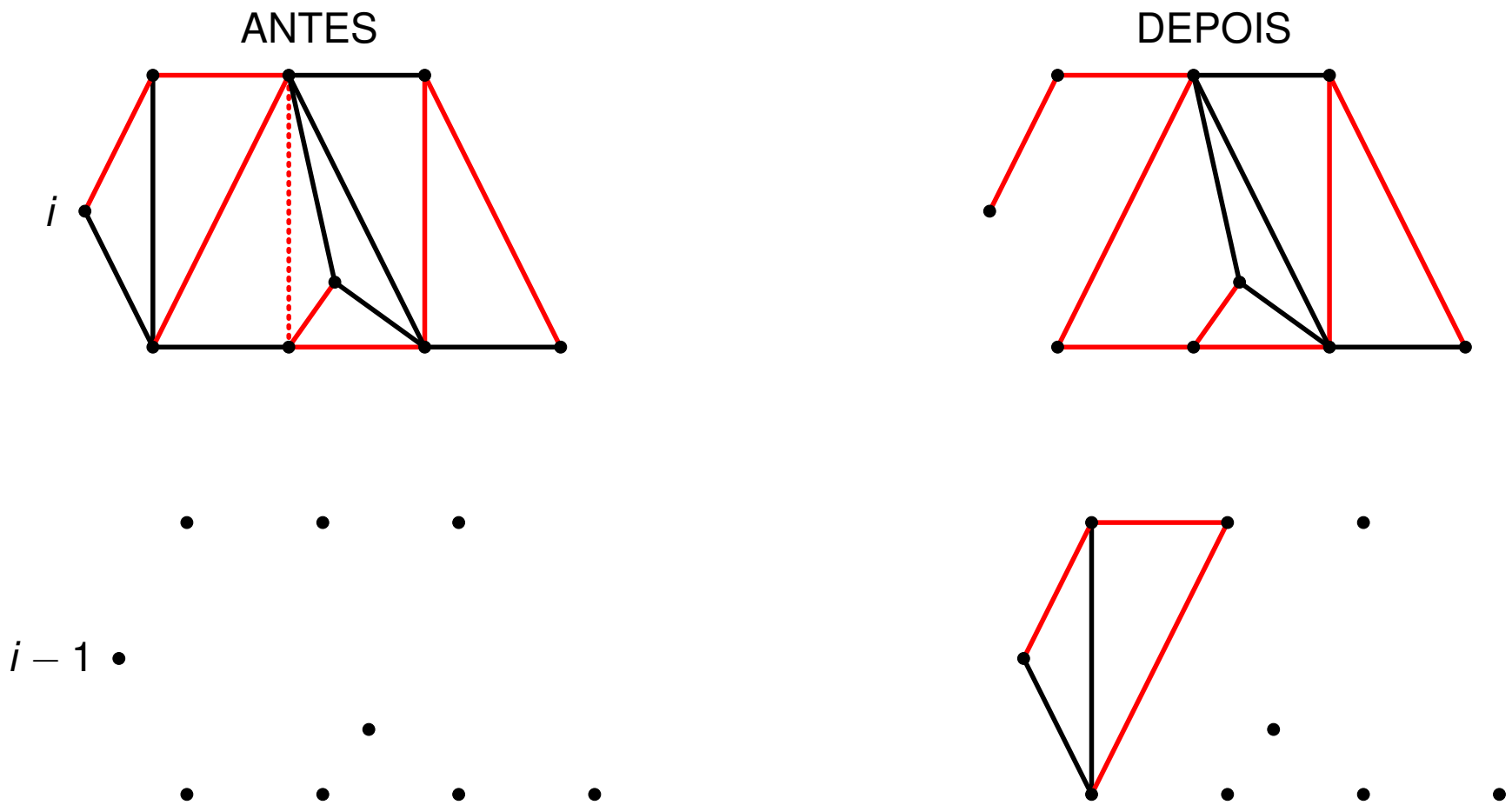
$$G.F \to G.F_{\leqslant \lceil \log n \rceil}$$
$$G.R \to G.R_{\lceil \log n \rceil}$$

Algorithm REMOVAGD(G, u, v)

```
1: i ← NÍVEL[u, v]
2: NÍVEL[u, v] ← NIL
3: se uv ∈ G.F_{⩽⌈lg n⌉} então
4:     para j ← i até ⌈lg n⌉ faça
5:         REMOVAFD(G.F_{⩽j}, u, v)
6:     SUBSTITUAGD(G, u, v, i)
7: senão
8:     REMOVAGLA(G.R_i, u, v)
```

REMOVAGD : $O(\lg^2 n)$ amortizado.

Estrutura de níveis



A rotina SUBSTITUAGD

Algorithm SUBSTITUAGD(G,u,v,niv)

1: **para** $i \leftarrow niv$ até $\lceil \lg n \rceil$ **faça**
2: $T_v \leftarrow$ RAIZ($F_{\leqslant i}[v, v]$)
3: $T_u \leftarrow$ RAIZ($F_{\leqslant i}[u, u]$)
4: **se** TAMANHO(T_v) < TAMANHO(T_u) **então** ▷ Garantimos que $|T_v| \geqslant |T_u|$
5: $u \leftrightarrow v$
6: $T_u \leftrightarrow T_v$
7: **para** xy em T_u com NÍVEL[x, y] = i **faça** ▷ Move T_u para o nível $i - 1$
8: NÍVEL$[x, y] \leftarrow i - 1$
9: LIGUEFD($G.F_{\leqslant i-1}$, x, y)
10: **para** xy em $G.R_i$ com x em T_u **faça** ▷ Procura substituta para uv
11: REMOVAGLA($G.R_i$, x, y)
12: **se não** CONECTADOFD($F_{\leqslant i}$, x, y **então**
13: **para** $j \leftarrow i$ até $\lceil \lg n \rceil$ **faça**
14: LIGUEFD($G.F_{\leqslant j}$, x, y)
15: **retorne**
16: **senão**
17: NÍVEL$[x, y] \leftarrow i - 1$
18: LIGUEGLA($G.R_{i-1}$, x, y)

Implementação

- Implementamos HDT em Python3;
- O código dessa implementação está livremente disponível;
- O código LaTeXdo dissertão também está disponível.

Floresta maximal de peso mínimo em grafos planos ponderados dinâmicos

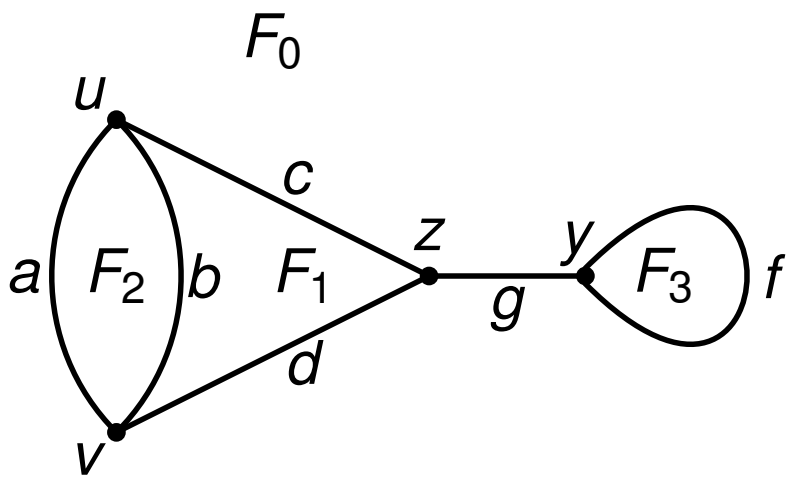
Grafo plano

Grafo plano

Um **grafo plano** é um grafo $G = (V, E)$ com as seguintes propriedades:

1. $V \subset \mathbb{R}^2$;
2. Toda aresta é um arco entre dois vértices;
3. O interior de uma aresta não contém vértices nem intersecta outras arestas.

aresta	peso
a	2
b	7
c	3
d	1
f	2
g	4

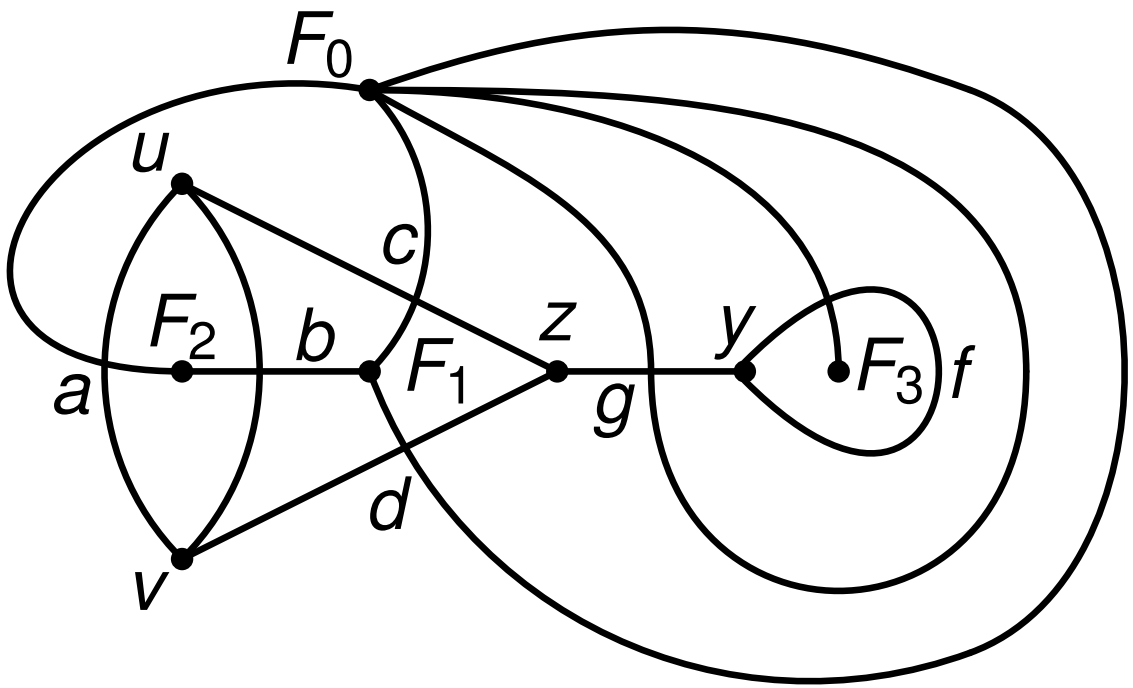


Grafo dual

Grafo dual

Dado um grafo plano G, o grafo **dual** de G é o grafo $G^\star = (F, E^\star)$, onde

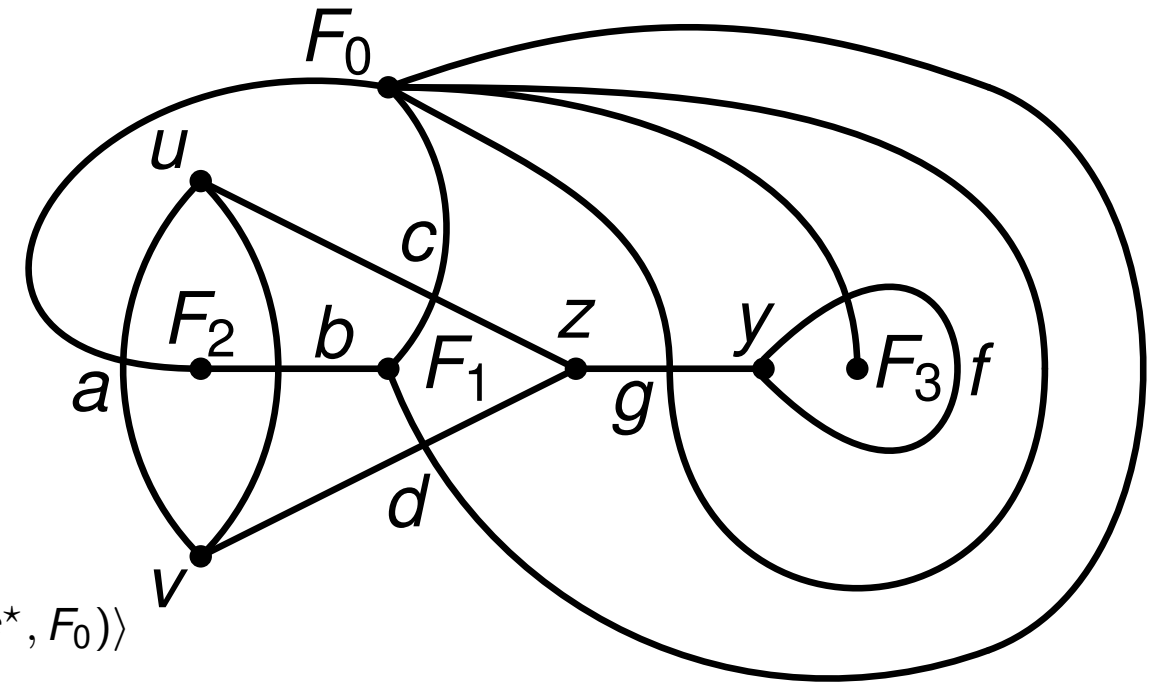
- F é o conjunto de faces de G;
- $E^\star$ é o conjunto de arestas duais de G.



Descrição combinatória plana

$D(u) = \langle (a, v), (b, v), (c, z) \rangle$
$D(v) = \langle (a, u), (d, z), (b, u) \rangle$
$D(z) = \langle (c, u), (d, v), (g, y) \rangle$
$D(y) = \langle (g, z), (f, y), (f, y) \rangle$

$D(F_0) = \langle (a^\star, F_2), (c^\star, F_1),$
$(g^\star, F_0), (y^\star, F_3),$
$(g^\star, F_0), (d^\star, F_1) \rangle$
$D(F_1) = \langle (b^\star, F_2), (d^\star, F_0), (c^\star, F_0) \rangle$
$D(F_2) = \langle (a^\star, F_0), (b^\star, F_1) \rangle$
$D(F_3) = \langle (f^\star, F_0) \rangle$



Floresta maximal de peso mínimo

O problema de floresta maximal de peso mínimo em grafos planos

- NOVOGDP(n): Cria e devolve um grafo plano ponderado G com n vértices isolados.
- LIGUEGDP(G, e, u, e_u, v, e_v, w): Insere em G uma nova aresta e com peso w ligando os vértices u e v. A nova aresta e é sucessora das arestas e_u e e_v nas ordens cíclicas de u e v, respectivamente.
- REMOVAGDP(G, e): Remove a aresta e de G.
- MUDAPESOGDP(G, e, w): Altera o peso da aresta e de G para o valor w.
- PESOGDP(G): Devolve o peso de uma MSF de G.

Árvores dinâmicas planas

Apresentaremos a solução proposta por Eppstein, Italiano, Tamassia, Tarjan, Westbrook e Yung para esse problema. O nome da estrutura de dados introduzida por esses autores é *edge-ordered dynamic tree* que traduzimos para **árvores dinâmicas planas**.

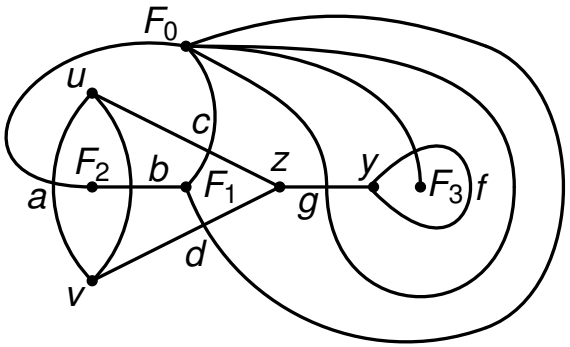
Árvore maximal e sua dual

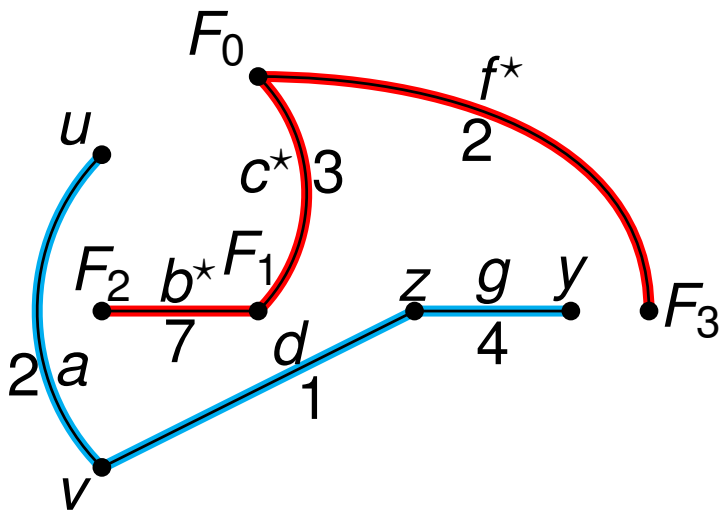
Teorema

Seja T uma árvore geradora de um grafo plano conexo G. O conjunto

$$T^\star = \{e^\star : e \notin T\}$$

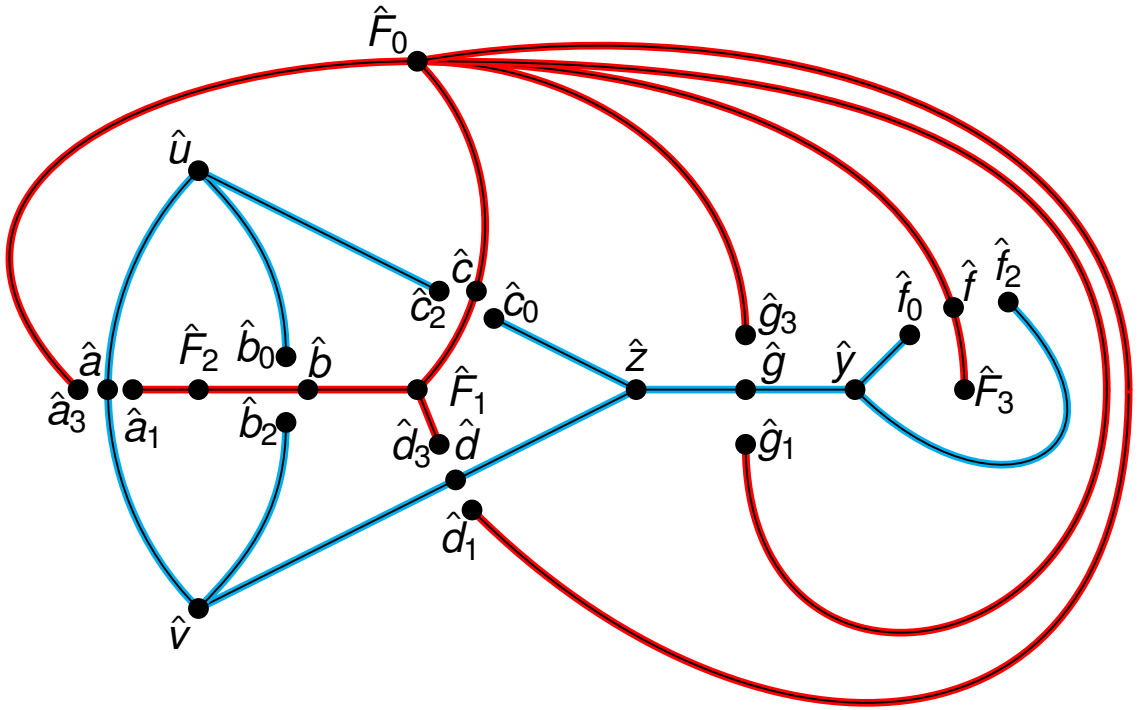
é uma árvore geradora de $G^\star$. Além disso, se G for ponderado e adotarmos $w(e^\star) = w(e)$, então T será de peso mínimo em G se e somente se $T^\star$ for de peso máximo em $G^\star$.



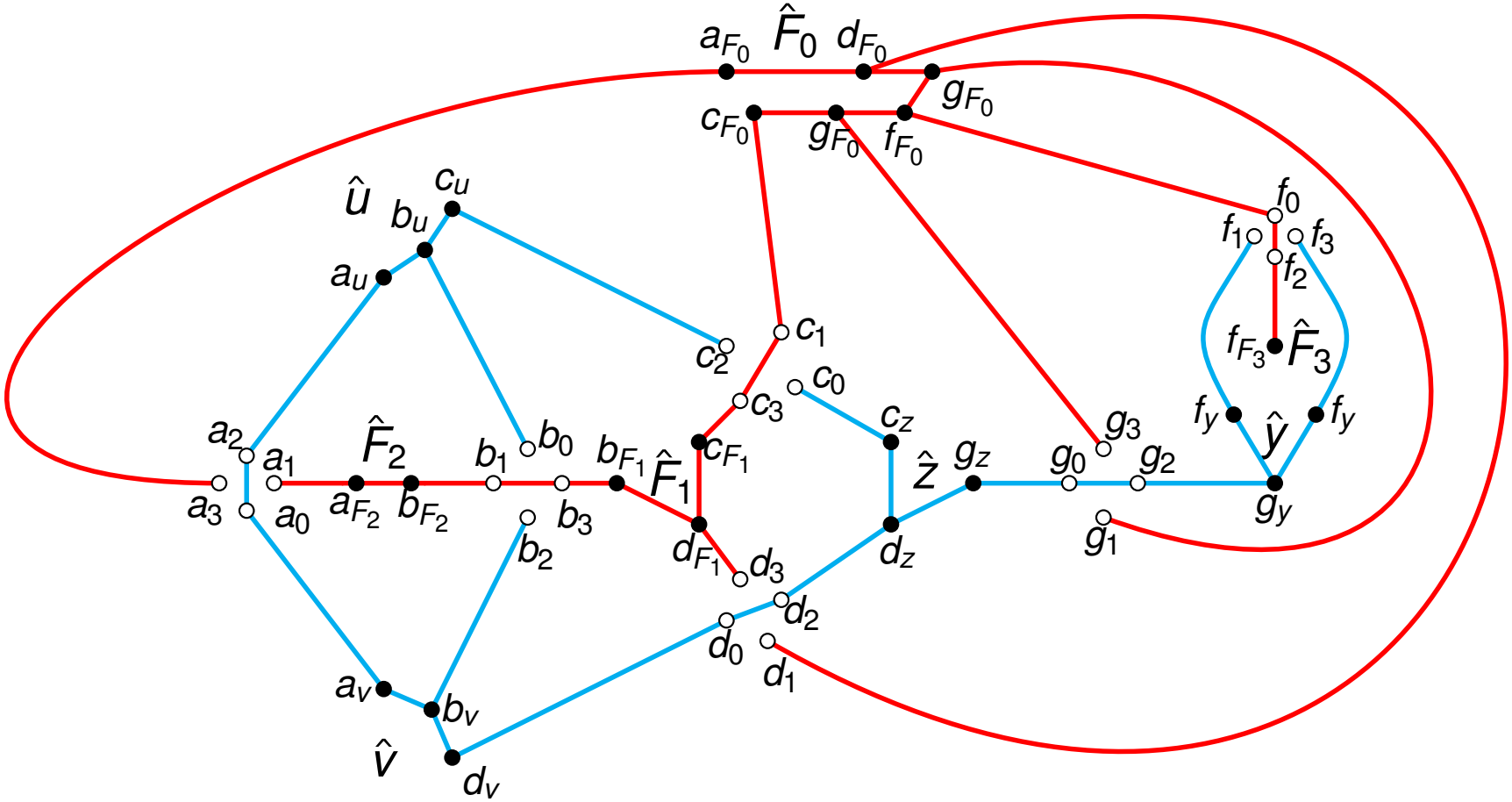


Árvores modificadas

vértices	pesos
$\hat{a}$, $\hat{a}_1$, $\hat{a}_3$	2
$\hat{b}$, $\hat{b}_0$, $\hat{b}_2$	7
$\hat{c}$, $\hat{c}_0$, $\hat{c}_2$	3
$\hat{d}$, $\hat{d}_0$, $\hat{d}_2$	2
$\hat{f}$, $\hat{f}_1$, $\hat{f}_3$	1
$\hat{g}$, $\hat{g}_1$, $\hat{g}_3$	4
$\hat{u}$, $\hat{v}$, $\hat{y}$, $\hat{z}$	$-\infty$
$\hat{F}_0$, $\hat{F}_1$, $\hat{F}_2$, $\hat{F}_3$	∞



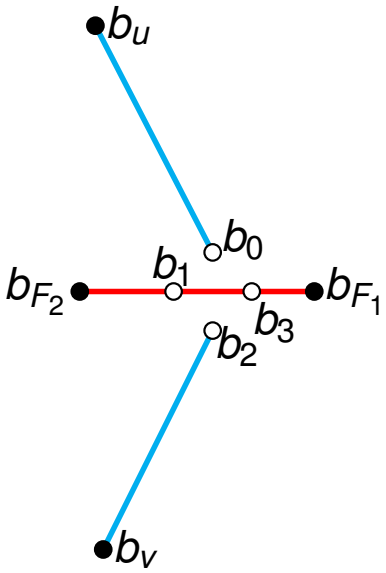
Exemplo de ADP

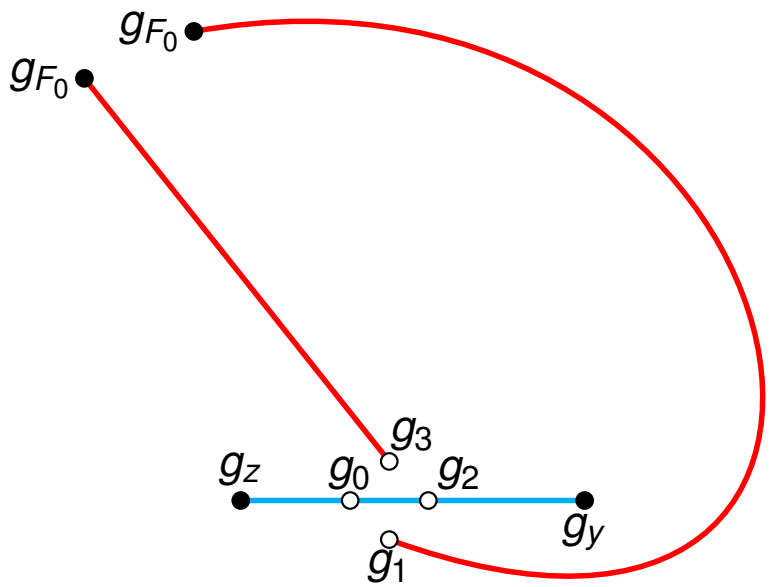


Exemplo de ADP

Óctupla de e

Há 8 nós de link cut tree para cada aresta e de G. Chamaremos esses 8 vértices de **óctupla de** e.





Criação de grafo plano ponderado dinâmico e obtenção de peso

Algorithm NOVOGDP(n)

1: $G.H \leftarrow$ NOVODICIO(n)
2: $G.p \leftarrow 0$
3: **retorne** G

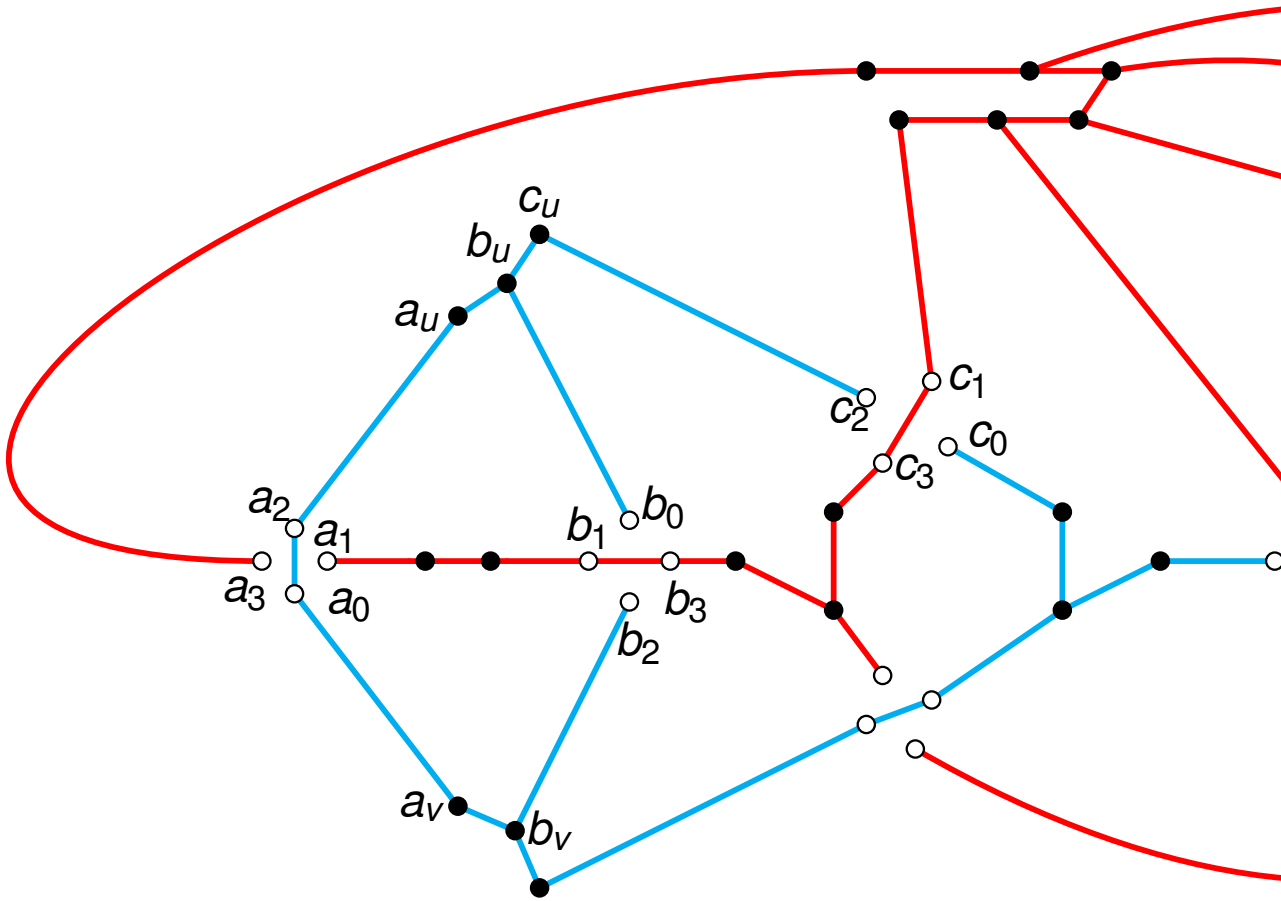
Algorithm PESOGDP(G)

1: **retorne** $G.p$

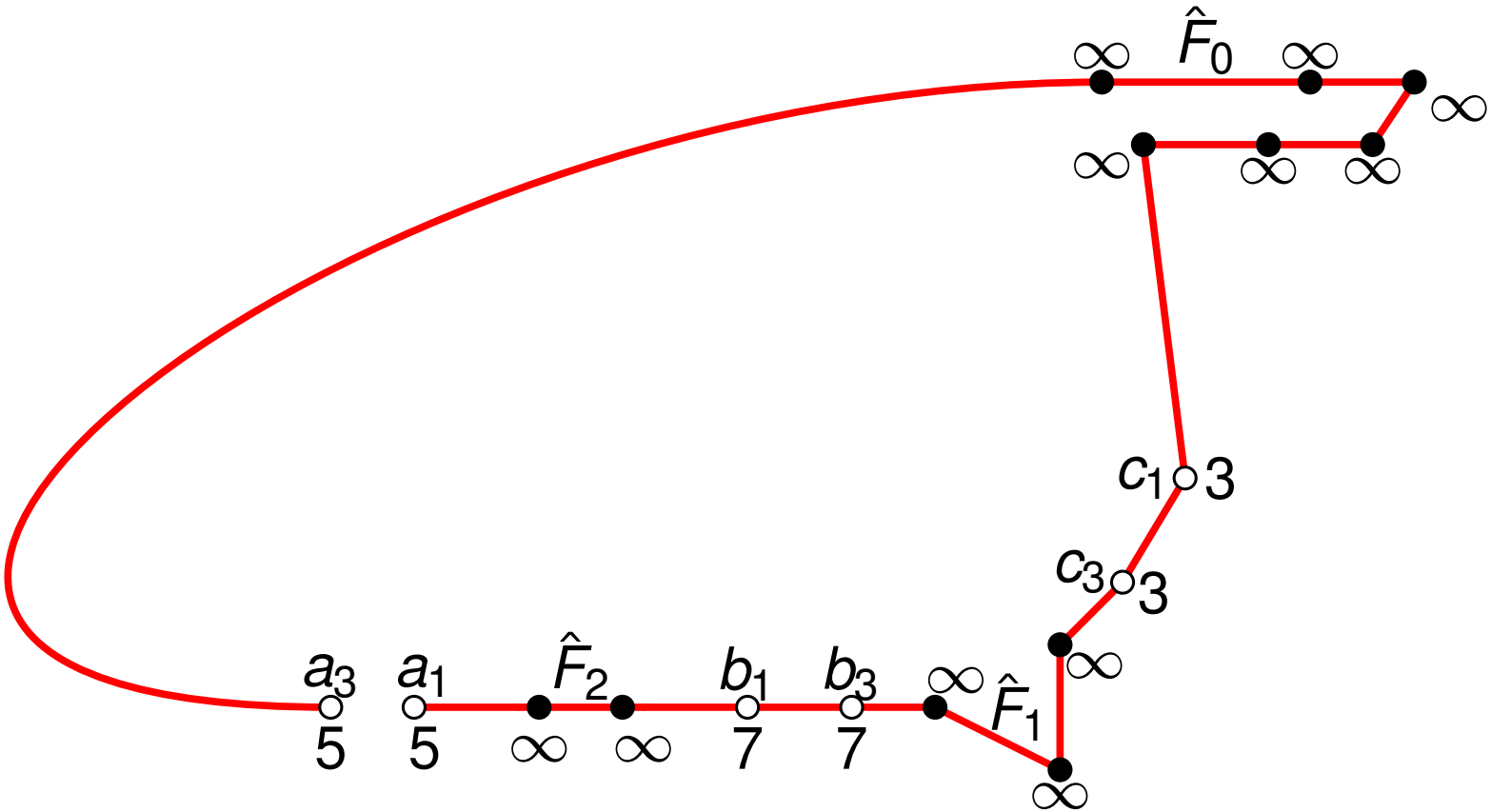
NOVOGDP: $O(n)$.
PESOGDP: $O(1)$.
LIGUEGDP: $O(\lg m)$
REMOVAGDP: $O(\lg m)$.
MUDAPESOGDP: $O(\lg m)$.

Execução de Mudança de peso MUDAPESOGDP(G, a, 5)

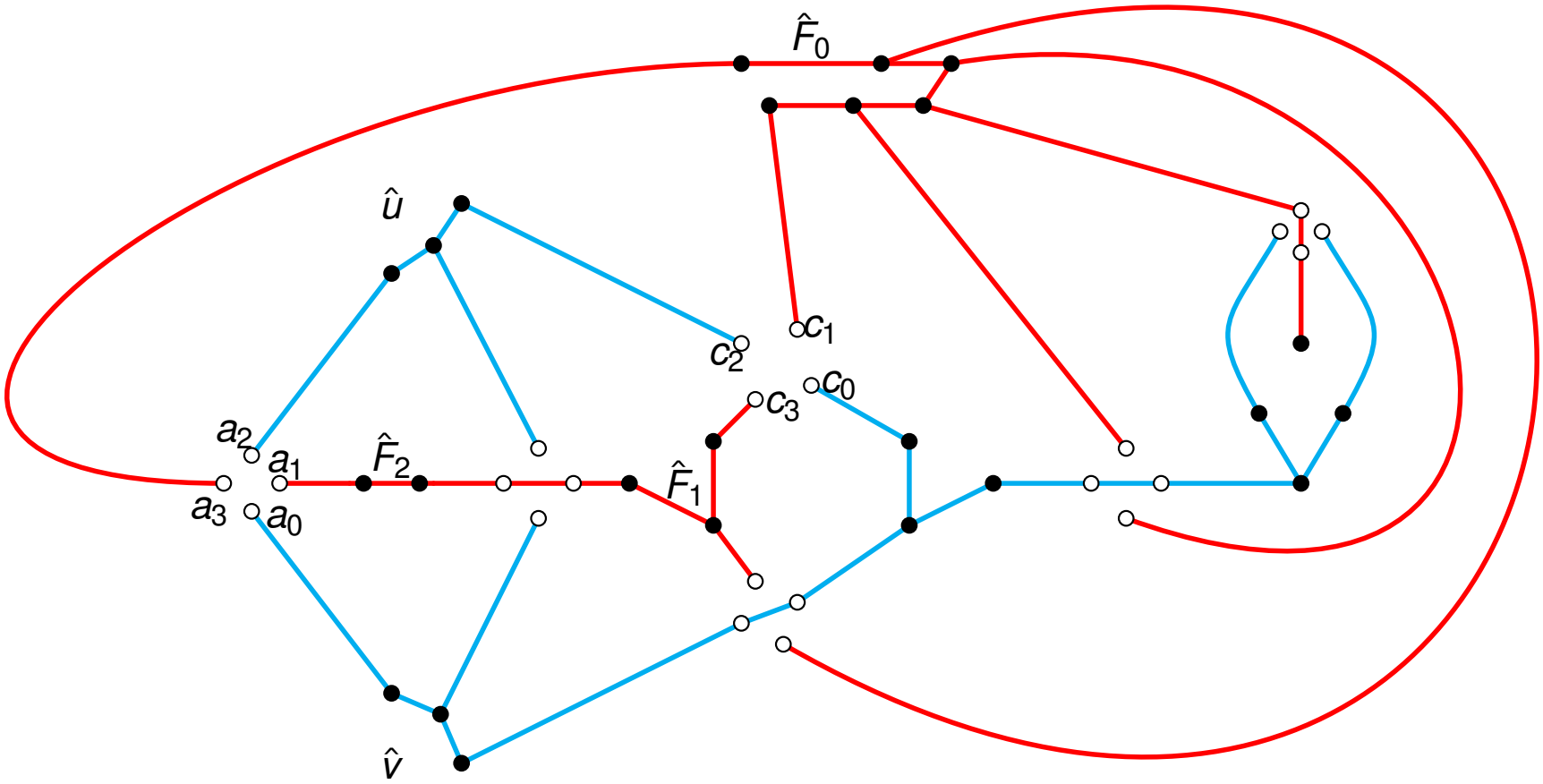
aresta	peso
a_i	2
b_i	7
c_i	3
d_i	1
f_i	2
g_i	4



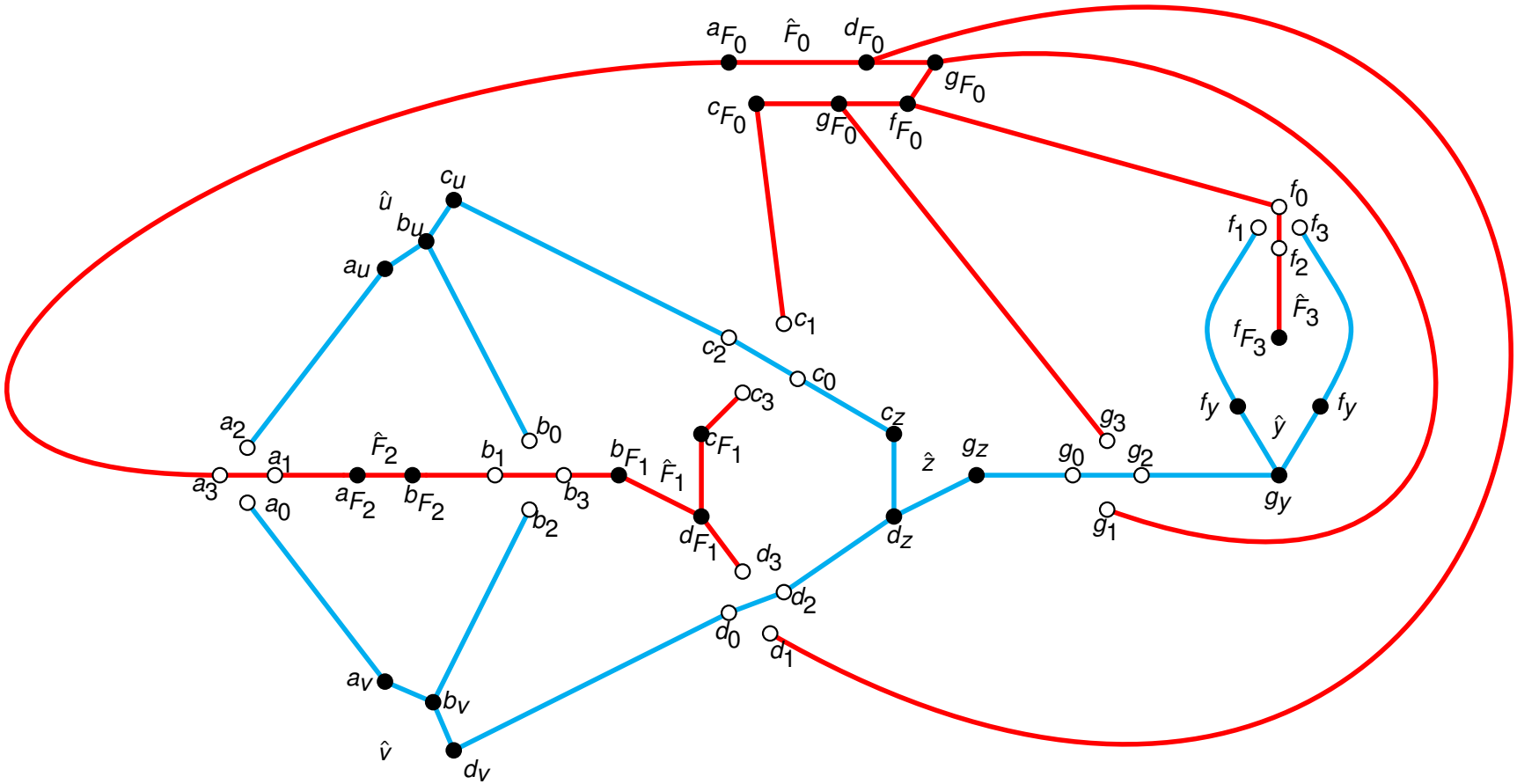
Execução de Mudança de peso MUDAPESOGDP(G, a, 5)



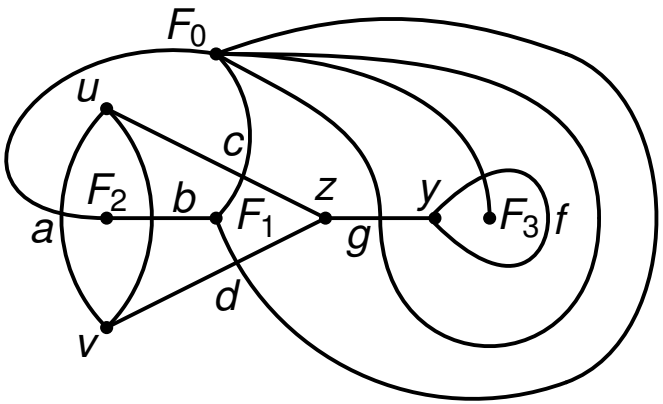
Execução de Mudança de peso MUDAPESOGDP(G, a, 5)

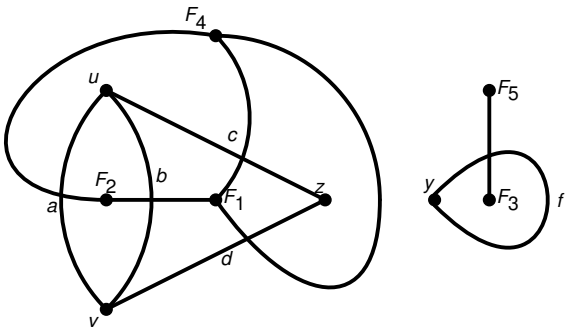


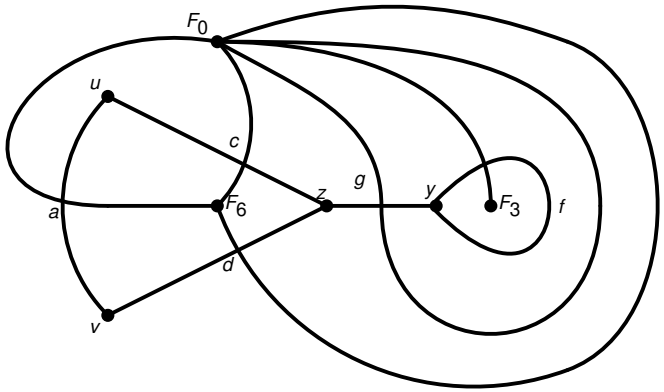
Execução de Mudança de peso MUDAPESOGDP(G, a, 5)



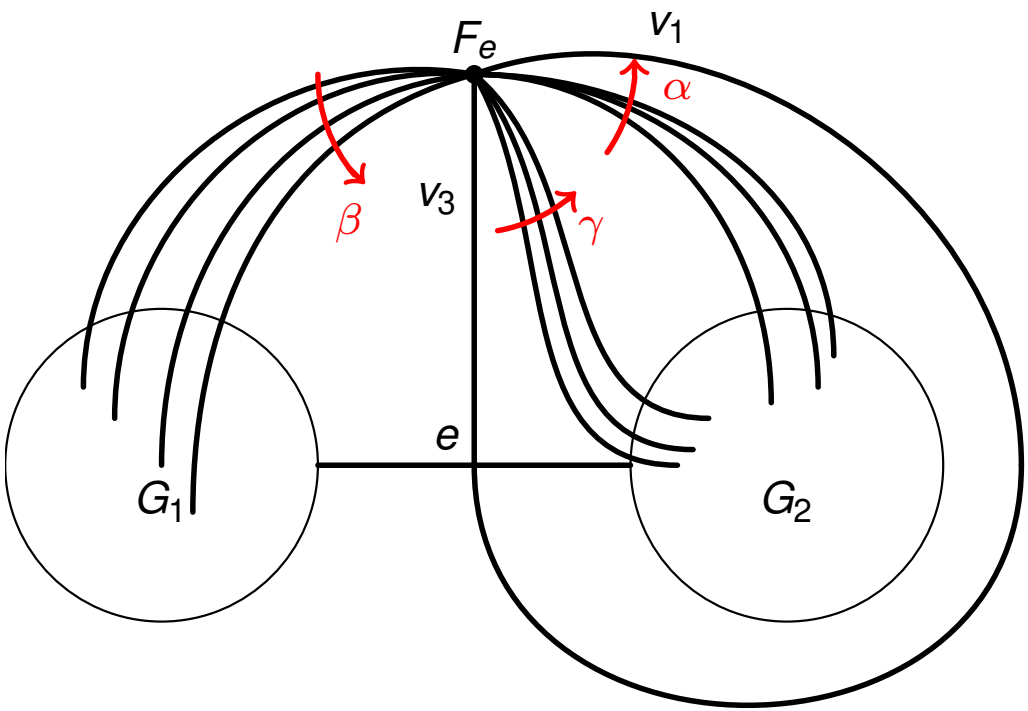
Dois casos de remoção de aresta



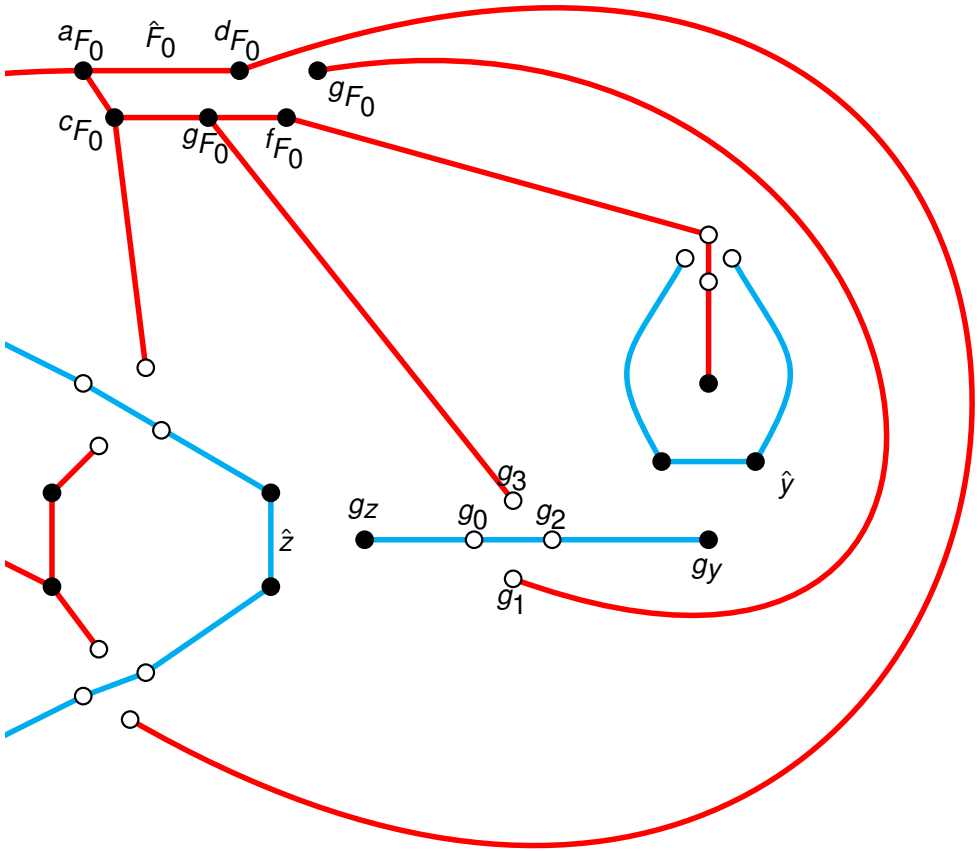




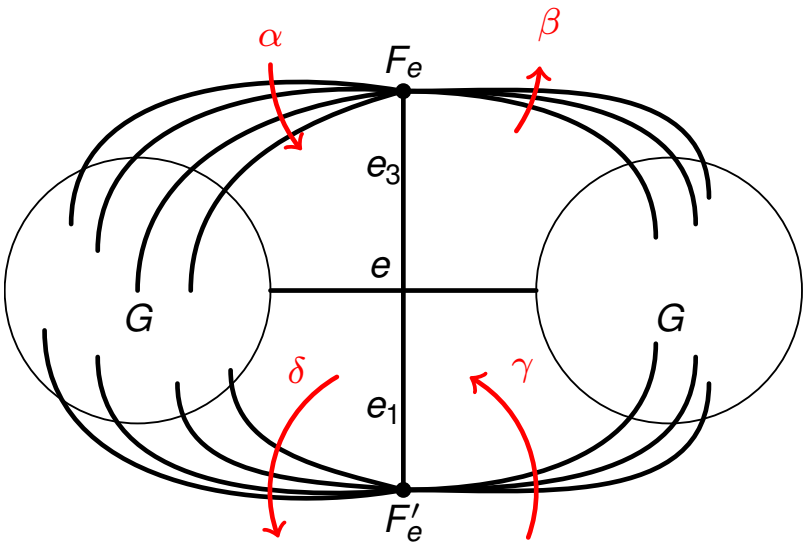
Remoção de ponte

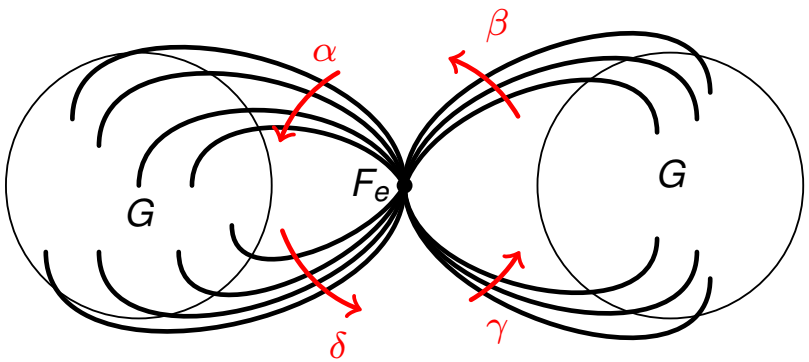


Remoção de ponte - REMOVAGDP(G, g)

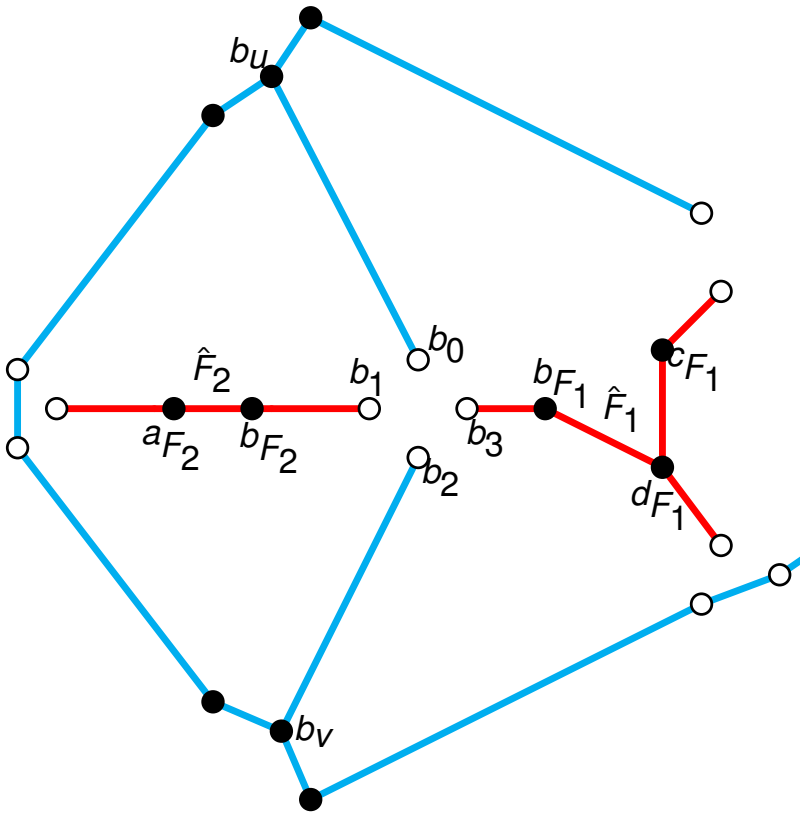


Remoção de não ponte

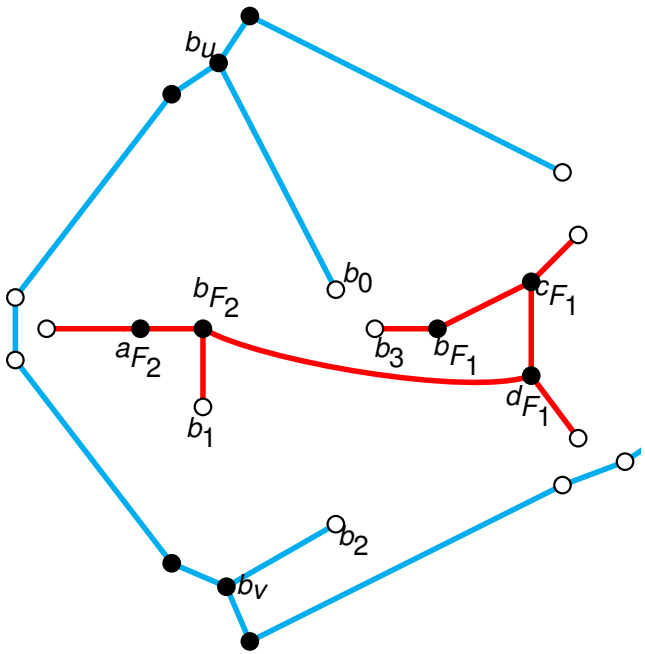




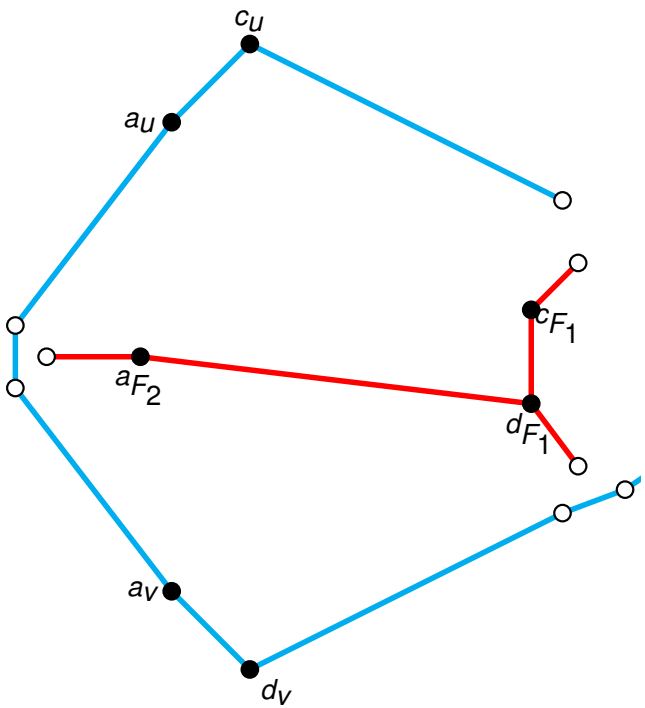
Remoção de não ponte - REMOVAGDP(G, b)



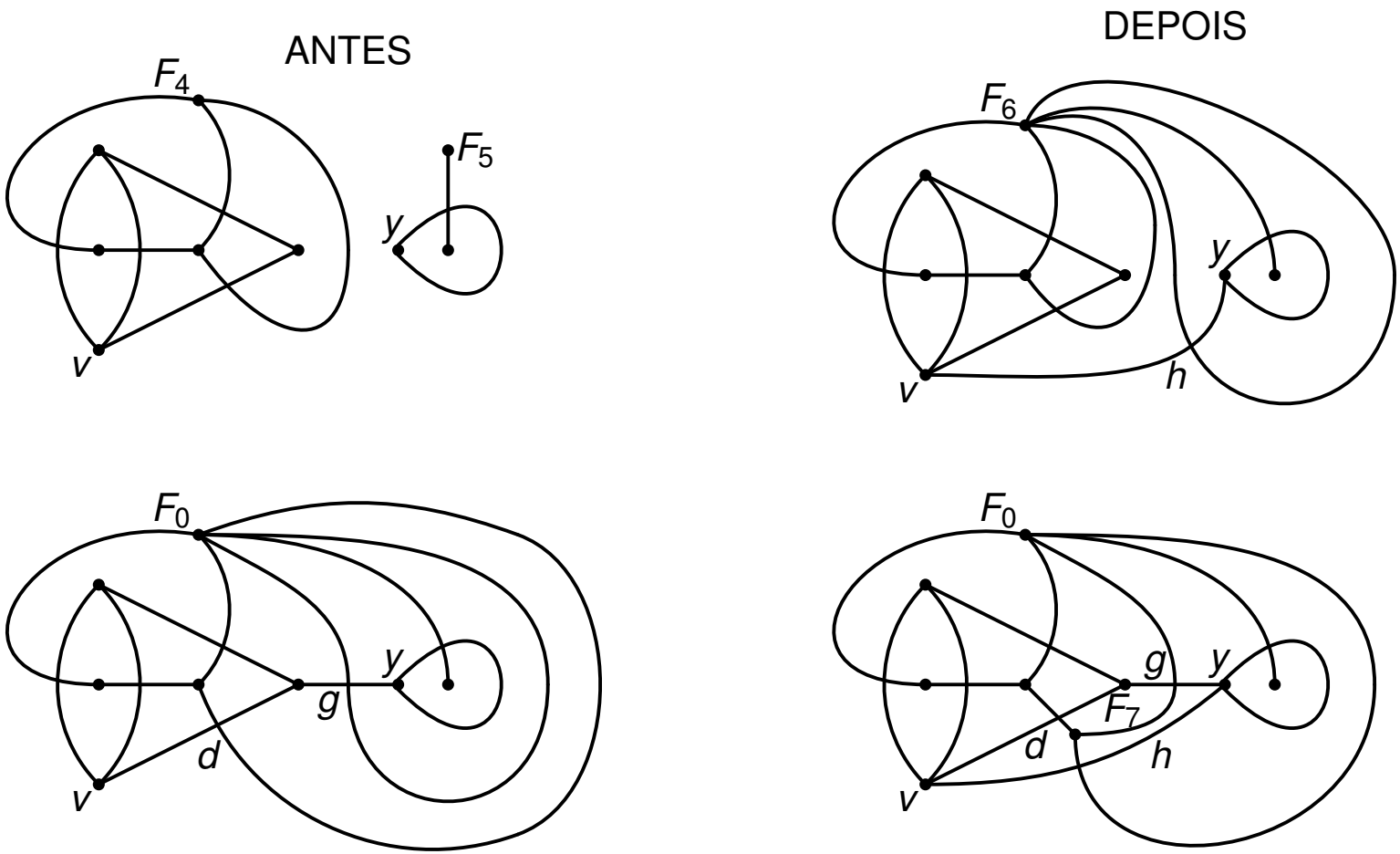
Remoção de não ponte - REMOVAGDP(G, b)



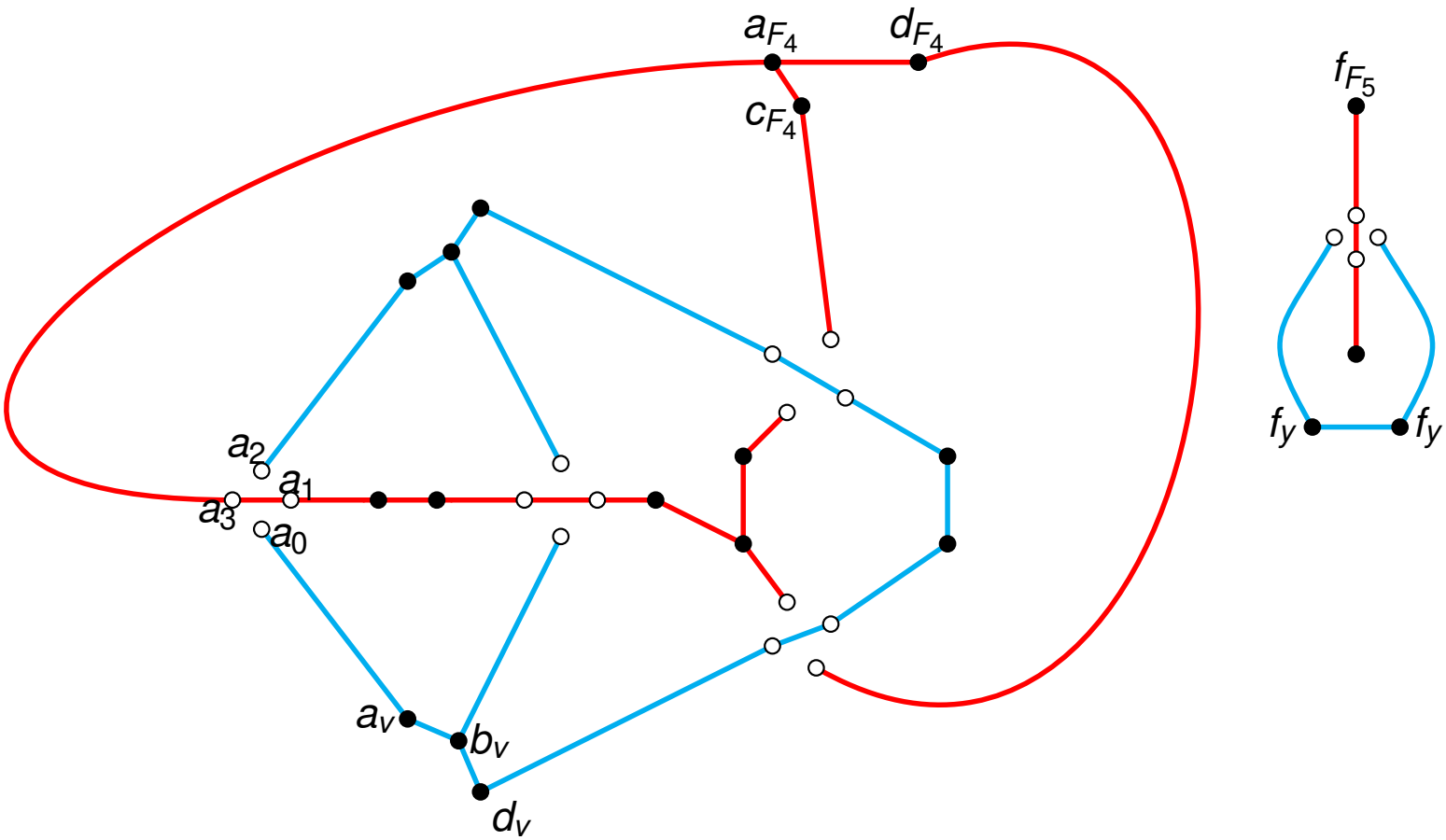
Remoção de não ponte - REMOVANÃOPONTE(G, b)



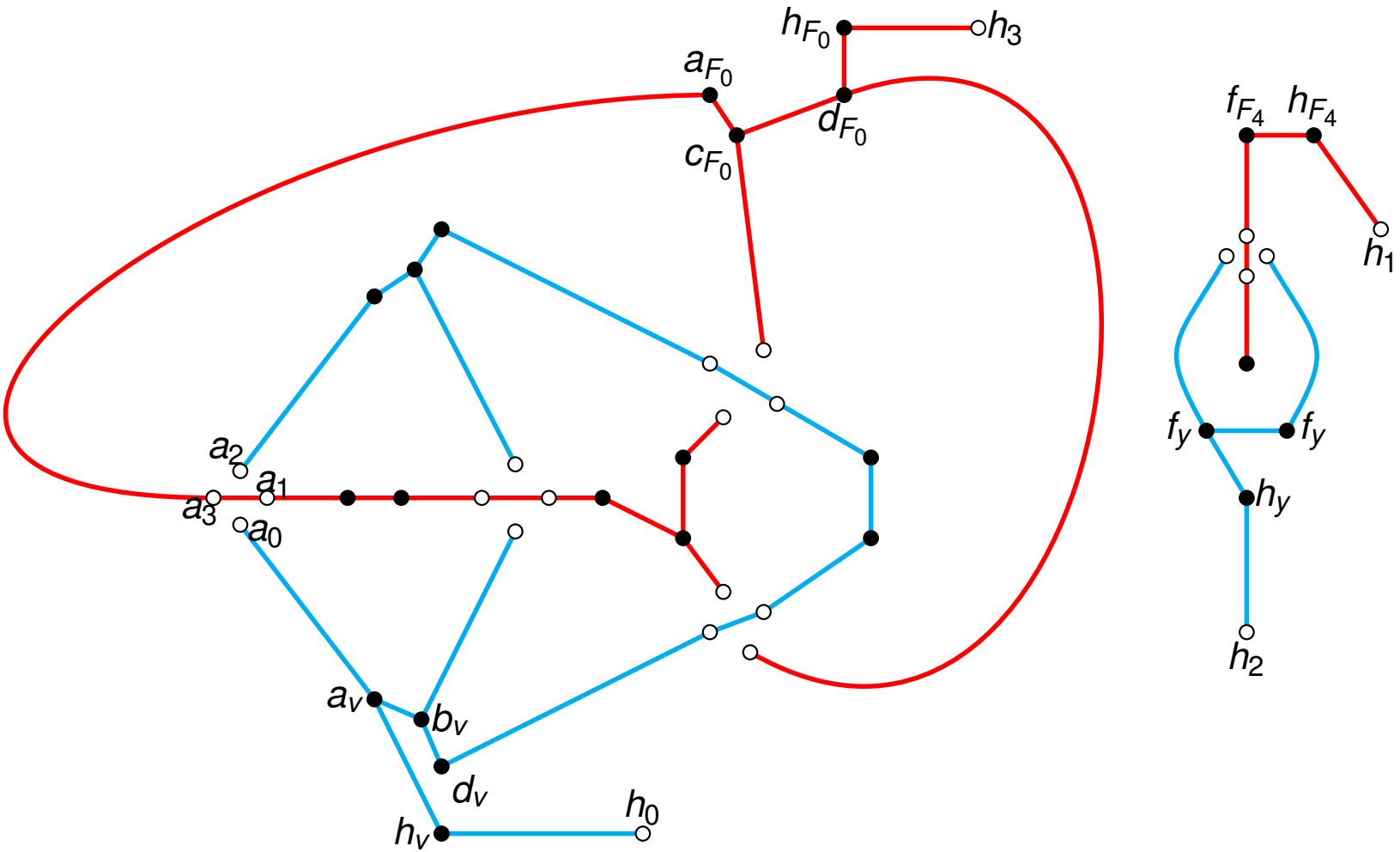
Dois casos de adição de aresta



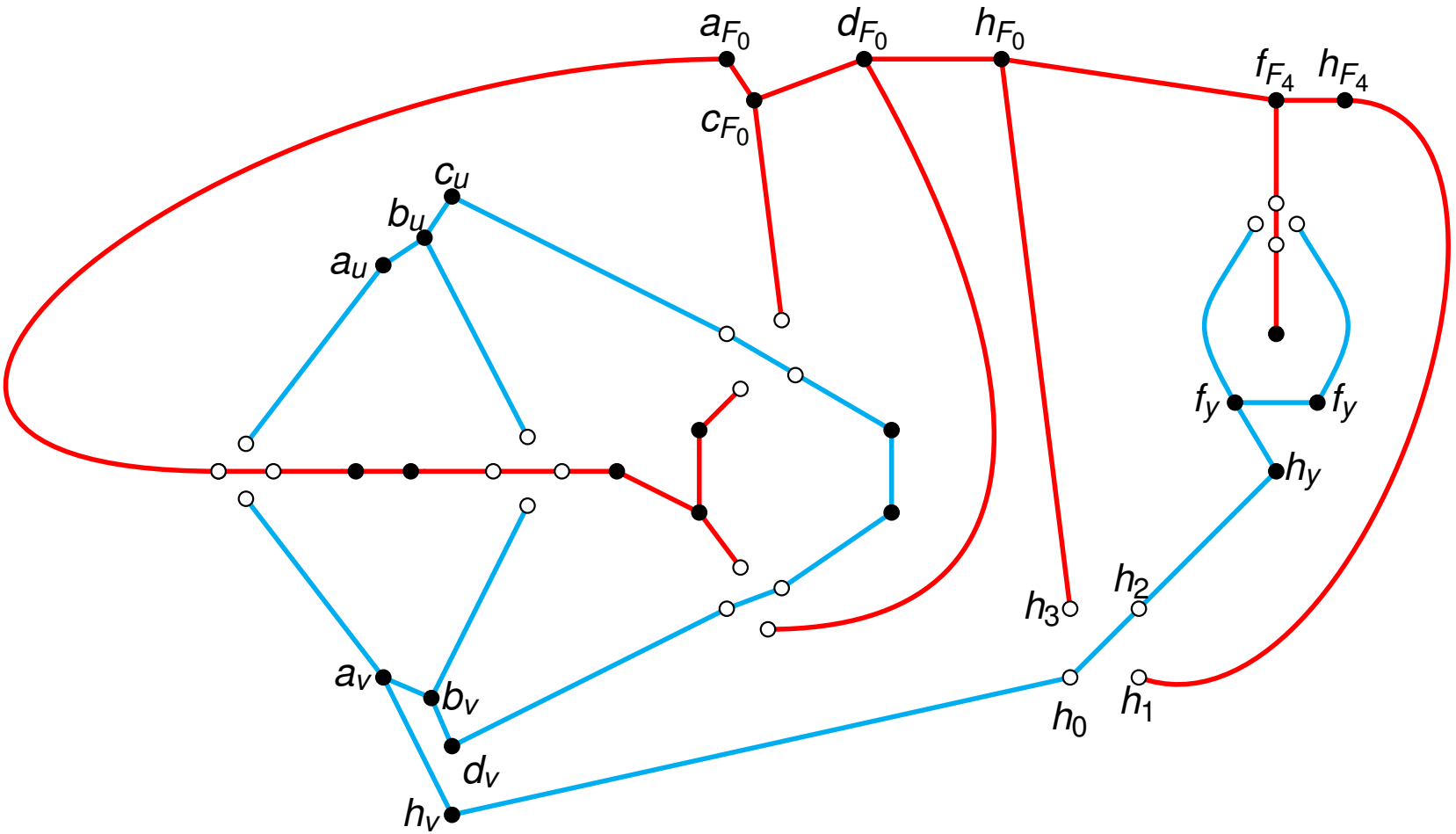
Adição de aresta - LIGUEGDP(G, h, v, a, y, f, 7)



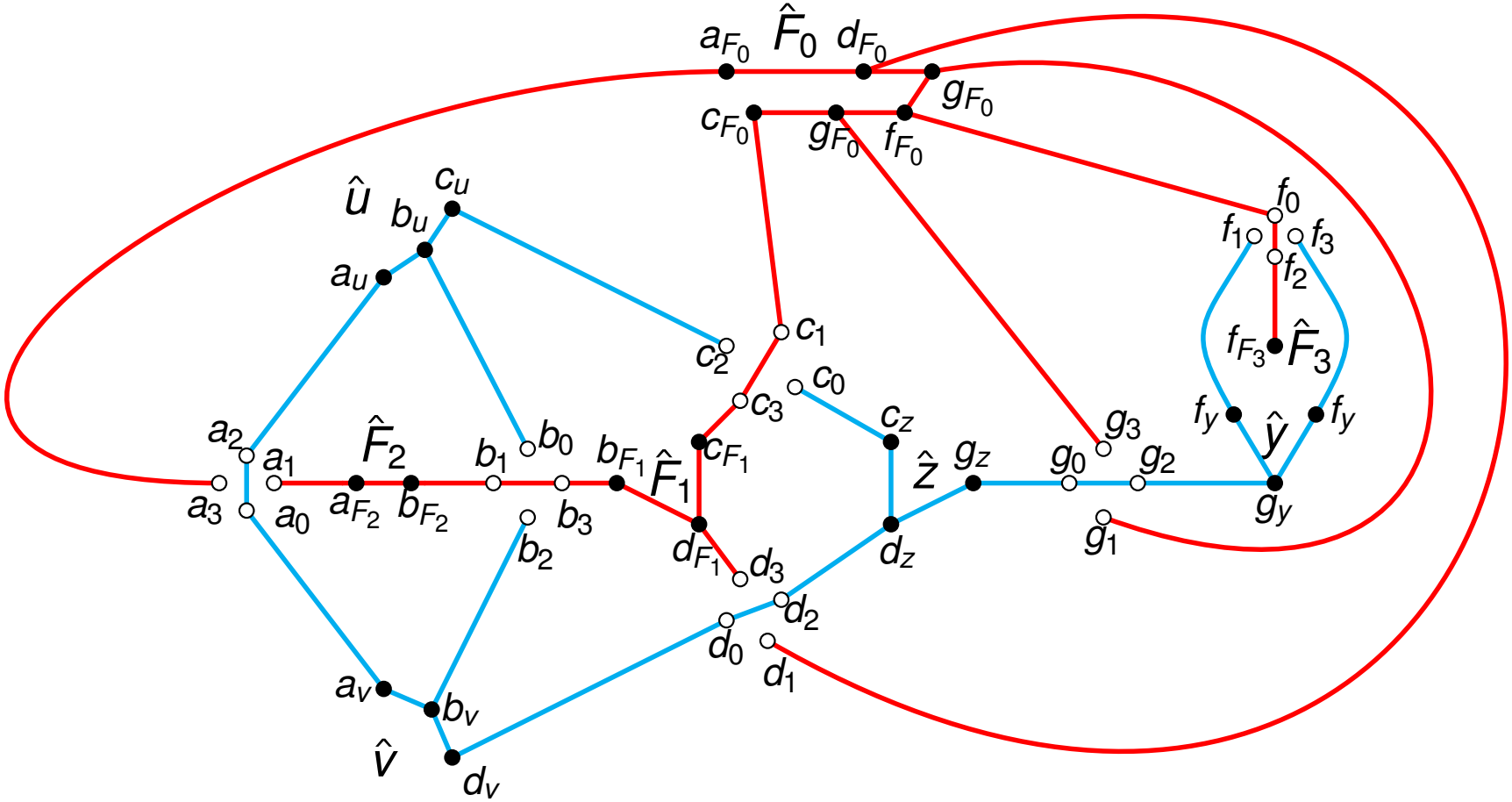
Adição de aresta - LIGUEGDP(G, h, v, a, y, f, 7)



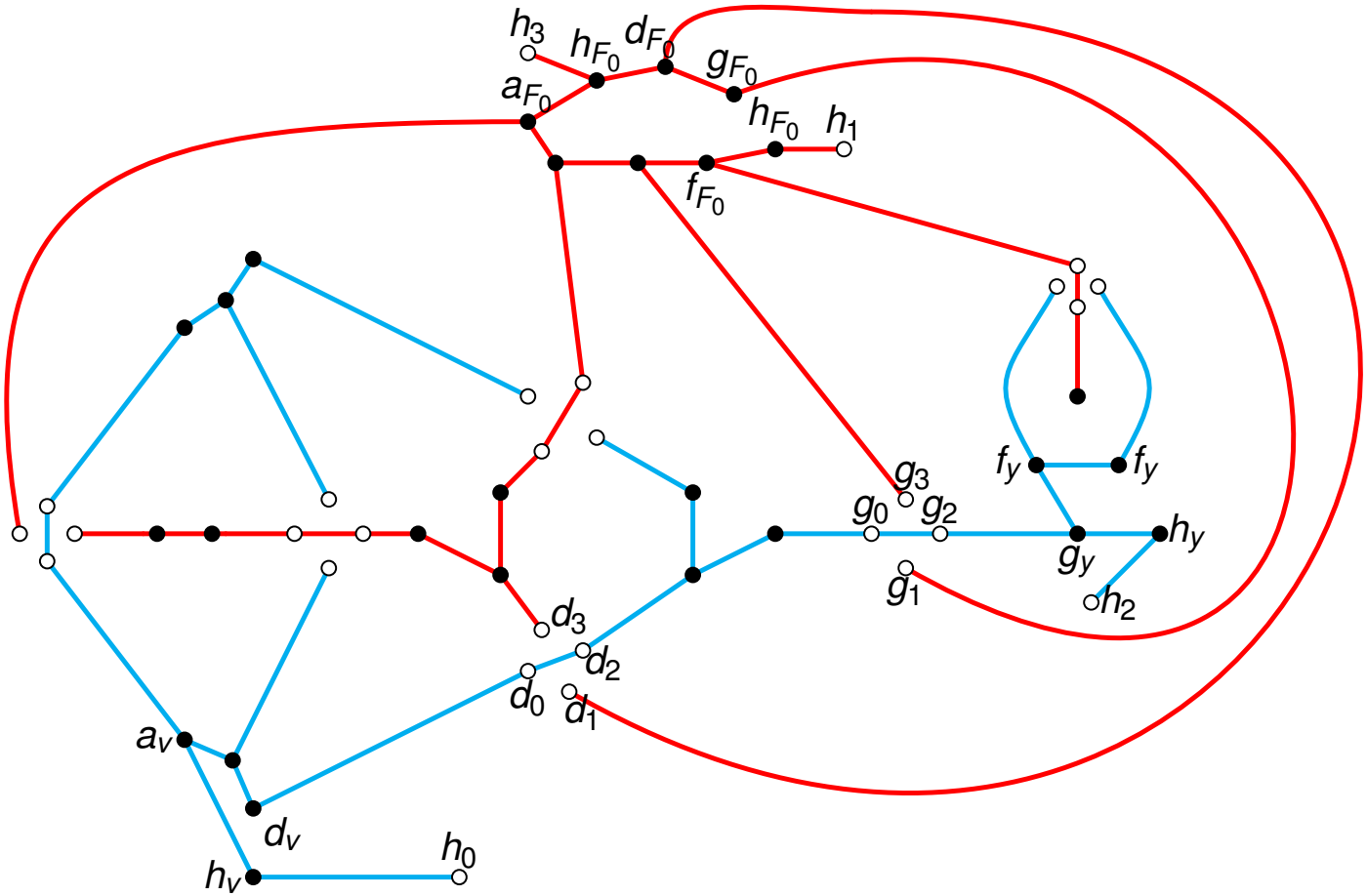
Adição de aresta - LIGUEGDP(G, h, v, a, y, f, 7)



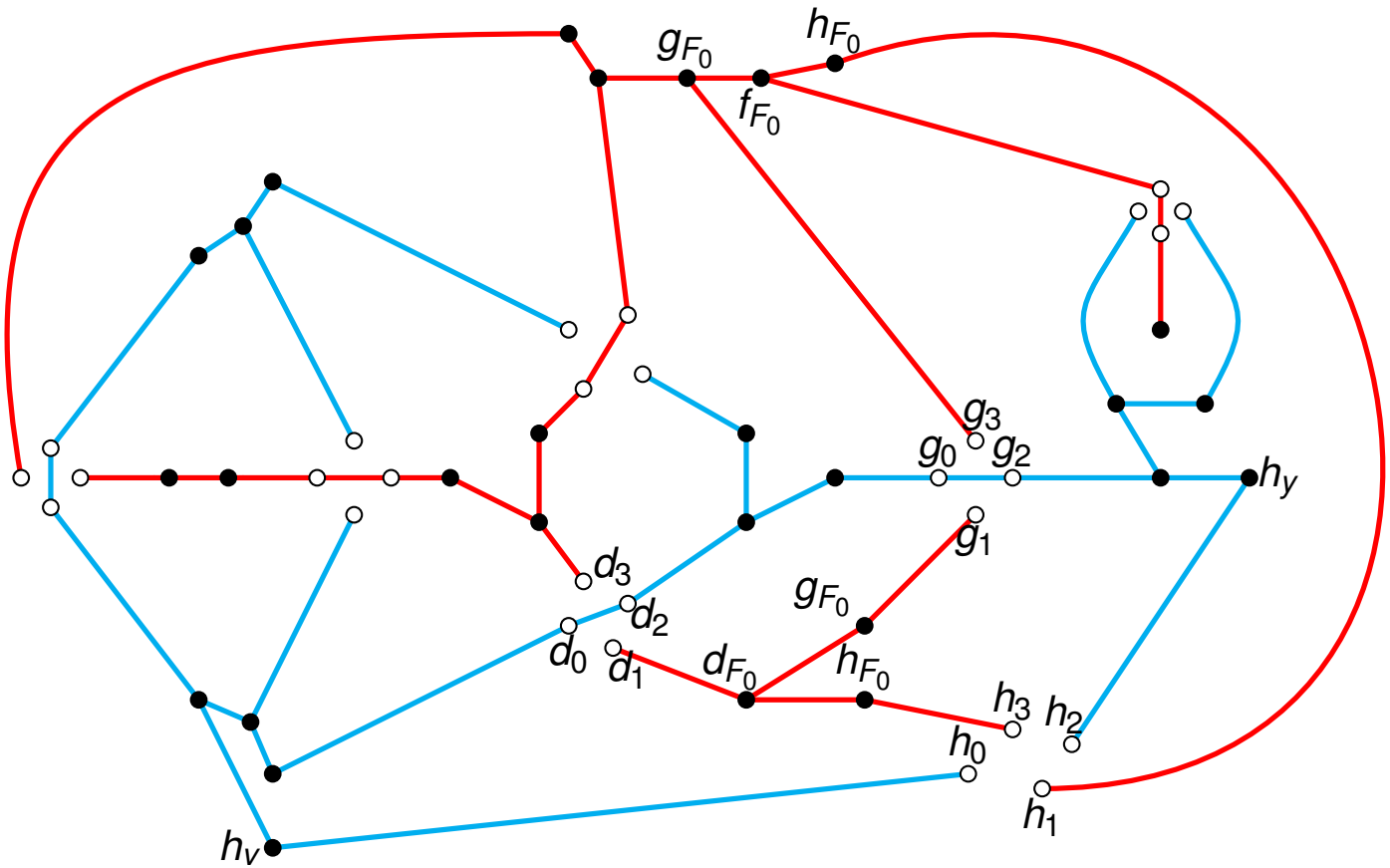
Adição de aresta - LIGUEGDP(G, h, v, a, y, f, 7)



Adição de aresta - LIGUEGDP(G, h, v, a, y, f, 7)



Adição de aresta - LIGUEGDP(G, h, v, a, y, f, 7)



O limitante inferior de $\Omega(\lg n)$

O limitante inferior de $\Omega(\lg n)$

Mihai Patrascu e Erik D. Demaine provaram o seguinte limitante inferior:

Teorema

Seja t_m o consumo de tempo de LIGUEGD *ou* REMOVAGD *e t_c o consumo de tempo de* CONECTADOGD*, então*

$$\min\{t_m, t_c\} \lg\left(\frac{\max\{t_m, t_c\}}{\min\{t_m, t_c\}}\right) = \Omega(\lg n).$$

Esse limitante é válido mesmo para implementações aleatorizadas e/ou amortizadas de LIGUEGD*,* REMOVAGD *e* CONECTADOGD *e mesmo restringindo a classe de grafos do problema para caminhos.*

Bibliografia

Bibliografia I


Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
Introduction to Algorithms.
The MIT Press, 2 edition, 2001.


Reinhard Diestel.
Graph Theory.
Graduate Texts in Mathematics. Springer, 6 edition, 2024.


David Eppstein, Giuseppe F Italiano, Roberto Tamassia, Robert E Tarjan, Jeffery Westbrook, and Moti Yung.
Maintenance of a minimum spanning forest in a dynamic plane graph.
Journal of Algorithms, 13(1):33–54, 1992.


Monika R. Henzinger and Valerie King.
Randomized fully dynamic graph algorithms with polylogarithmic time per operation.
Journal of ACM, 46(4):502–516, 1999.


Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup.
Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity.
Journal of the ACM, 48(4):723–760, 2001.

Bibliografia II

Mihai Patrascu and Erik D. Demaine.
Logarithmic lower bounds in the cell-probe model.
SIAM Journal on Computing, 35(4):932–963, 2006.