Análise e Síntese de Algoritmos
DFS.
CLRS Cap. 22

Prof. Pedro T. Monteiro

IST - Universidade de Lisboa

2024/2025

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica [CLRS, Cap.15]
  - Algoritmos greedy [CLRS, Cap.16]
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes [CLRS, Cap.23]
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais
  - Complexidade Computacional [CLRS, Cap.34]

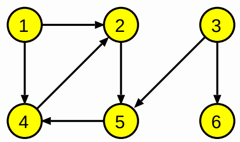
Grafos

Grafo $G = (V, E)$ é definido por:

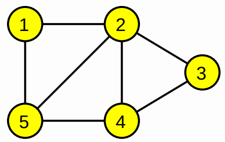
- um conjunto de V vértices
- um conjunto de $E \subseteq V \times V$ arcos
  - Se $|E| << |V \times V|$, o grafo diz-se esparso
  - Caso contrário, diz-se denso

Grafos podem ser ou não dirigidos

Grafo Dirigido



Grafo Não Dirigido

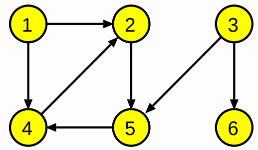


Grafos

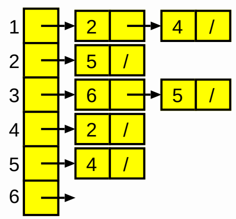
Representação dos arcos

- Matriz de adjacências: arcos representados por matriz
  $\rightarrow$ para grafos densos
- Listas de adjacências: arcos representados por listas
  $\rightarrow$ para grafos esparsos

Grafo Dirigido



Listas de Adjacências



Matriz de Adjacências

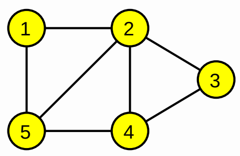
	1	2	3	4	5	6
1		1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	**0**	0	0	0	0	0

Grafos

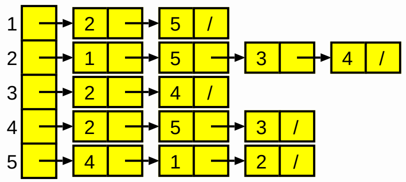
Representação dos arcos

- Matriz de adjacências: arcos representados por matriz
  $\rightarrow$ para grafos densos
- Listas de adjacências: arcos representados por listas
  $\rightarrow$ para grafos esparsos

Grafo Não Dirigido



Lista de Adjacências



Matriz de Adjacências

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

Grafos

Matriz de Adjacências

- $\Theta(V^2)$ para qualquer grafo

Listas de adjacências

- Tamanho das listas é $|E|$ para grafos dirigidos
- Tamanho das listas é $2|E|$ para grafos não dirigidos
- Tamanho total das listas de adjacências é $\Theta(V + E)$

Grafos pesados

- Existência de uma função de pesos $\omega : E \rightarrow IR$
- Função de pesos ω associa um peso a cada arco $(u, v) \in E$

Grafos

Questões

- Dado um grafo denso, qual a representação adequada para os arcos?
- E se a operação mais frequente for ler o peso dos arcos?
- E se quiser representar o grafo da World Wide Web?

Procura em profundidade primeiro (DFS)

Intuição

Grafo pesquisado dando prioridade aos arcos dos vértices visitados mais recentemente

Aplicações

- Resolução de labirintos
- Detecção de ciclos
- Ordenação topológica
- Testar se um grafo é bipartido
- Descobrir componentes fortemente ligados/conexos

Procura em profundidade primeiro (DFS)

Implementação

- $d[v]$: tempo de início (de visita do vértice)
- $f[v]$: tempo de fim (de visita do vértice)
- $\pi[v]$: predecessor de v na árvore DF
- $color[v]$: cor do vértice v: branco/cinzento/preto

Procura em profundidade primeiro (DFS)

DFS(G)

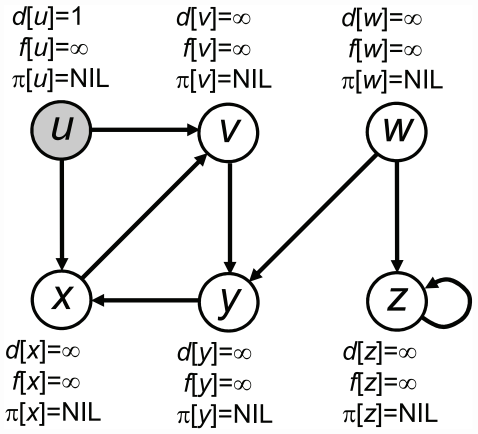
```
for u ∈ G.V do
  color[u] ← white
  d[u] ← ∞
  f[u] ← ∞
  π[u] ← NIL
end for
time ← 1
for u ∈ G.V do
  if color[u] == white then
    DFS-Visit(G, u)
  end if
end for
```

DFS-Visit(G,u)

```
color[u] ← gray
d[u] ← time
time ← time + 1
for v ∈ G.Adj[u] do
  if color[v] == white then
    π[v] ← u
    DFS-Visit(G, v)
  end if
end for
color[u] ← black
f[u] ← time
time ← time + 1
```

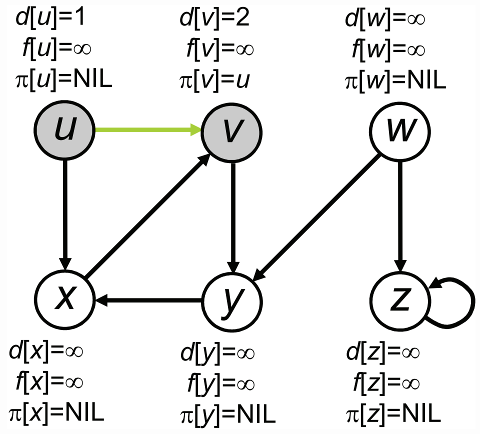

Procura em profundidade primeiro (DFS)

Exemplo



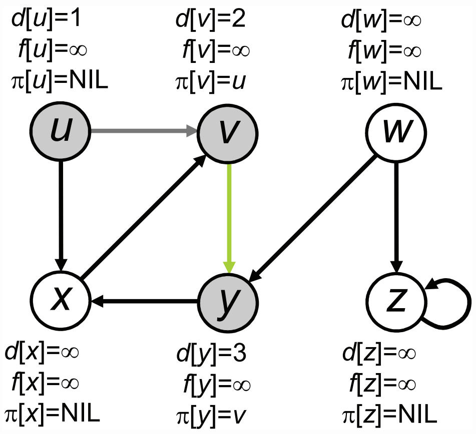
Procura em profundidade primeiro (DFS)

Exemplo



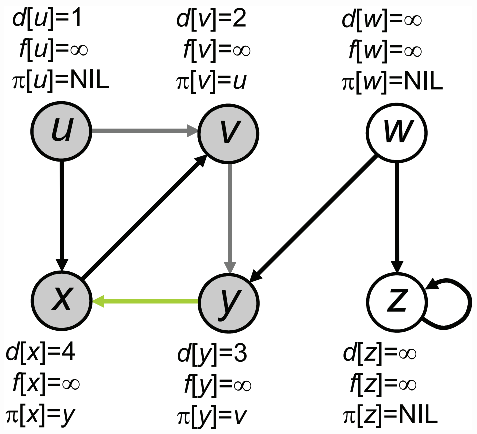
Procura em profundidade primeiro (DFS)

Exemplo



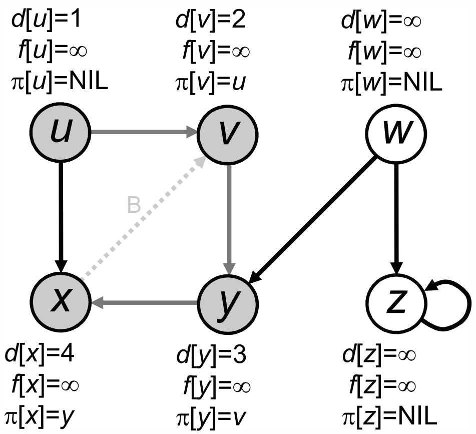
Procura em profundidade primeiro (DFS)

Exemplo



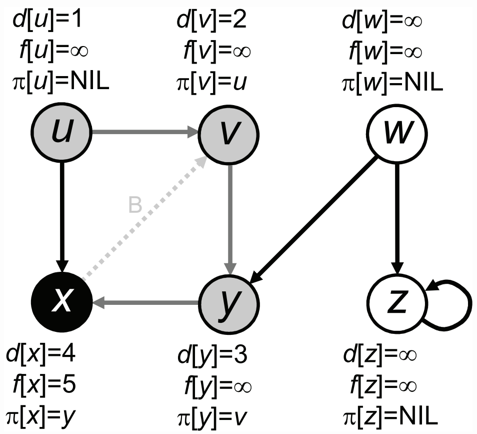
Procura em profundidade primeiro (DFS)

Exemplo



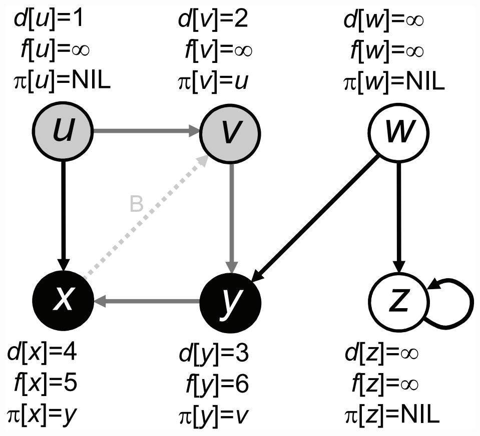
Procura em profundidade primeiro (DFS)

Exemplo



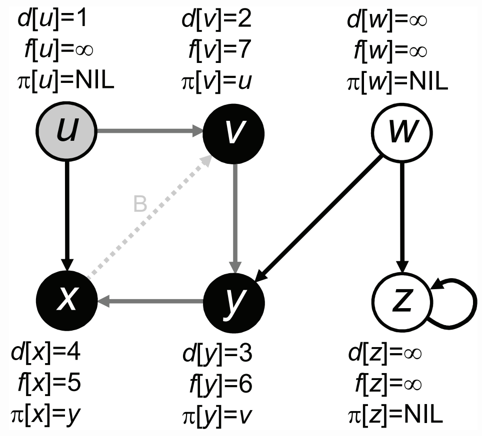
Procura em profundidade primeiro (DFS)

Exemplo



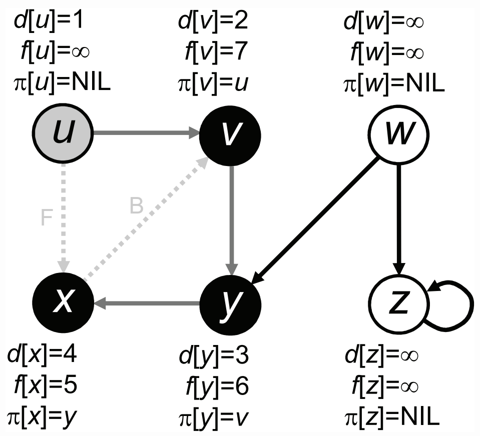
Procura em profundidade primeiro (DFS)

Exemplo



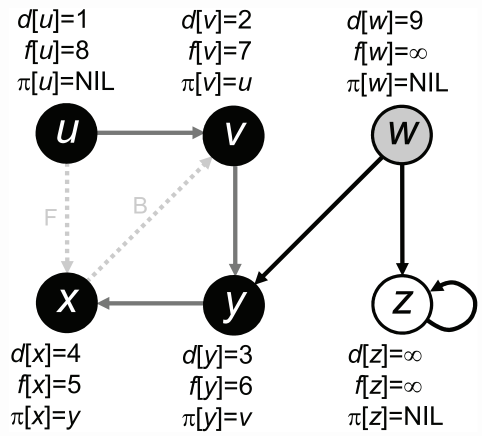
Procura em profundidade primeiro (DFS)

Exemplo

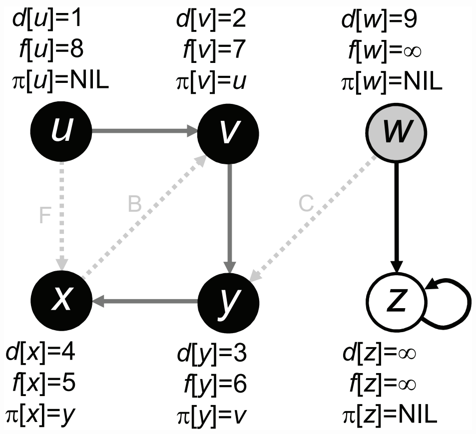


Procura em profundidade primeiro (DFS)

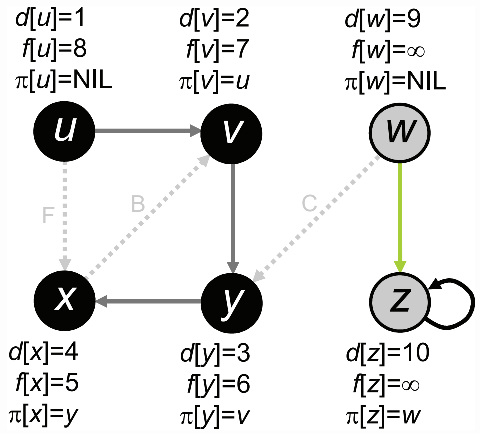
Exemplo



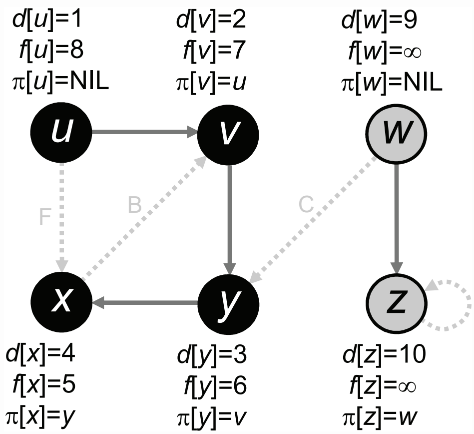
Exemplo



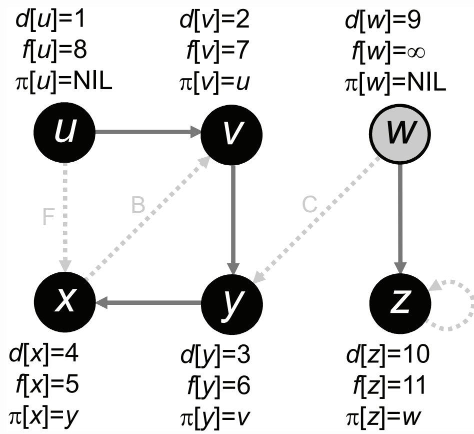
Exemplo



Exemplo

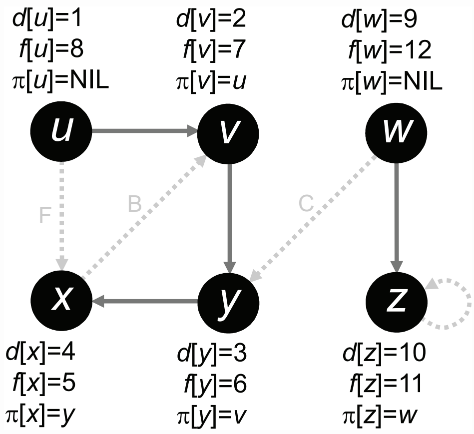


Exemplo



Procura em profundidade primeiro (DFS)

Exemplo



Procura em profundidade primeiro (DFS)

Complexidade

- Inicialização: $\Theta(V)$
- Chamadas a DFS-Visit dentro de DFS: $\Theta(V)$
- Arcos analisados em DFS-Visit: $\Theta(E)$
  - Chamadas a DFS-Visit dentro de DFS-Visit: $O(V)$
  - Mas $\sum_{v \in V} |Adj[v]| = \Theta(E)$

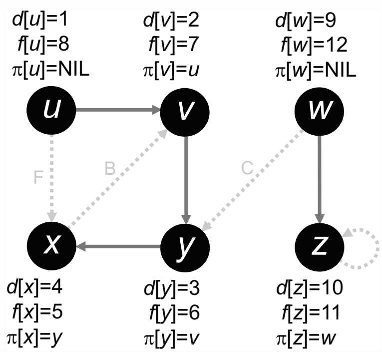
Tempo de execução: $\Theta(V + E)$

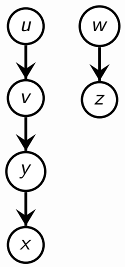
Procura em profundidade primeiro (DFS)

Resultado da DFS

Floresta Depth-First (DF)

- $G_\pi = (V, E_\pi)$
- $E_\pi = \{(\pi[v], v) : v \in V \wedge \pi[v] \neq NIL\}$
- Floresta DF composta por várias árvores DF





Procura em profundidade primeiro (DFS)

Propriedade: Estrutura de parêntesis

Se considerarmos que:

- $(u$ - representa a descoberta de u
- $u)$ - representa o fim de u

a história de descobertas e fim formam uma expressão bem formada com parêntesis aninhados

Exemplo

```
s z y x x y w w z s
( ( ( ( ) ) ( ) ) )
```

Procura em profundidade primeiro (DFS)

TÉCNICO LISBOA

Teorema dos parêntesis
Para qualquer DFS de $G = (V, E)$, para cada par de vértices u e v apenas um dos 3 casos seguintes é verdade:

- $[d[u], f[u]] \cap [d[v], f[v]] = \emptyset$
  u e v não são relacionados $\rightarrow$ u não ascendente/descendente de v
- $[d[u], f[u]] \subset [d[v], f[v]]$
  u é descendente de v na árvore DF
- $[d[v], f[v]] \subset [d[u], f[u]]$
  v é descendente de u na árvore DF

- $[d[u], f[u]]$ e $[d[v], f[v]]$ não se podem intersectar parcialmente

Procura em profundidade primeiro (DFS)

TÉCNICO LISBOA

Propriedade: **Classificação de arcos** (u, v)

- Arcos de árvore: (**tree edges**)
  - arcos na floresta DF, G_π
  - (u, v) é arco de árvore se v foi visitado devido ao arco (u, v) ser visitado
- Arcos para trás: (**back edges**)
  - ligam vértice u a vértice v antecessor na mesma árvore DF
- Arcos para a frente: (**forward edges**)
  - ligam vértice v a vértice descendente na mesma árvore DF
- Arcos de cruzamento: (**cross edges**)
  - na mesma árvore DF, se u (ou v) não antecessor de v (ou u)
  - ou entre árvores DF diferentes

Procura em profundidade primeiro (DFS)

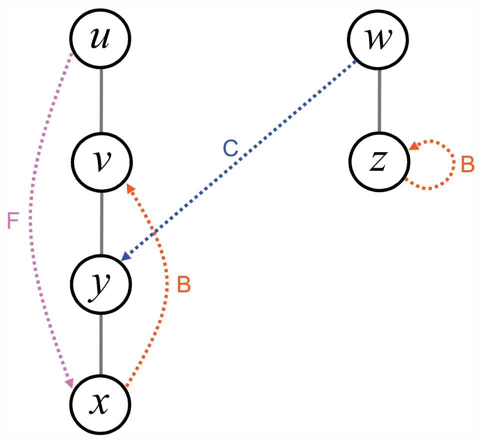
TÉCNICO LISBOA

Propriedade: **Classificação de arcos** (u, v)

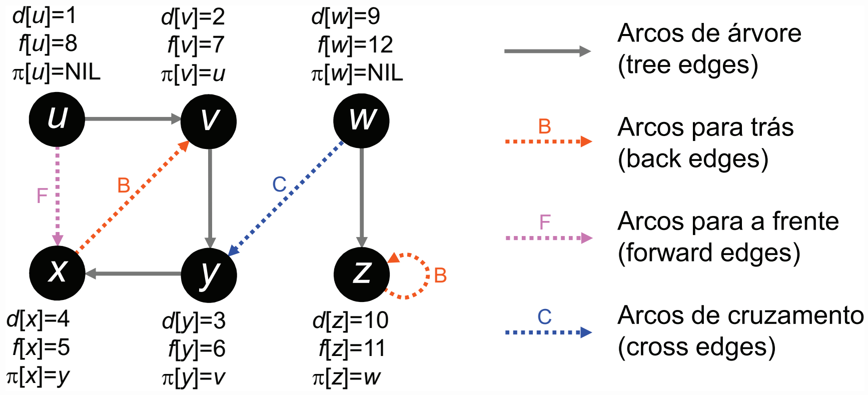
- Arcos de árvore: (**tree edges**)
  - $d[u] < d[v] < f[v] < f[u]$
  - $color[v] = white$ quando (u, v) é analisado
- Arcos para trás: (**back edges**)
  - $d[u] < d[v] < f[v] < f[u]$
  - $color[u] = gray$ quando (v, u) é analisado
- Arcos para a frente: (**forward edges**)
  - $d[u] < d[v] < f[v] < f[u]$
  - $color[v] = black$ quando (u, v) é analisado
- Arcos de cruzamento: (**cross edges**)
  - $d[v] < f[v] < d[u] < f[u]$
  - $color[v] = black$ quando (u, v) é analisado

Procura em profundidade primeiro (DFS)

TÉCNICO LISBOA



Procura em profundidade primeiro (DFS)



Procura em profundidade primeiro (DFS)

Propriedade

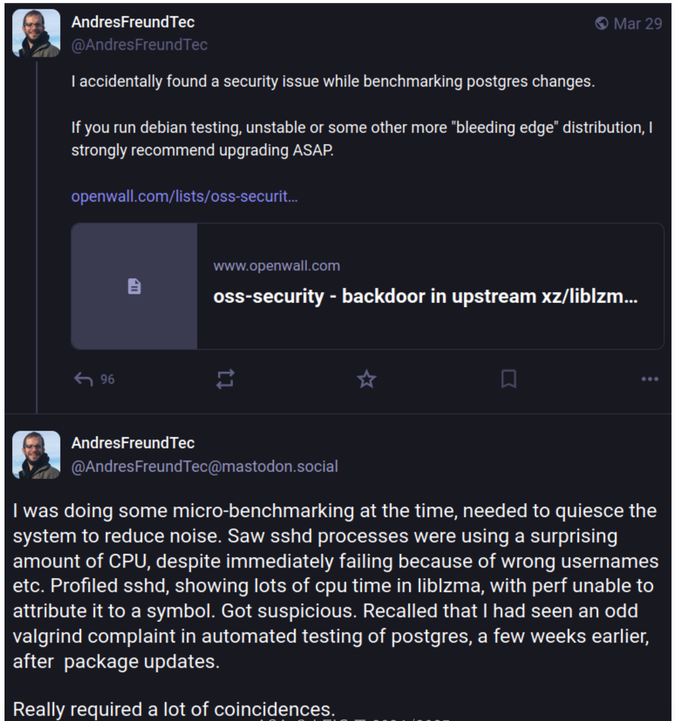
Dado $G = (V, E)$ **não dirigido**, cada arco é arco de árvore ou para trás i.e., não existem arcos para a frente e de cruzamento

Teorema caminho branco

Numa floresta DF (grafo dirigido ou não dirigido):

v descendente de $u \Leftrightarrow$ existe caminho de vértices brancos de u para v

- Qualquer vértice w descendente de u verifica $[d[w], f[w]] \subset [d[u], f[u]]$, pelo que w é branco quando u é descoberto


AndresFreundTec
@AndresFreundTec
Mar 29

I accidentally found a security issue while benchmarking postgres changes.

If you run debian testing, unstable or some other more "bleeding edge" distribution, I strongly recommend upgrading ASAP.

openwall.com/lists/oss-securit...

www.openwall.com
oss-security - backdoor in upstream xz/liblzm...

96

AndresFreundTec
@AndresFreundTec@mastodon.social

I was doing some micro-benchmarking at the time, needed to quiesce the system to reduce noise. Saw sshd processes were using a surprising amount of CPU, despite immediately failing because of wrong usernames etc. Profiled sshd, showing lots of cpu time in liblzma, with perf unable to attribute it to a symbol. Got suspicious. Recalled that I had seen an odd valgrind complaint in automated testing of postgres, a few weeks earlier, after package updates.

Really required a lot of coincidences.

Questões?

Dúvidas?