CI218: Gerenciamento de Transações

Profa. Carmem Hara

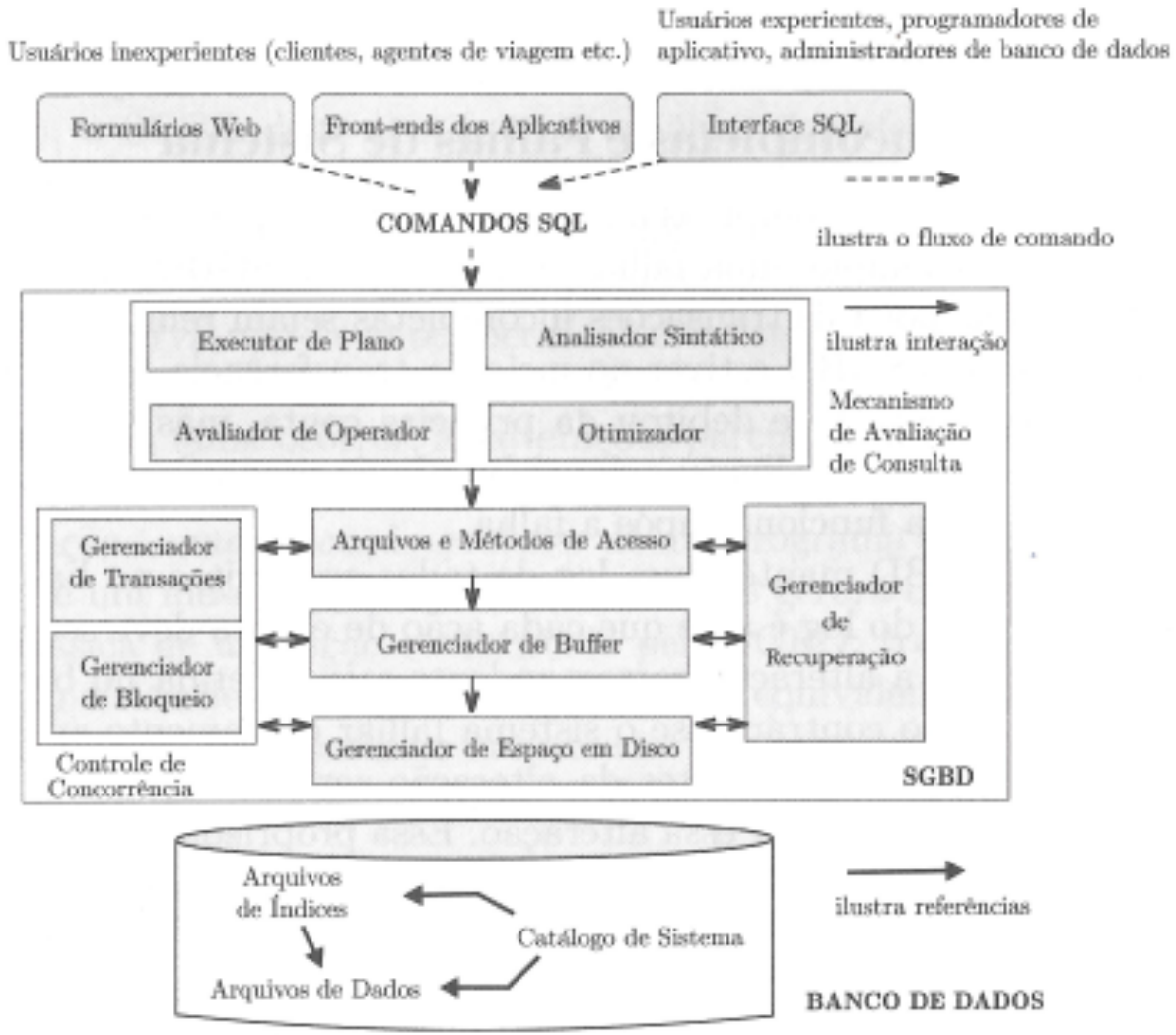
Departamento de Informática/UFPR

29 de outubro de 2023

Transações

- Arquitetura de um SGBD
- Transações
- Propriedades ACID
- Controle de concorrência
- Gerenciador de recuperação
- Escalonamentos
- Seriabilidade de escalonamentos

Arquitetura de um SGBD



Transações

Motivação: desempenho

- acessos a disco são lentos: é importante manter a CPU ocupada executando vários programas/transações concorrentemente

Transação

- conceito básico para controle de concorrência e recuperação
- uma sequência de operações que são consideradas uma unidade atômica (indivisível) de trabalho

Operações no SGBD

Operações elementares:

- leitura (read)
- escrita (write)

Operações especiais:

- iniciar transação (begin)
- efetivar transação (commit)
- cancelar transação (abort)

Exemplo

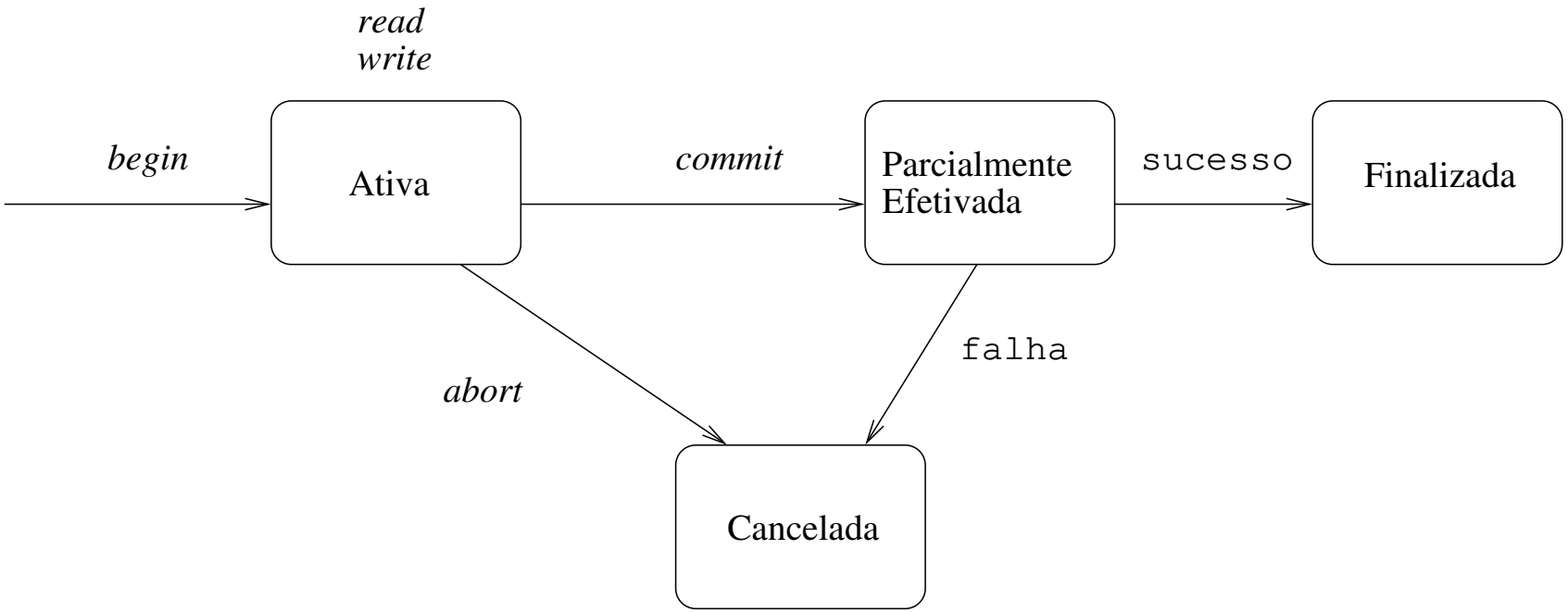
T1: transferência de R$100 da conta X para conta Y

T2: depósito de R$200 na conta X

```
  T1                T2
---------         ---------
begin               begin
read(X)             read(X)
X:=X-100            X:=X+200
write(X)            write(X)
read(Y)             commit
Y:=Y+100
write(Y)
commit
```

As operações que compõem T1 e T2 podem ser intercaladas, desde que o **efeito visível final seja o equivalente a executar as transações serialmente** (T1 depois T2) ou (T2 depois T1).

Estados da transação



Propriedade ACID

Propriedades desejáveis de uma transação

- **Atomicidade**: uma transação é realizada integralmente ou não é realizada de modo algum
- **Consistência**: uma transação começa com o BD em um estado consistente e termina com o BD em estado consistente
- **Isolamento**: o resultado de cada transação deve ser como se estivesse executando sozinha no sistema, ou seja, isoladada de outras transacoes
- **Durabilidade**: se uma transação é terminada com sucesso (committed), as alterações devem persistir no BD

As propriedades ACID e os componentes do SGBD

- Controle de concorrência
  - Garante consistência e isolamento
  - Gerencia como as operações das transações podem ser intercaladas
- Gerenciador de recuperação (de falhas)
  - Garante atomicidade e durabilidade
  - Tipos de falhas
    - falha do computador: erro de hardware ou software
    - erro de transação ou de sistema: divisão por zero
    - condições de exceção detectados pela transação: saldo insuficiente
    - imposição do controle de concorrencia: *deadlock*
    - falha de disco
    - problemas físicos e catástrofes

Escalonamento de Transações

Um **escalonamento** S de n transações $T_1, T_2, ..., T_n$ é um ordenamento das operações das transações sujeitas à restrição de que, para cada transação T_i que participa em S, as operações de T_i em S devem aparecer na mesma ordem em que ocorrem em T_i.

$\rightarrow$ Não pode mudar a ordem na qual as operações são executadas em cada transação

$\rightarrow$ Os escalonamentos devem preservar as propriedades ACID.

Operações conflitantes

Duas operações em um escalonamento são **conflitantes** se:

- pertencem a transações diferentes
- acessam o mesmo item X
- pelo menos uma das operações é um `write`

Problema: Perda de Atualização

```
Tempo    T1          T2
-----  ---------   -----------
  1    read(X)
  2    X:=X-100
  3                read(X)
  4                X:=X+200
  5    write(X)
  6    read(Y)
  7                write(X)
  8    Y:=Y+100
  9    write(Y)
```

- Violação das propriedades: isolamento, consistência
- Conflito Write-Write - Atualização perdida

Problema: Atualização Temporária

```
   T1          T2
--------   -----------
 read(X)
 X:=X-100
 write(X)
              read(X)
              X:=X+200
              write(X)
              commit
 abort
```

- Violação das propriedades: isolamento, consistência, durabilidade
- Conflito Write-Read - Leitura suja

Problema: Agregação incorreta (soma, resumo)

```
  T1              T3
---------    ------------
 read(X)
 X:=X-200
 write(X)
                read(X)
                read(Y)
                soma:= X + Y
                print( soma )
 read(Y)
 Y:=Y+200
 write(Y)
```

- Violação da propriedade: isolamento
- Conflito Read-Write - Leituras não repetíveis

Tipos de Escalonamentos - Recuperável

Um escalonamento é **recuperável** se nunca for necessário desfazer uma transação efetivada.

- Condição: uma transação T não pode ser efetivada até que todas as transações T' que tenham gravado um item lido por T tenham sido efetivadas.

Tipos de Escalonamentos - Recuperável

```
Escalonamento S1
    T1          T2
--------------------
 read(X)
             read(X)
 write(X)
 read(Y)
             write(X)
             commit
 write(Y)
 commit
```

```
Escalonamento S2
   T3              T4
----------------------
 read(X)
 write(X)
                 read(X)
 read(Y)
                 write(X)
                 commit
 abort
```

S1 é **recuperável**, mas tem o problema de atualização perdida.

S2 **não** é recuperável

Escalonamento sem Cancelamentos em Cascata

```
   T1                T2
-----------------------
 read(X)
 write(X)
                   read(X)
 read(Y)
                   write(X)
 write(Y)
 abort
                   abort
```

- Condição: não pode ter leitura suja
- read(X) de T2 tem que ser adiado até a efetivação de T1

Escalonamento Estrito

Um escalonamento é **estrito** se nenhuma transação puder ler e nem gravar um item X até que a última transação que gravou X seja efetivada.

Escalonamento Serial

Um escalonamento é **serial** se, para todas as transações T participantes do escalonamento, todas as operações de T forem executadas consecutivamente no escalonamento; caso contrário, o escalonamento é denominado não-serial.

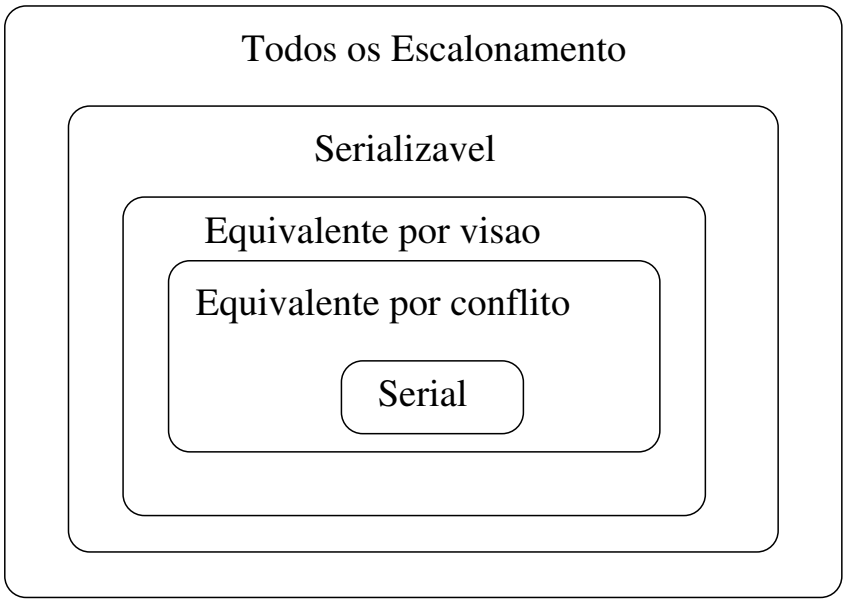
- possui somente uma transação ativa de cada vez
- não permite nenhum entrelaçamento de transações
- é considerado correto, independente da ordem de execução das transações
- limita a concorrência
- na prática, é inaceitável

Escalonamento Serializável

Um escalonamento é **serializável** se for equivalente a um escalonamento serial.

Diferentes critérios de equivalência:

- Equivalente por conflito
- Equivalente por visão



Equivalência por Conflito

Um escalonamento é **serializável por conflito** se a ordem das operações conflitantes é a mesma de um escalonamento serial.

Escalonamento S1

T1	T2
read(X)	
	read(X)
write(X)	
read(Y)	
	write(X)
write(Y)	

Escalonamento S2

T1	T2
read(X)	
write(X)	
	read(X)
	write(X)
read(Y)	
write(Y)	

S1 não é serializável por conflito

S2 é serializável por conflito

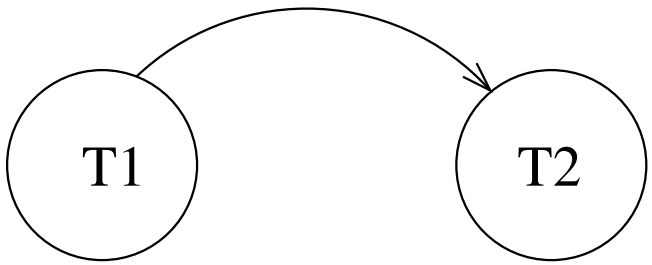
Teste: Serialização por Conflito

Utilizando um **grafo de dependências**
Algoritmo:

1. Crie um vértice para cada transação no escalonamento
2. Para cada transação conflitante op_i da transação T_i e op_j da transação T_j, se op_i é executada antes de op_j no escalonamento então crie uma aresta de T_i para T_j
3. se houver um ciclo no grafo então o escalonamento não é serializável por conflito; caso contrário, o escalonamento é equivalente a um escalonamento serial

Exemplo: Grafo de Dependências

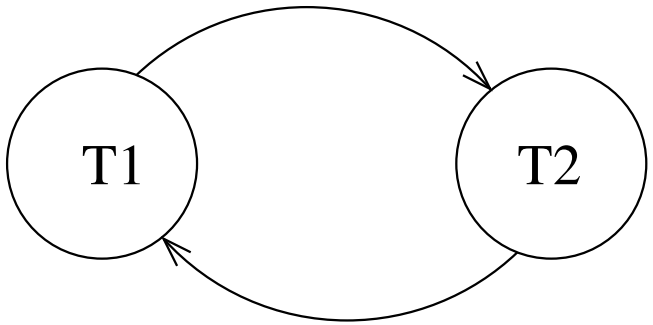
```
  T1            T2
--------------------
read(X)
write(X)
              read(X)
read(Y)
              write(X)
write(Y)
              read(Y)
              write(Y)
```



- Esta é a condição que é tipicamente assegurada pelos SGBDs
- Ela e' suficiente (mas não necessária) para que o escalonamento seja serializável.

Exemplo: Escalonamento Serializável

```
  T1              T2
--------------------
read(X)
X:=X-50
write(X)
                read(Y)
                Y:=Y-10
                write(Y)
read(Y)
Y:=Y+50
write(Y)
                read(X)
                X:=X+10
                write(X)
```



- O escalonamento não é serializável por conflito
- É equivalente a um escalonamento serial

Escalonamento Serializável por Visão

Dois escalonamentos S e S' sao equivalentes por visão se:

- Se a transação T_i lê o valor inicial do item de dado X em S, ele também lê o valor inicial de X em S'
- Se T_i lê o valor de X escrito por T_j em S, o mesmo acontece em S'
- Se T_i é o último a gravar X em S, ele tambem é o último a gravar Xem S'

Um escalonamento é **serializavel por visão** se ele é equivalente por visão a um escalonamento serial.

Material sobre serialização por visão: *Database Systems: The Complete Book (DS:CB)*, de Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom, 1^a edição

Exemplo: Escalonamento Serializável por Visão

```
  T1          T2          T3
-------------------------------
read(X)
            write(X)
write(X)
                        write(X)
```

- Equivalente ao escalonamento serial $< T1, T2, T3 >$
- Um escalonamento serializável por conflito sempre é serializável por visão
- Um escalonamento serializável por visão nem sempre é serializável por conflito

Exemplo: Escalonamento Serializável por Visão

```
  T1          T2          T3          T4
-------------------------------------------
              read(A)
read(A)
write(C)
                          read(C)
write(B)
                                      read(B)
                          write(A)
                                      read(C)
              write(D)
              read(B)
                                      write(A)
                                      write(B)
```

- Equivalente ao escalonamento serial $< T1, T2, T3, T4 >$

Escalonamento Serializável por Visão - Definição Alternativa

- Considere que existe uma transação T_0 que escreve todos os itens de dados
- Considere que existe uma transação T_f que lê todos os itens de dados
- Condição para dois escalonamentos S e S' serem equivalentes por visão: Se T_i lê o valor de X escrito por T_j em S, o mesmo acontece em S'

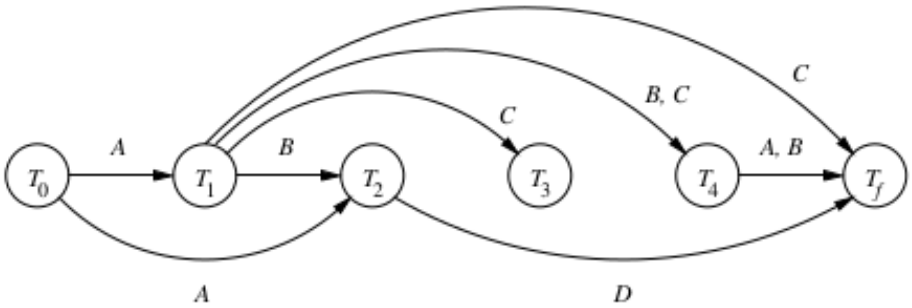
Algoritmo: Escalonamento Serializável por Visão

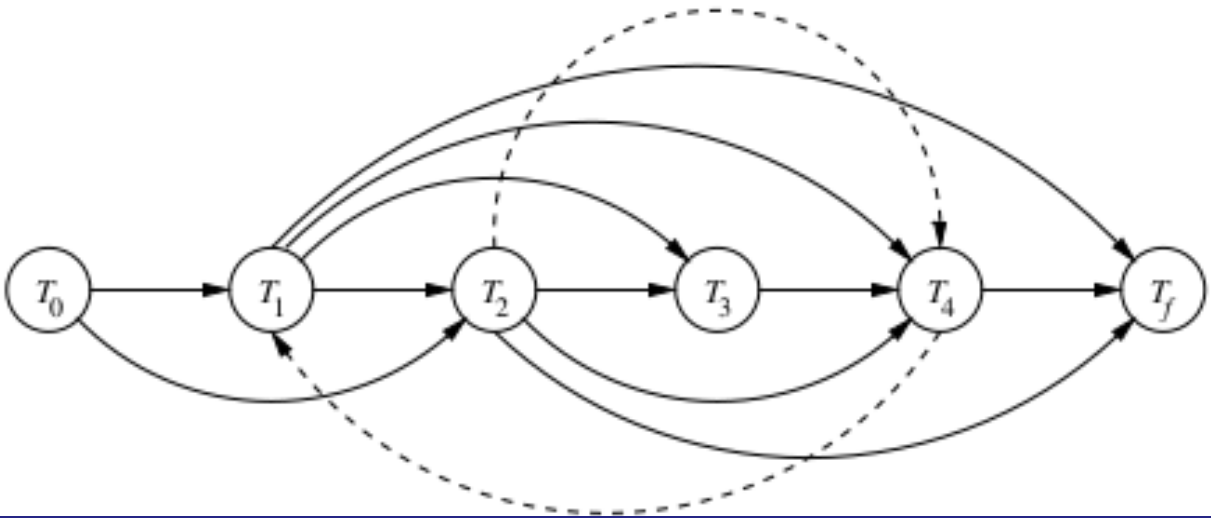
1. Crie um vértice para T_0, T_f e para cada transação no escalonamento
2. Se a transação T_j lê um item de dado X escrito pela transação T_i e $i \neq j$, crie uma aresta de T_i para T_j
3. Se uma transação T_k ($k \neq i$ e $k \neq j$) também escreve o mesmo item de dado X, crie um **par** de arestas de T_k para T_i e de T_j para T_k
   - Se T_i for T_0 não crie a aresta de T_k para T_0
   - Se T_j for T_f não crie a aresta de T_f para T_k
4. Considere os grafos gerados considerando **apenas um elemento de cada par de arestas** do passo anterior. Se em todos os grafos houver um ciclo, então o escalonamento não é serializável por visão; caso contrário, o escalonamento é equivalente a um escalonamento serial

Exemplo: Escalonamento Serializável por Visão

```
  T1          T2          T3          T4
-------------------------------------------
              read(A)
read(A)
write(C)
                          read(C)
write(B)
                                      read(B)
                          write(A)
                                      read(C)
              write(D)
              read(B)
                                      write(A)
                                      write(B)
```

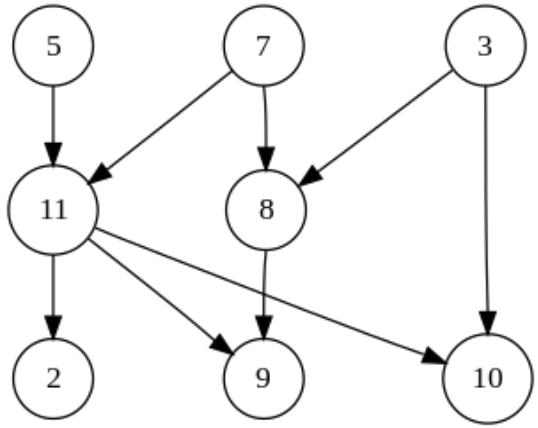
Passos 1 e 2:





Escalonamento Serial Equivalente

Ordenação Topológica do grafo acíclico direcionado (DAG): ordenação linear dos vértices, na qual cada vértice aparece antes de seus descendentes



Várias ordenações topológicas possíveis:

- 5, 7, 3, 11, 8, 2, 9, 10
- 3, 5, 7, 8, 11, 2, 9, 10
- 5, 7, 3, 8, 11, 10, 9, 2
- 7, 5, 11, 3, 10, 8, 9, 2
- 5, 7, 11, 2, 3, 8, 9, 10
- 3, 7, 8, 5, 11, 10, 2, 9

Algoritmo de Ordenação Topológica

Algoritmo de Kahn (1962): gera a ordenação (L) a partir dos vértices sem arestas incidentes (S)

```
L:= vazio;
S:= vértices sem arestas incidentes;

enquanto S não estiver vazio faça
        remova um vértice n de S
        insira n em L
        para cada vértice m com uma aresta e = (n, m) faça
                remova a aresta e do grafo
                se m não tiver mais arestas incidentes então
                        insira m em S

se o grafo não estiver vazio então
        o grafo contém pelo menos um ciclo
senão
        returne L
```