Fundamentos de Linguagens de Programação

Python: Memória e Garbage Colector

Nelson Carvalho Sandes

Centro de Ciências Tecnológicas - CCT
Universidade Federal do Cariri

2022

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Tópicos

1. Contextualização
2. Stack, Heap e Objetos
3. Garbage Colector
   - Contador de Referências
   - Rastreamento

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Tópicos

1 Contextualização

2 Stack, Heap e Objetos

3 Garbage Colector
- Contador de Referências
- Rastreamento

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Descritores

- Um descritor é responsável por armazenar um conjunto de atributos de uma variável.
- Algumas Strings são classificadas de acordo com o tamanho e os seus descritores armazenam informações relevantes.
  1. Uma String de tamanho dinâmico e limitado possui um descritor que informa o seu tamanho atual, tamanho máximo e endereço.
  2. O descritor de uma String de tamanho estático precisa armazenar apenas o seu endereço e o seu tamanho.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Memória

- Vimos nas aulas anteriores que a medida que uma função é executada, suas variáveis são alocadas na memória **stack**. Após a execução da função, as variáveis são desalocadas da memória.
- Também vimos que é possível alocar variáveis na memória **Heap**.
- Em C e C++ é necessário desalocar explicitamente as variáveis dessa região. Porém, em algumas linguagens, o desalocamento dessas variáveis é feito de forma automática pelo **Garbage Colector**.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Alocação de Memória

- Vimos detalhadamente como funciona a alocação e desalocação de memória na linguagem C, que é compilada e não possui Garbage Colector.
- Apesar disso, a dinâmica de alocação e desalocação de memória em outras linguagens pode ser diferente.
- Nessa aula, veremos o comportamento da Stack e da Heap em Python. Além de ver como funciona o mecanismo de Garbage Colector nessa linguagem.

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Tópicos

1 Contextualização

2 Stack, Heap e Objetos

3 Garbage Colector
- Contador de Referências
- Rastreamento

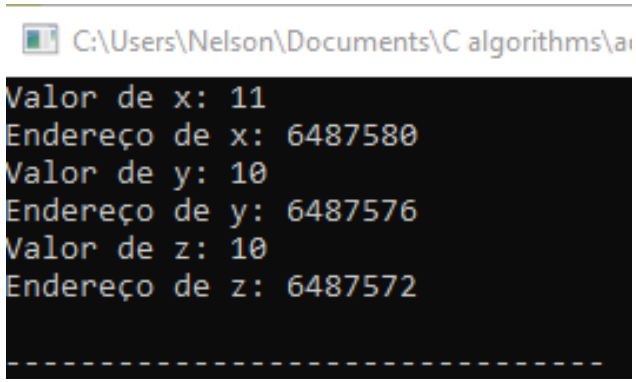
UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Memória Stack

Exemplo em C

```c
#include <stdio.h>
#include <locale.h>

int main(){
    setlocale(LC_ALL,"Portuguese");
    int x = 10;
    int y = x;
    x = x + 1;
    int z = 10;
    printf("Valor de x: %d \n", x);
    printf("Endereço de x: %d \n", &x);
    printf("Valor de y: %d \n", y);
    printf("Endereço de y: %d \n", &y);
    printf("Valor de z: %d \n", z);
    printf("Endereço de z: %d \n", &z);
    return 0;
}
```



UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Memória Stack e Heap

- Python é uma linguagem em que todos os seus tipos são objetos.
- Como vimos em aulas anteriores, geralmente objetos são alocados na memória Heap.
- Vamos ver a execução de um código em Python similar ao código do slide anterior em C e ver o funcionamento da memória.

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Memória Stack e Heap

Exemplo em Python

```python
if __name__ == "__main__":
    x = 10
    y = 10
    x = x + 1
    z = 10
    print("----------------------")
    print(f"Valor de x: {x}")
    print(f"Valor de y: {y}")
    print(f"Valor de z: {z}")
    print(f"Endereço do objeto na heap que x faz referência: {id(x)}")
    print(f"Endereço do objeto na heap que y faz referência: {id(y)}")
    print(f"Endereço do objeto na heap que z faz referência: {id(z)}")
```

```
Valor de x: 11
Valor de y: 10
Valor de z: 10
Endereço do objeto na heap que x faz referência: 140704056483904
Endereço do objeto na heap que y faz referência: 140704056483872
Endereço do objeto na heap que z faz referência: 140704056483872
```

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Memória Stack e Heap

Exemplo em C (Funções)

```c
#include <stdio.h>
#include <locale.h>

int f2(int x) {
    x = x + 1;
    return x;
}

int f1(int x) {
    x = x*2;
    int y = f2(x);
}

int main(){
    setlocale(LC_ALL,"Portuguese");
    int y = 5;
    int z = f1(y);
    printf("%d \n", z);
    return 0;
}
```

- Na linguagem C, ao chamar as funções, as variáveis são alocadas na memória Stack.
- Cada variável nessa região possui um valor associado que é associado a ela.
- Esse valor é na própria região Stack.

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Memória Stack e Heap

Exemplo em Python (Funções)

```
def f2(x):
    x = x + 1
    return x

def f1(x):
    x = x * 2
    y = f2(x)
    return y

if __name__ == "__main__":
    y = 5
    z = f1(y)
    print(z)
```

- Em Python, as variáveis também são alocadas na Stack a medida que chamamos funções.
- Apesar disso, é importante salientar que as variáveis na Stack fazem referências para objetos criados na memória Heap.
- Vamos executar o código ao lado passo à passo.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Memória Stack e Heap

Exemplo em Python (Objetos)

```python
class Pessoa:
    def __init__(self, idade, altura):
        self.idade = idade
        self.altura = altura

if __name__ == "__main__":
    p = Pessoa(35, 1.69)
```

- Vamos ver a execução desse código.
- Nesse caso, o programador cria a classe Pessoa e, na função principal, cria uma variável p na Stack.
- A variável 'p' faz referência à um objeto pessoa alocado na memória Heap.

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Objetos

Tipo de objetos em Python

Simples

- Inteiros
- Ponto flutuantes
- Booleanos
- Strings

Containers

- Listas
- Dicionários
- Classes definidas por usuários.

Tópicos

1 Contextualização

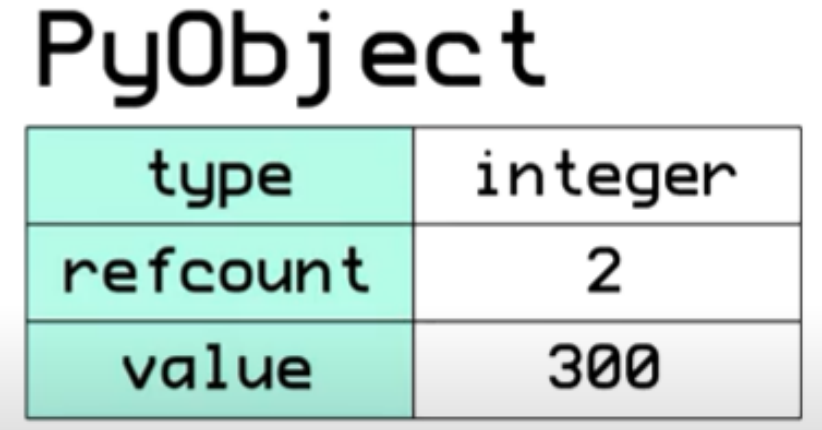
2 Stack, Heap e Objetos

3 Garbage Colector
- Contador de Referências
- Rastreamento

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

PyObjects

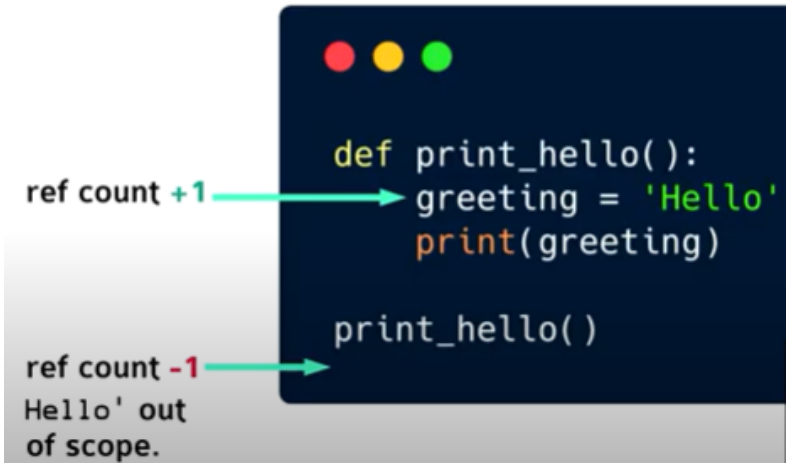


- Para cada variável em Python, é criado um PyObject.
- Um PyObject trabalha como um descritor. Ele armazena o tipo da variável, o seu valor e a quantidade de referências que apontam para ele.
- Suponha que tenhamos o código:
  - x = 300
  - y = 300

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Contador de referências: Exemplo



- Vamos ver esse exemplo simples para entender a contagem de referências.
- Nesse caso, ao chamar a função, o contador da string 'Hello' vai para 1. Como a variável de referência é desalocada no término da função, o contador volta para 0.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Exemplo em Python (Funções)

```
def f2(x):
    x = x + 1
    return x

def f1(x):
    x = x * 2
    y = f2(x)
    return y

if __name__ == "__main__":
    y = 5
    z = f1(y)
    print(z)
```

- Vamos ver mais uma vez a execução desse exemplo.
- Dessa vez, vamos ficar atentos à contagem de referências das variáveis.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Contador de referências

- Uma das formas que o Garbage Colector utiliza para remover objetos da Heap é verificar se o **contador de referências** é 0.
- Nesse caso, como não existe nenhuma referência ao objeto, ele pode ser removido da memória Heap sem prejuízo para a aplicação.
- É uma estratégia de fácil implementação, mas tem algumas desvantagens.

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Contador de referências: Desvantagens

1. Ocupa espaço extra para armazenar os PyObjects.
2. Faz mais execuções, já que para cada atribuição é preciso atualizar o contador de referência.
3. Não consegue detectar **referências cíclicas** (que podem ocorrer com **objetos containers**) e removê-las da memória Heap.
4. Para entender sobre **referências cíclicas**, vamos aprender outras formas de decrementar o contador de referências de um objeto.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

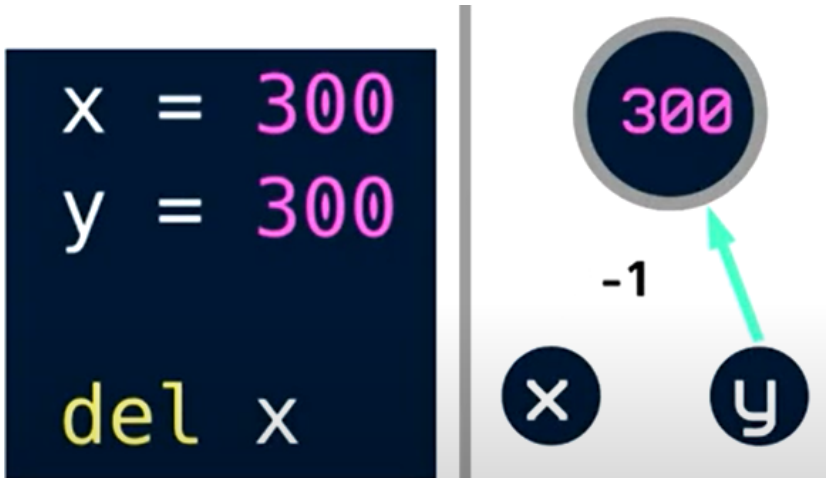
Atribuir outros valores para variáveis

- Nessa situação, o objeto 300 inicia com 2 referências apontado para ele.
- Quando outros valores são atribuidos para as variáveis, o contador de 300 cai para 0.

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Instrução del



- A instrução **del** remove a referência de um objeto.
- Nesse caso, o objeto 300 inicia a execução com duas referências.
- Após a instrução **del**, x deixa de referenciar 300 e o contador passa a ter valor 1.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Instrução del

```
class No:
    def __init__(self, valor):
        self.valor = valor
        self.next = None

    def setNext(self, next):
        self.next = next;


if __name__ == "__main__":
    raiz = No("raiz")
    no1 = No("no1")
    no2 = No("no2")
    raiz.setNext(no1)
    no1.setNext(no2)
    no2.setNext(no1)
    del raiz
    del no1
    del no2
```

- Agora que sabemos da instrução **del**, vamos ver como fica a memória heap nesse caso.
- Apesar de não ter como acessar no1 e no2, os contadores de referência deles estão acima de 0.
- Como o Garbage Colector age nessa situação?

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Rastreamento: introdução

1. Para tratar as referências cíclicas, o Garbage Colector precisa executar algoritmos para identificar essas referências.
2. Pode ser muito custoso executar esse algoritmo em todos objetos. Por conta disso, Python adota algumas estratégias para focar em apenas alguns.
3. A linguagem classifica objetos **containers** em gerações e parte do princípio que esses objetos "morrem jovens".

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Rastreamento: gerações

- Python mantém três listas de gerações:
  - Geração 0
  - Geração 1
  - Geração 2
- Inicialmente, todos os objetos **containers** são colocados na geração 0 (Vale salientar que apenas objetos com contador acima de 0 e containers são alocados em gerações).

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Rastreamento: execução

- Depois da criação da lista de gerações, o Garbage Colector é acionado quando uma geração ultrapassa um valor pré-determinado (**threshold**);
- Quando ele é acionado, os seguintes passos são executados:
  1. Uma lista de descarte é criada.
  2. Referências cíclicas são detectadas e os objetos que fazem parte dessas referências são adicionados na lista de descarte.
  3. Os objetos na lista de descarte são liberados da memória.
  4. Os objetos que sobreviverem passam para a próxima geração (menos os da geração 2).
- O valor do *threshold* pode aumentar a medida que objetos que não precisam ser descartados acumulam na memória.

UFCA UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

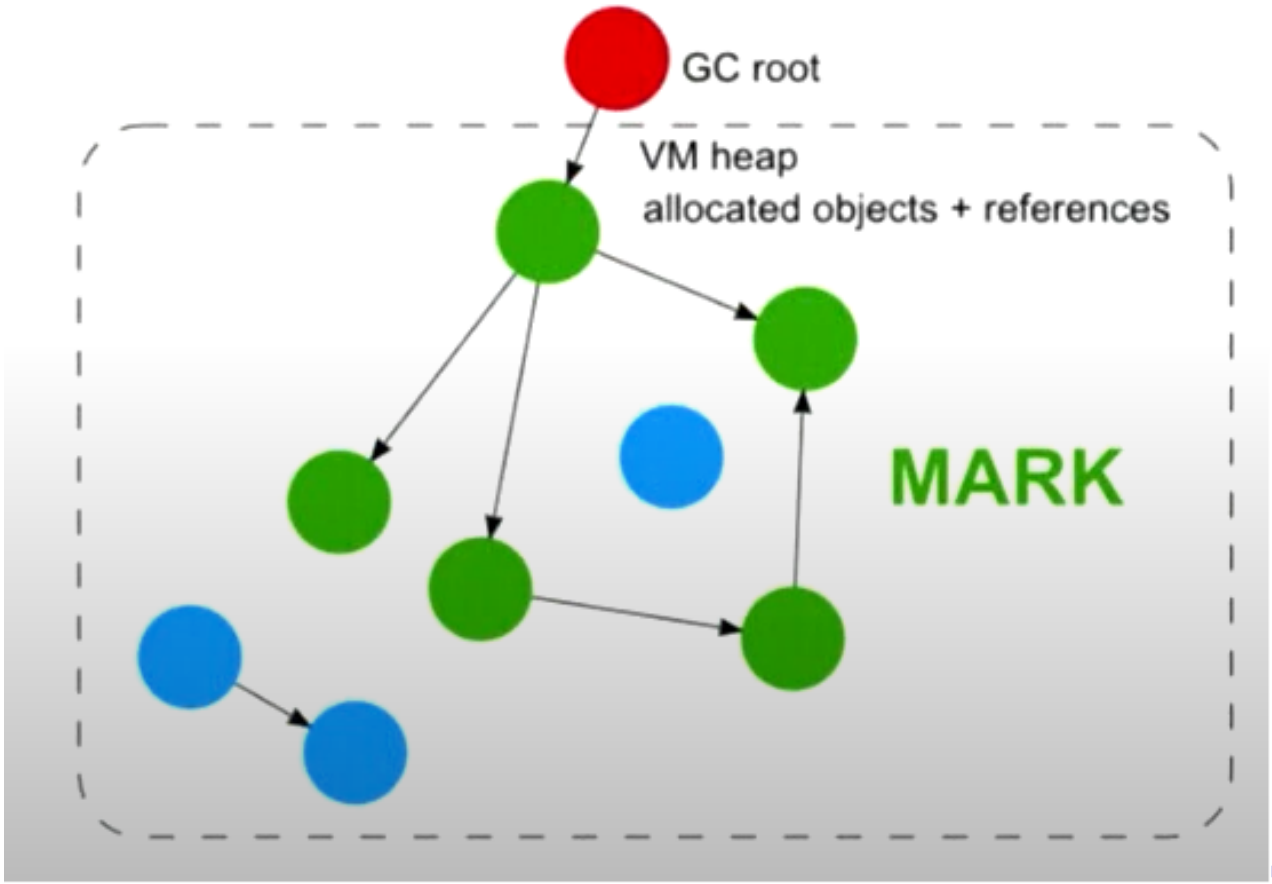
Rastreamento: Mark and Sweep

- Como detectar as referências cíclicas?
- Pode existir diversas abordagens.
- Vamos ver uma abordagem simples de forma intuitiva. Ela se chama **mark and sweep** (marcar e varrer).

UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

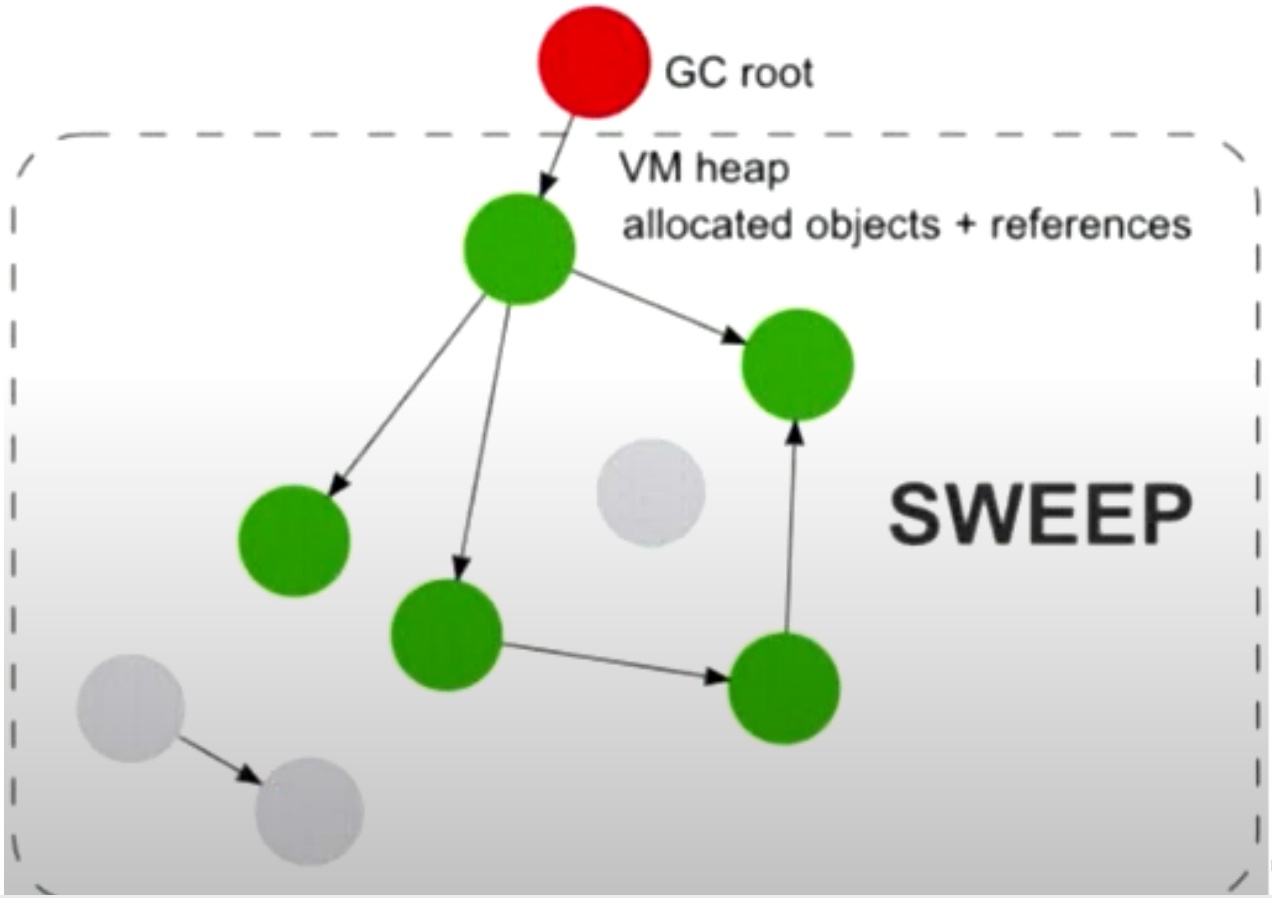
Rastreamento: Mark and Sweep



UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

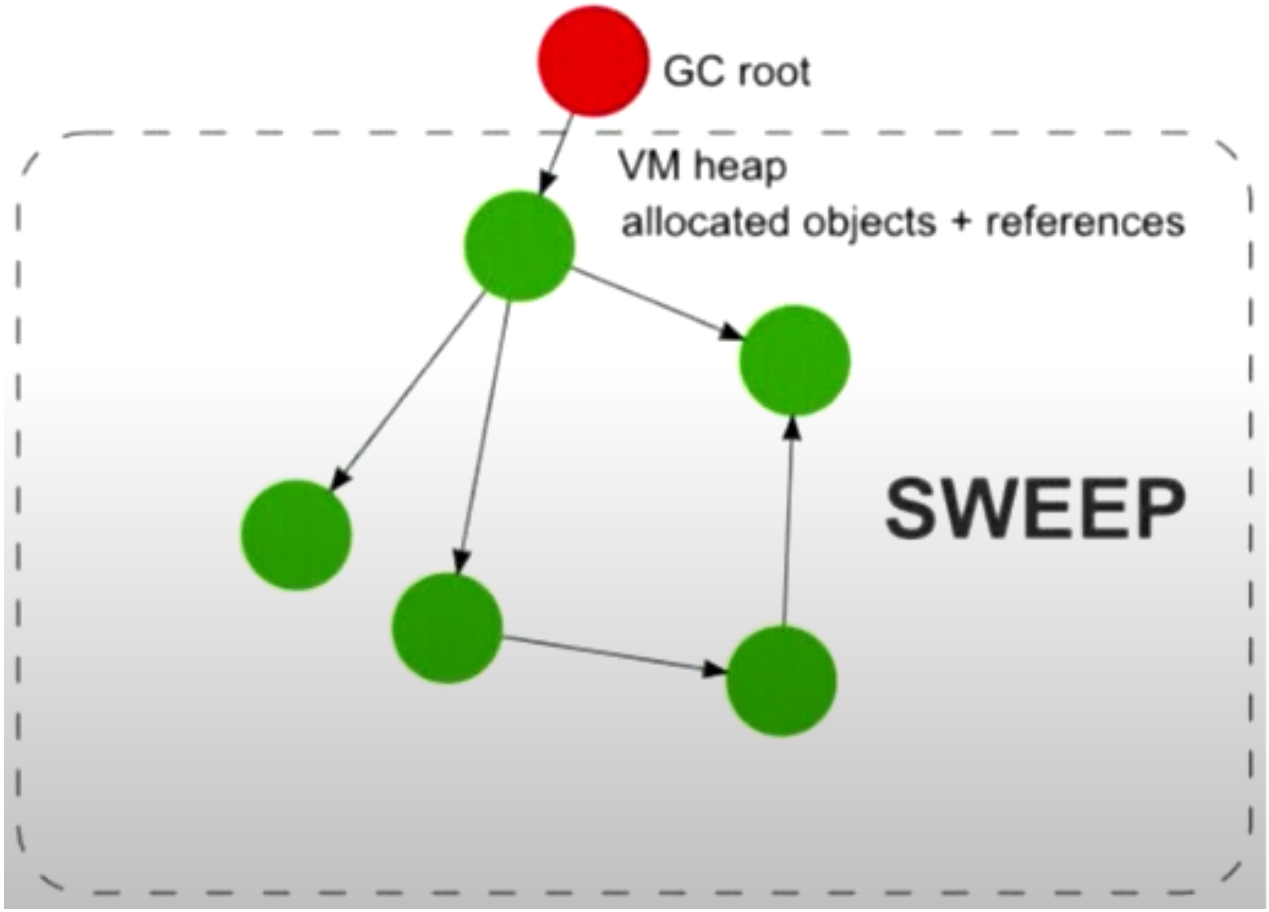
Rastreamento: Mark and Sweep



UFCA
UNIVERSIDADE FEDERAL DO CARIRI

Garbage Colector

Rastreamento: Mark and Sweep



UFCA
UNIVERSIDADE FEDERAL DO CARIRI