Systems Programming

Spring 2023

7 주차

- Q&A
  - 08-allocation-basic
  - 09-allocation-advanced

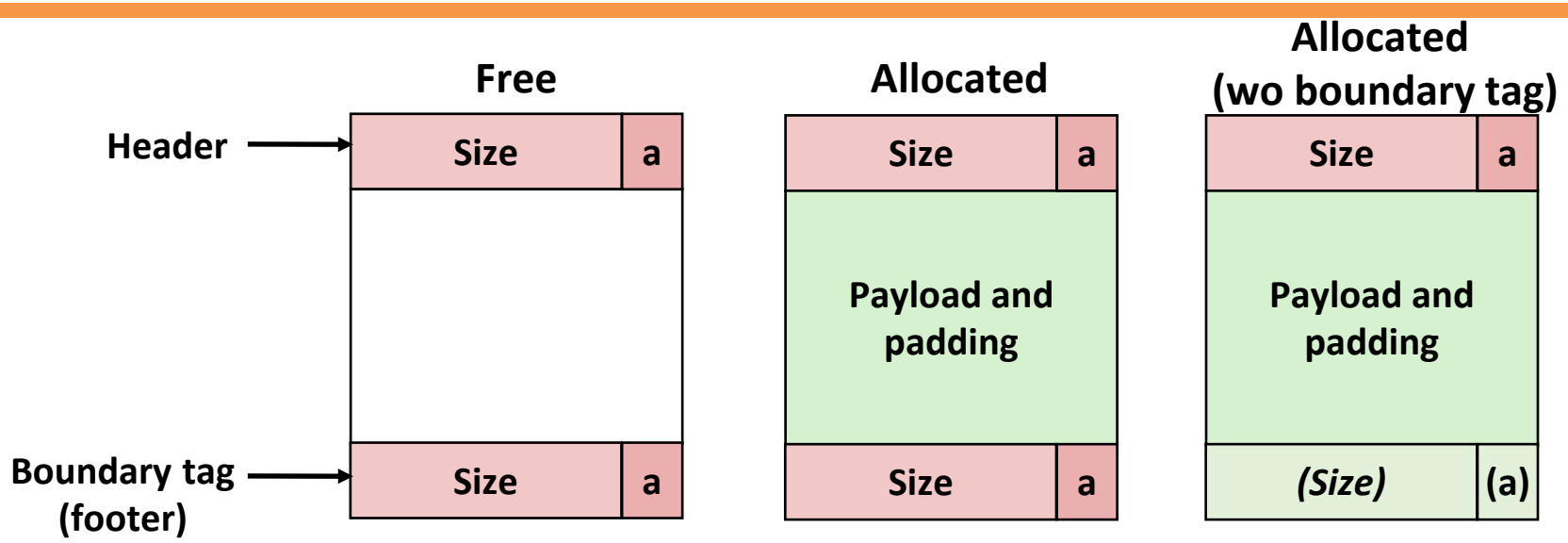
08-allocation-basic

Questions

Implicit List: Bidirectional Coalescing에서 boundary tag가 doubly linked list와 같은 역할을 하여, 특정 block을 free할때 previous/next block을 모두 확인하여 합쳐야할지 여부를 결정할 수 있게 한다는 점은 이해하였습니다. 하지만 08-31쪽에서 free block에만 boundary tag를 포함시켜 성능을 최적화할 수 있다는 설명을 해주셨는데, 이경우 기존에 할당되었던 block을 free할때 previous block을 향하는 포인터가 없어 이전 block과 합쳐야 할지를 결정할 수 없는 상황이 되는 것이 아닌지 혼동됩니다. 관련하여, allocate block에 boundary tag를 사용하지 않을 경우 오류가 날 확률에 대해서도 간략하게 강의에서 설명해주신 것 같은데, 이와 관련하여 조금만 추가적으로 설명해주시면 감사하겠습니다!

allocation - basic의 p.31에서 질문이 있습니다.
allocated block의 footer가 필요없는 이유는, 만약 어떤 블록을 할당해제할때
1. 바로 앞 블록의 footer를 보고 header로 갔을 때 그 앞 블록이 해제되어있던 블록이라면 실제로 우리가 확인한 부분이 각각 footer, header가 맞고 그 값이 같을 거라는 의미이고,
2. 그 앞 블록이 할당되어있던 블록이라면 우리가 footer라고 생각하고 참조했던 부분은 사실 그냥 프로그램의 데이터이고 따라서 거기서 비트연산으로 블록사이즈를 구해 참조한, 우리가 header라고 생각했던 부분도 그냥 무작위의 메모리 공간이며, 이 둘이 같을 확률은 사실상 0이다.
그러므로 할당되어있는 블록은 footer를 넣지 않고 그 공간을 그냥 활용해도된다.. 그래도 coalescing할 때 구분 가능하다는게 결론이라 이해했는데 잘 이해한건지 궁금합니다.

Questions



- If block is free,
  - allocated(a) = 0 && data(header) == data(boundary tag)
- If block is allocated & without boundary tag
  - If area(allocated) == 1, block is allocated
  - If area(allocated) == 0,
    - area(Size) should be aligned value
    - and area(boundary tag) should be the same value as data(header)

09-allocation-advanced

Questions

09-allocation-advanced에서 질문이 있습니다.

39p page Garbage collection in python에서 보면 아래쪽에 reference cycle을 찾는 방법이 나오는데 이 부분이 이해가 잘 가지 않아 질문드립니다. (왜 ref count < 2이면 unreachable로 판단하는지 이해하지 못했습니다)

Garbage collection in Python

- **Memory management**
  - Python does not necessarily release memory back to OS
  - For small objects(~512B), object allocator keeps the memory for future use
- **Standard GC algorithms**
  - Reference counting collector
    - efficient, simple, but reference cycles can survive
  - Generational garbage collector
- **How to find reference cycles**
  - GC iterates over each container object and temporarily removes all references to container objects it references.
  - After full iteration, all objects which reference count lower than two are unreachable from Python's code and thus can be collected.

Questions

09-allocation-advanced에서 질문이 있습니다.

39p page Garbage collection in python에서 보면 아래쪽에 reference cycle을 찾는 방법이 나오는데 이 부분이 이해가 잘 가지 않아 질문드립니다. (왜 ref count < 2이면 unreachable로 판단하는지 이해하지 못했습니다)

- reference count

```
>>> x = object()
>>> sys.getrefcount(x)
2
>>> y = x
>>> sys.getrefcount(x)
3
>>> del y
>>> sys.getrefcount(x)
2
```

```
>>> container = []
>>> container.append(container)
>>> sys.getrefcount(container)
3
>>> del container
```

container holds a reference to itself
reference count never falls to 0

https://devguide.python.org/internals/garbage-collector/

Questions

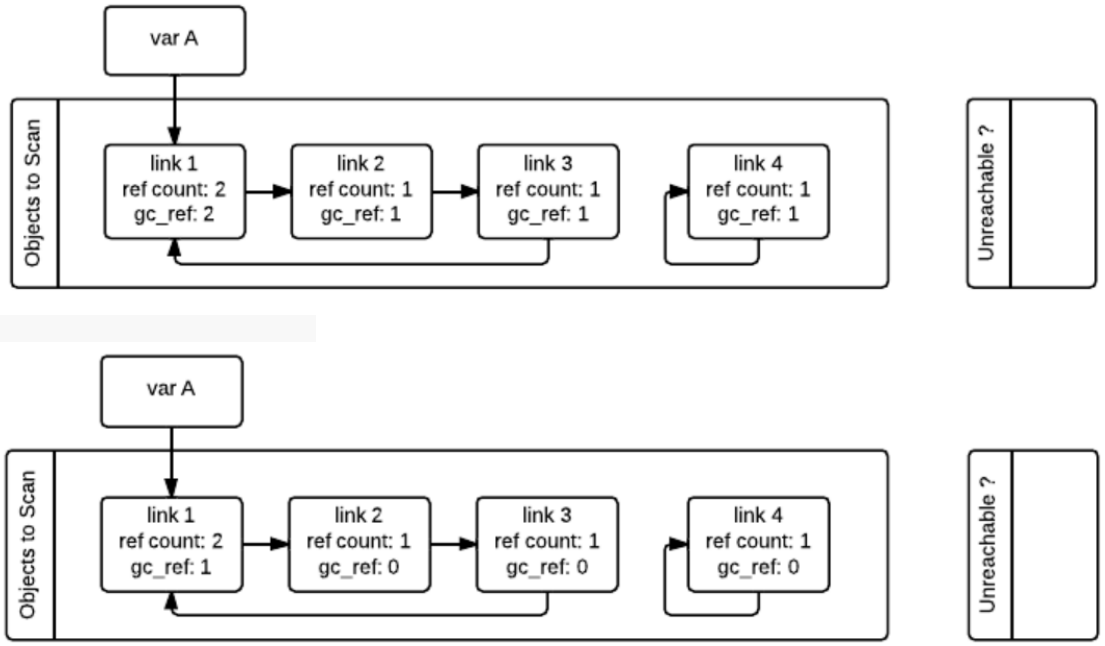
```
>>> import gc

>>> class Link:
...     def __init__(self, next_link=None):
...         self.next_link = next_link

>>> link_3 = Link()
>>> link_2 = Link(link_3)
>>> link_1 = Link(link_2)
>>> link_3.next_link = link_1
>>> A = link_1
>>> del link_1, link_2, link_3

>>> link_4 = Link()
>>> link_4.next_link = link_4
>>> del link_4

# Collect the unreachable Link object (and its .__dict__ dict).
>>> gc.collect()
2
```



https://devguide.python.org/internals/garbage-collector/

Questions

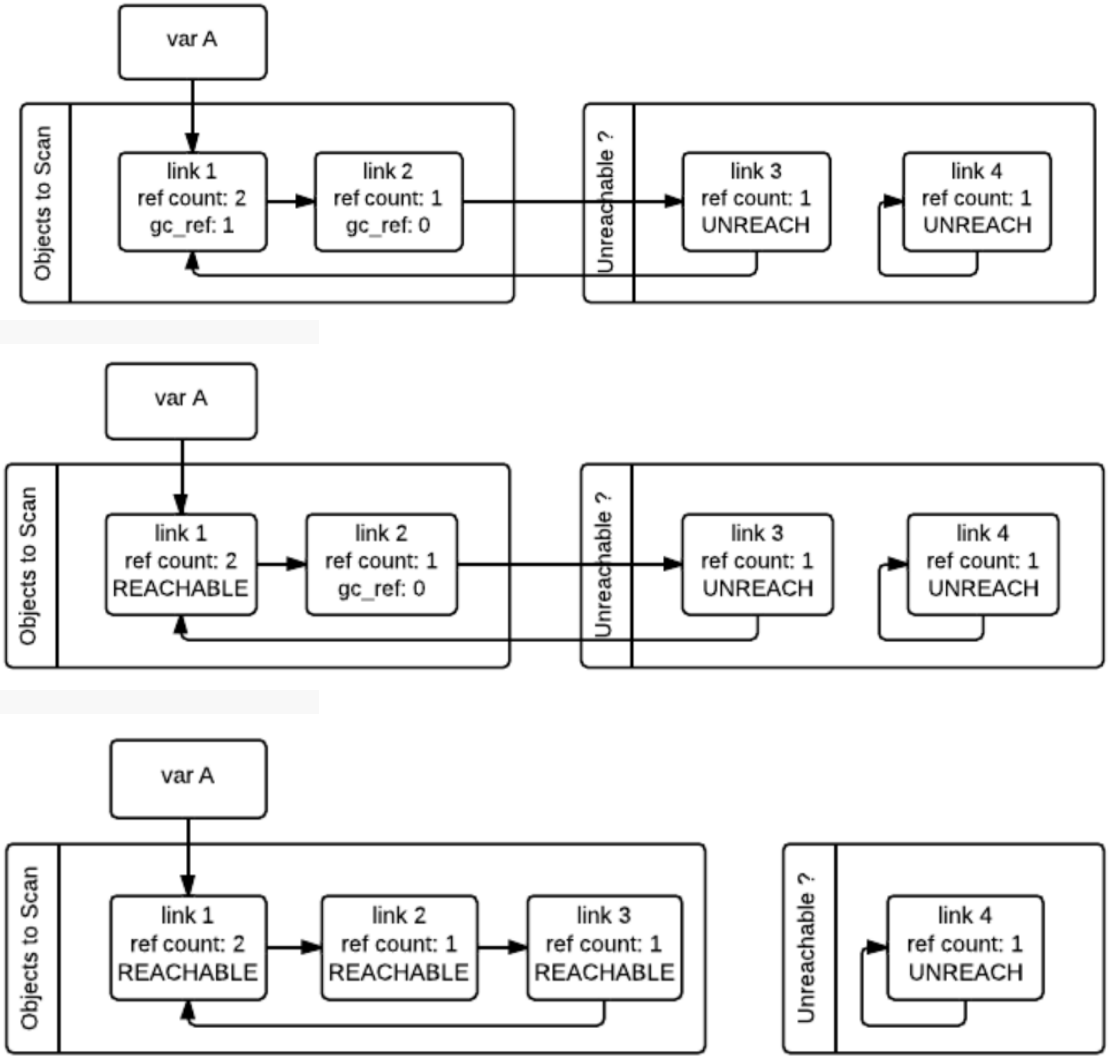
```
>>> import gc

>>> class Link:
...     def __init__(self, next_link=None):
...         self.next_link = next_link

>>> link_3 = Link()
>>> link_2 = Link(link_3)
>>> link_1 = Link(link_2)
>>> link_3.next_link = link_1
>>> A = link_1
>>> del link_1, link_2, link_3

>>> link_4 = Link()
>>> link_4.next_link = link_4
>>> del link_4

# Collect the unreachable Link object (and it
>>> gc.collect()
2
```



https://devguide.python.org/internals/garbage-collector/