Free Space Management

CS 301, Fall 2019, Operating System
Guest Lecture

Date: 04 September 2019 1100 Hours

Rishiraj Adhikary, Ph.D. Student, Sustainability Lab, IIT Gandhinagar

Motivation

- What if you need (k* 1 MB) space from the heap?

Motivation

- What if you need (k* 1 MB) space from the heap?

- `int *y = malloc(k*(1024*1024))`

Motivation

- What if you need (k* 1 MB) space from the heap?

- `int *y = malloc(1*(1024*1024))`
- `mmap(NULL, 1052672, ...)`

Motivation

- What if you need (k* 1 MB) space from the heap?

- `int *y = malloc(1*(1024*1024))`
- `mmap(NULL, 1052672, ...)`

- 1052672 B - 1024*1024 B = 4096 B

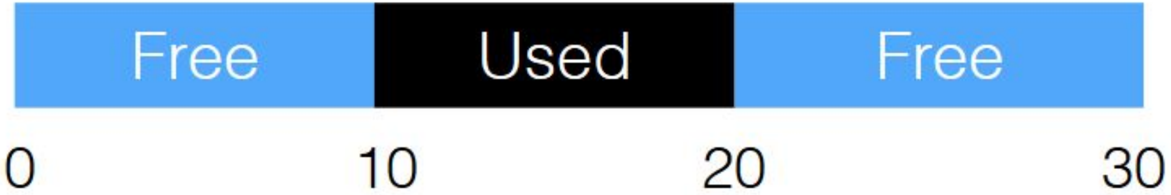
Motivation

- What if you need (k* 1 MB) space from the heap?

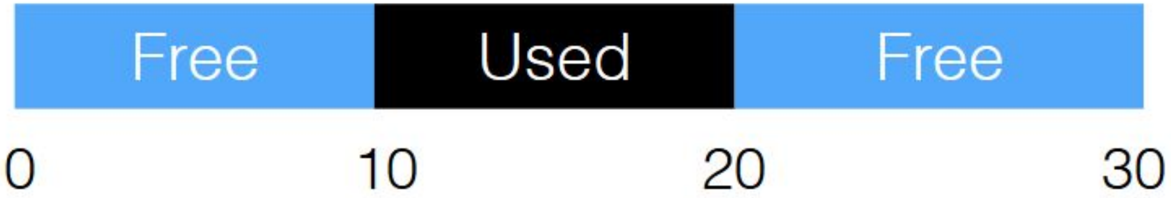
```
● int *y = malloc(1*(1024*1024))
● mmap(NULL, 1052672, ...)
```

- 1052672 B - 1024*1024 B = 4096 B
- Why do we see this difference in size ?

Revisiting External Fragmentation

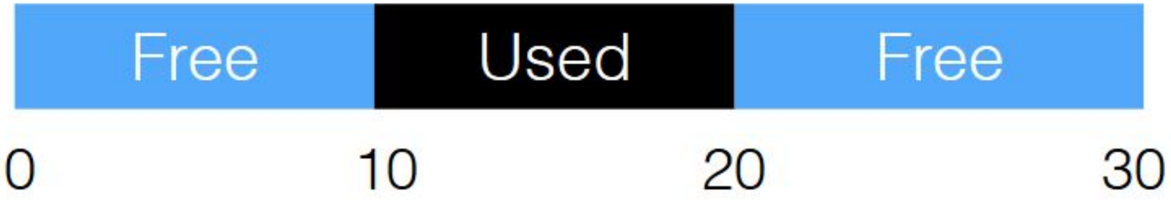


Revisiting External Fragmentation



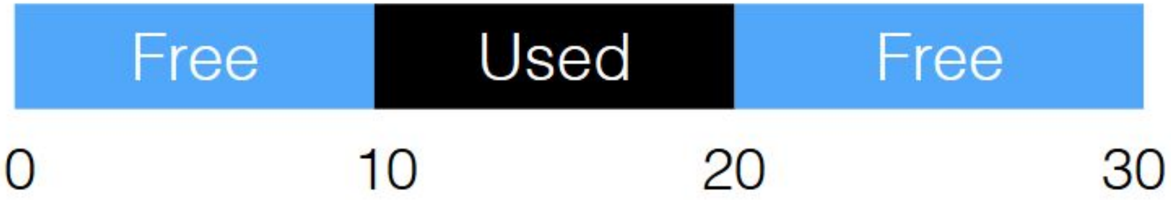
A request for 15 Bytes will fail

Revisiting External Fragmentation



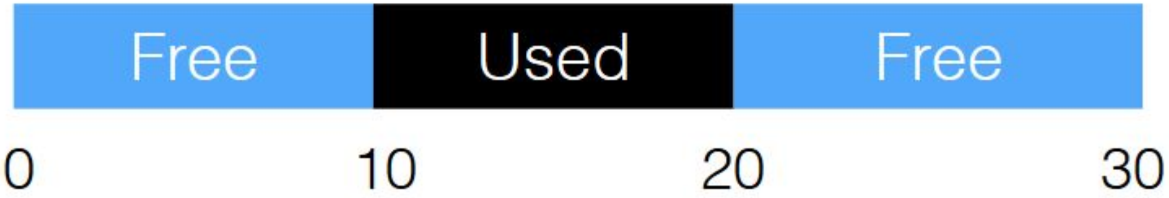
- A good idea will be to combine multiple free space to make a bigger free space

Revisiting External Fragmentation



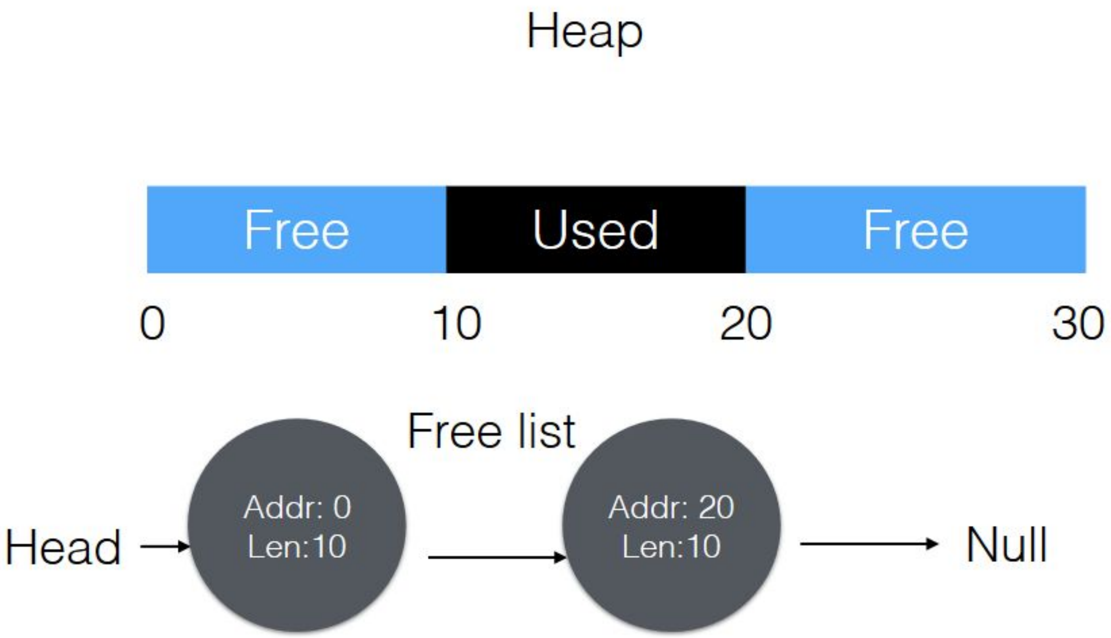
- A good idea will be to combine multiple free space to make a bigger free space
- We will need a data structure to represent free space

Free List



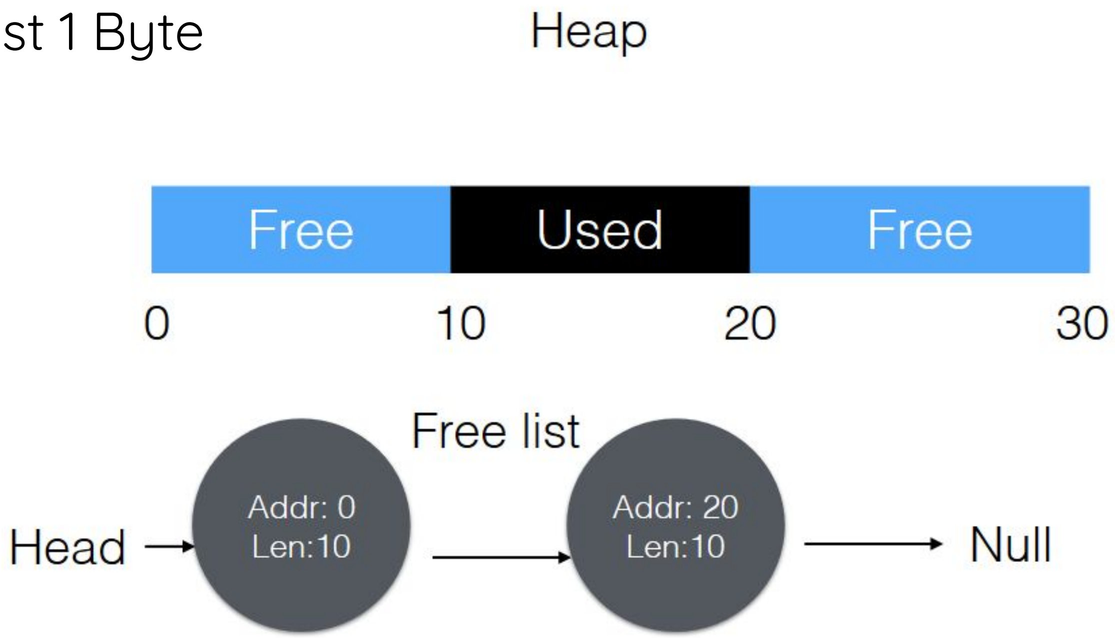
- A good idea will be to combine multiple free space to make a bigger free space
- We will need a data structure to represent free space.
- A List !

Free List


Heap
Free
Used
Free
0
10
20
30
Free list
Head
Addr: 0
Len:10
Addr: 20
Len:10
Null

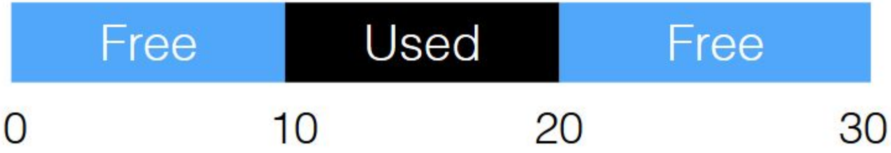
Free List

Request 1 Byte



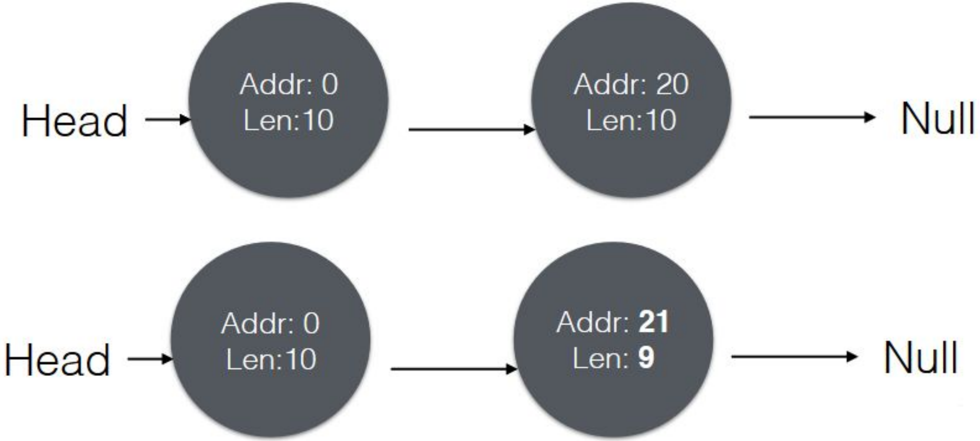
Split

After Request of 1 Byte

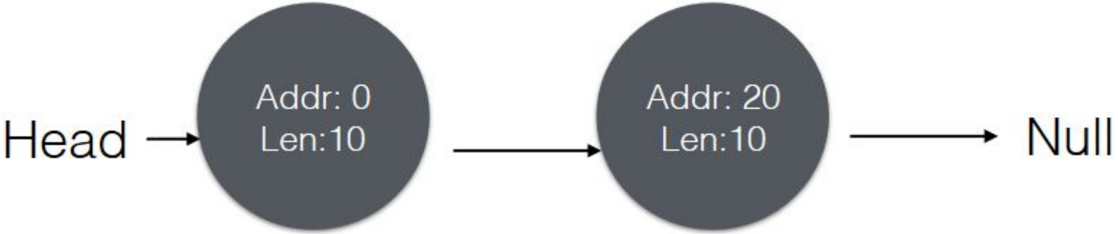


Before

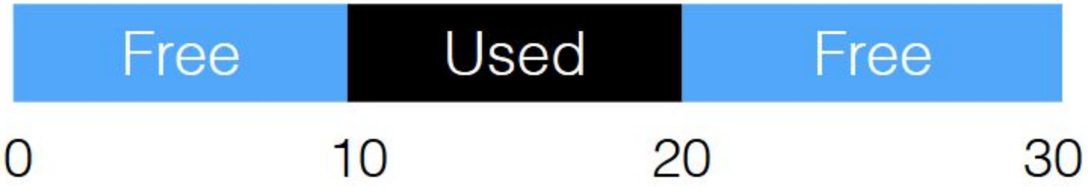
After



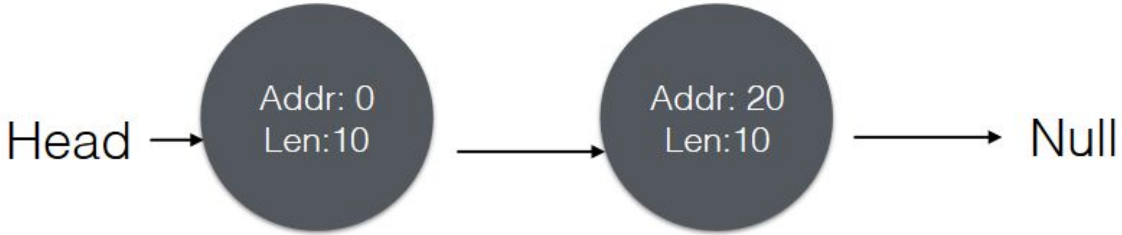
Free Space



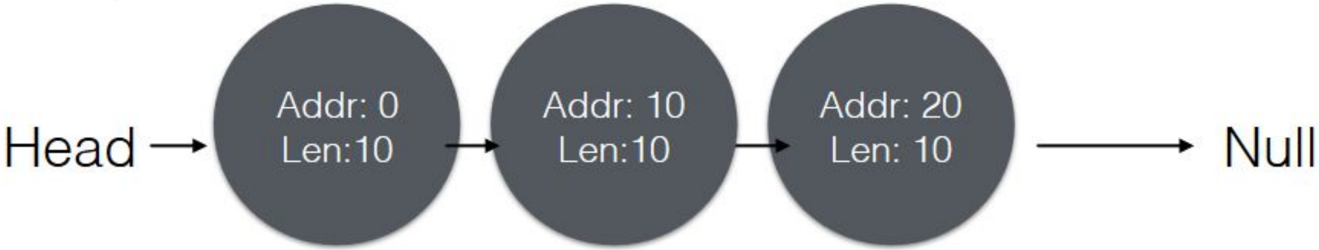
Free 10 Bytes



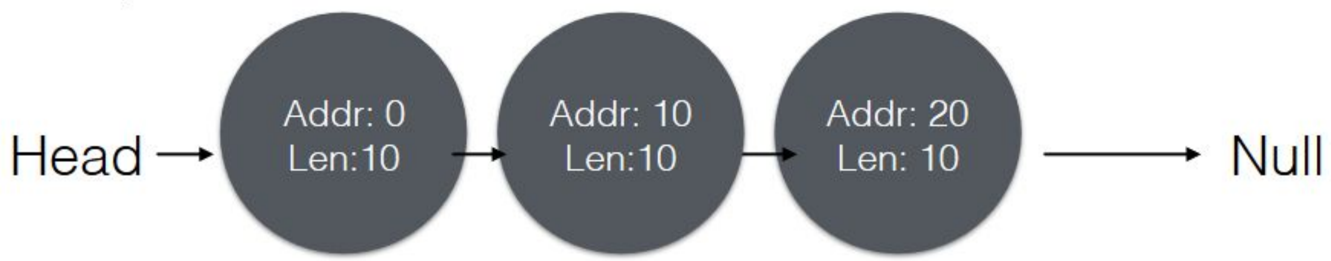
Free Space



Free 10 Bytes

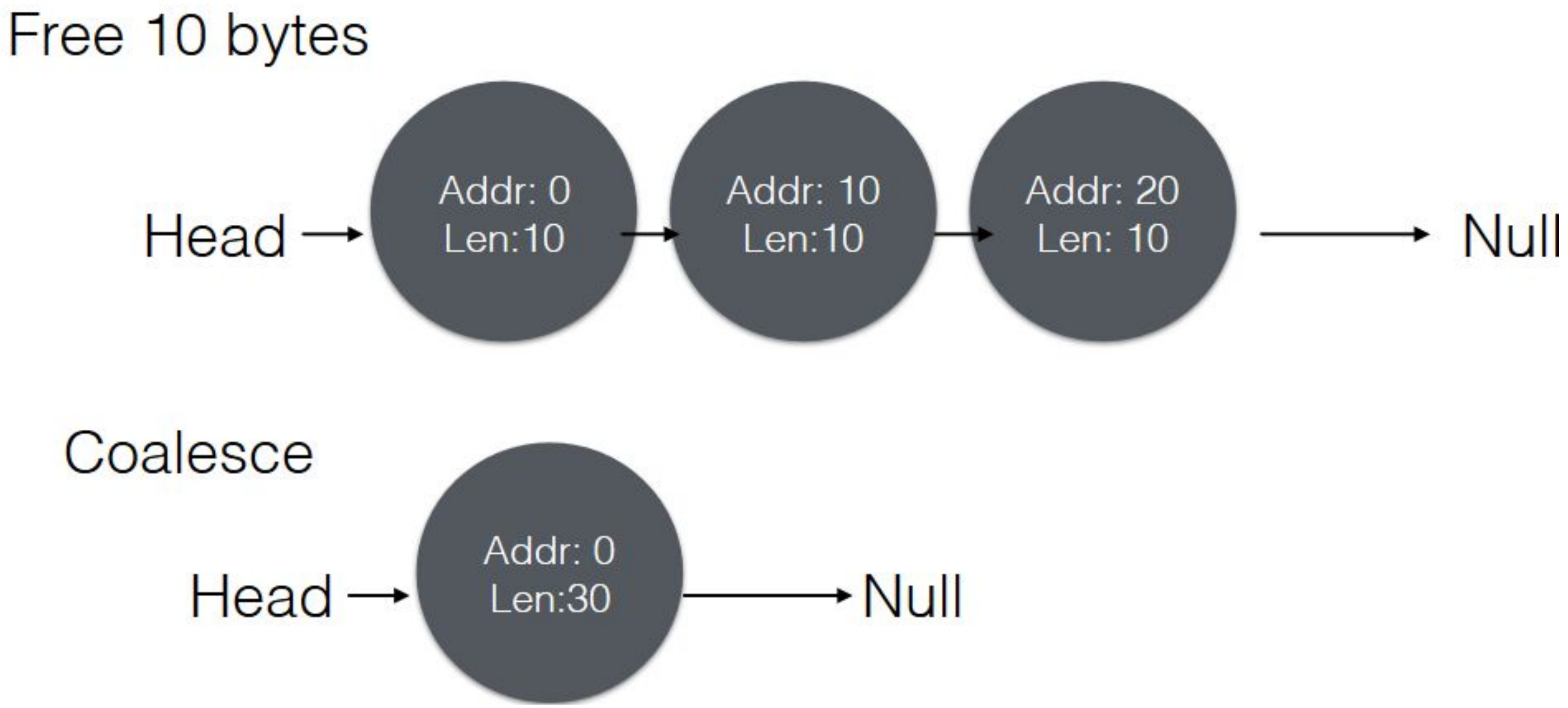


Free Space



Can we allocate 20 Bytes of memory?

Coalescing



Free 10 bytes

Head → Addr: 0 Len:10 → Addr: 10 Len:10 → Addr: 20 Len: 10 → Null

Coalesce

Head → Addr: 0 Len:30 → Null

Coalescing

When do you think coalescing happens ?

Coalescing

When do you think coalescing happens ?

Coalescing can happen each time any memory is free and then we look for empty free spaces.

malloc() and free() interface

- `int *ptr = malloc(1024*1024)`
- `free(ptr)`

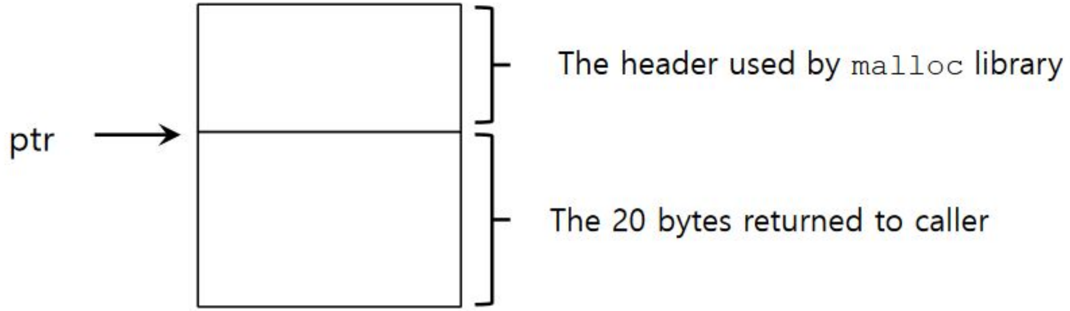
Tracking size of allocations

How does free(void *ptr) know the size of memory region that will be back into free list?

Tracking size of allocations

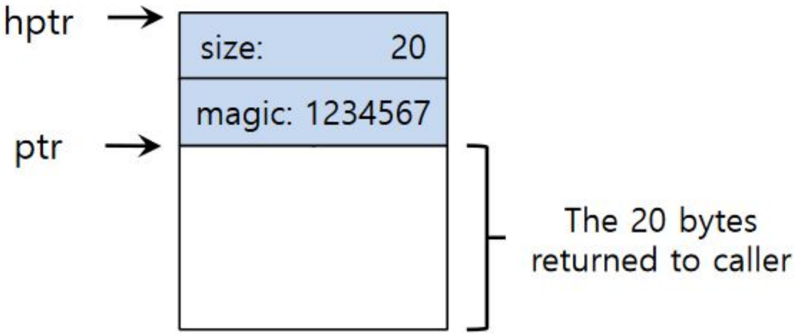
How does free(void *ptr) know the size of memory region that will be back into free list?

```
ptr = malloc(20);
```


The header used by malloc library
ptr
The 20 bytes returned to caller

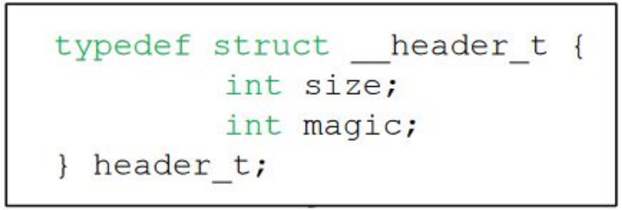
An Allocated Region Plus Header

Tracking size of allocations



The 20 bytes returned to caller

Specific Contents Of The Header



```
typedef struct __header_t {
            int size;
            int magic;
} header_t;
```

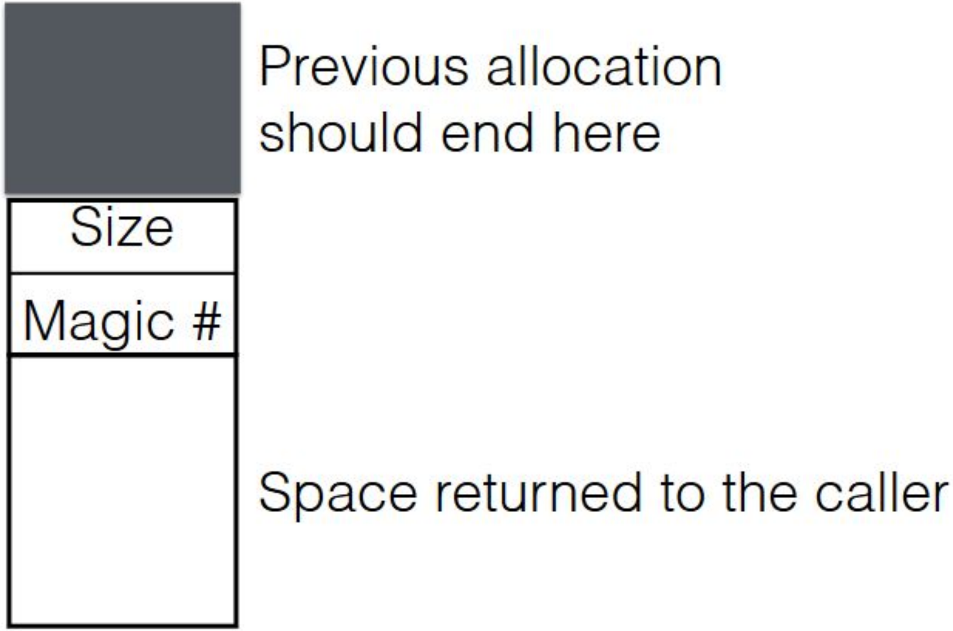
A Simple Header

Magic Number

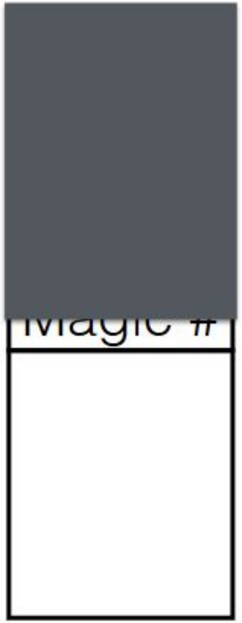
Magic numbers are used for integrity checking

Magic Number

Magic numbers are used for integrity checking


Previous allocation should end here
Size
Magic #
Space returned to the caller

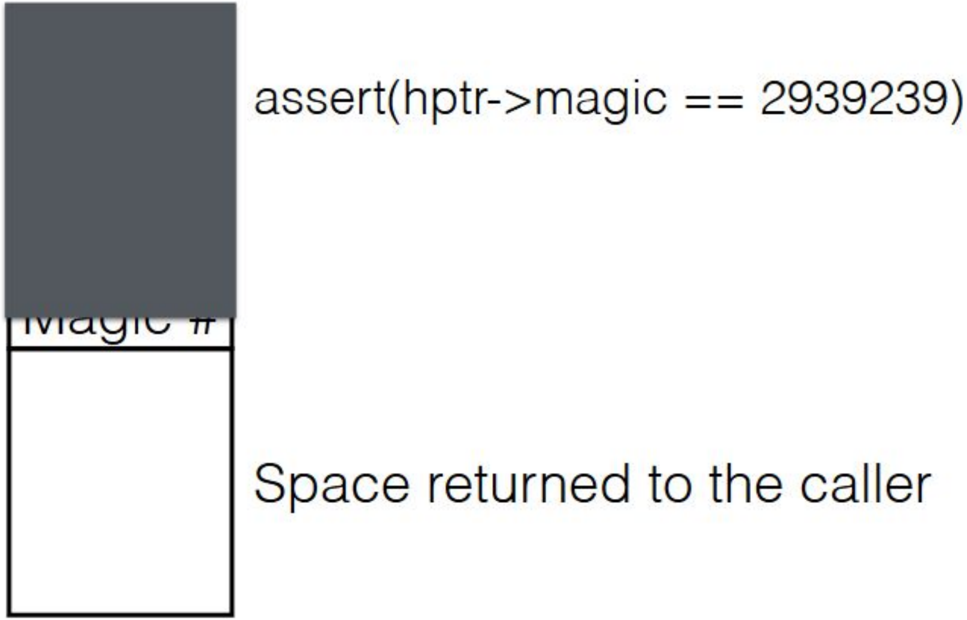
Magic Number



But, instead ends here …

Space returned to the caller

Magic Number



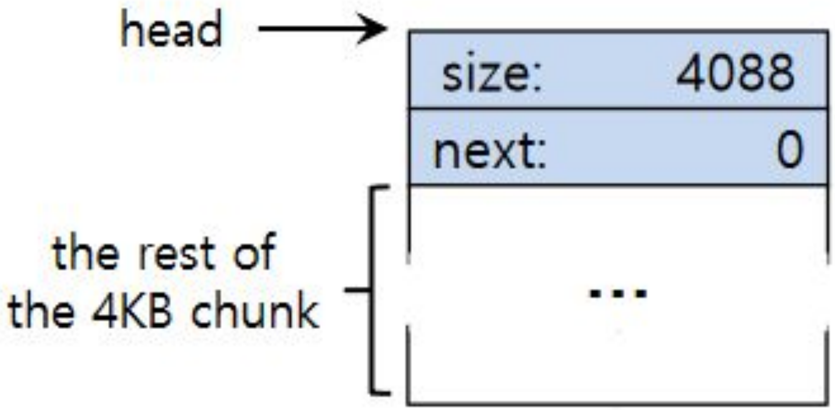
assert(hptr->magic == 2939239)

Space returned to the caller

Magic number can be used for debugging. Set it to some constant when memory is allocated. https://danluu.com/malloc-tutorial/

Example

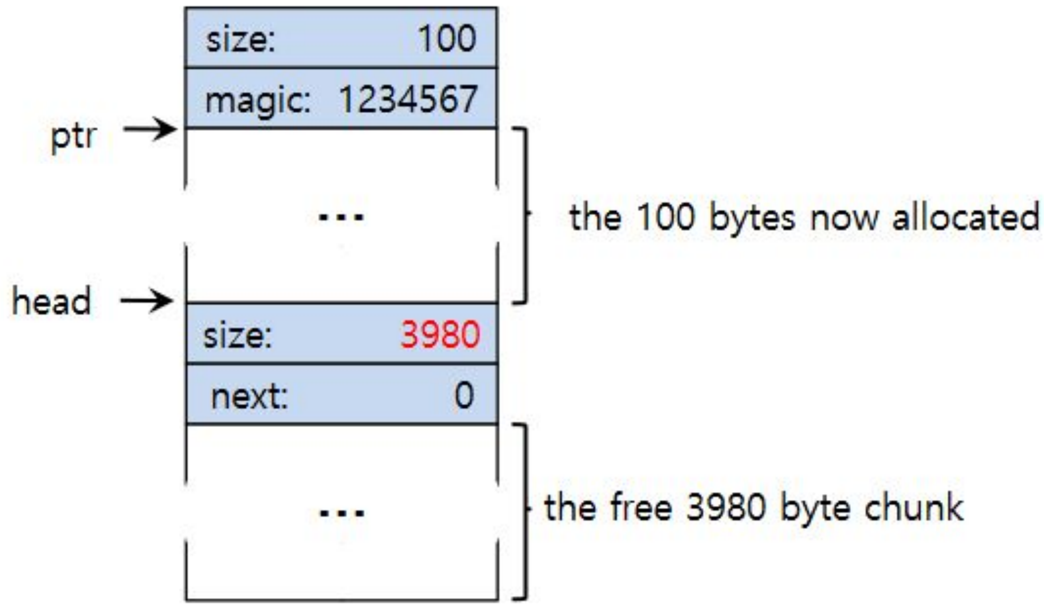
A 4KB Heap With One Free Chunk



Example

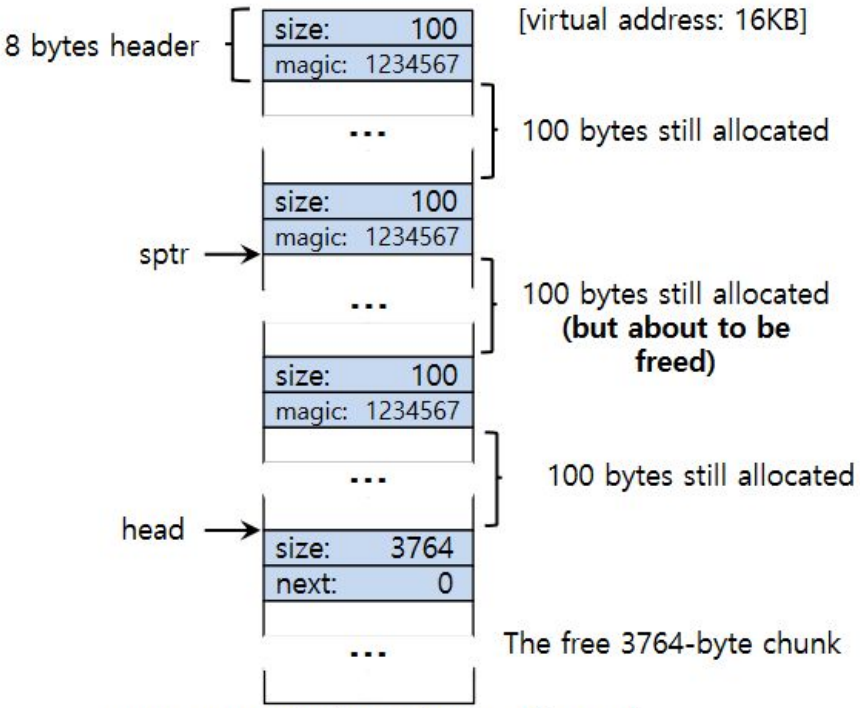
ptr = malloc(100)

: A Heap : After One Allocation



Example

free(sptr)

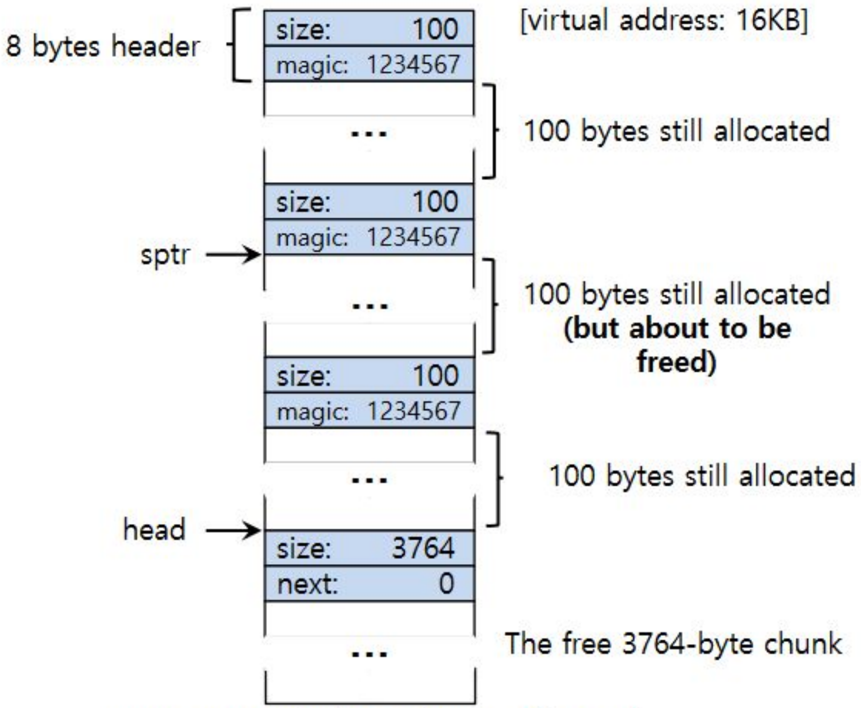


Free Space With Three Chunks Allocated

Example

free(sptr)

What is sptr ?



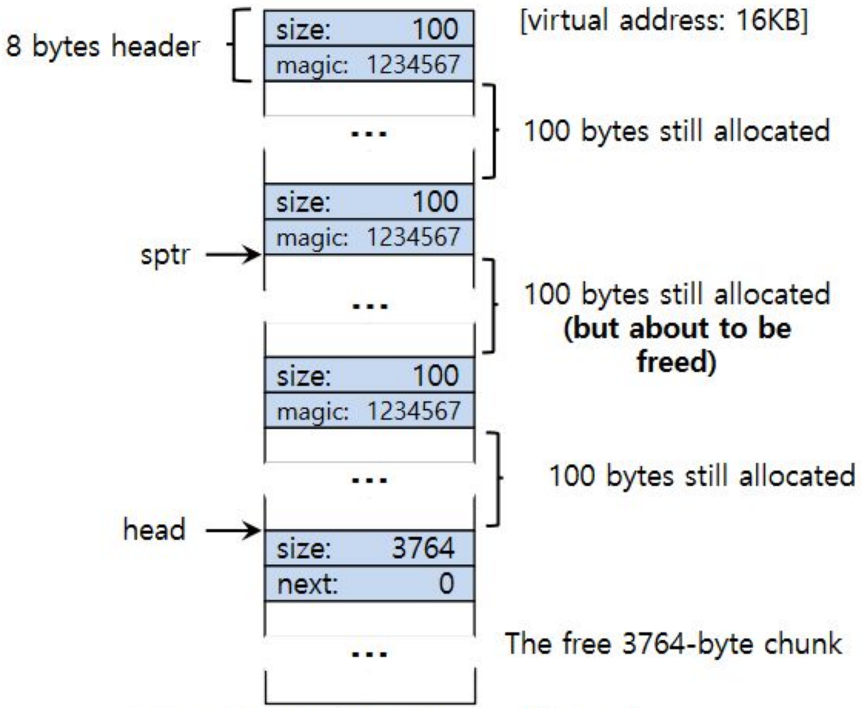
Free Space With Three Chunks Allocated

Example

free(sptr)

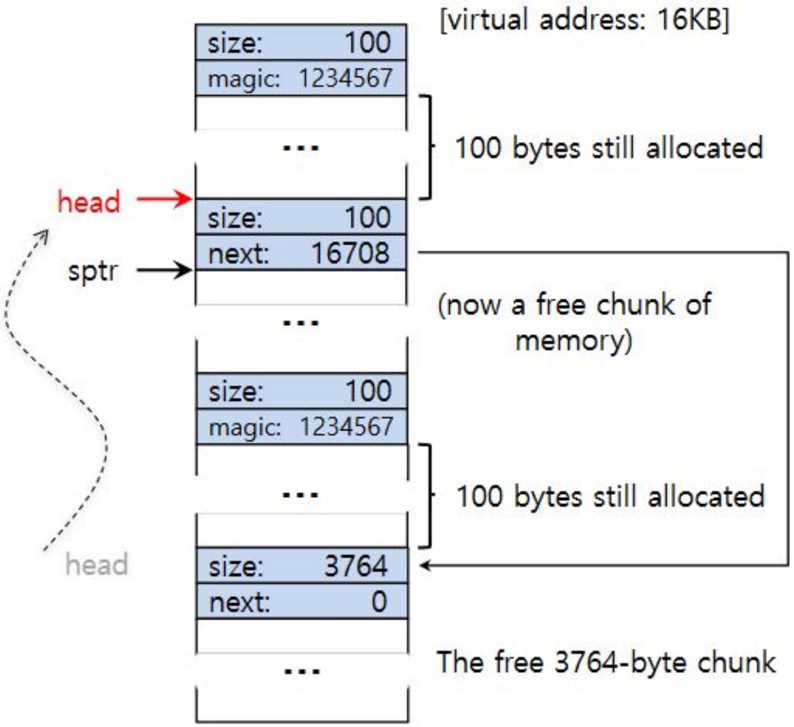
What is sptr ?

16KB+8+100+8

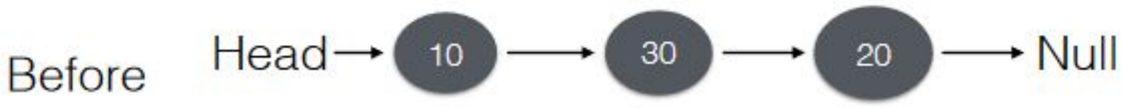


Free Space With Three Chunks Allocated

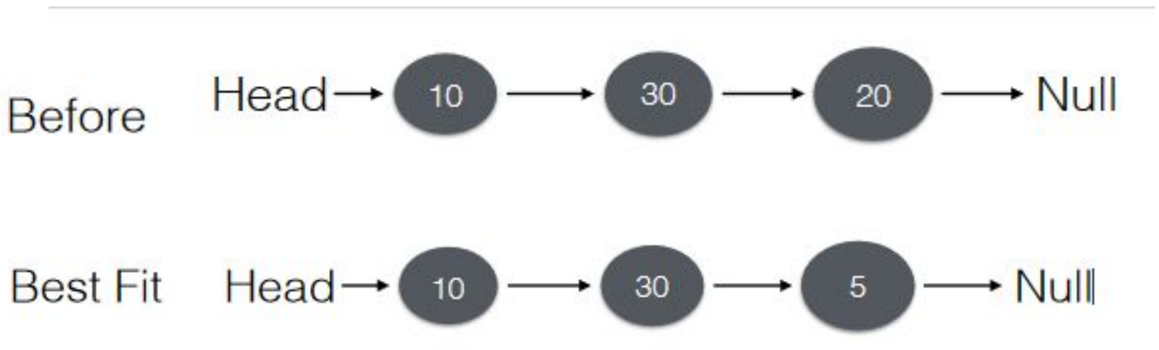
Example


[virtual address: 16KB]
size: 100
magic: 1234567
...
100 bytes still allocated
head
size: 100
next: 16708
sptr
...
(now a free chunk of memory)
size: 100
magic: 1234567
...
100 bytes still allocated
head
size: 3764
next: 0
...
The free 3764-byte chunk

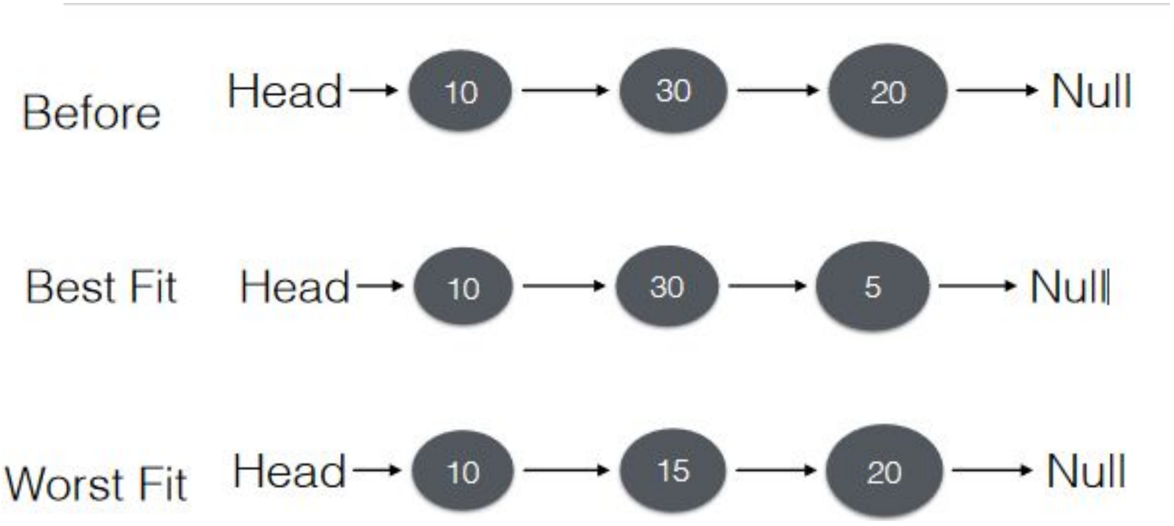
Allocation Strategies



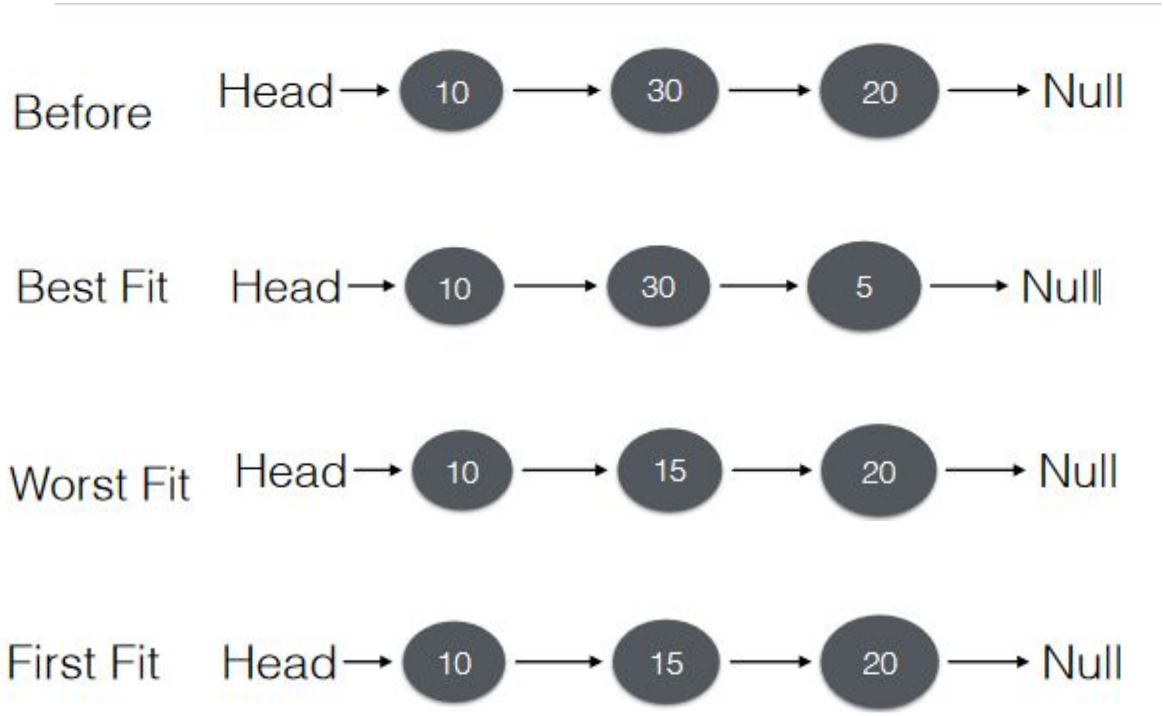
Allocation Strategies



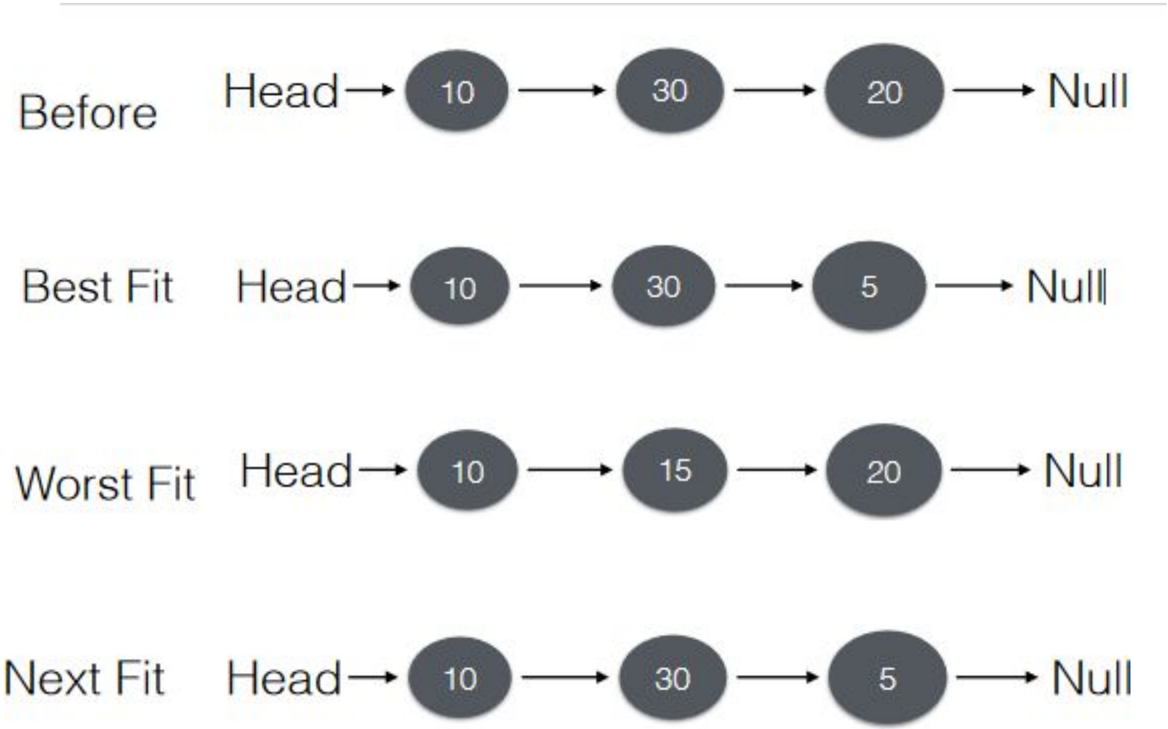
Allocation Strategies



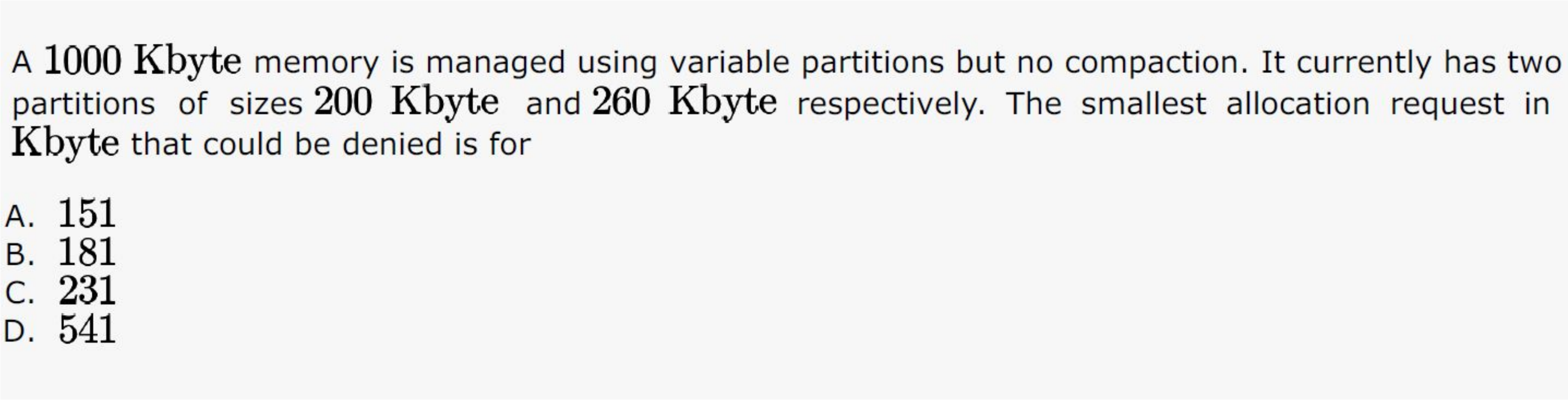
Allocation Strategies


Before
Head → 10 → 30 → 20 → Null
Best Fit
Head → 10 → 30 → 5 → Null
Worst Fit
Head → 10 → 15 → 20 → Null
First Fit
Head → 10 → 15 → 20 → Null

Allocation Strategies


Before
Head → 10 → 30 → 20 → Null
Best Fit
Head → 10 → 30 → 5 → Null
Worst Fit
Head → 10 → 15 → 20 → Null
Next Fit
Head → 10 → 30 → 5 → Null

Question



A $1000\ \mathrm{Kbyte}$ memory is managed using variable partitions but no compaction. It currently has two partitions of sizes $200\ \mathrm{Kbyte}$ and $260\ \mathrm{Kbyte}$ respectively. The smallest allocation request in Kbyte that could be denied is for

A. 151

B. 181

C. 231

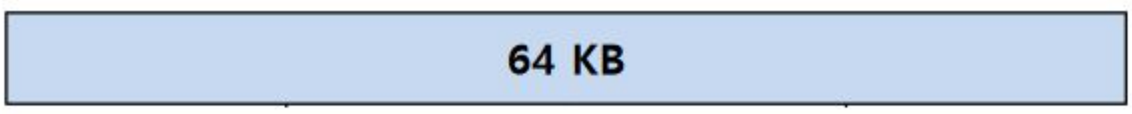
D. 541

Binary Buddy Allocation

The allocator divides free space by two until a block that is big enough to accommodate the request is found.

Binary Buddy Allocation

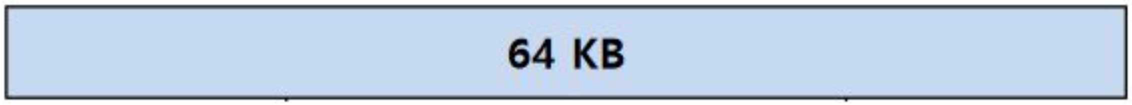
The allocator divides free space by two until a block that is big enough to accommodate the request is found.



64 KB

Binary Buddy Allocation

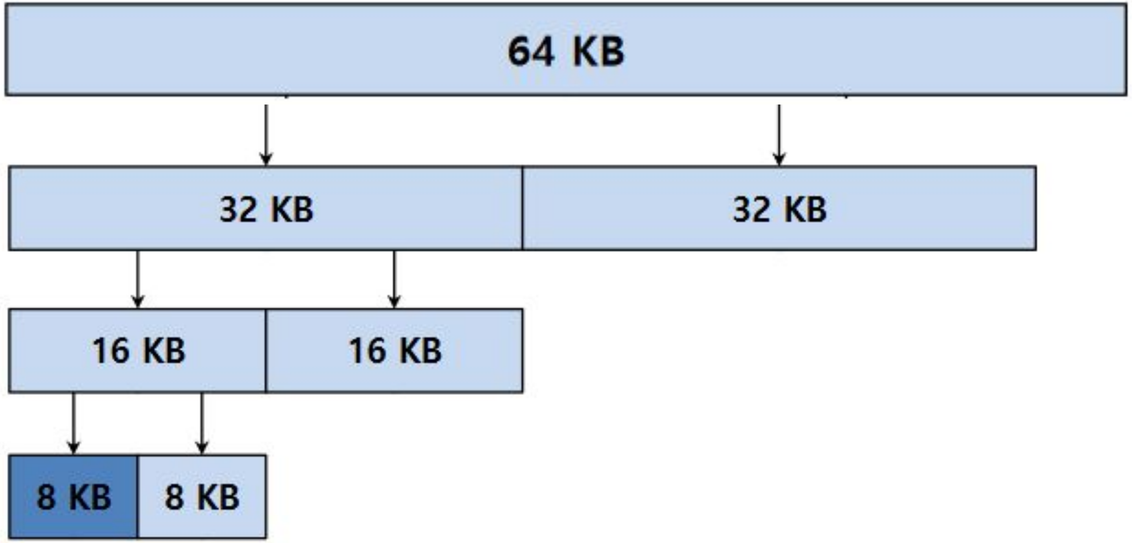
The allocator divides free space by two until a block that is big enough to accommodate the request is found.



A 7 KB request

Binary Buddy Allocation

The allocator divides free space by two until a block that is big enough to accommodate the request is found.



Reference and Credit

- OSTEP, Chapter 17.

- Prof. Nipun Batra: Slides on free space management: https://nipunbatra.github.io/teaching/os-fall-18/lectures/16-swapping-free-memory.pdf
- Some images in the slides are courtesy of Prof. Youjip Won, SSRC, Baskin Engineering, Santa Cruz. https://www.ssrc.ucsc.edu/person/youjip.html