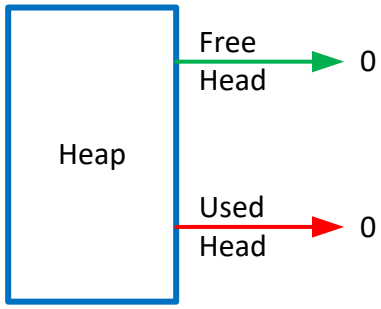
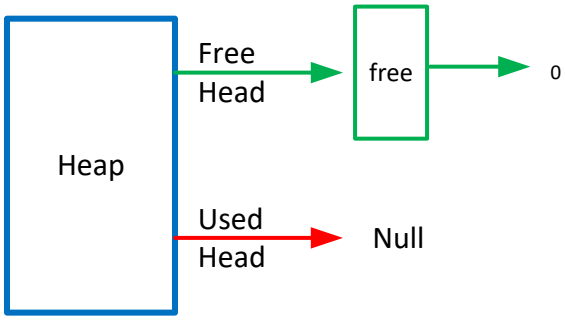
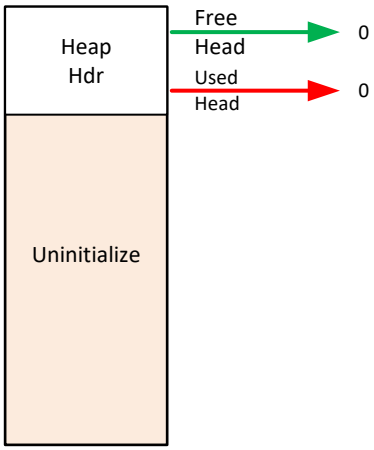


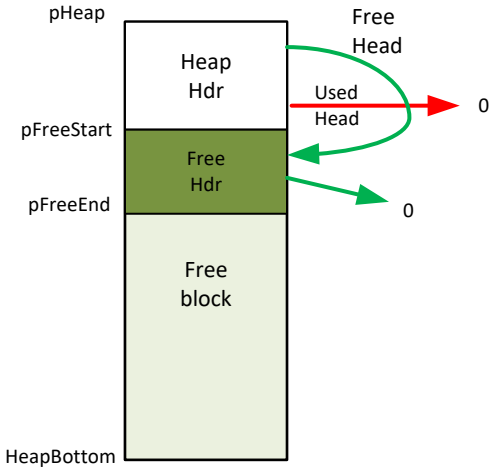
Initialize



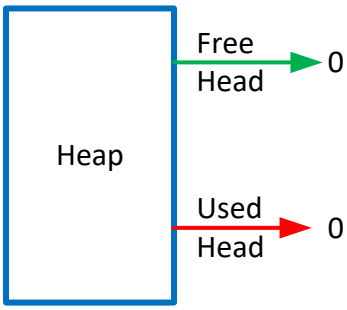
Test1



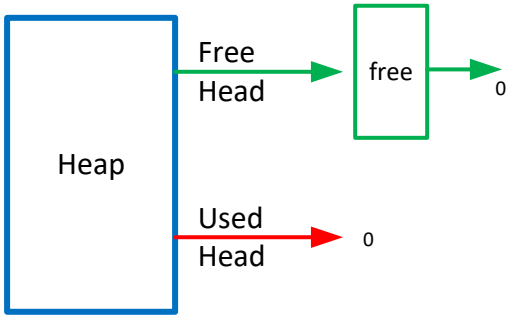
Physical
Layout



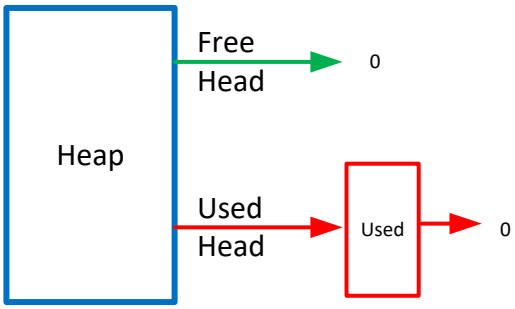
Physical
Layout



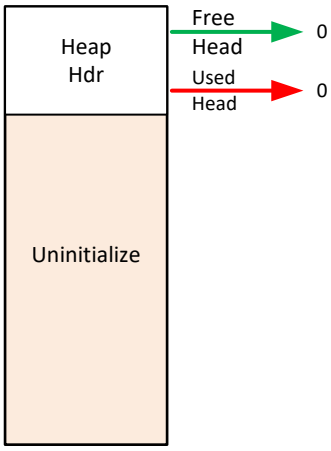
Mem()



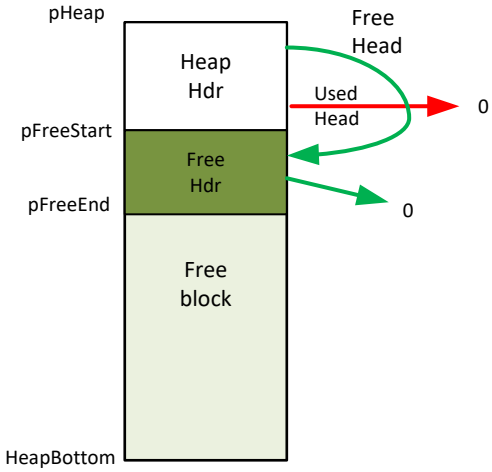
Mem::Initialize()



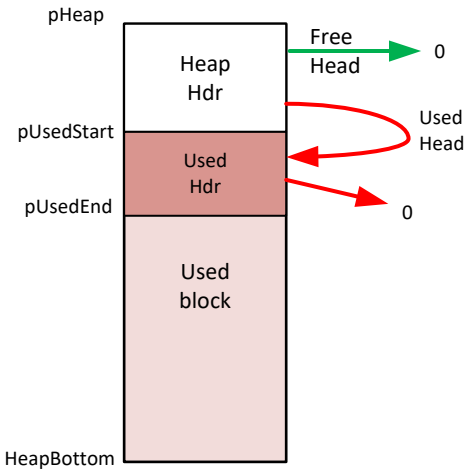
Mem::malloc(0xC7C0)



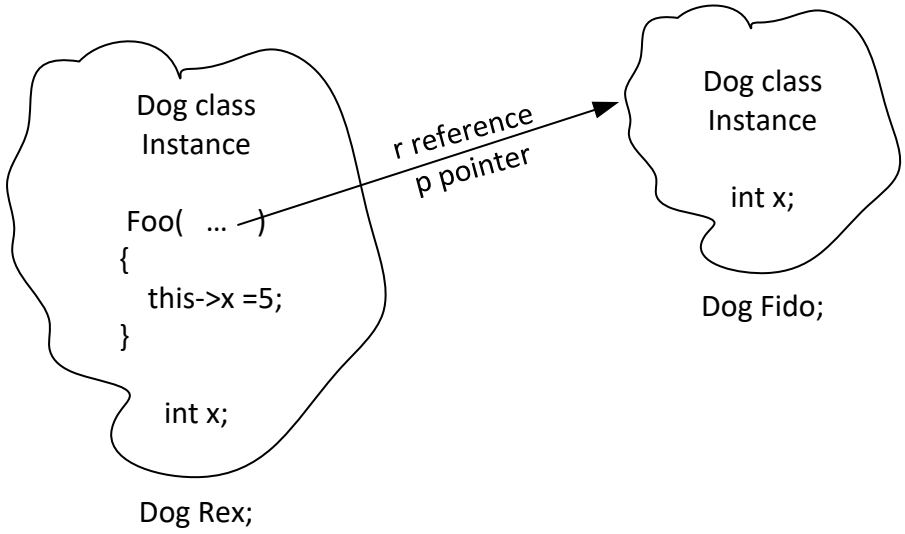
Physical
Layout



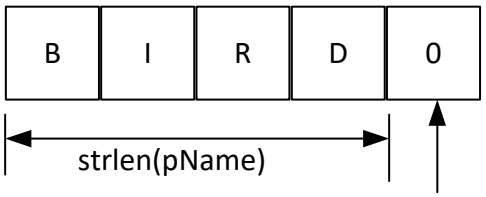
Physical
Layout

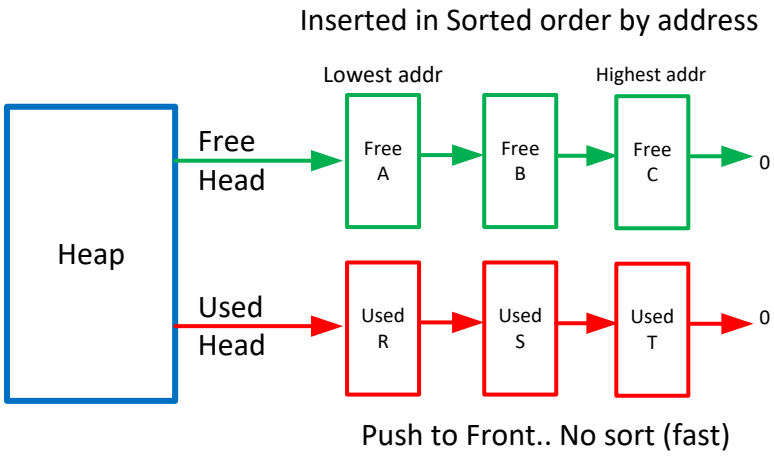


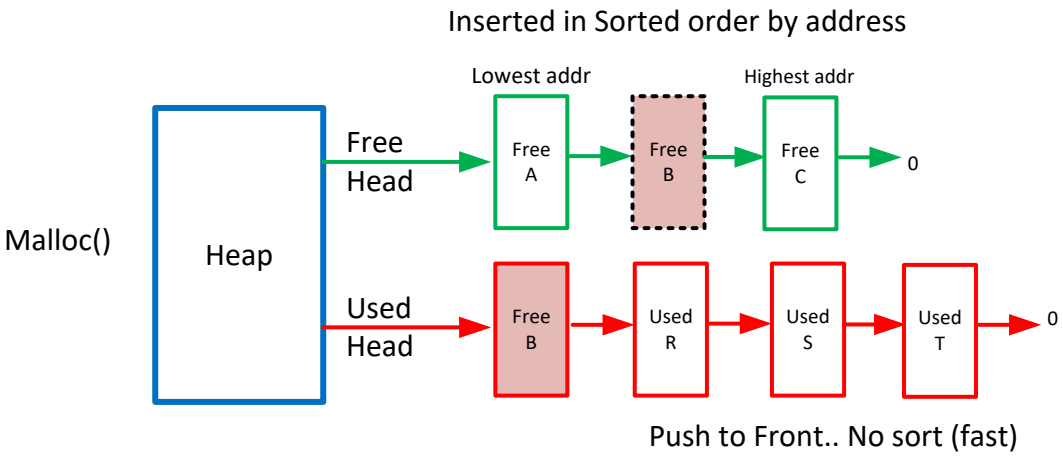
Physical
Layout

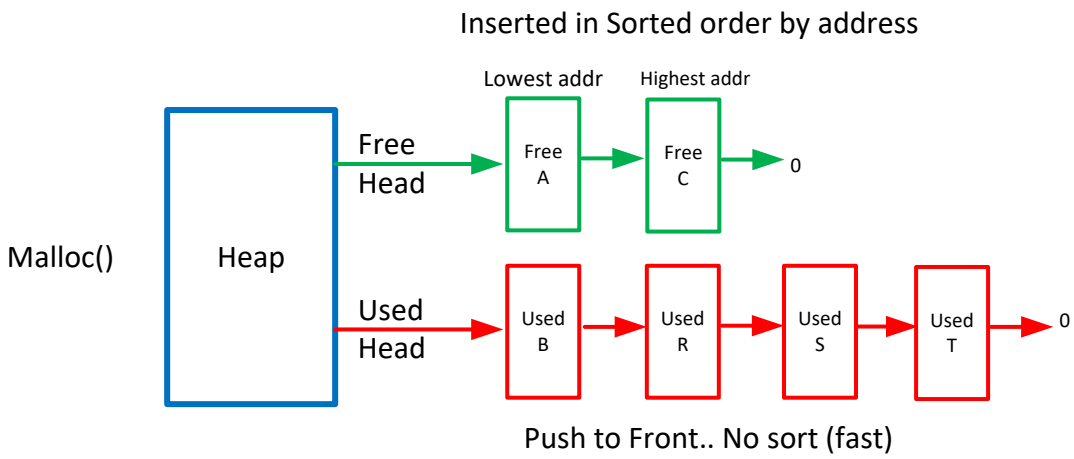

Dog class
Instance
Foo(…)
{
this->x =5;
}
int x;
Dog Rex;
r reference
p pointer
Dog class
Instance
int x;
Dog Fido;

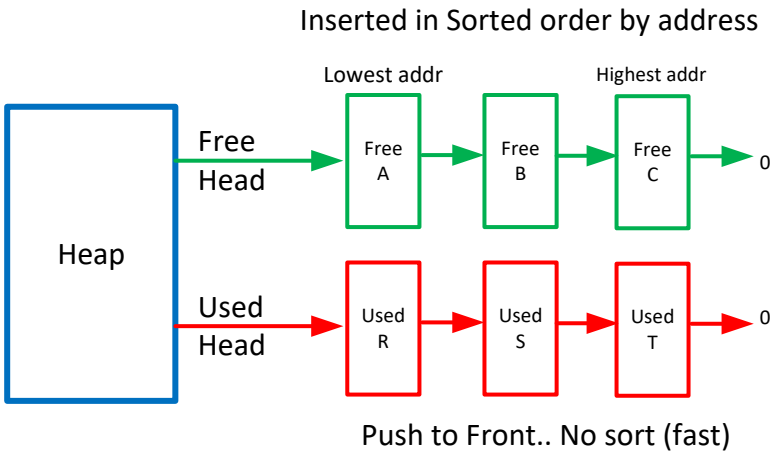
```
char pName[5];
strcpy(pName, “bird”);
```

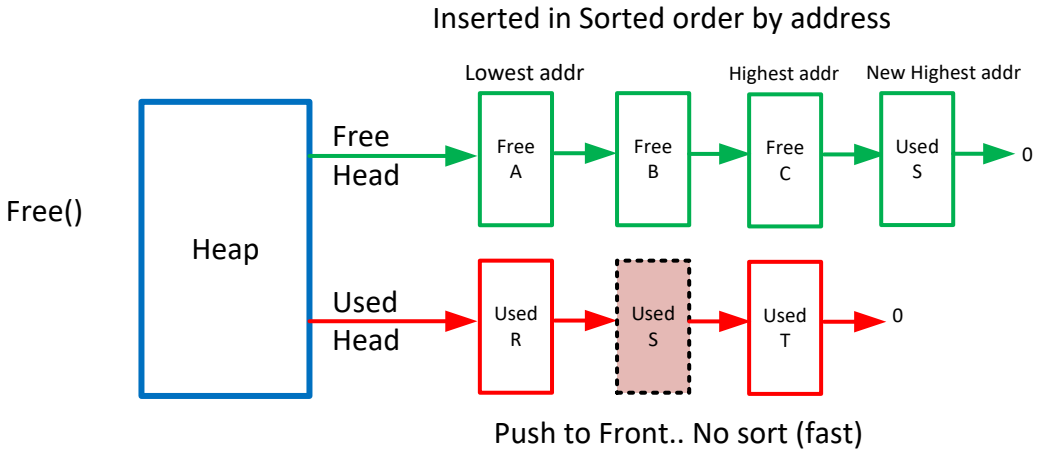


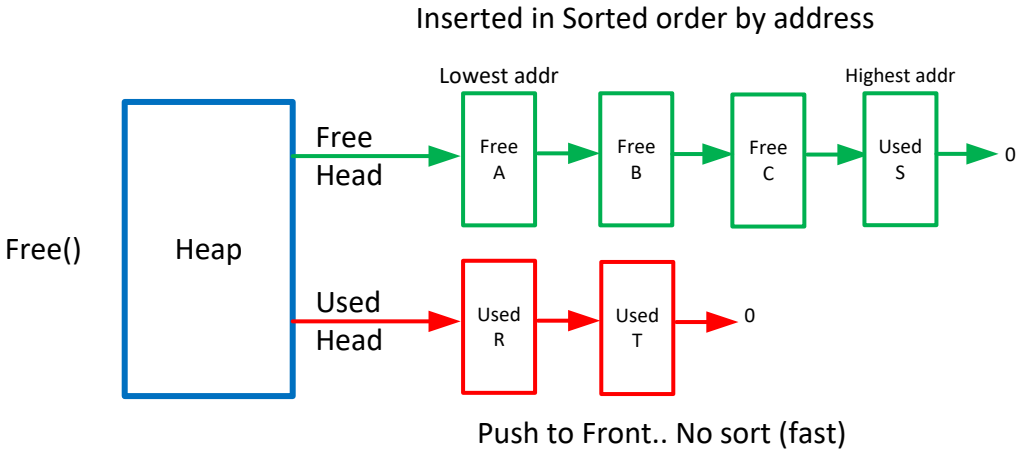

Inserted in Sorted order by address
Lowest addr
Highest addr
Heap
Free
Head
Free
A
Free
B
Free
C
0
Used
Head
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)

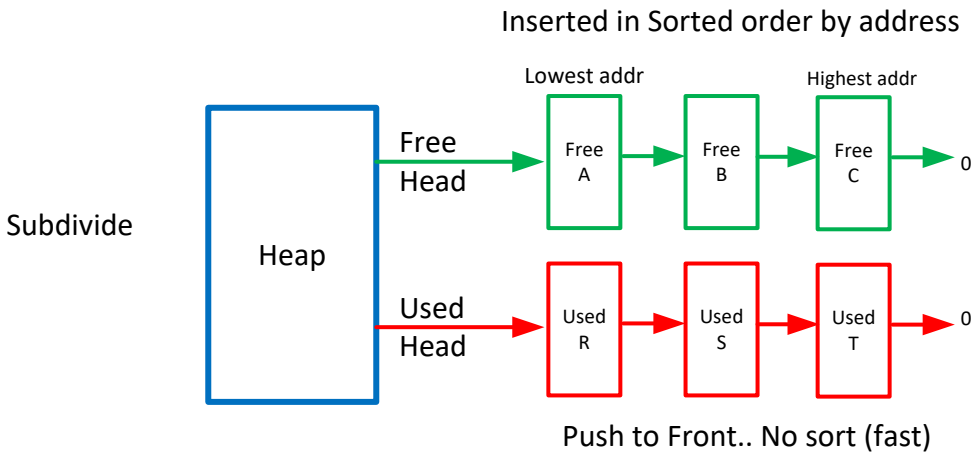

Inserted in Sorted order by address
Lowest addr
Highest addr
Malloc()
Heap
Free
Head
Free
A
Free
B
Free
C
0
Used
Head
Free
B
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)

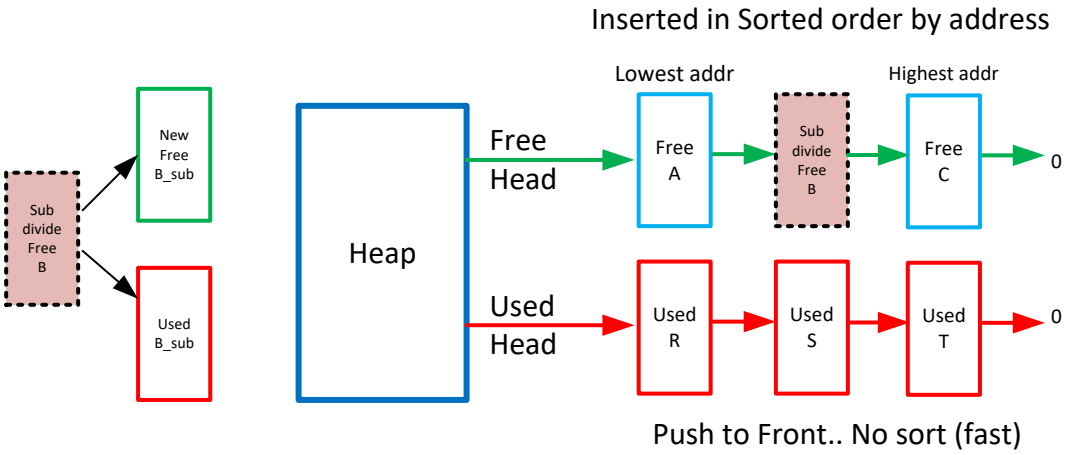

Inserted in Sorted order by address
Lowest addr
Highest addr
Malloc()
Heap
Free
Head
Free
A
Free
C
0
Used
Head
Used
B
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)

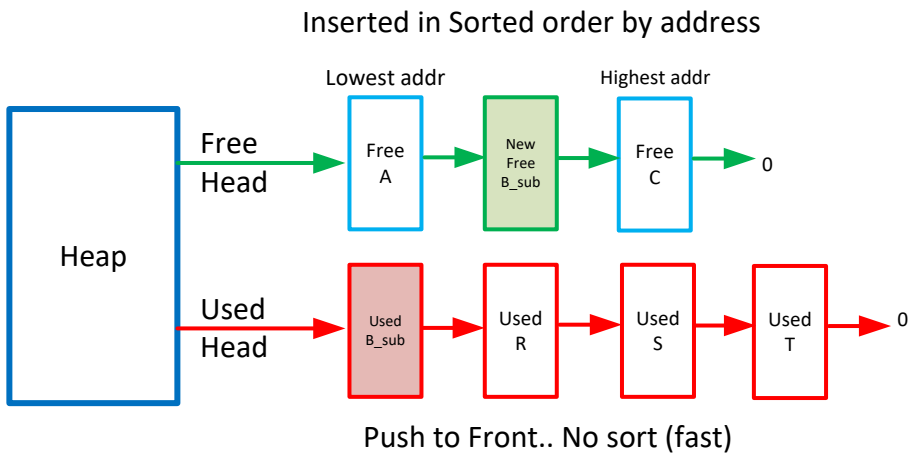

Inserted in Sorted order by address
Lowest addr
Highest addr
Heap
Free
Head
Free
A
Free
B
Free
C
0
Used
Head
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)


Inserted in Sorted order by address
Lowest addr
Highest addr
New Highest addr
Free()
Heap
Free
Head
Free
A
Free
B
Free
C
Used
S
0
Used
Head
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)


Inserted in Sorted order by address
Lowest addr
Highest addr
Free()
Heap
Free
Head
Free
A
Free
B
Free
C
Used
S
0
Used
Head
Used
R
Used
T
0
Push to Front.. No sort (fast)

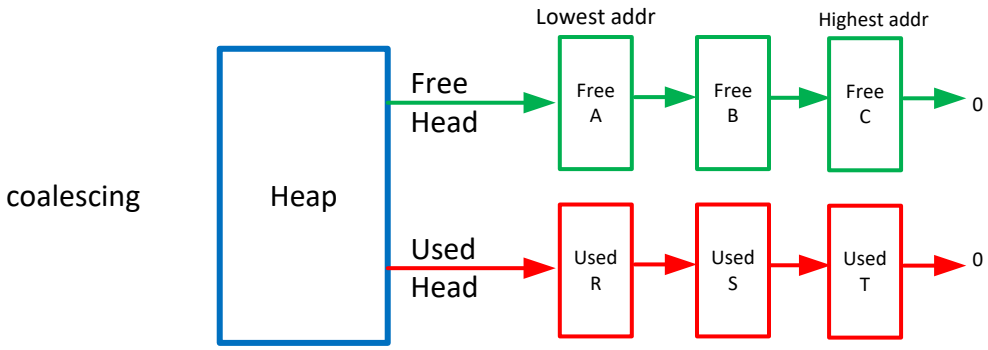

Inserted in Sorted order by address
Lowest addr
Highest addr
Subdivide
Heap
Free
Head
Free
A
Free
B
Free
C
0
Used
Head
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)


Inserted in Sorted order by address
Lowest addr
Highest addr
New
Free
B_sub
Sub
divide
Free
B
Used
B_sub
Heap
Free
Head
Free
A
Sub
divide
Free
B
Free
C
0
Used
Head
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)


Inserted in Sorted order by address
Lowest addr
Highest addr
Heap
Free
Head
Free
A
New
Free
B_sub
Free
C
0
Used
Head
Used
B_sub
Used
R
Used
S
Used
T
0
Push to Front.. No sort (fast)

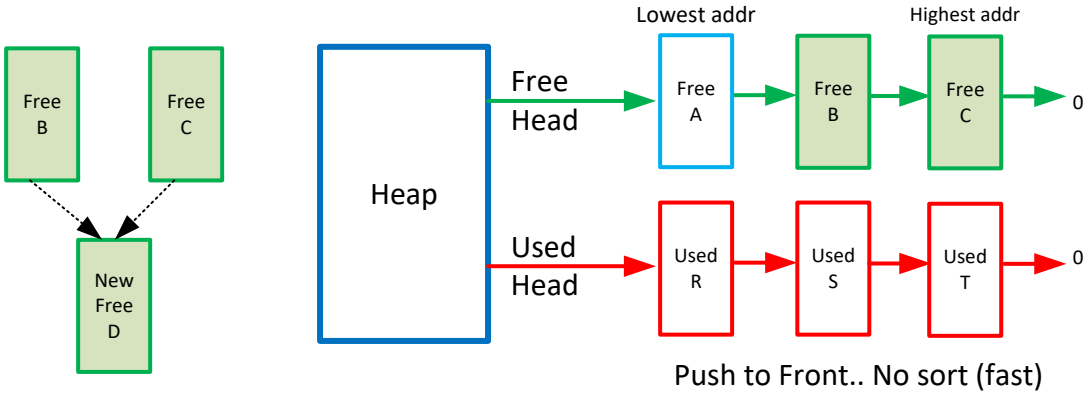
Inserted in Sorted order by address

coalescing



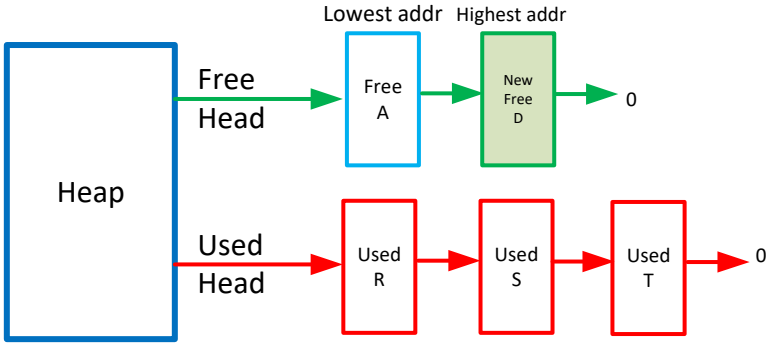
Push to Front.. No sort (fast)

Inserted in Sorted order by address

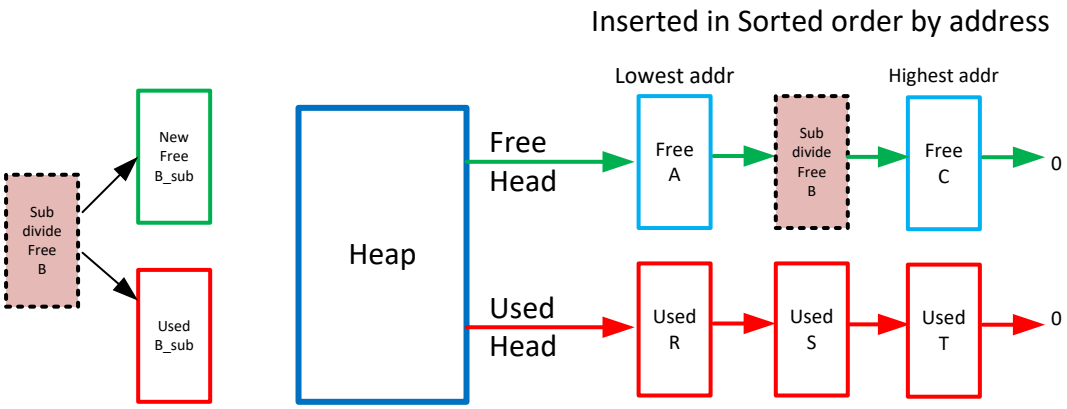


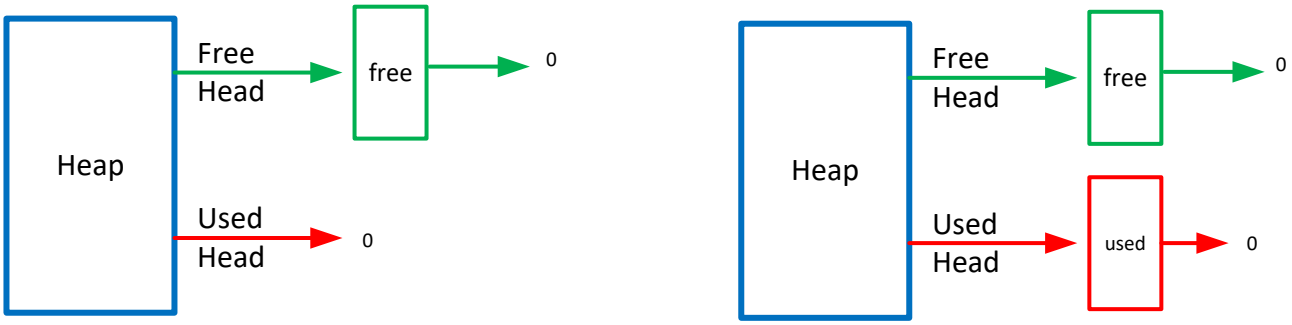
Push to Front.. No sort (fast)

Inserted in Sorted order by address

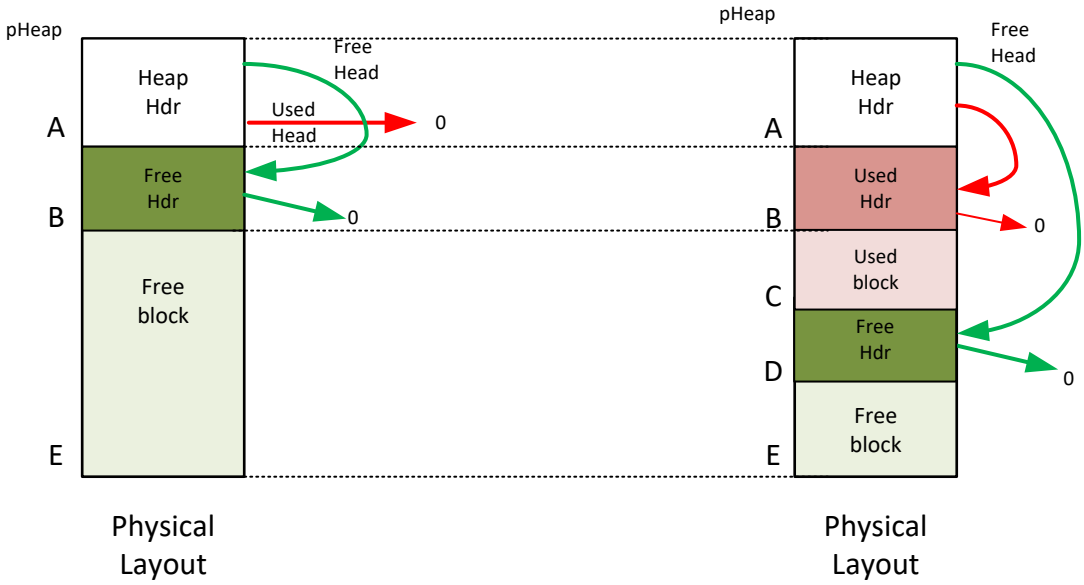


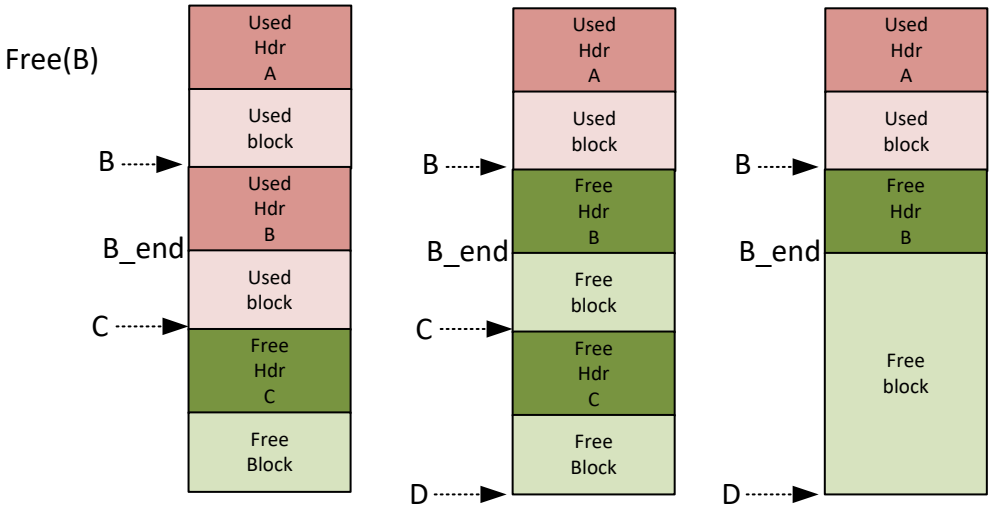
Push to Front.. No sort (fast)



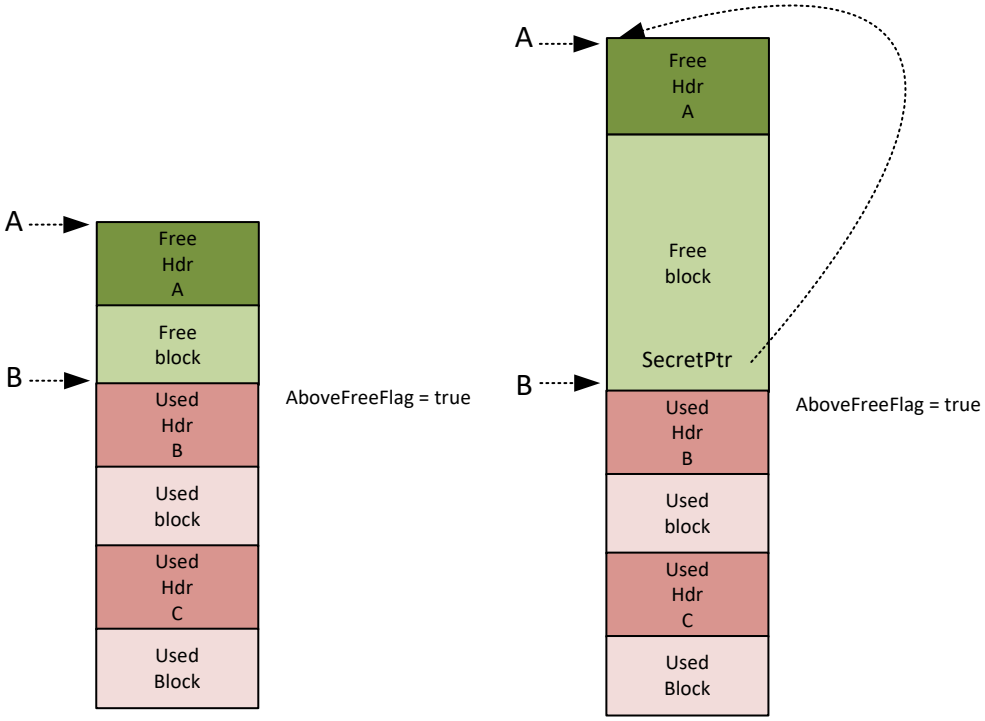


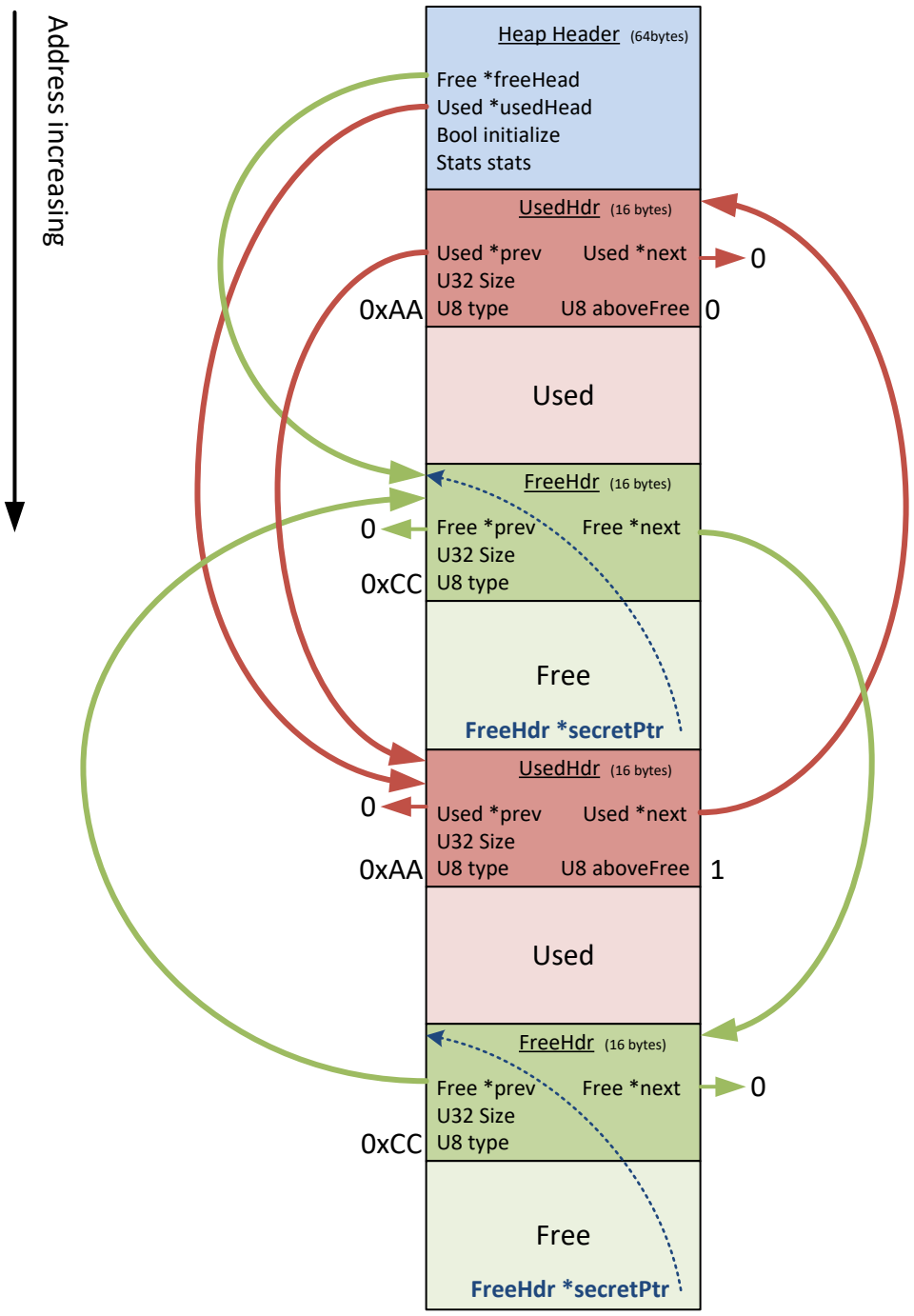
Mem::Initialize()




Free(B)
Used
Hdr
A
Used
block
B
Used
Hdr
B
B_end
Used
block
C
Free
Hdr
C
Free
Block
Used
Hdr
A
Used
block
B
Free
Hdr
B
B_end
Free
block
C
Free
Hdr
C
Free
Block
D
Used
Hdr
A
Used
block
B
Free
Hdr
B
B_end
Free
block
D

Free(B)


A
Free
Hdr
A
Free
block
B
Used
Hdr
B
Used
block
Used
Hdr
C
Used
Block
AboveFreeFlag = true
A
Free
Hdr
A
Free
block
SecretPtr
B
Used
Hdr
B
Used
block
Used
Hdr
C
Used
Block
AboveFreeFlag = true



Notes:

* Used blocks are unsorted, pushed to the head
* Free blocks are sorted smallest address at the front of list
* Used block size, Free block size does not include the header size
* Minimum allocation is multiple of 16 bytes
* Heap is aligned on creation, no need to align the heap after it has been initialized
* Two adjacent free blocks are coalesced into one large free block
* secretPtr is place at the bottom 32 bits of the free block, it points back to the freeHdr
* types – 0xAA used, 0xCC free