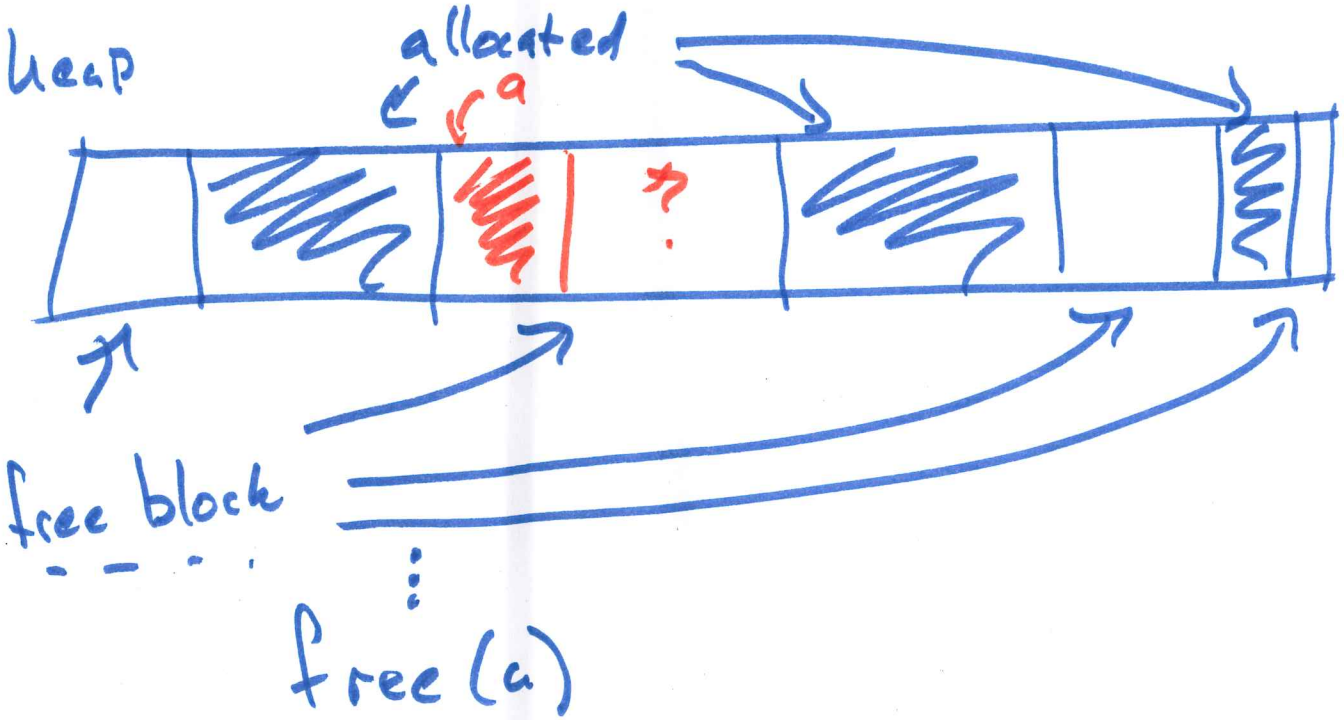
Memory allocator algorithms

four points to cover

1) How to track free blocks
2) How to choose free blocks when allocating
3) What to do w/ extra space after allocation
4) What to do w/ blocks when they are ~~free'd~~ freed

~~malloc~~ int* a = malloc(100)



- algorithms

cover

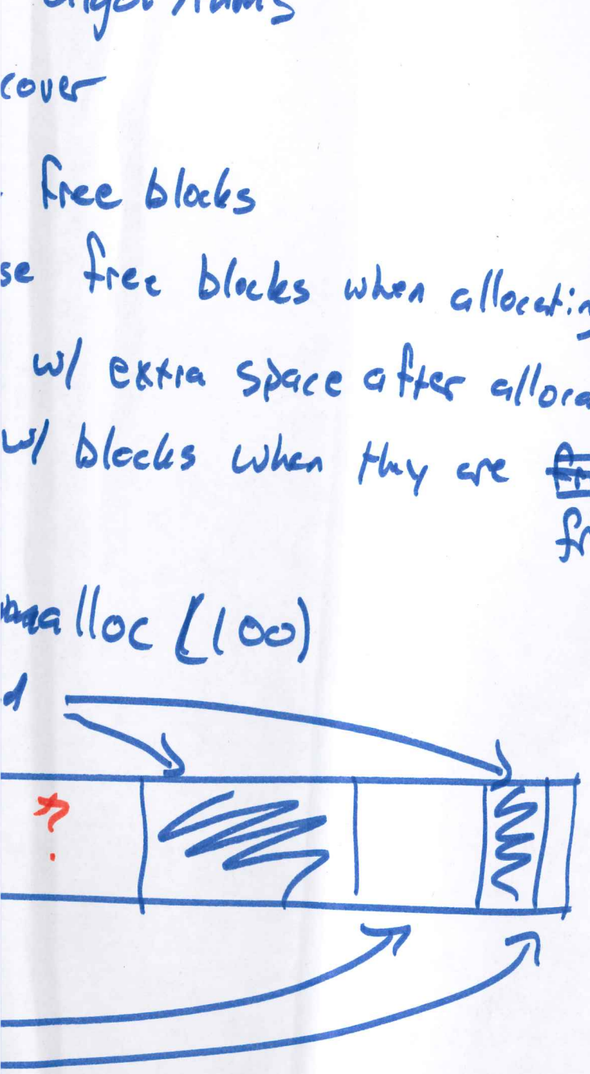
- free blocks

se free blocks when allocating

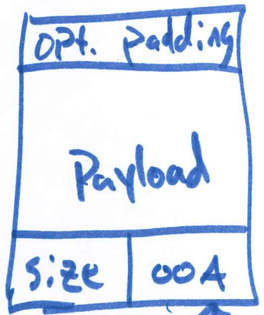
w/ extra space after allocation

w/ blocks when they are ~~free'd~~ freed

malloc (100)



block



addr

encoding allocation in size field

→ 8-byte aligned

1011010110
&1111111000
-> -1<<3

allocated bit

if size divisible by 8 (size & 1 == 0)

↳ unallocated

if size not divisible 8 or size & 1 == 1

↳ allocated

the "real" size of an allocated block

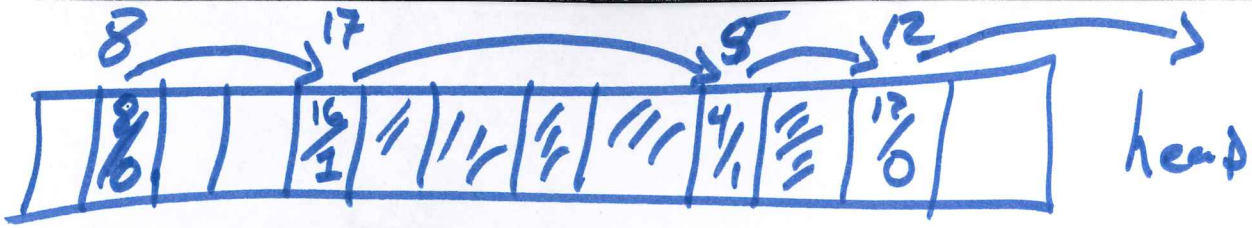
size & (-1<<3)

```
struct block{
    void * start;
    int size;
}
```

```
struct block * b;
b->size
```

↳ ~~100~~ 104

↳ 73 (real size)
72



size / allocated

---

2) how to choose a block?

Policy

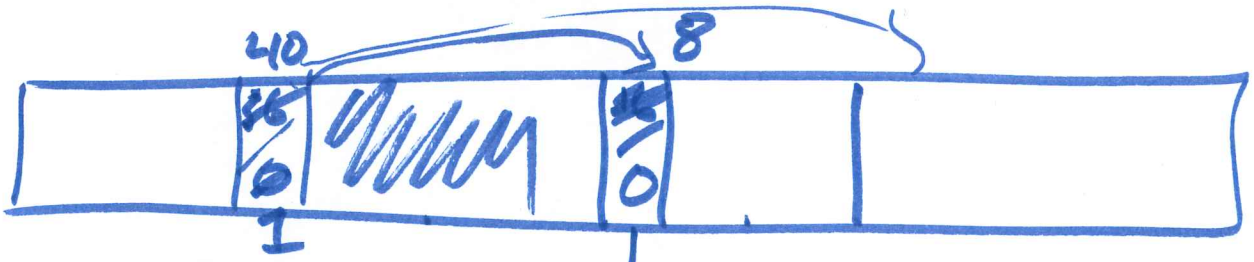
1) first-fit → first block that's big enough
2) best-fit → block big enough w/ least extra space
3) next-fit → "next" block that's big enough
   → start where you last allocated instead of beginning

- + fast
  - can be slow
  - fragmentation
- + least frag.
  - slow
- ~ ok fragmentation
  + fast

---

3) Extra space after allocation

- use the whole block → internal fragmentation
- split block into 2 blocks



a = malloc (40)

free(a)

↳ can't use whole 16 bytes because need space for header

---

4)

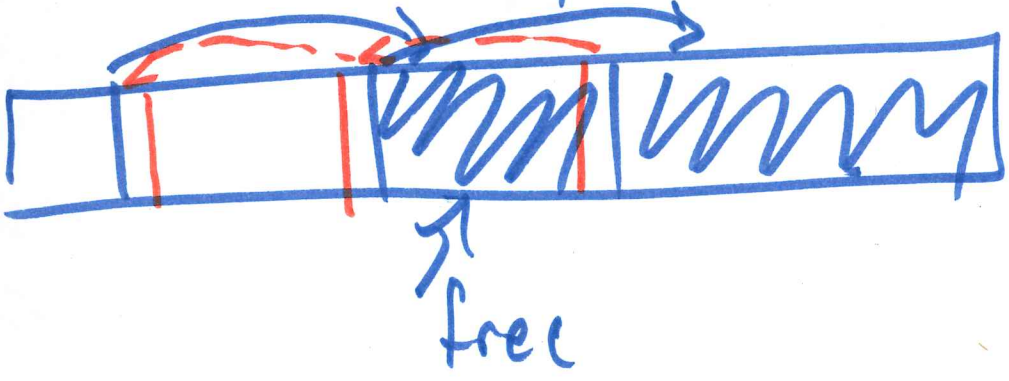
What do we do when freeing

- Coalesce free blocks together

Straight forward if <u>next</u> block is also free.

- alloc speed
- coalesce on allocation
- Can have footer to point back
  - ↳ space
  - ↳ only in free blocks

What about prev. block?



Don't coalesce? → <u>false</u> fragmentation

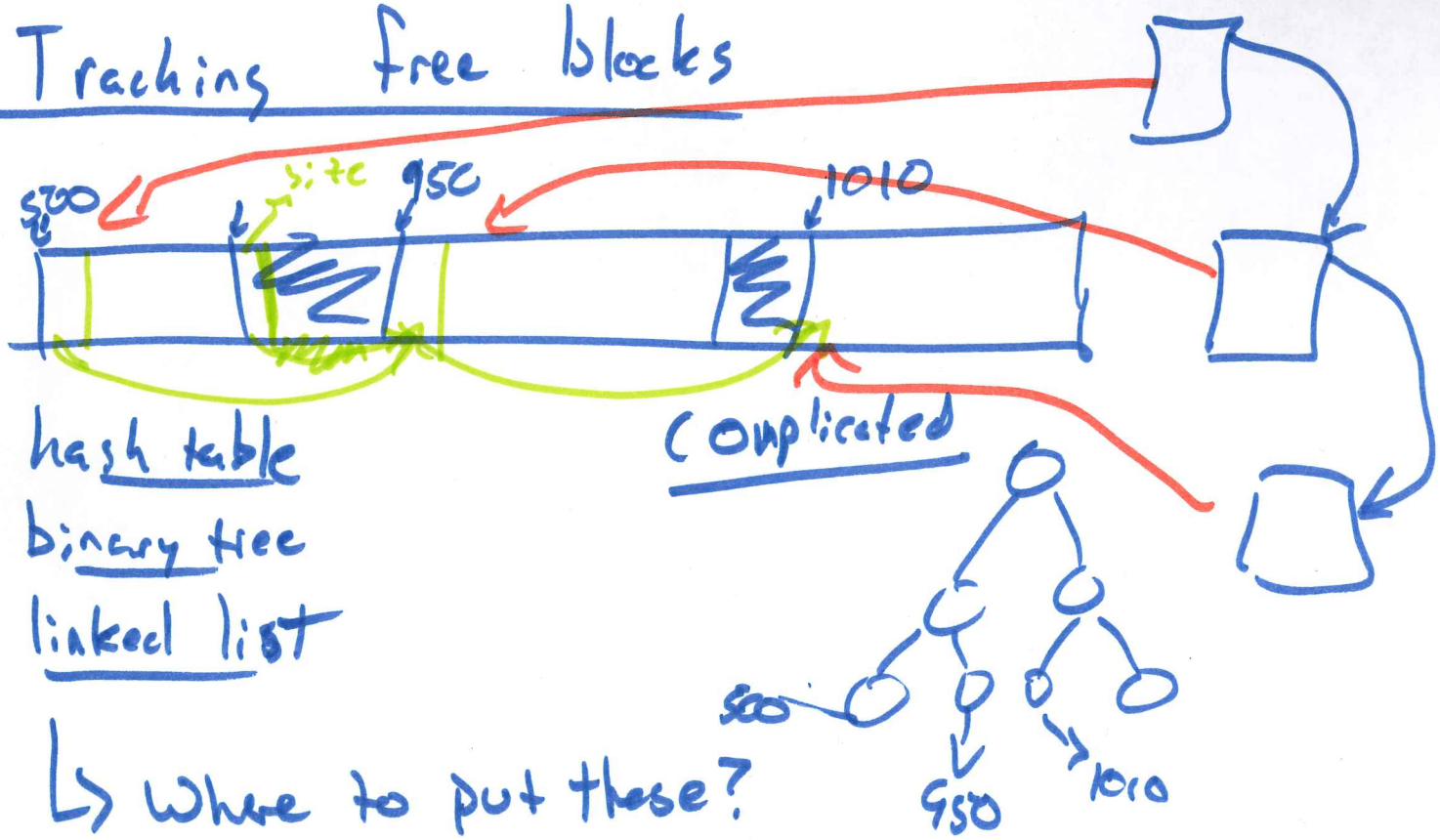
- Traverse the whole list

Putting it all together -> assignment 5

Mem_alloc } you implement
Mem_free }

- ↳ variation of implicit free list
- ↳ Do NOT change headers/structs
- ↳ test-driven development
- ↳ best-fit algorithm
- ↳ linked-lists
- ↳ I did it ~ 100 LOC
  - ↳ lots of tricky pointer arithmetic

Tracking free blocks



↳ Where to put these?

```
struct free_node {
    struct free_node *next;
    void * free_block;
    int size;
}
```

Explicit free lists

Implicit free lists