interfaces

size_t

malloc(sz) → ptr

free(ptr) → ?

Q malloc(0) → ptr

Q malloc(-1) → NULL

Q. ptr = malloc(sz); *ptr

Q. *(ptr + sz)

] ← Ouf

Q. free(NULL); free(ptr) ?

Q. free(ptr); free(ptr) ← double free (DF)

Q. free(ptr); ptr ≠ NULL? *ptr (use-after-free)

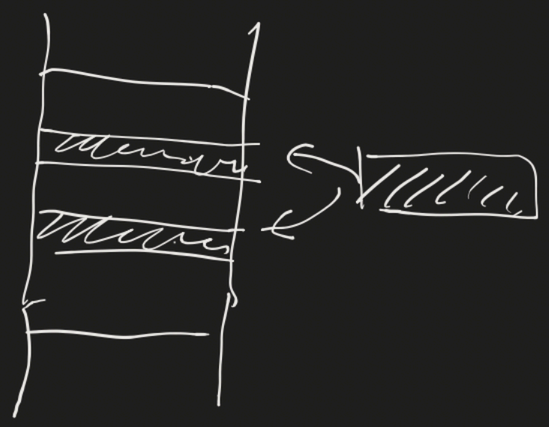
goals

1) performace

2) utilization ; fragmentation

- internal

- external



3) security

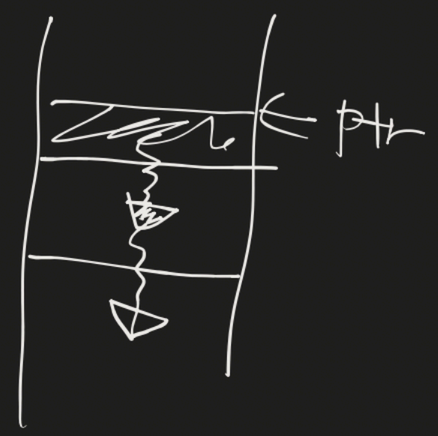
V0. simplest (secure)

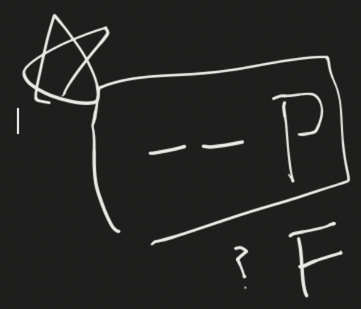
10 B

malloc() → mmap() ↙ ?

free(ptr) → mUnmap()

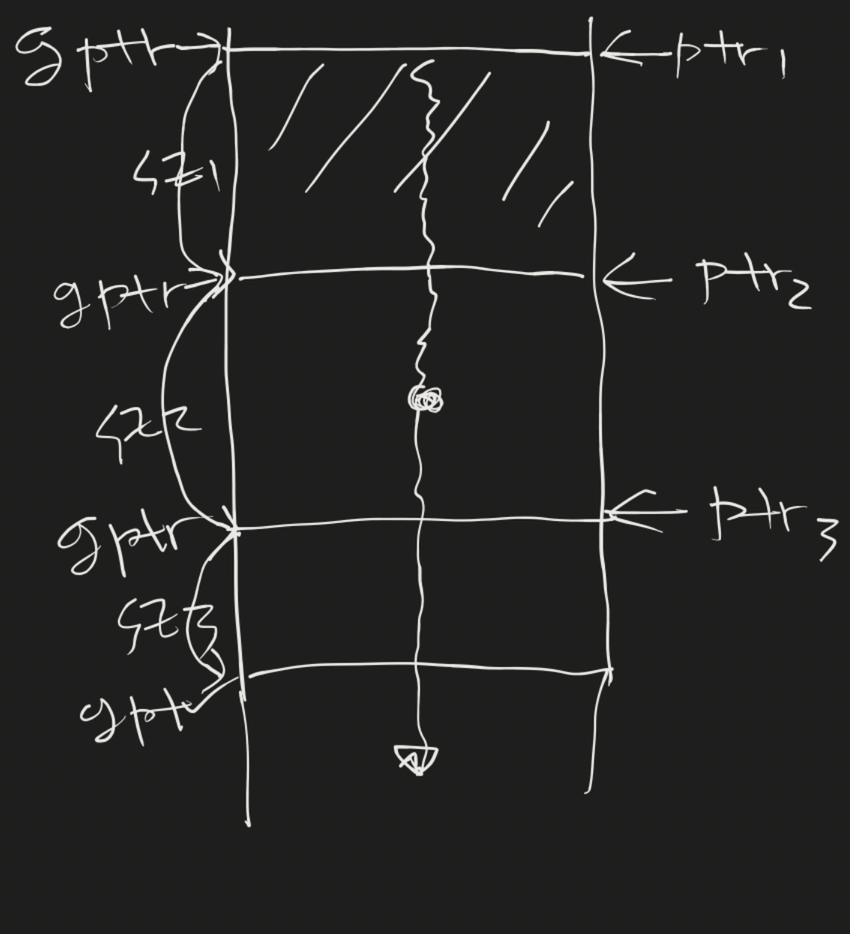
4KB





++ S

V1. fastest heap allocator



ptr1 = malloc(sz1)

ptr2 = malloc(sz2)

~~free(ptr2)~~ nop

ptr3 = malloc(sz3)

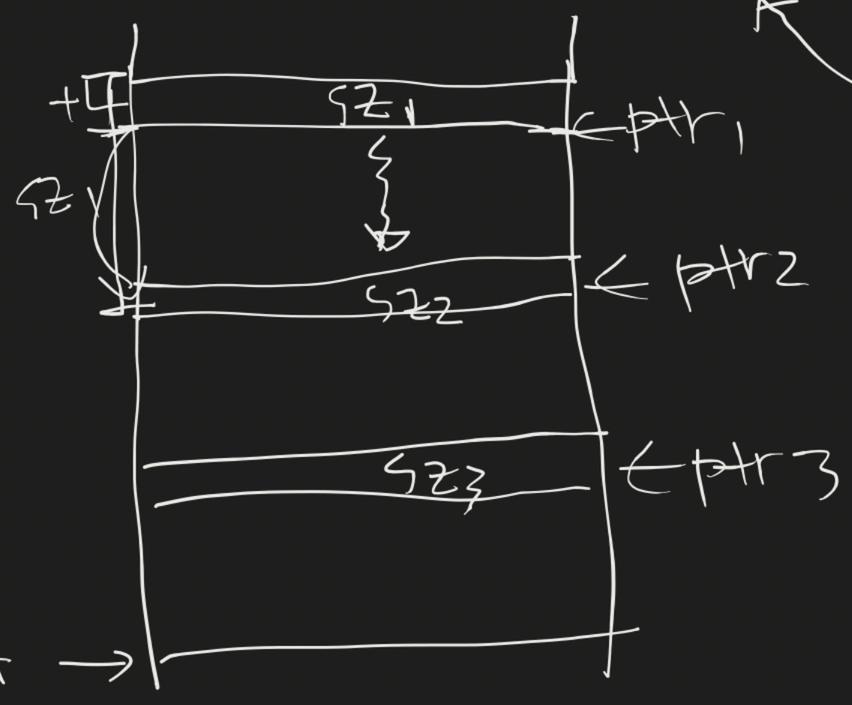
++p.

--F. (+I, -E)

+S. (-O, NDF, NUA)

L2. reusing freed memory

freelist = [ptr2, ...]




```
ptr1 = malloc(sz1)
ptr2 = malloc(sz2)
free(ptr2)
ptr3 = malloc(sz3)
free(ptr3)
```

def malloc(sz):

```
for f in freelist:
    if sz <= f->sz:
        unlink(f)
        return f
ptr = gptr
gptr += sz
return ptr
```

def free(ptr):

```
append(ptr)
```

1) in-place
2) out-of-band

- P

F (-IF, -EF)

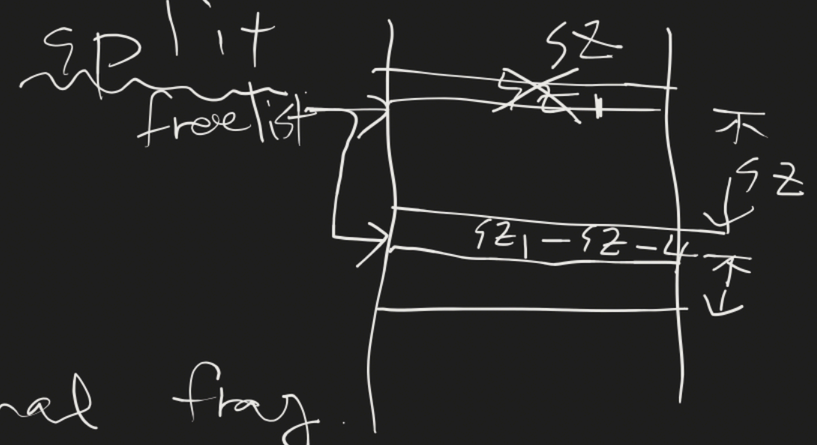
O (Ovfl, -UAF, -DF)

V3. handling fragmentation

1) Internal frag

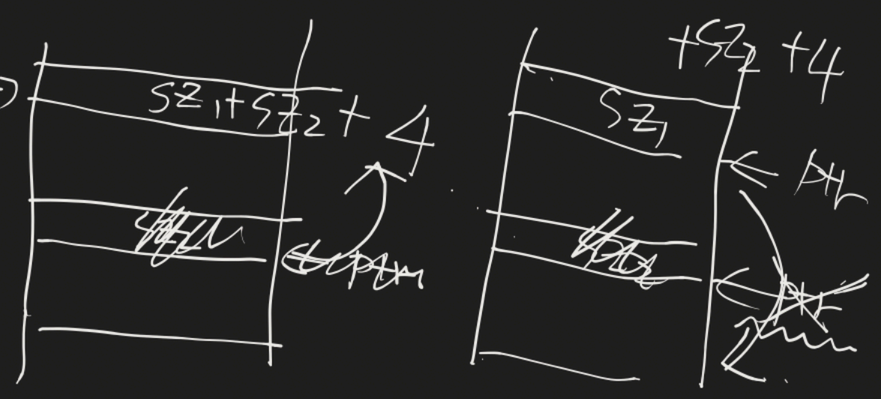
)) iterate entire freelist;

2) split



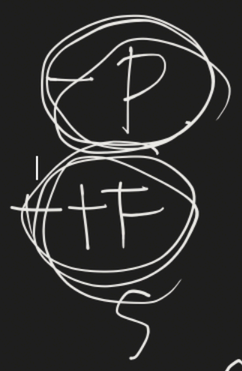
de malloc(sz):

2) External frag



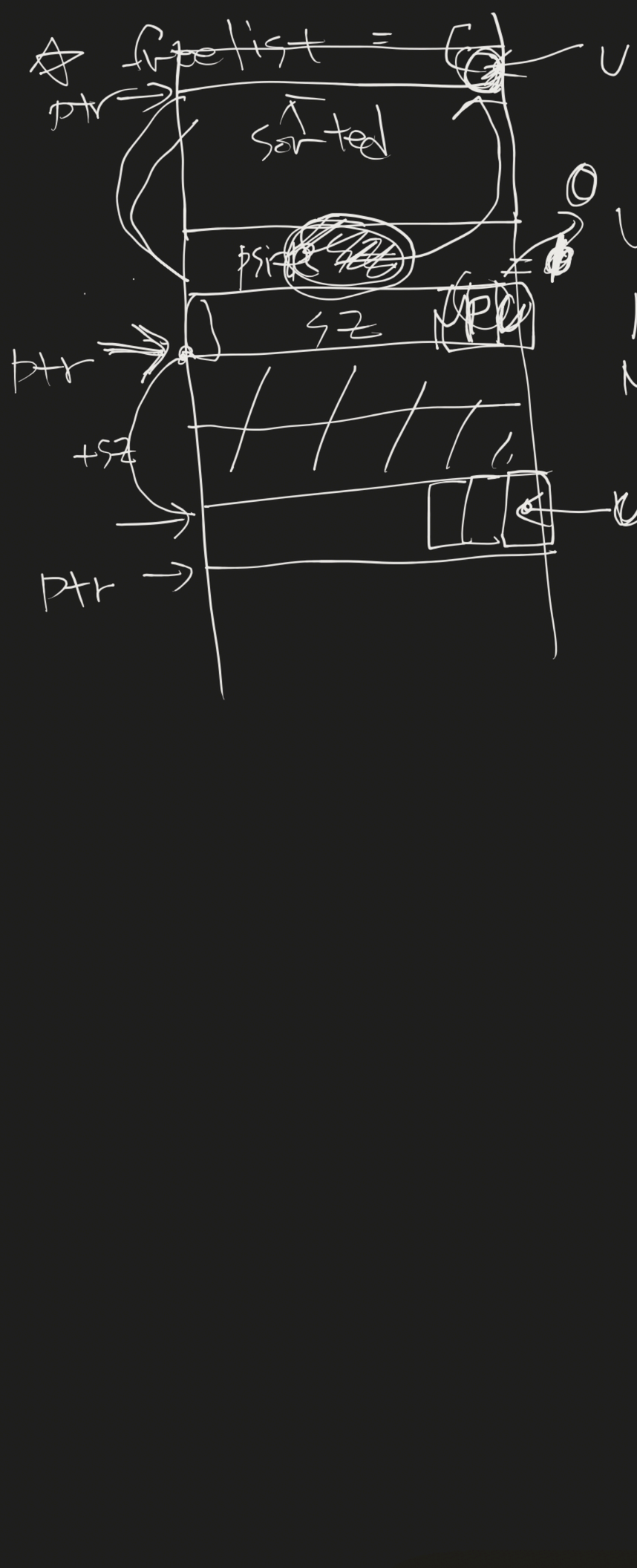
f/bk consolidate

def free(ptr):



OV -DF -UAF

V4. Optimizations



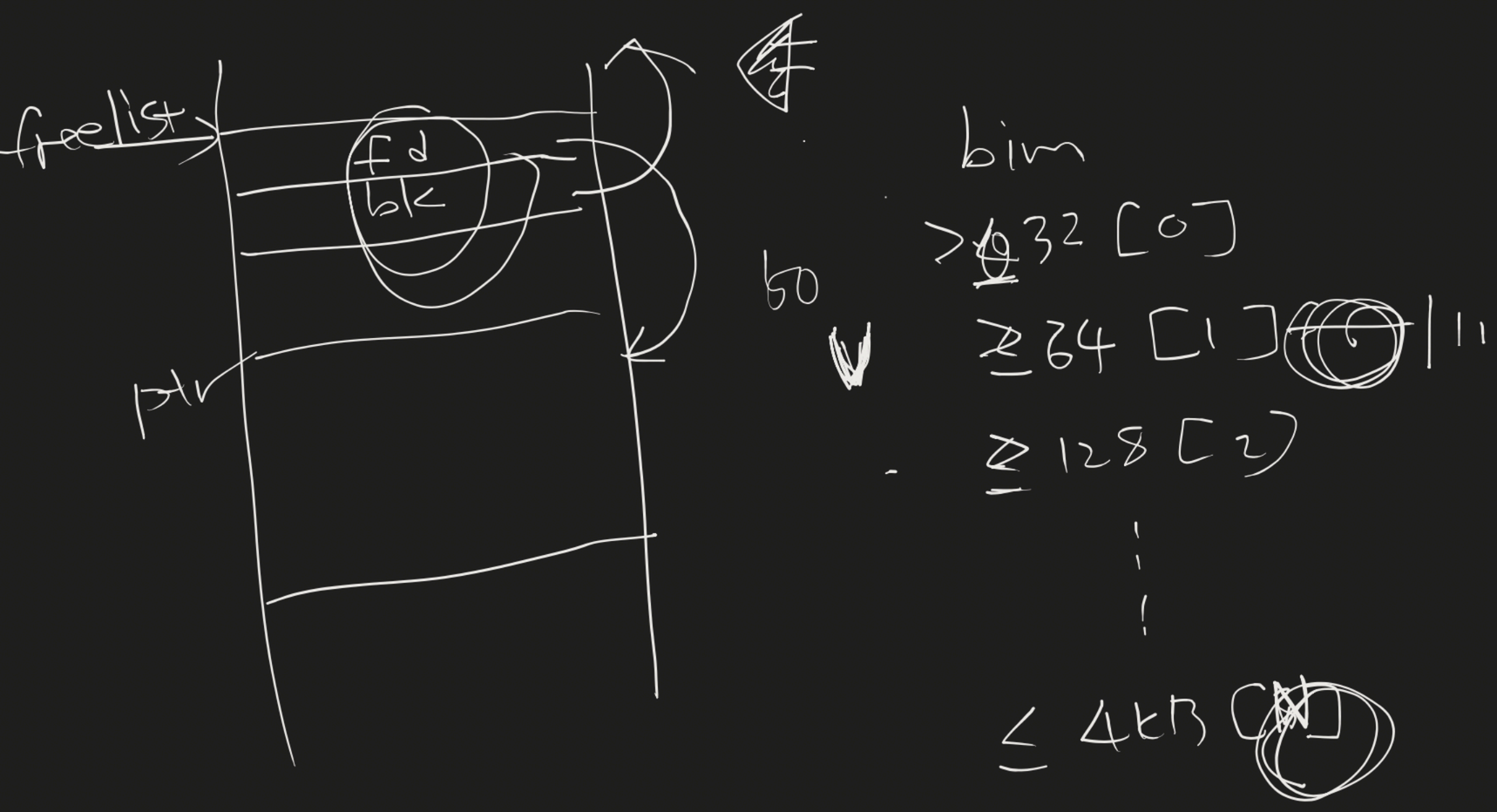
] O(log n)

U : In-use bit (=1, =0)

P : prev In-use (=1, =0)

M : Mmagic

*(ptr + sz) & 1 = U



- bitmap
- fastbin ≪ cache tcache