Compacting the Uncompactable!

MESH

Automatically Compacting
Your C/C++ Application's Memory (& more!)

Emery Berger
UMass Amherst

with Bobby Powers, David Tench, & Andrew McGregor
University of Massachusetts Amherst
http://libmesh.org

[PLDI 2019]

PIANO • VOCAL • GUITAR

The Impossible Dream
(The Quest)

Lyrics by JOE DARIEN
Music by MITCH LEIGH

PIANO · VOCAL · GUITAR

The Impossible Dream
(The Quest)

Lyrics by JOE DARIEN
Music by MITCH LEIGH

Reconquer
all of Spain!

MALLOC

Reconquer
all of Spain!

MALLOC

~~Reconquer~~ Allocate all of Spain!


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MALLOC
OF LA MANCHA


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MALLOC
MAN OF LA MANCHA


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARIEN
Music by MITCH LEIGH
MAN OF LA MANCHA

MALLOC


(malloc)


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARIEN
Music by MITCH LEIGH
MALLOC
MAN OF LA MANCHA


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARIEN
Music by MITCH LEIGH
MALLOC
MAN OF LA MANCHA


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MALLOC
MAN OF LA MANCHA


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MAN
OF LA MANCHA

MALLOC


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MALLOC
MAN OF LA MANCHA


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MAN OF LA MANCHA

MALLOC


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MAN
OF LA MANCHA

MALLOC


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MAN OF LA MANCHA

MALLOC


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MALLOC
MAN OF LA MANCHA

PIANO · VOCAL · GUITAR

The Impossible Dream
(The Quest)

Lyrics by JOE DARIEN
Music by MITCH LEIGH

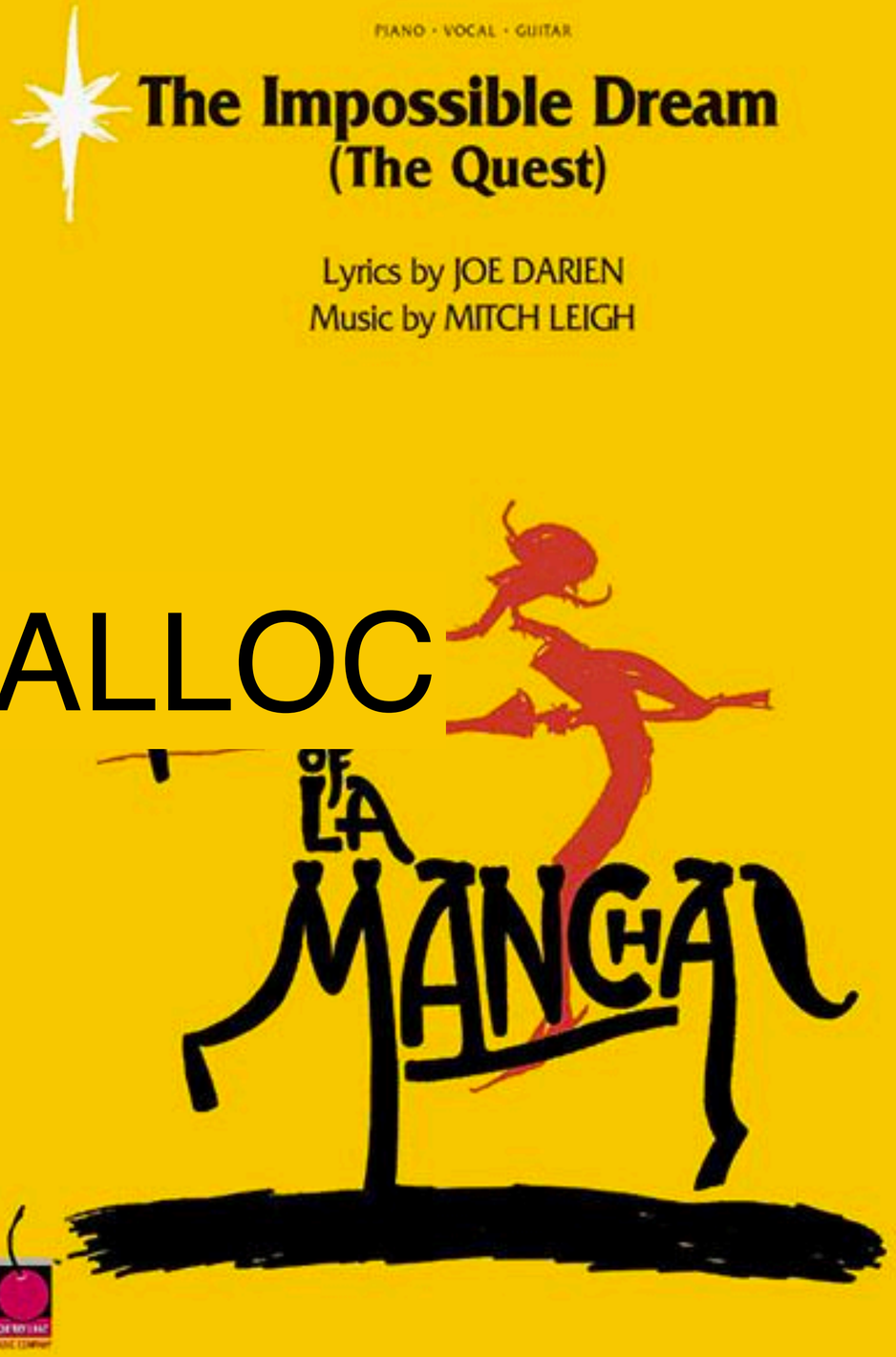
MALLOC

PIANO · VOCAL · GUITAR

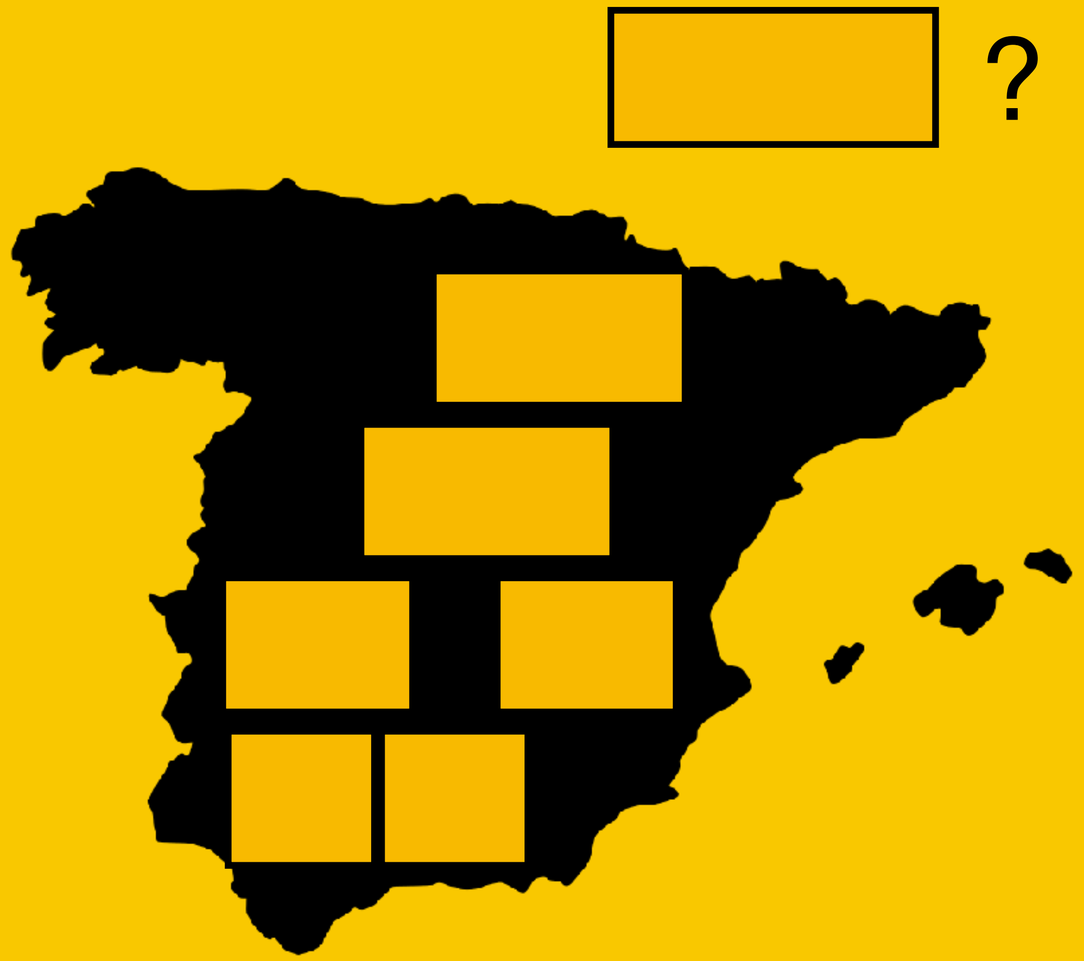
The Impossible Dream
(The Quest)

Lyrics by JOE DARION
Music by MITCH LEIGH

MALLOC

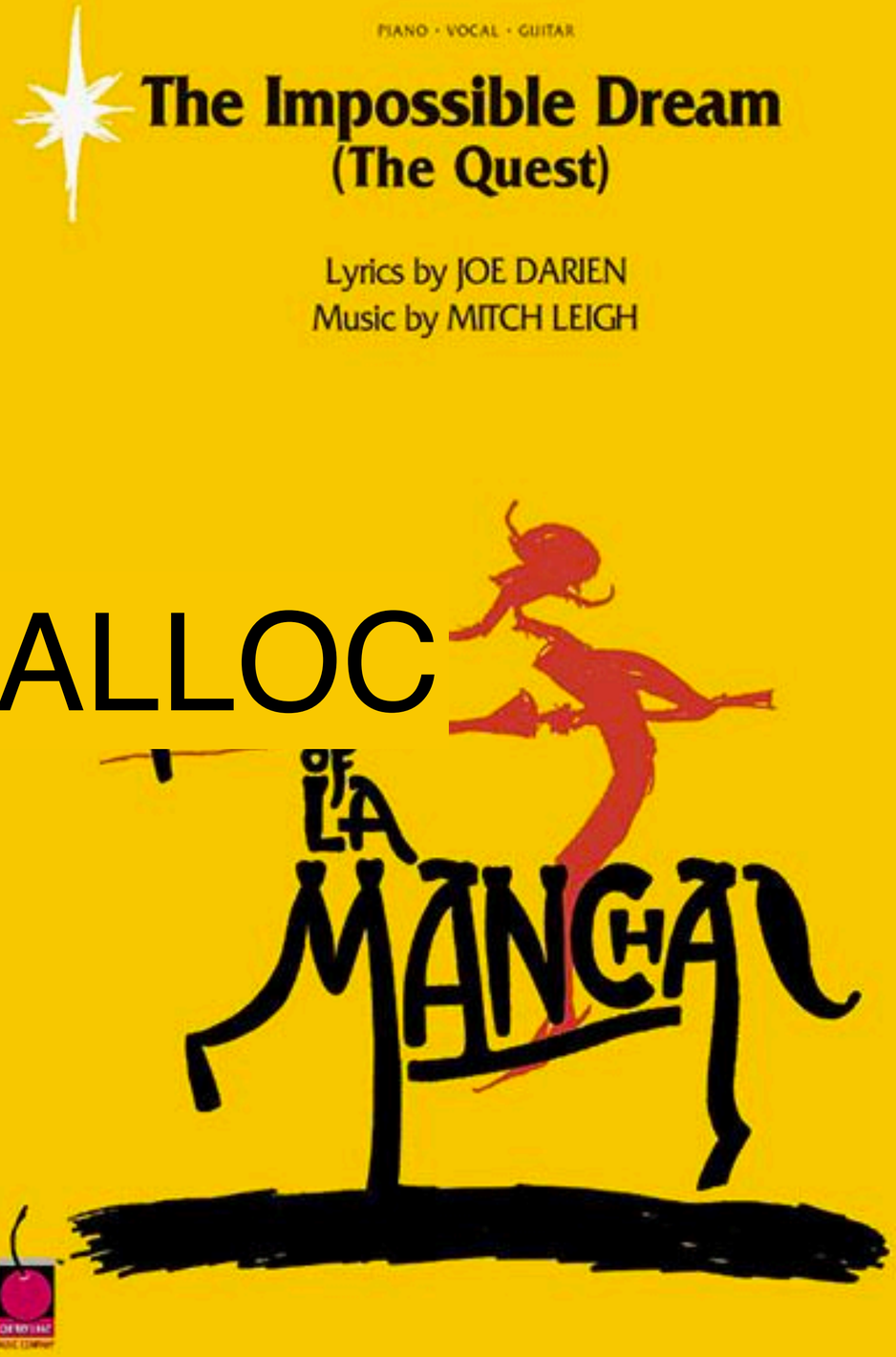

PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MAN
OF LA MANCHA

MALLOC


?


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MALLOC
MAN OF LA MANCHA
?

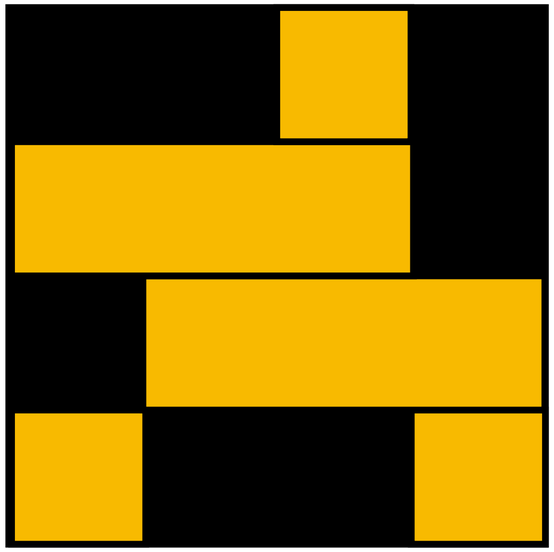

PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MALLOC
!
MAN OF LA MANCHA


PIANO · VOCAL · GUITAR
The Impossible Dream
(The Quest)
Lyrics by JOE DARION
Music by MITCH LEIGH
MAN OF LA MANCHA

MALLOC

¡ !

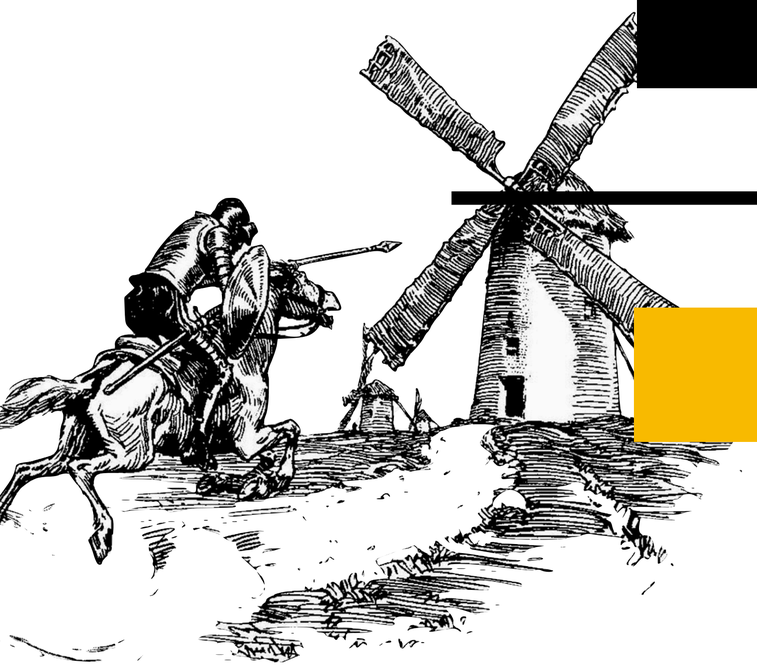
¡Fragmentación!

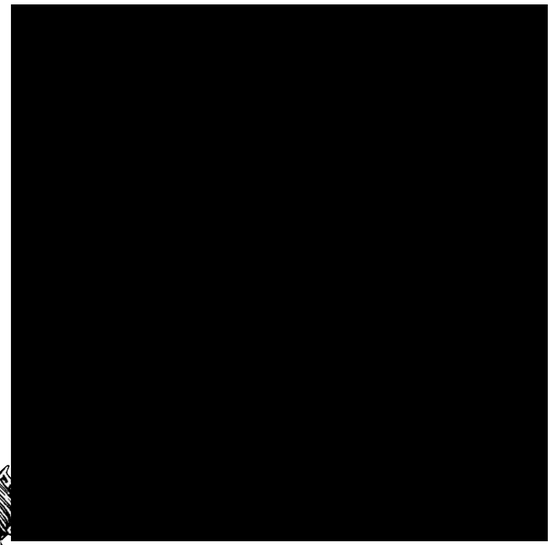


OS Page (“Spain”)

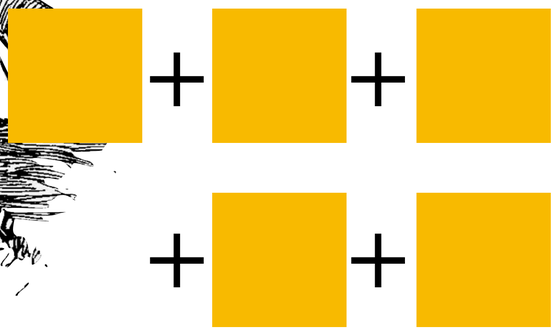
Live objects (malloc’d)

¡Fragmentación!



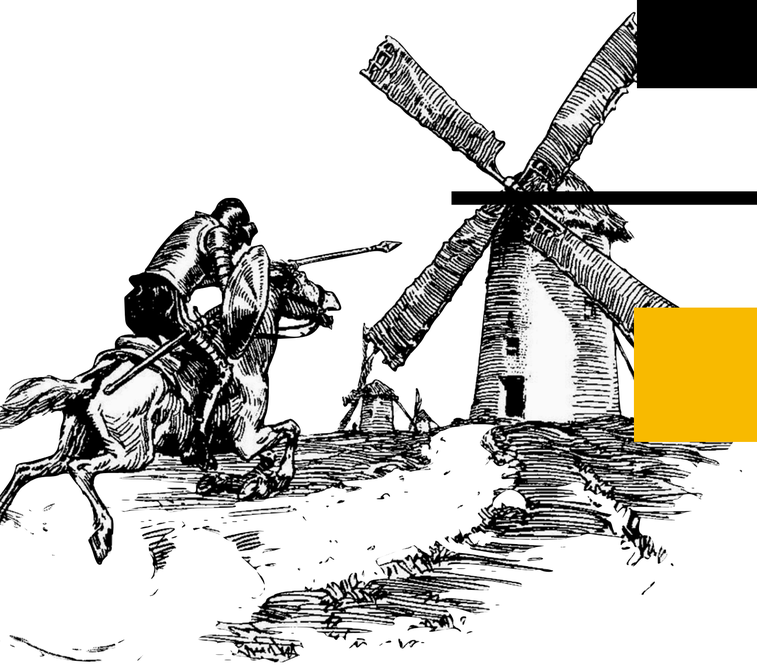


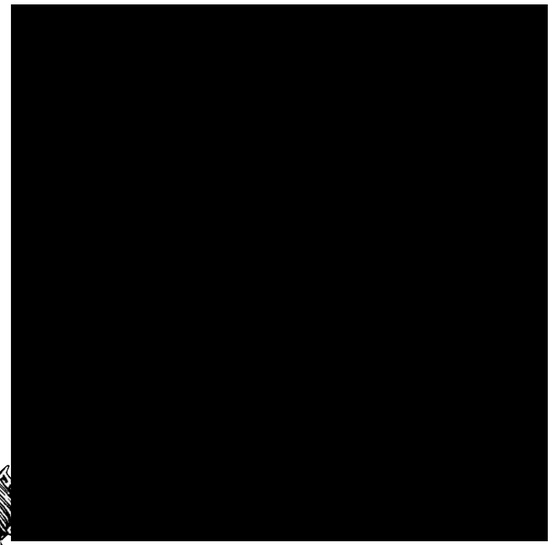
OS Pages



Live objects

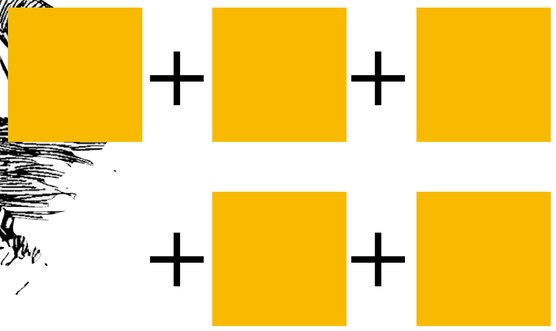
¡Fragmentación!





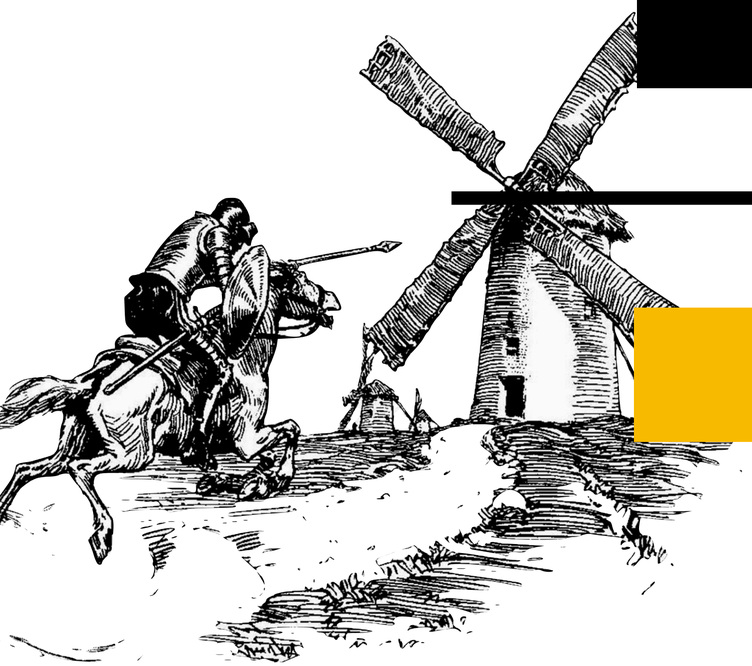
OS Pages

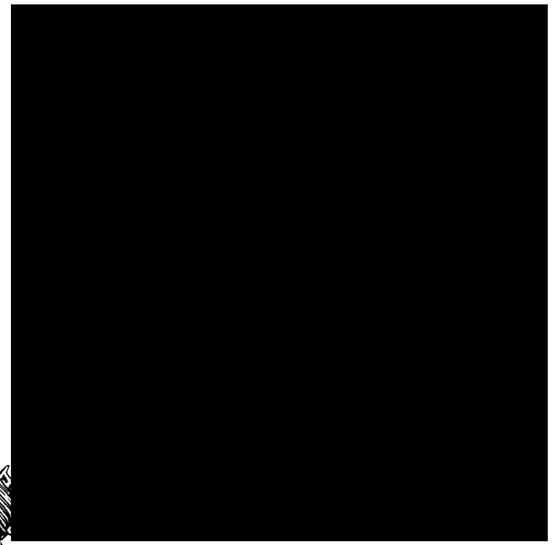
≫ 1



Live objects

¡Fragmentación!

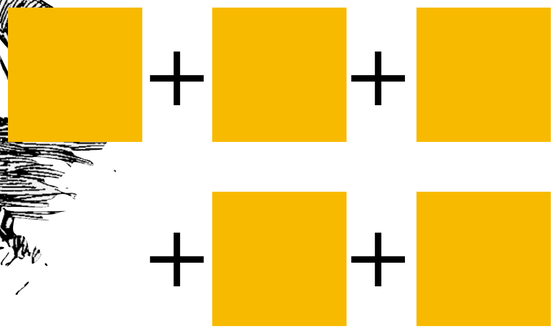




OS Pages

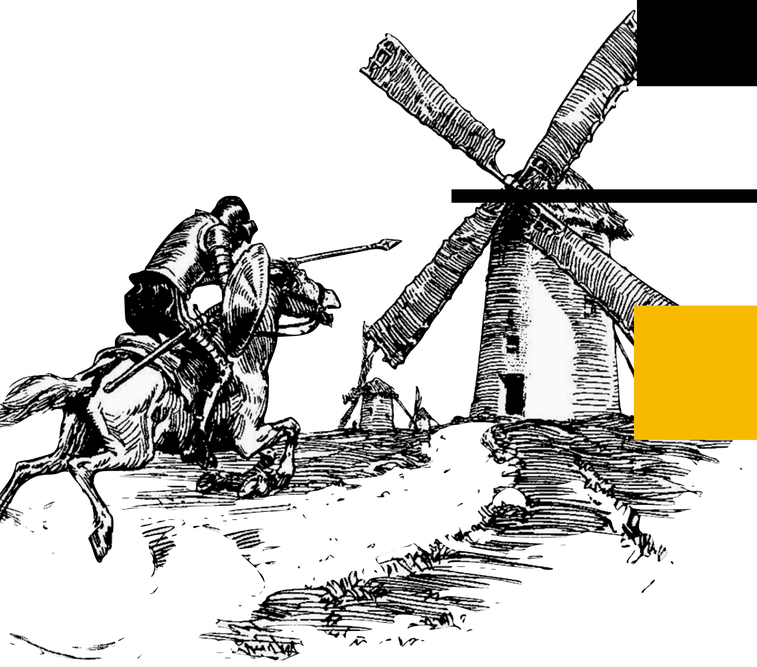
O(log ) 13x!

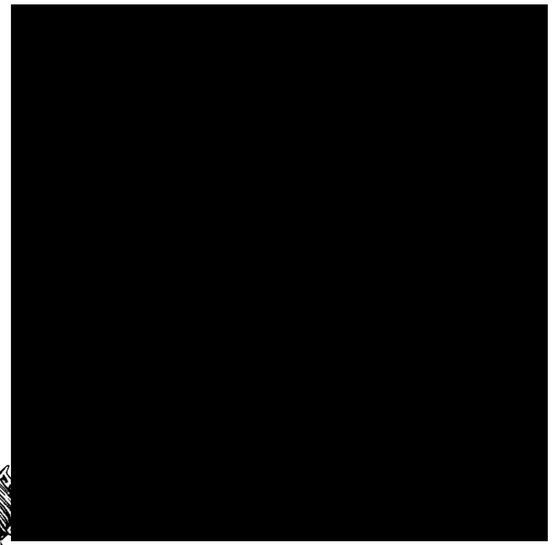
[Robson ’77]



Live objects

¿Compactación?

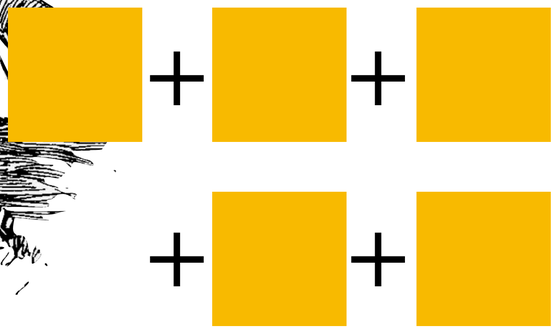




OS Pages

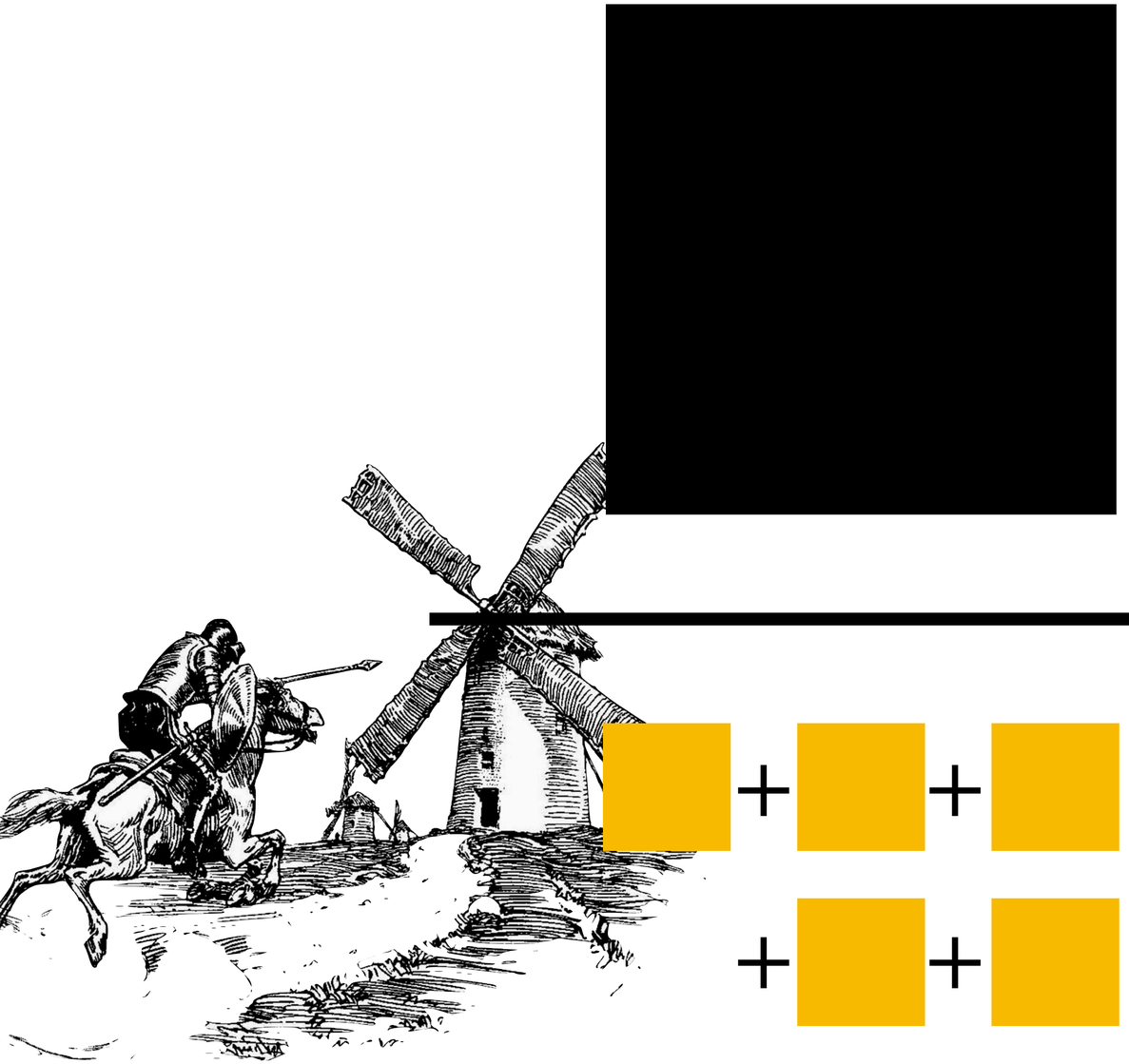
O(log ) 13x!

[Robson ’77]



Live objects

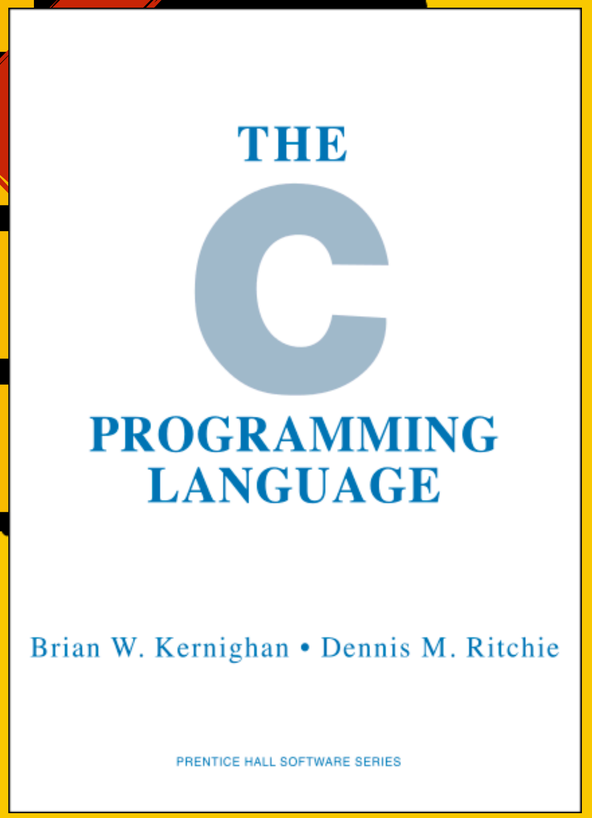
¿Compactación?

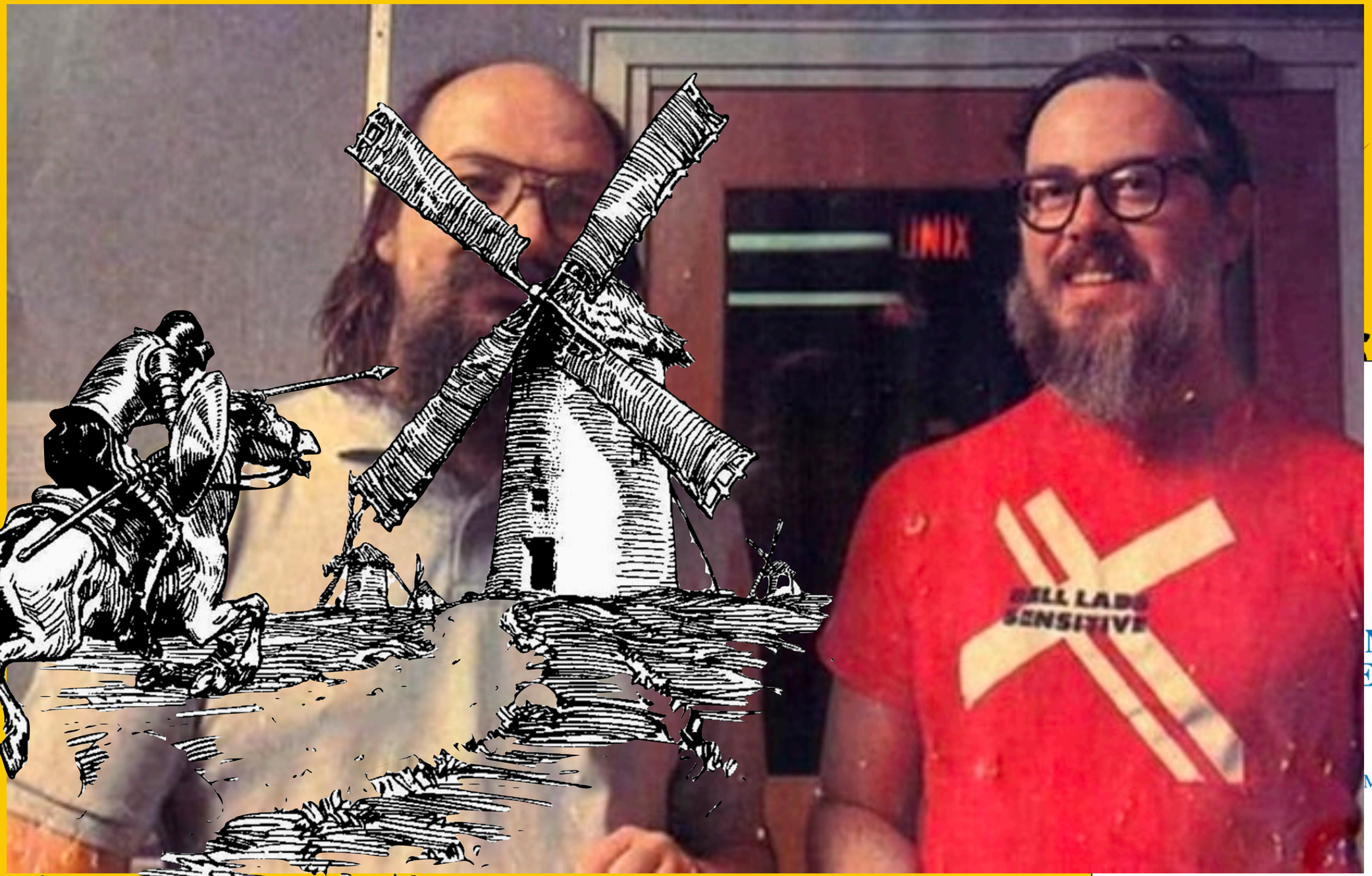


OS Pages

O(1)

Live objects


THE
C
PROGRAMMING
LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES



NG
E

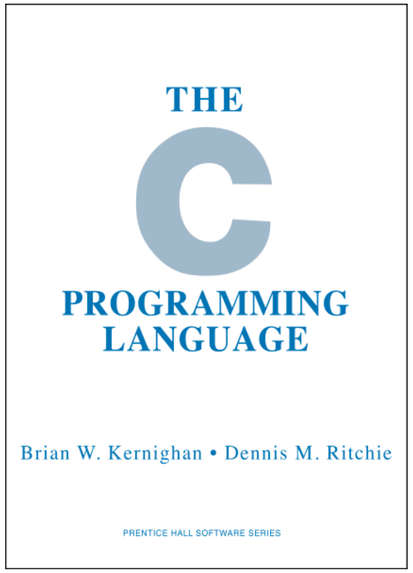
M. Ritchie

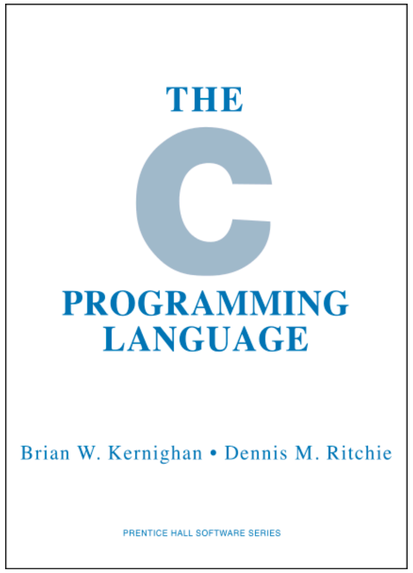
NG
E

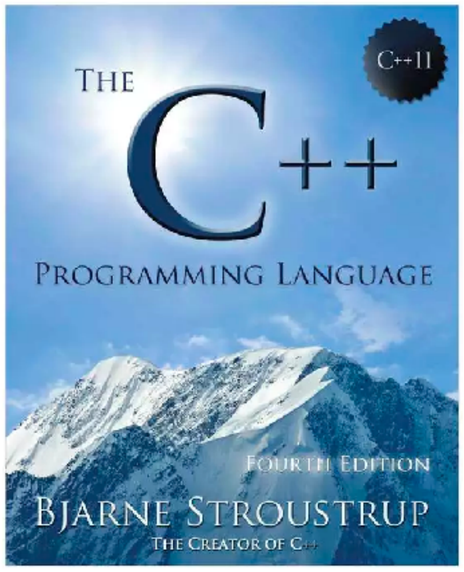
M. Ritchie

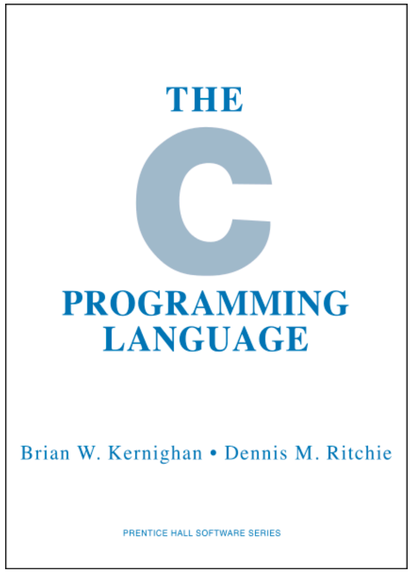
NG
E

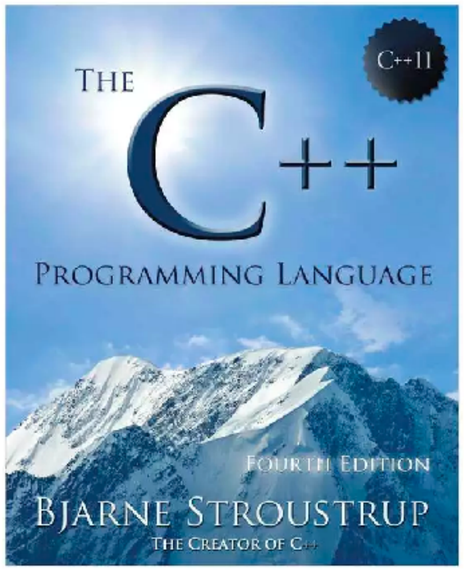
M. Ritchie


THE
C
PROGRAMMING
LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES

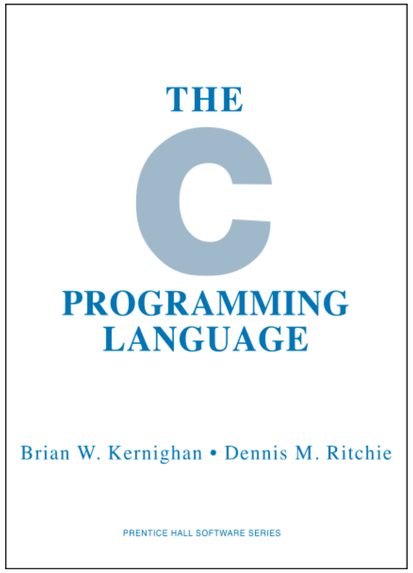

THE
C
PROGRAMMING
LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES


THE
C++
C++11
PROGRAMMING LANGUAGE
FOURTH EDITION
BJARNE STROUSTRUP
THE CREATOR OF C++


THE
C
PROGRAMMING
LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES


THE
C++
C++11
PROGRAMMING LANGUAGE
FOURTH EDITION
BJARNE STROUSTRUP
THE CREATOR OF C++


The Swift
Programming
Language
Swift 5 Edition



[Lattner, 2016]

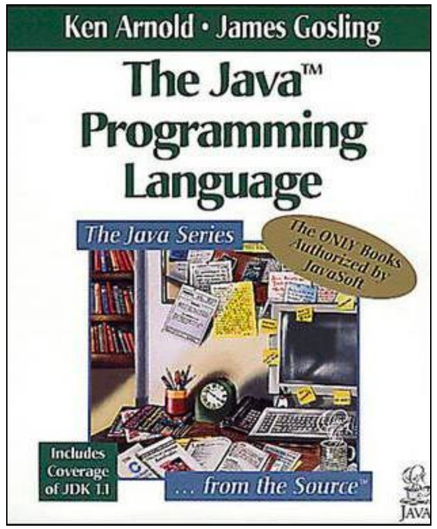
Why not a tracing GC?

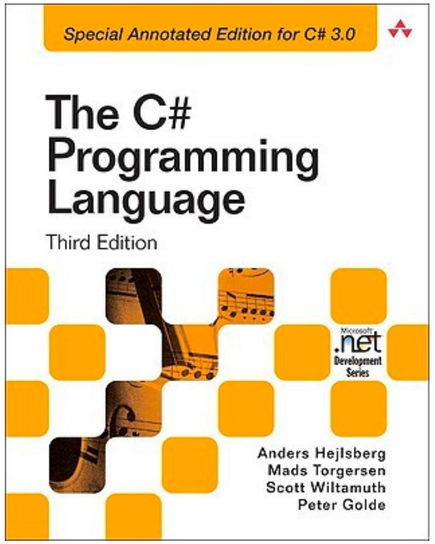
- Native interoperability with unmanaged code
- Deterministic destruction provides:
  - No "finalizer problems" like resurrection, threading, etc.
  - Deterministic performance: can test/debug performance stutters
- Performance:
  - GC use ~3-4x more memory than ARC to achieve good performance
  - Memory usage is very important for mobile and cloud apps
  - Incremental/concurrent GCs slow the mutator like ARC does

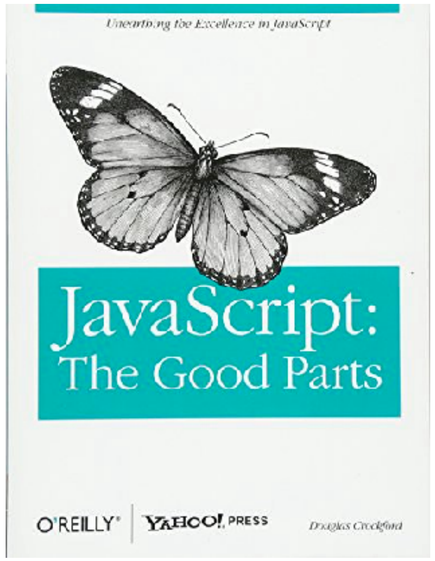
Quantifying the Performance of Garbage Collection vs. Explicit Memory Management
Matthew Hertz, Emery D. Berger. OOPSLA'05

- Performance:
  - GC use ~3-4x more memory than ARC to achieve good performance
  - Memory usage is very important for mobile and cloud apps
  - Incremental/concurrent GCs slow the mutator like ARC does

Quantifying the Performance of Garbage Collection vs. Explicit Memory Management
Matthew Hertz, Emery D. Berger. OOPSLA'05

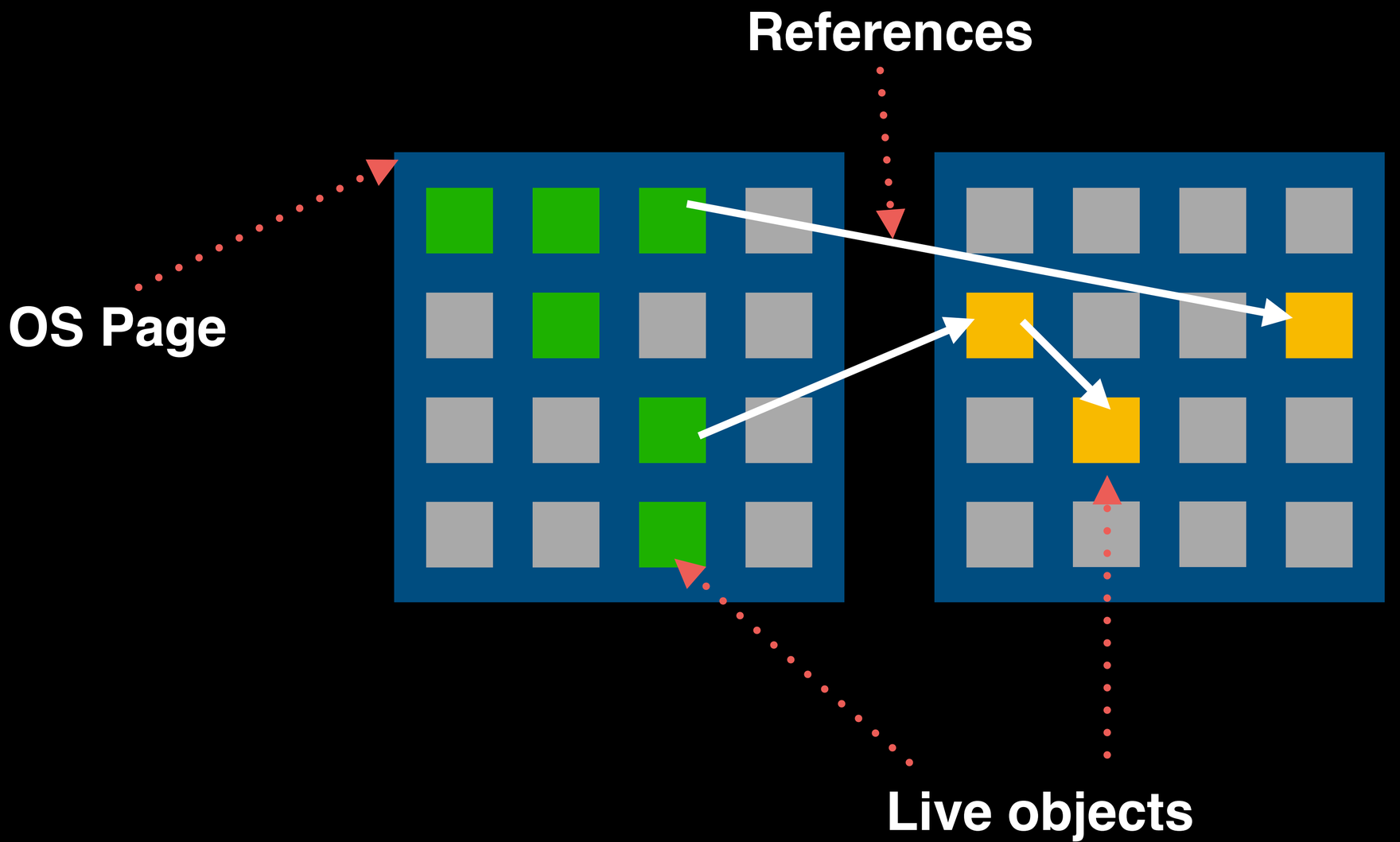


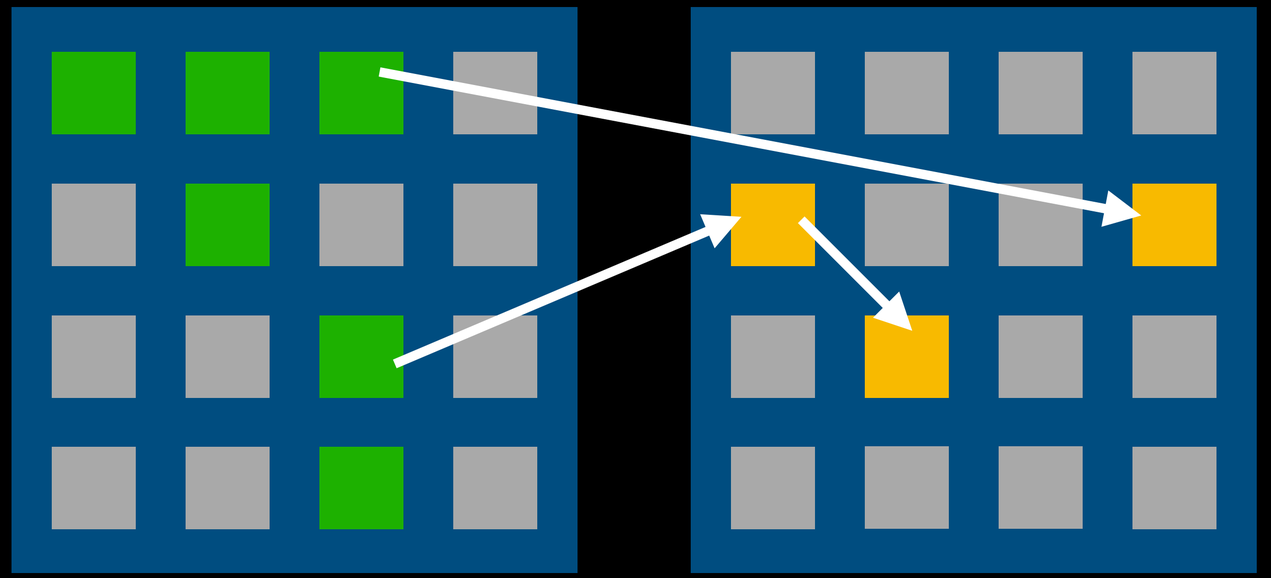


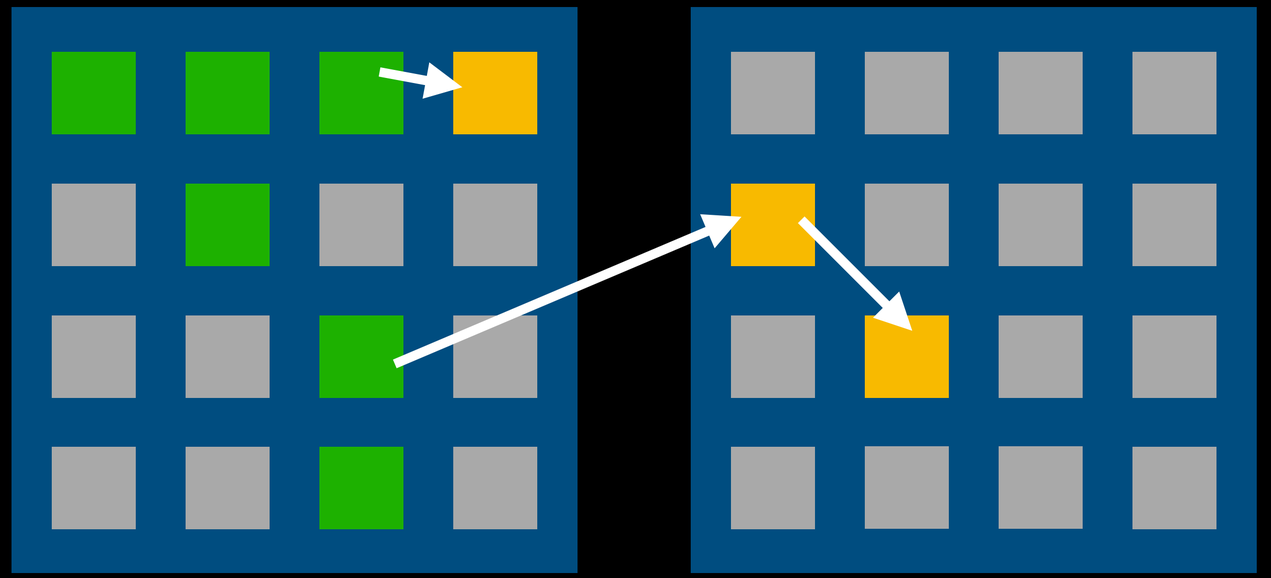


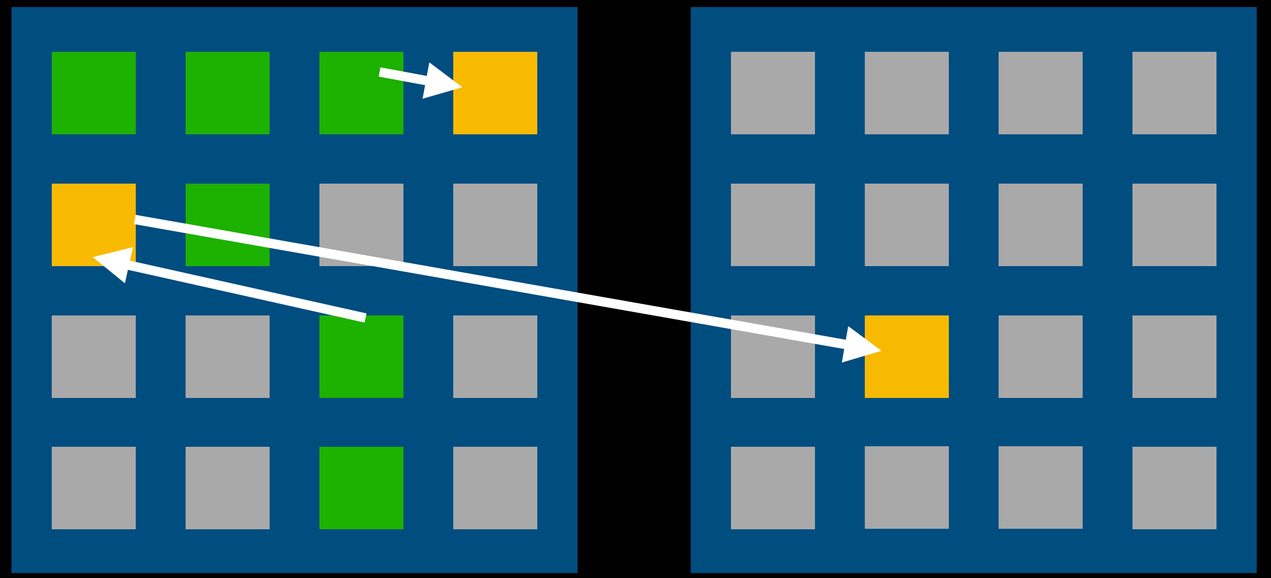
¡Compactación!

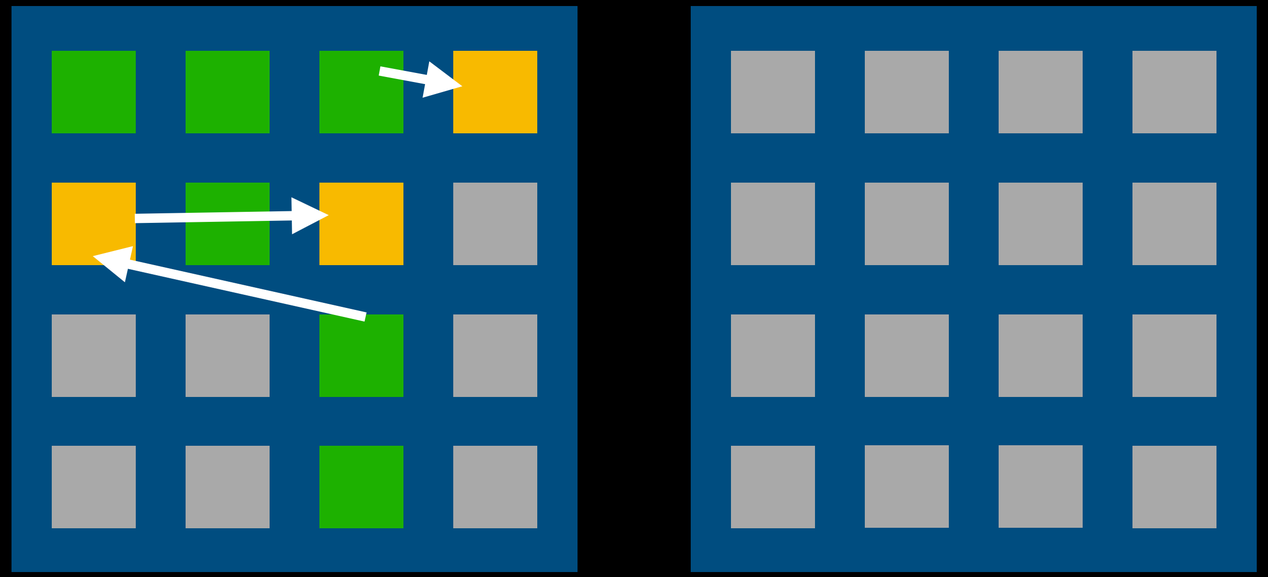
Compaction In Action

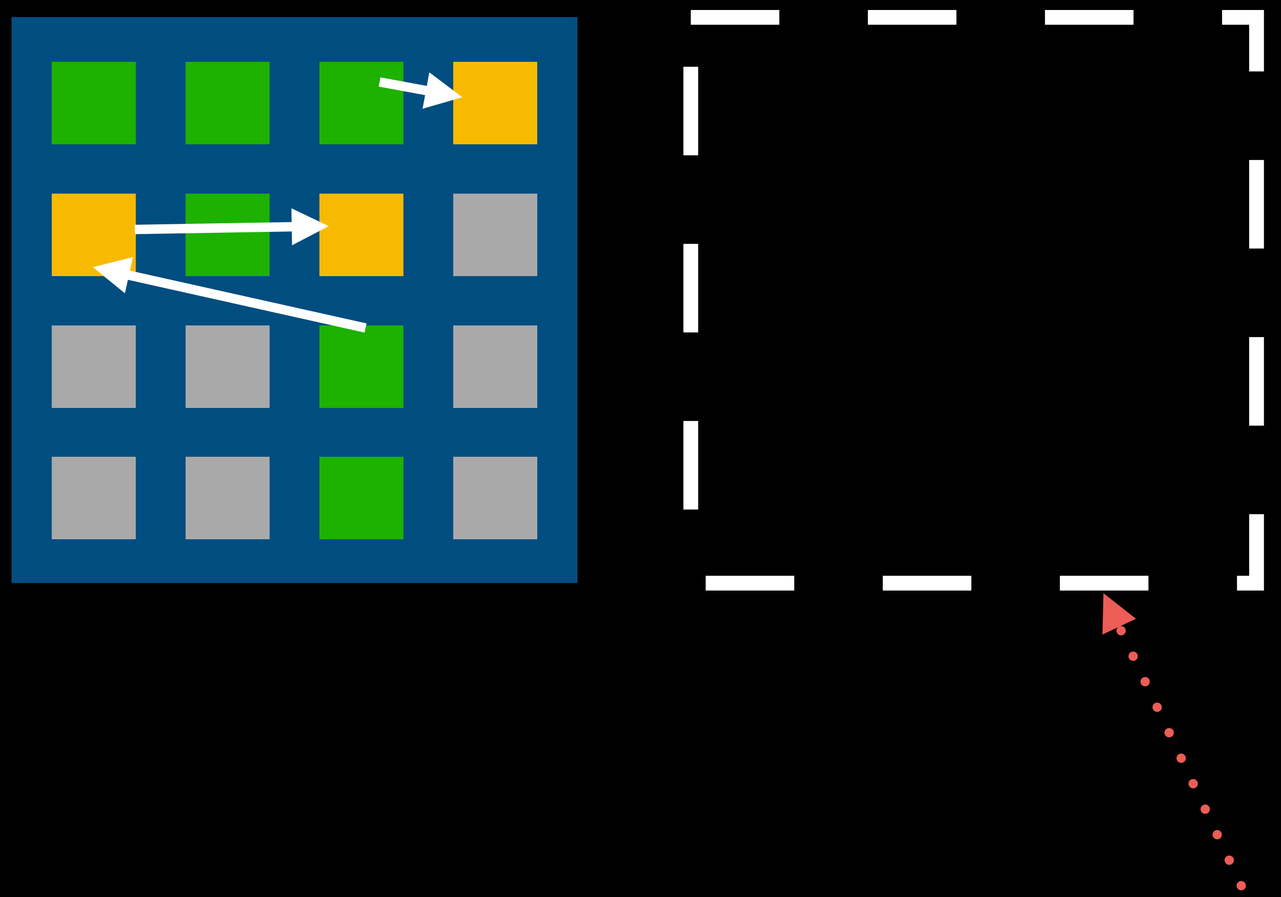












Page returned
to the OS

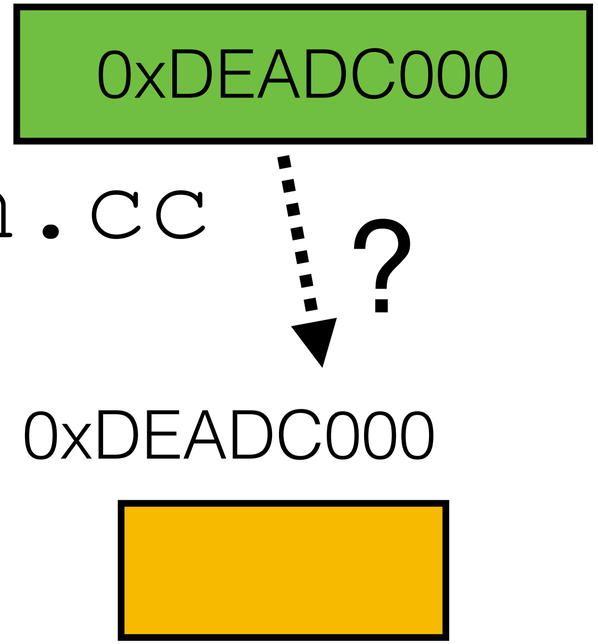
```
$ clang++ -o yolo main.cc
$ strip yolo
$ ./yolo
```

No way to precisely distinguish pointers from integers

```
$ clang++ -o yolo main.cc
$ strip yolo
$ ./yolo
```

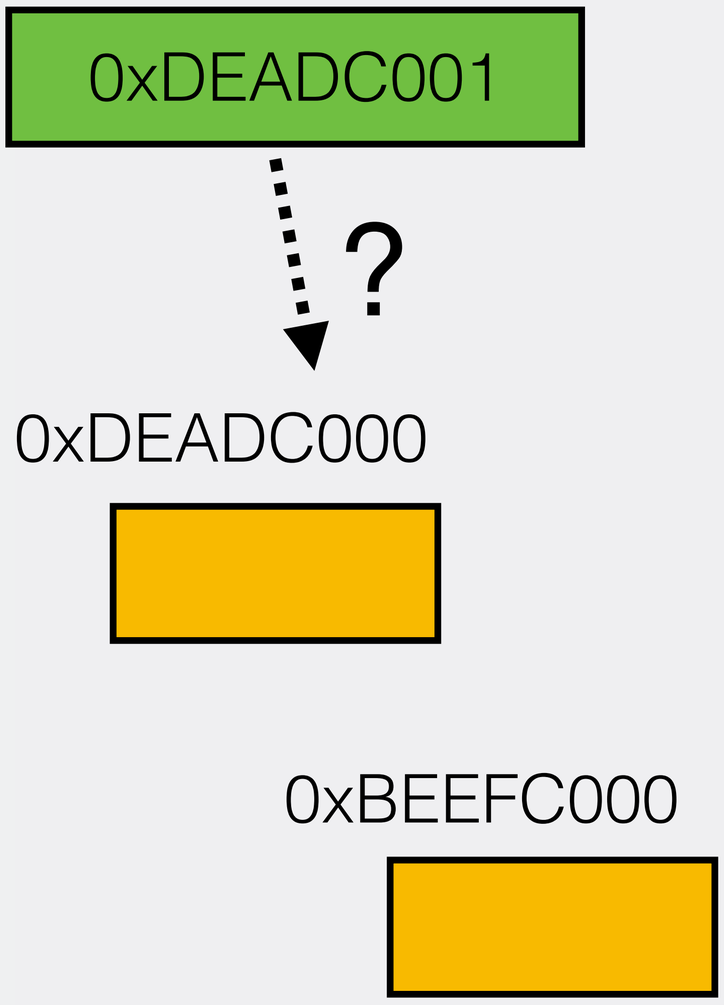
No way to precisely distinguish pointers from integers

```
$ clang++ -o yolo main.cc
$ strip yolo
$ ./yolo
```



No way to precisely distinguish pointers from integers

```
union tiny
{
 int * ptr;
 uintptr_t flag;
};
tiny x;

// initialize
x.ptr = new int;

// set flag true
x.flag |= 1;
```



MESH

Compaction without Relocation!

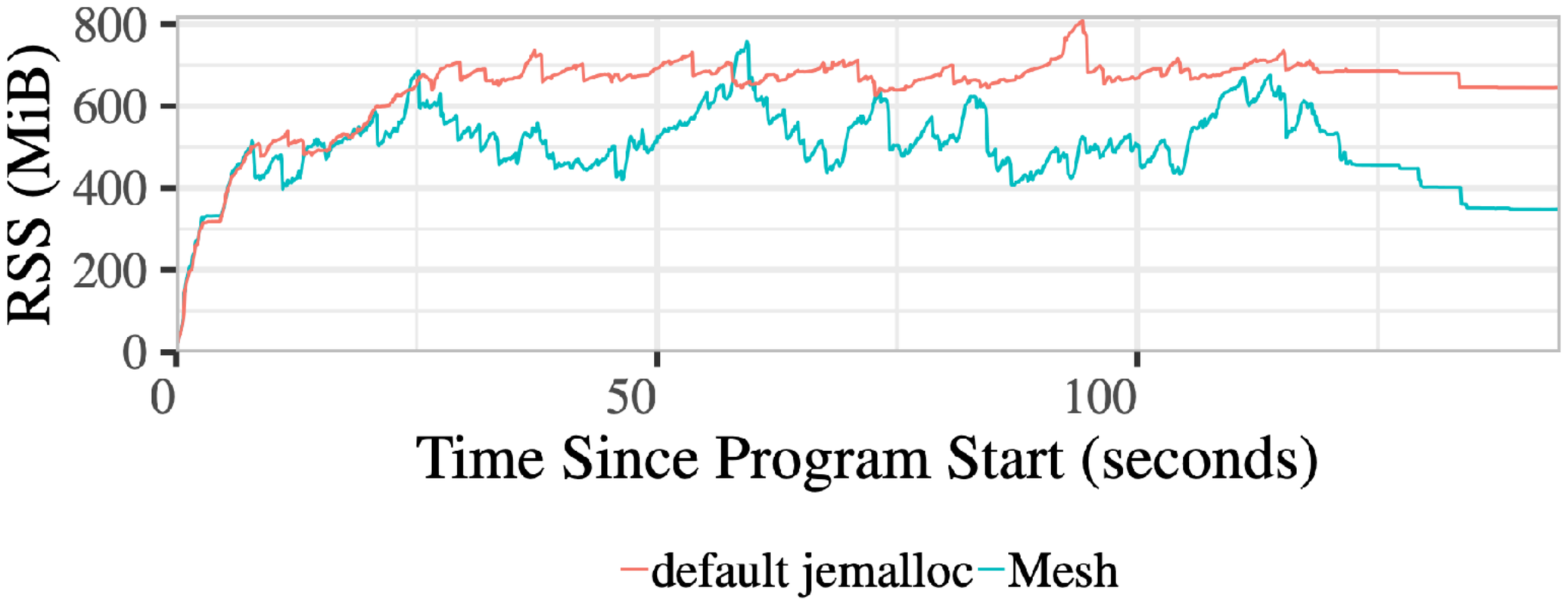
No code changes

No recompilation

(`LD_PRELOAD`)

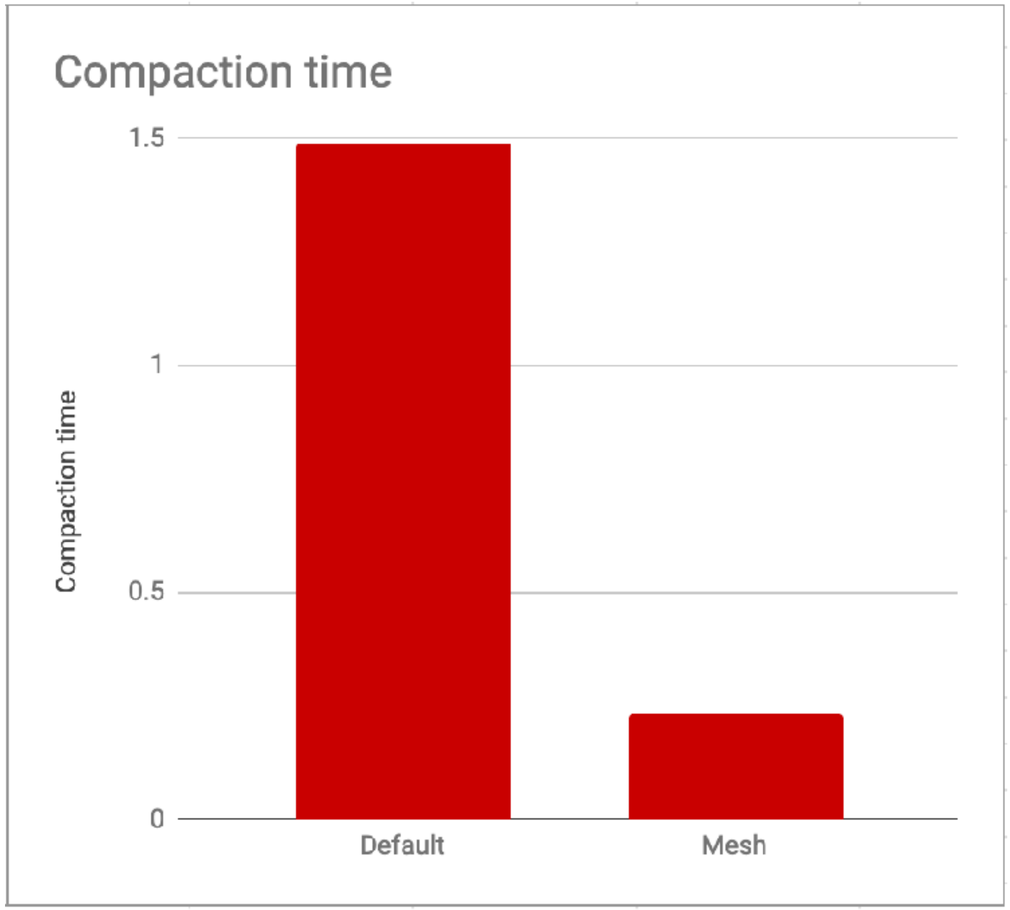
17% heap size reduction

< 1% performance overhead

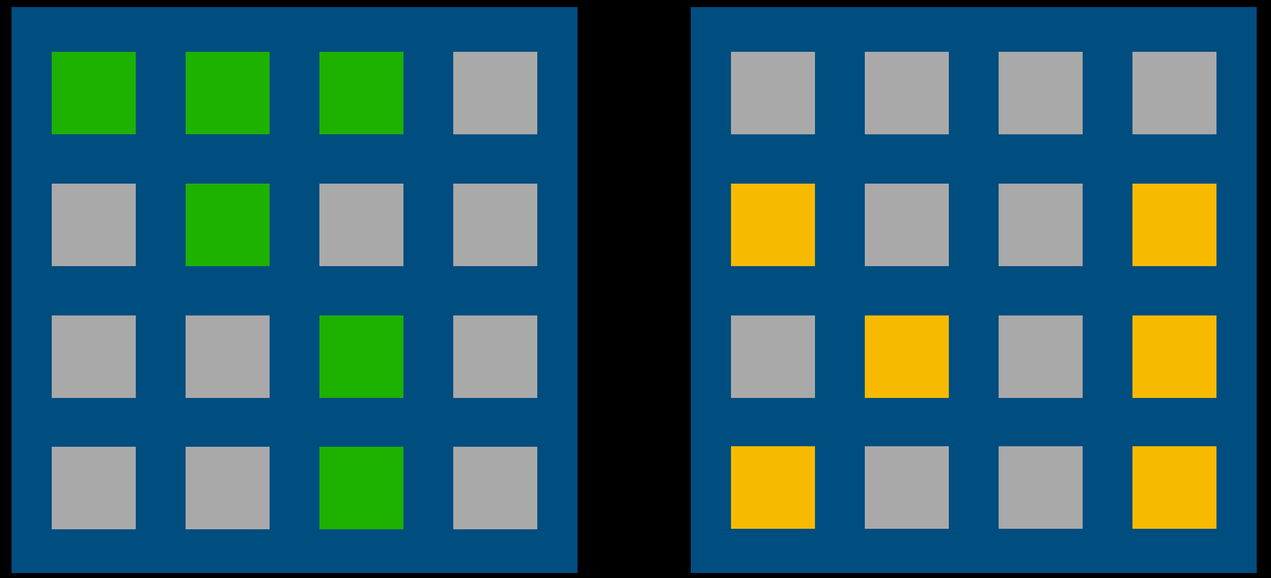


redis


jemalloc + activedefrag
Mesh
no compaction
RSS (MiB)
300
200
100
0
0
1
2
3
4
5
6
Time Since Program Start (seconds)

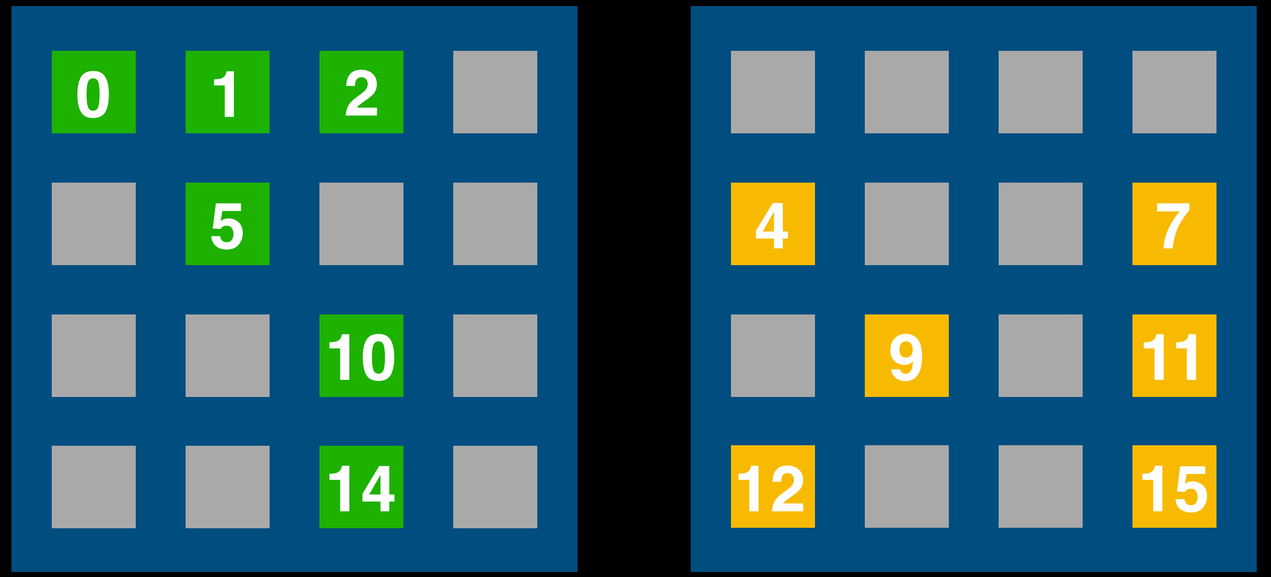


5x reduction in time spent compacting



Pages are ***meshable*** when they:

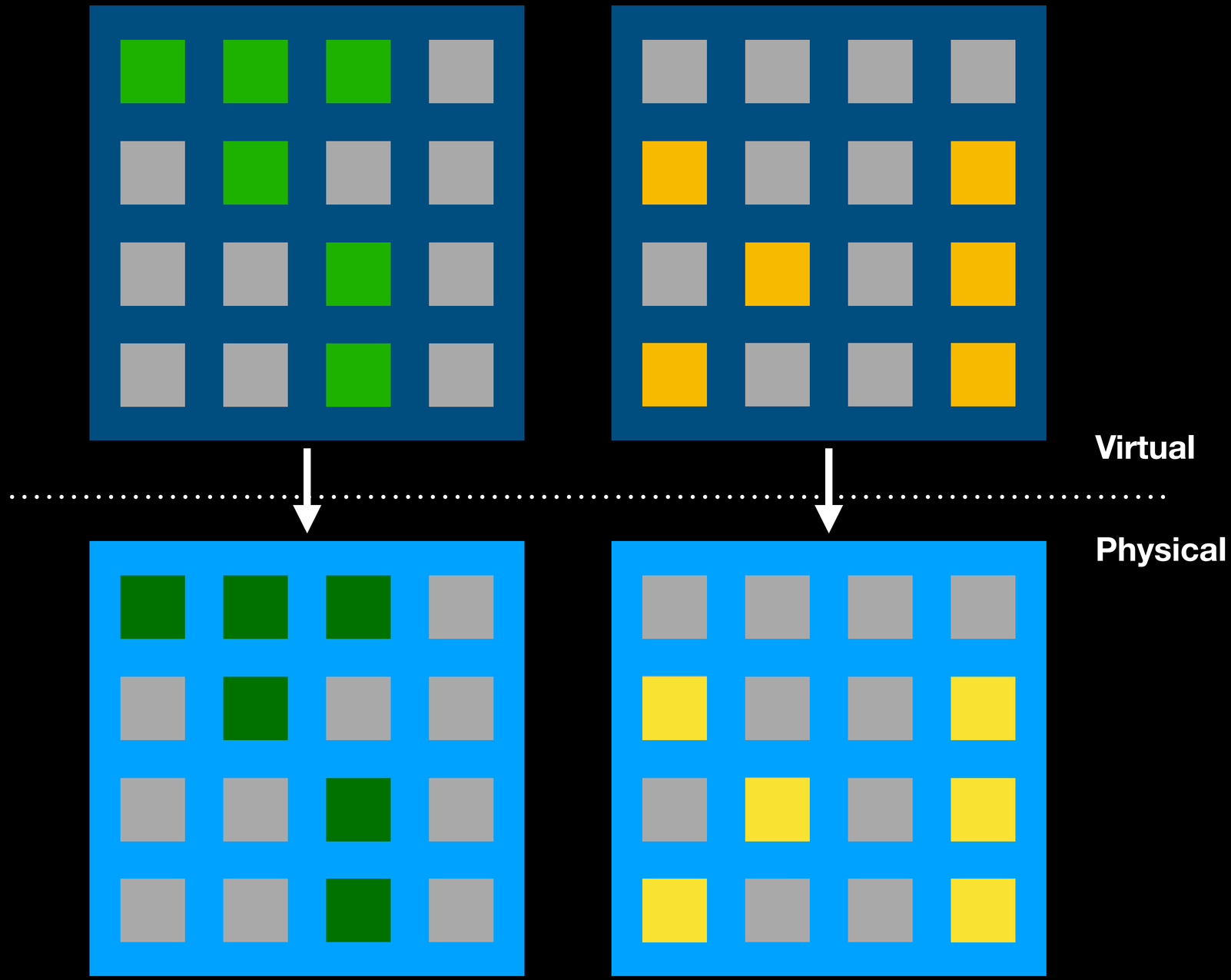
- Hold objects of the same size class



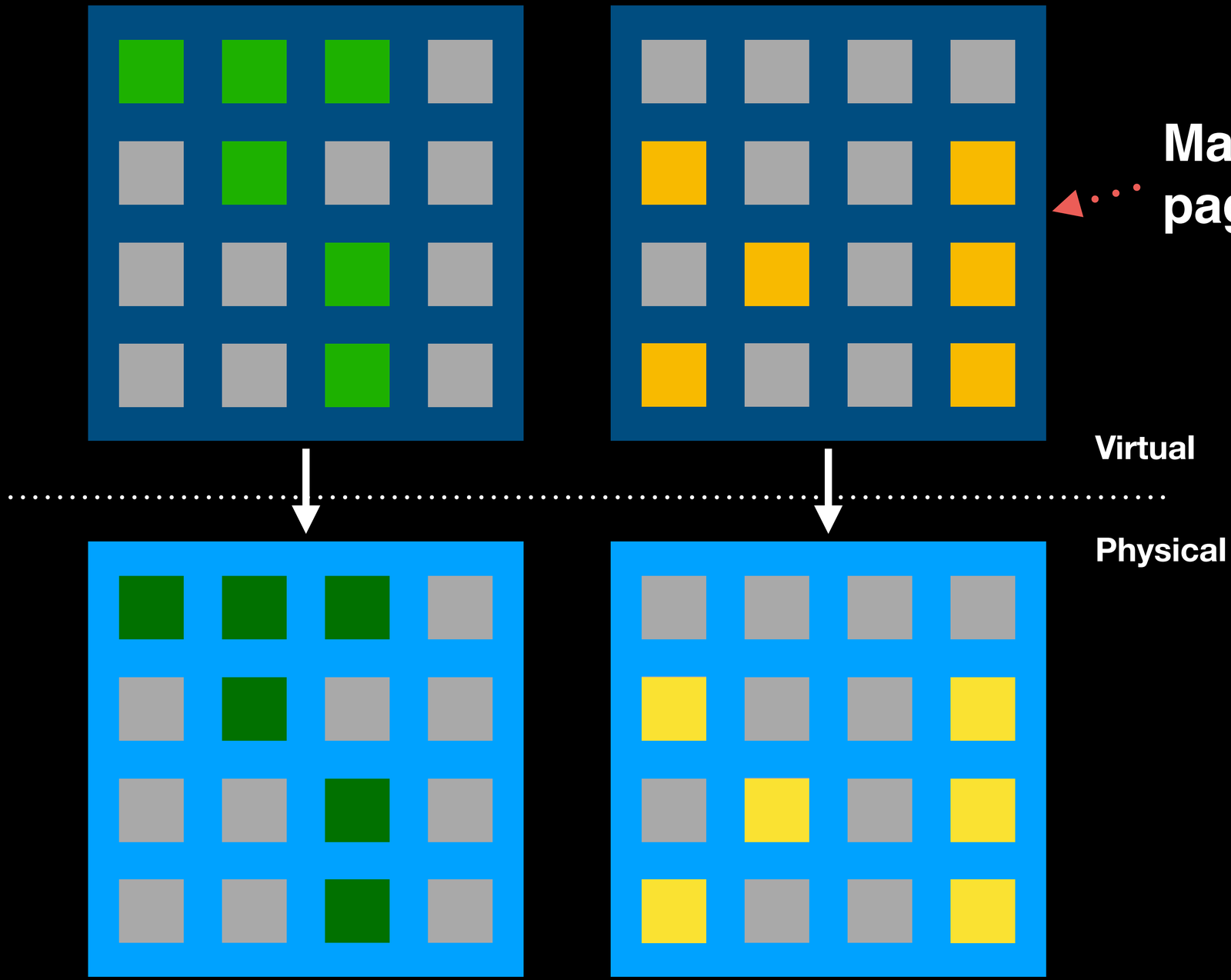
Pages are ***meshable*** when they:

- Hold objects of the same size class
- Have non-overlapping object offsets

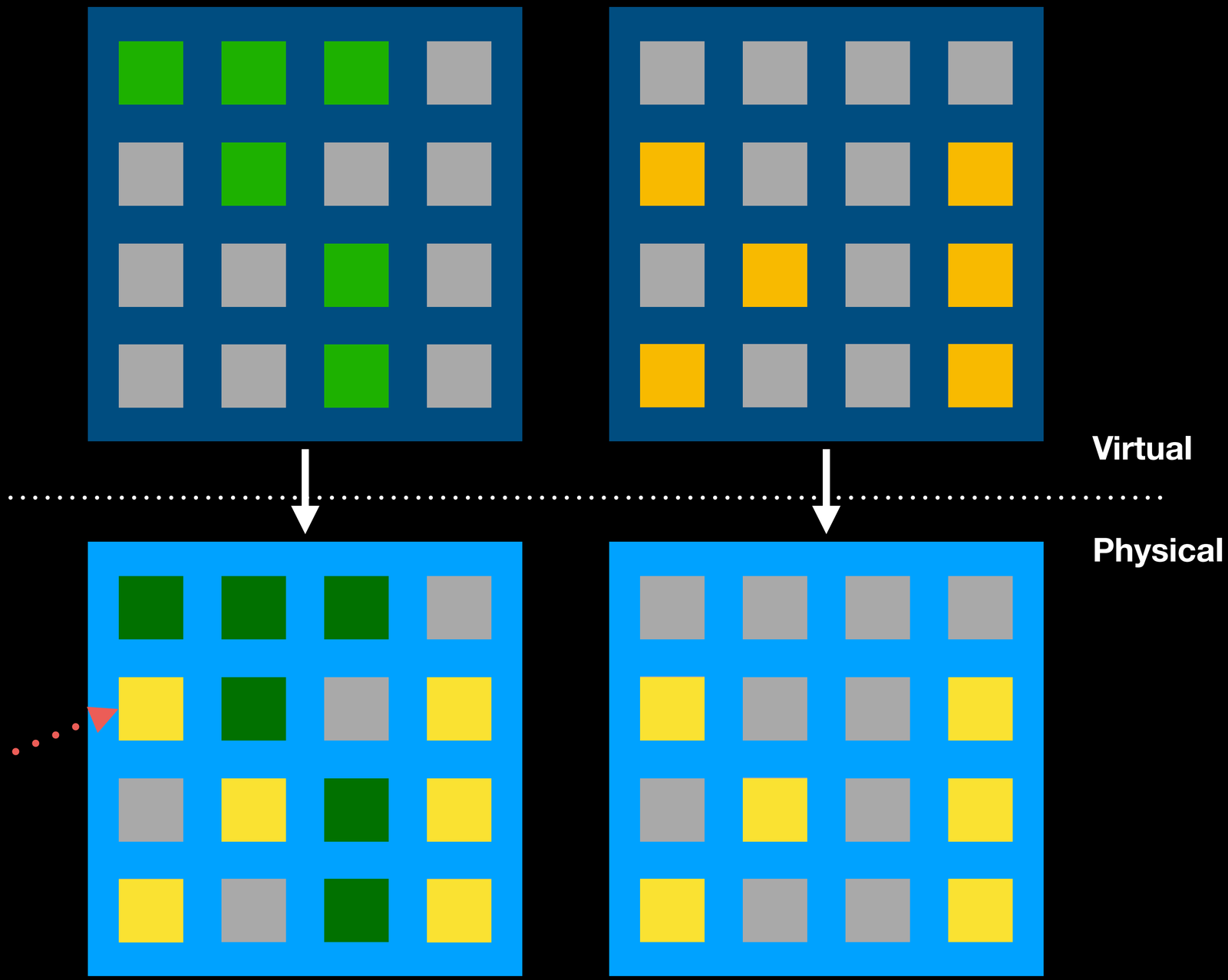
Meshing



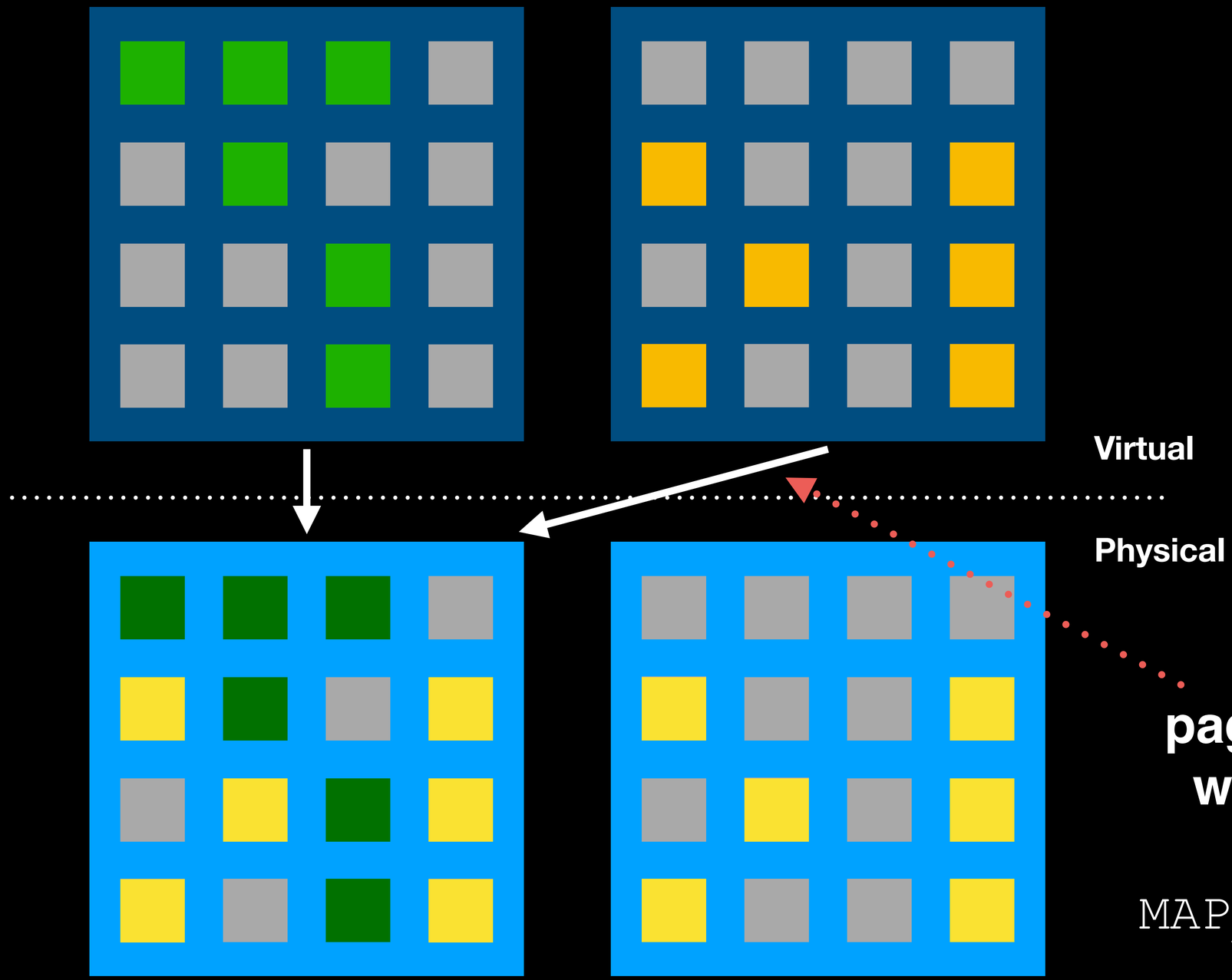
Meshing



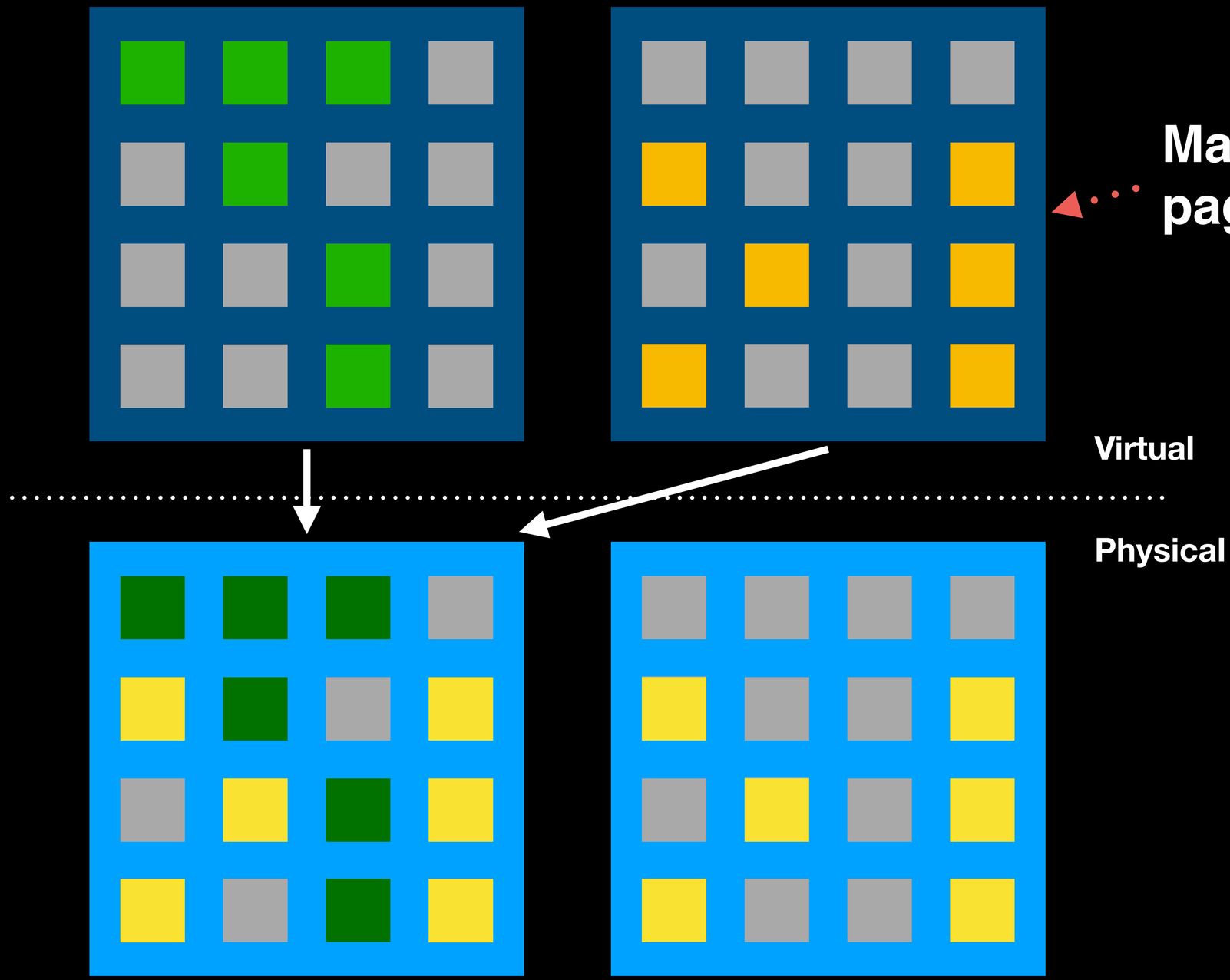
Meshing



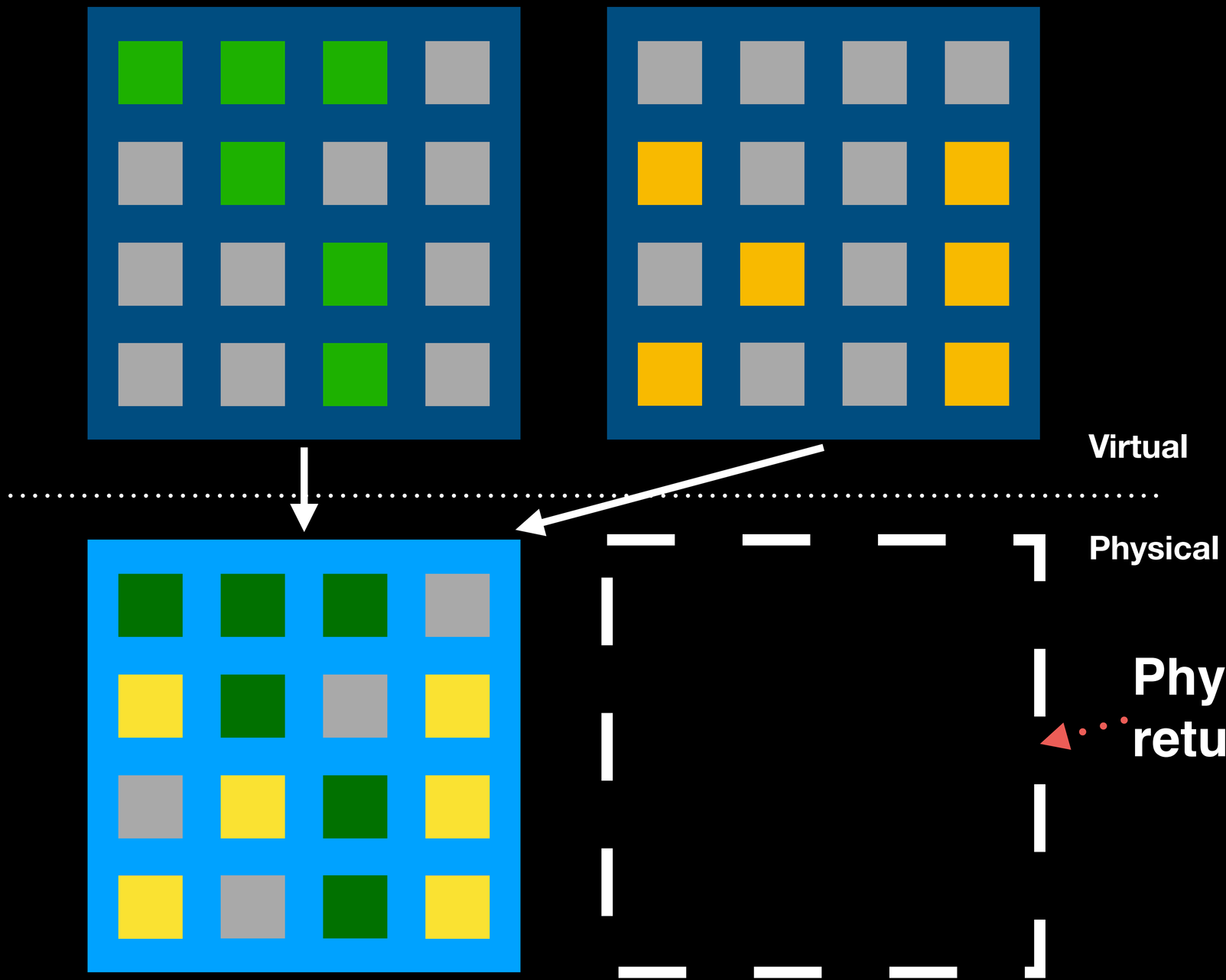
Meshing


Virtual
Physical
Update page tables with mmap, memfd, MAP_SHARED

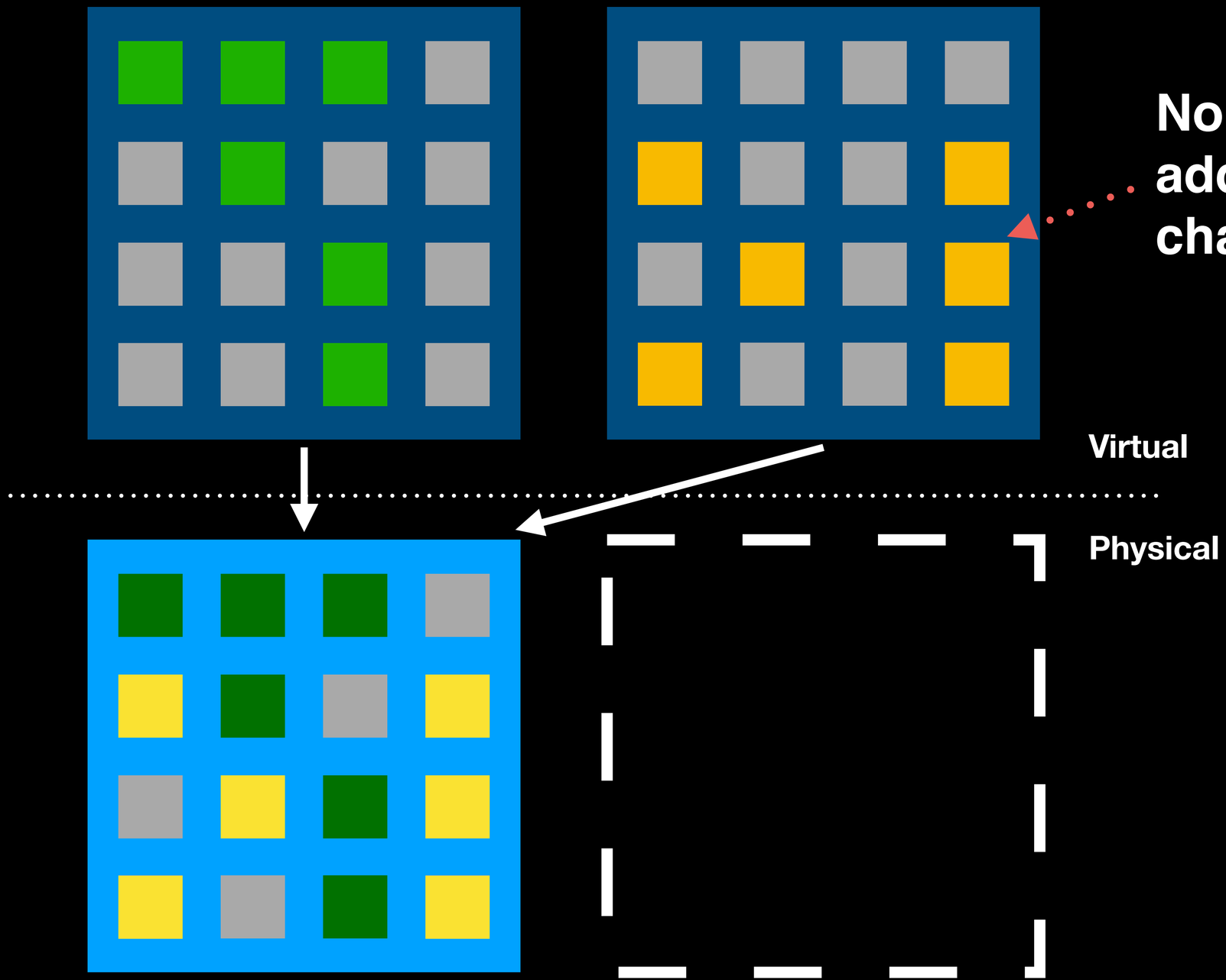
Meshing



Meshing



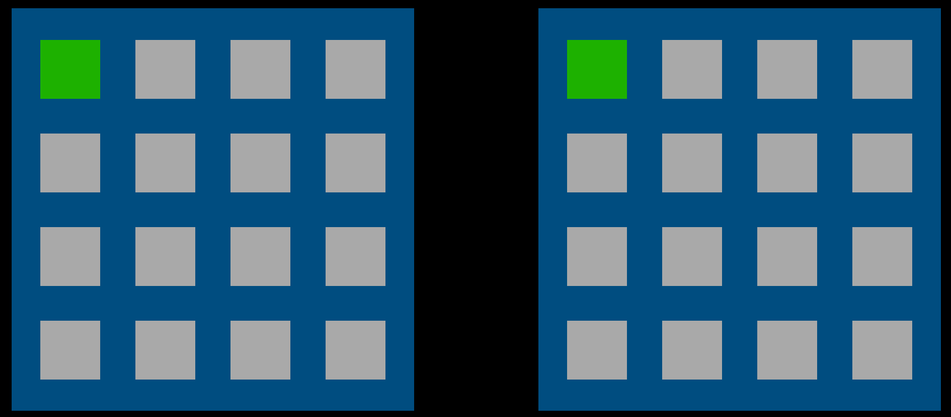
Meshing



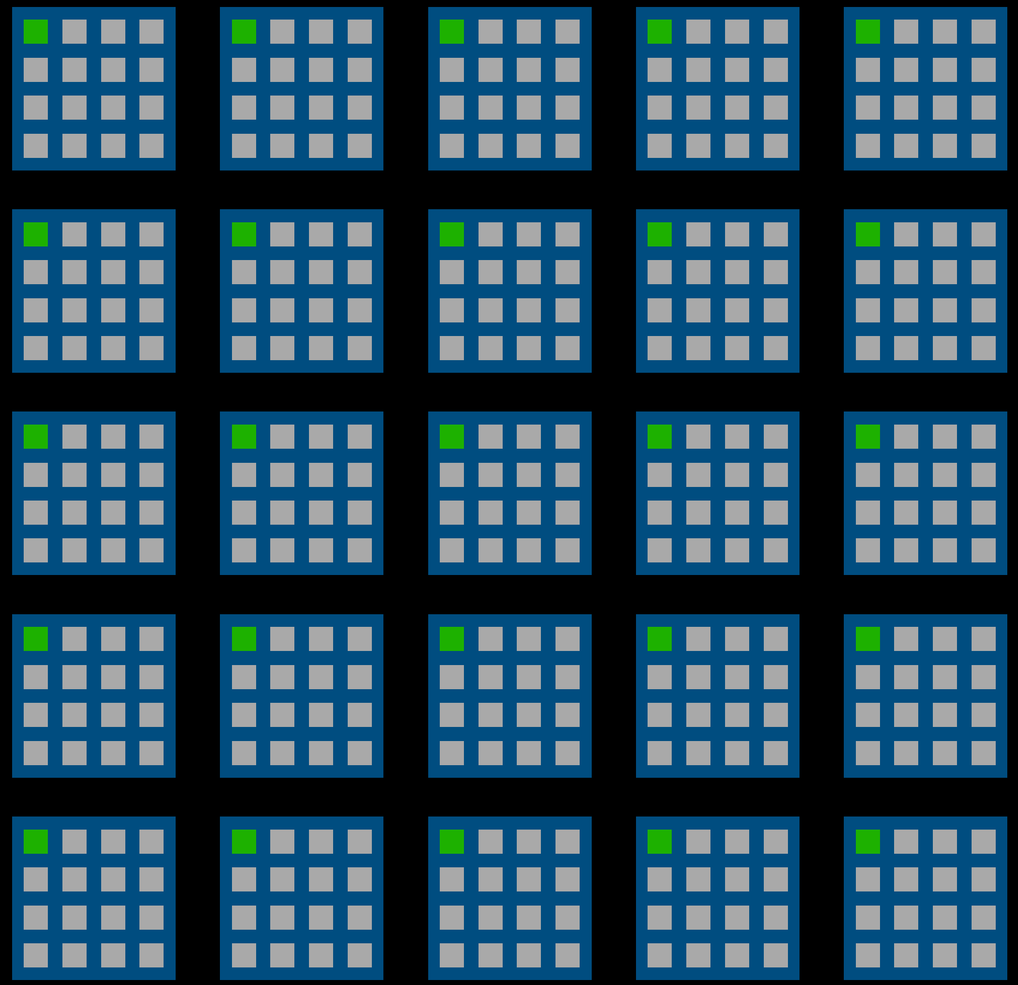
Worst Case:

Worst Case:

low occupancy, non-meshable pages

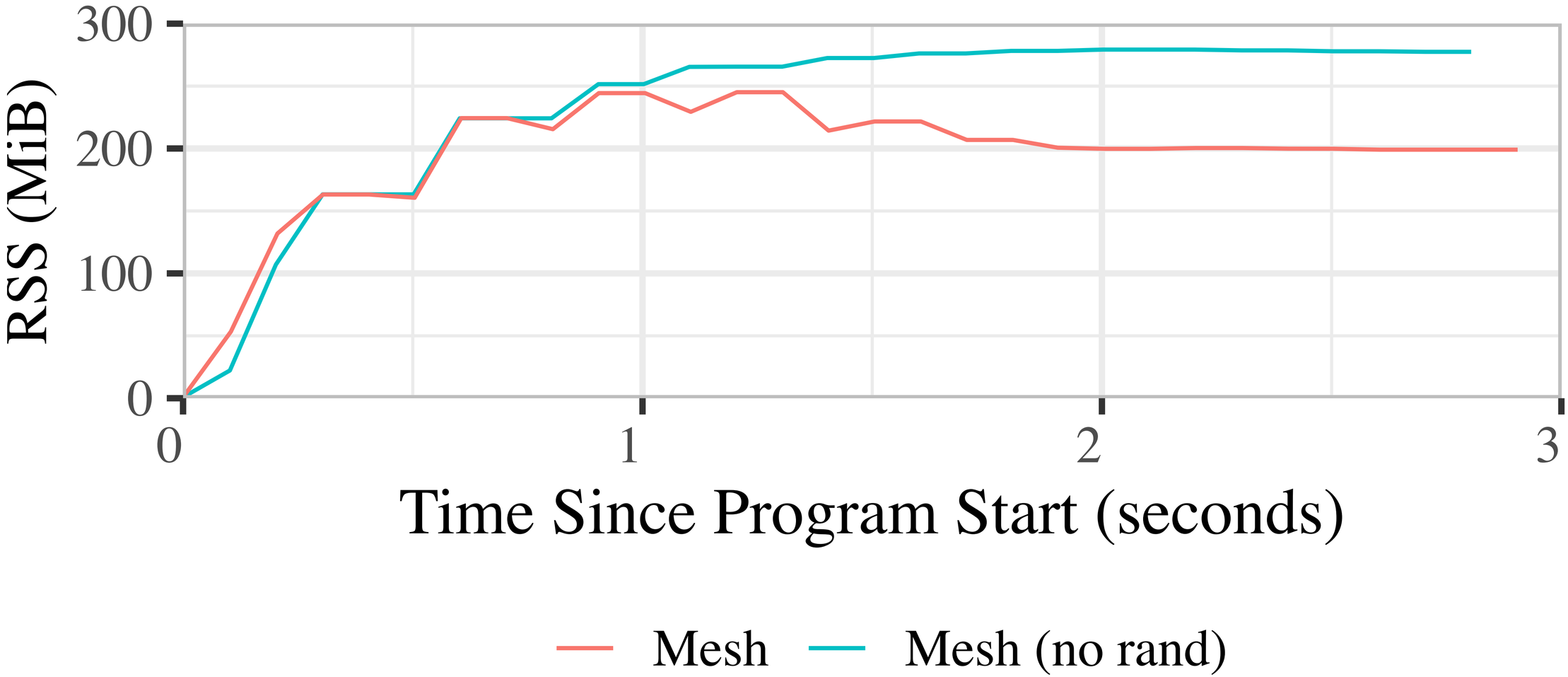


Worst Case:
many
low occupancy,
non-meshable
pages



Mesh uses **randomization** to ensure live objects are uniformly distributed

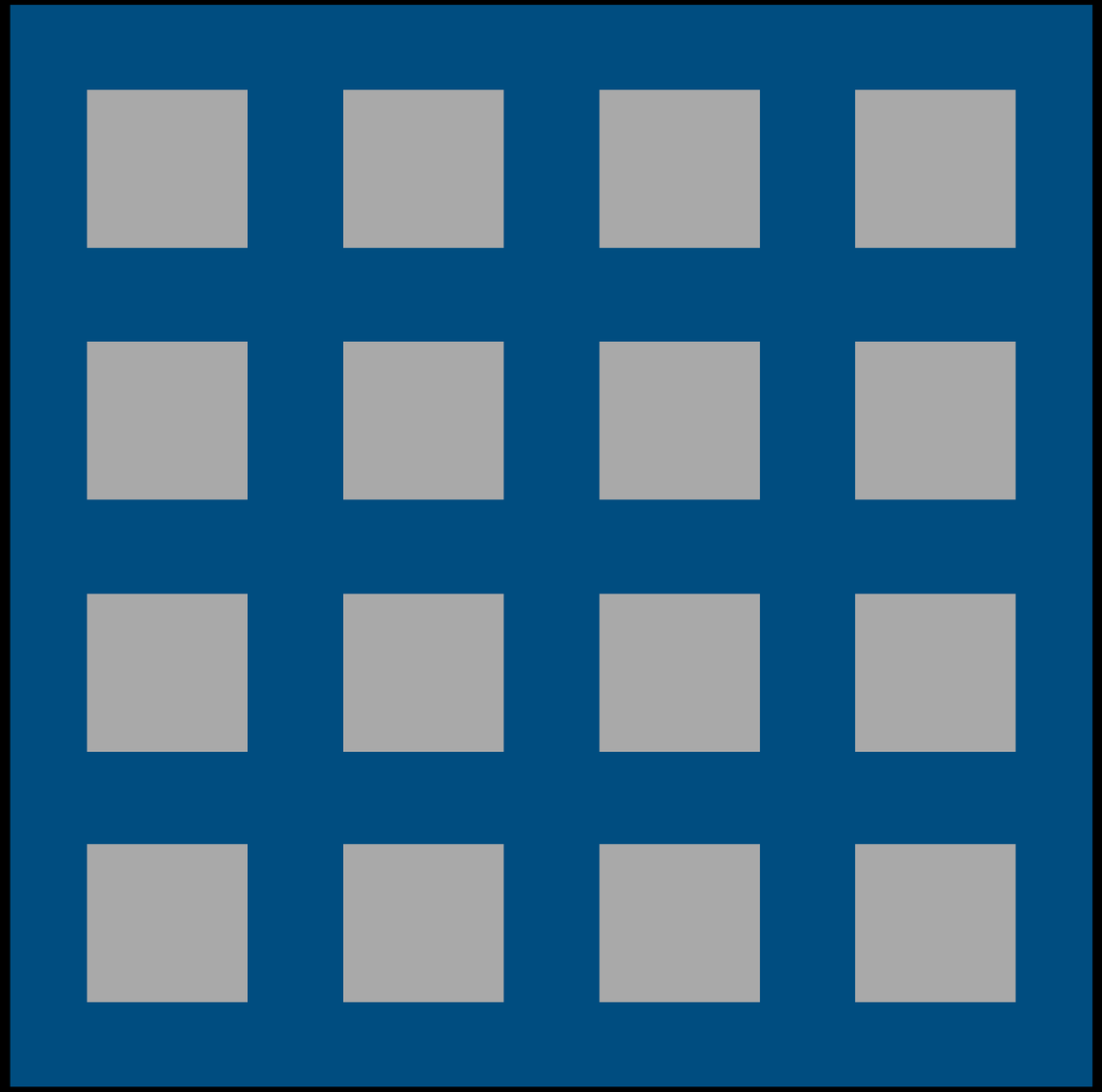
Impact of randomized allocation



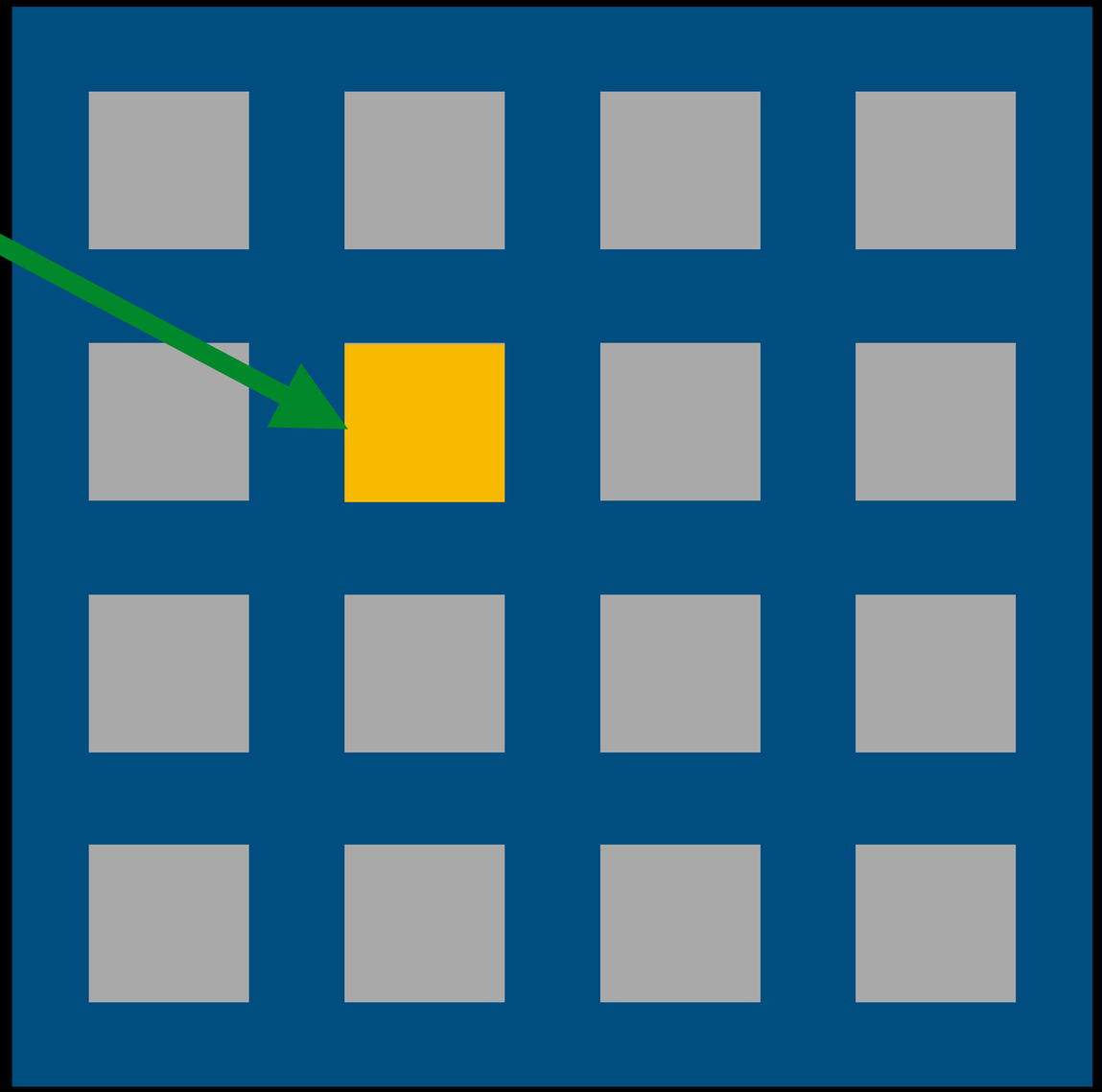
How does Mesh *efficiently* **randomize** allocation?

Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```



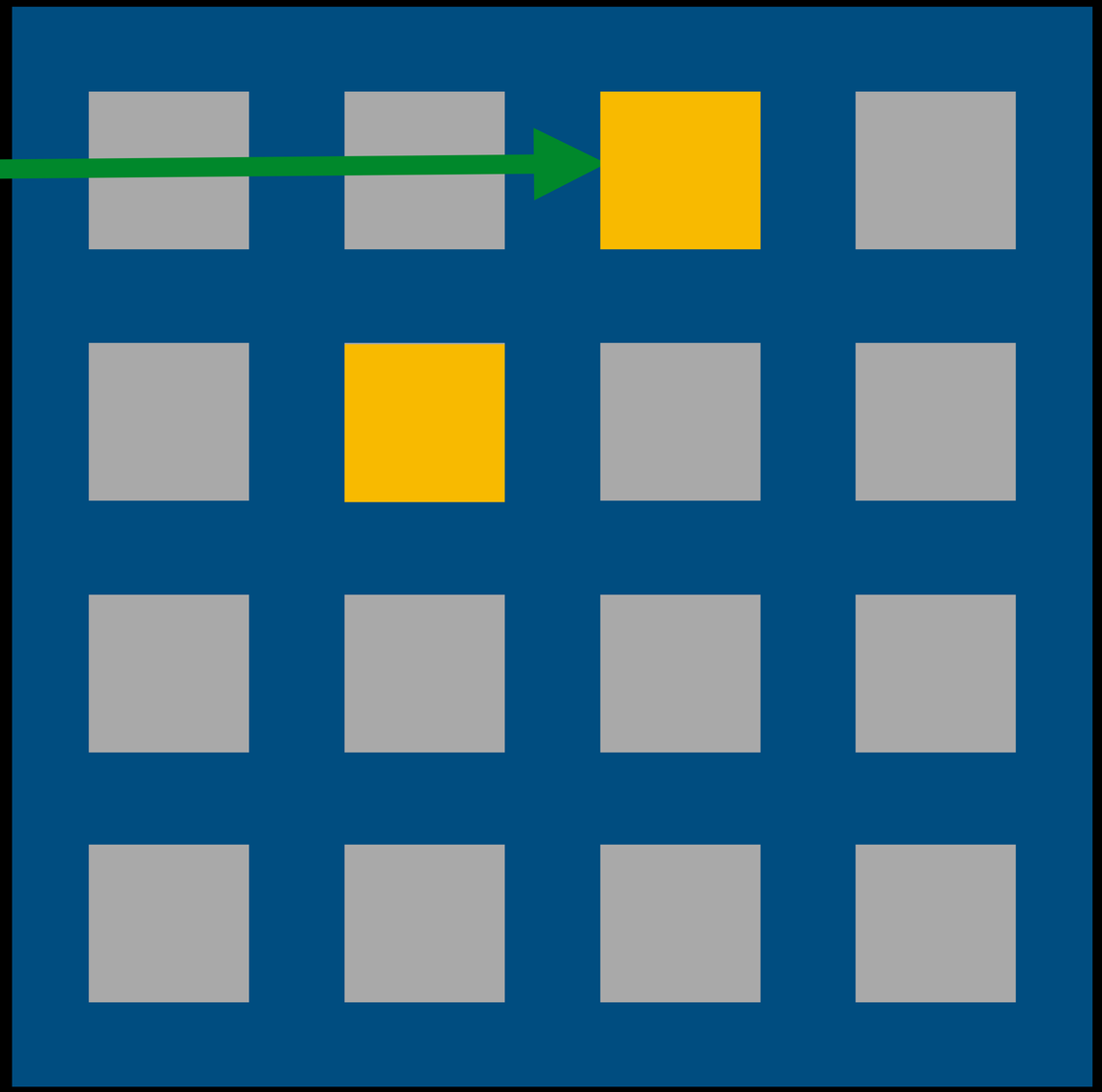
```
malloc(256)
```



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

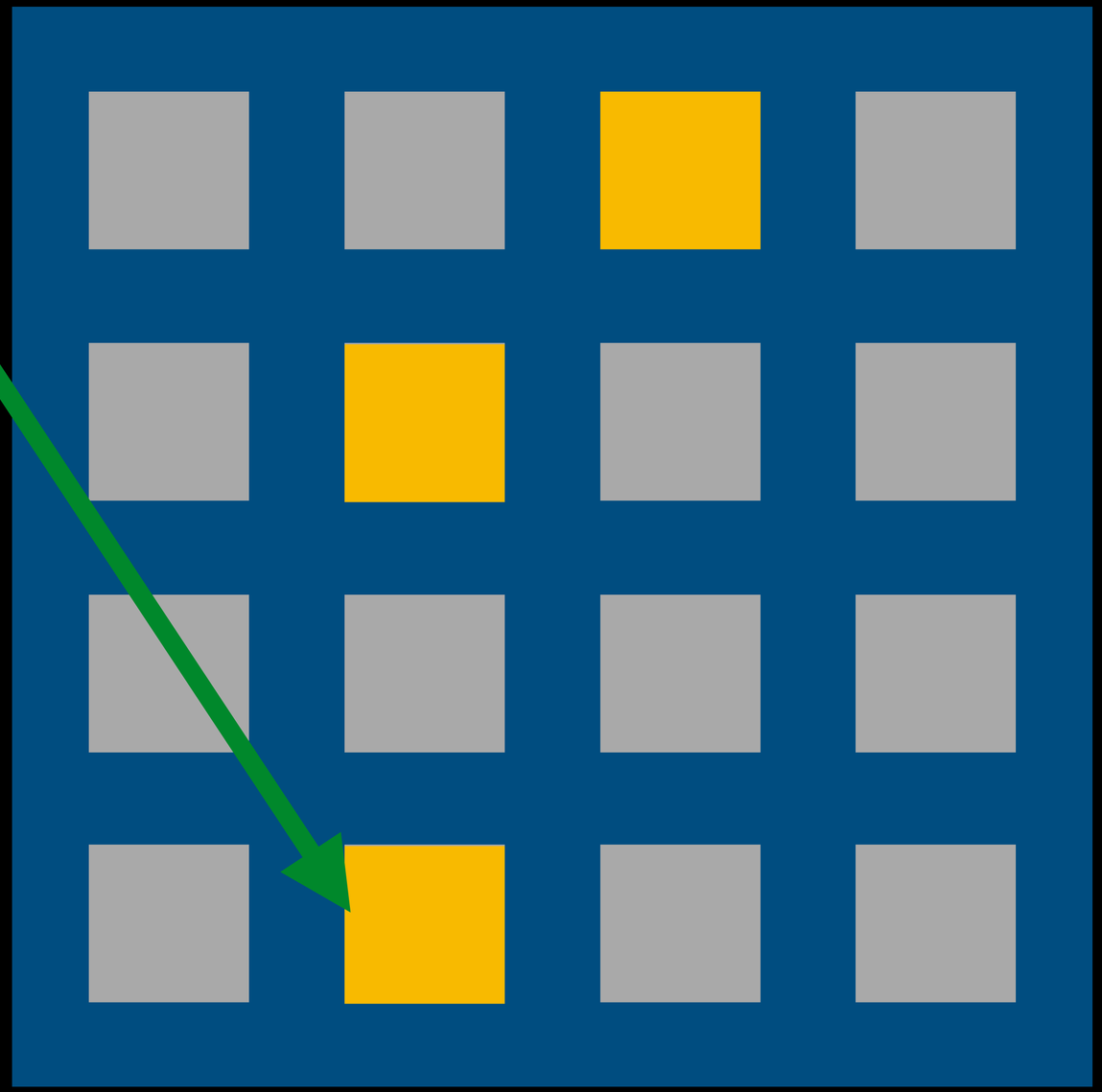
```
malloc(256)
```



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

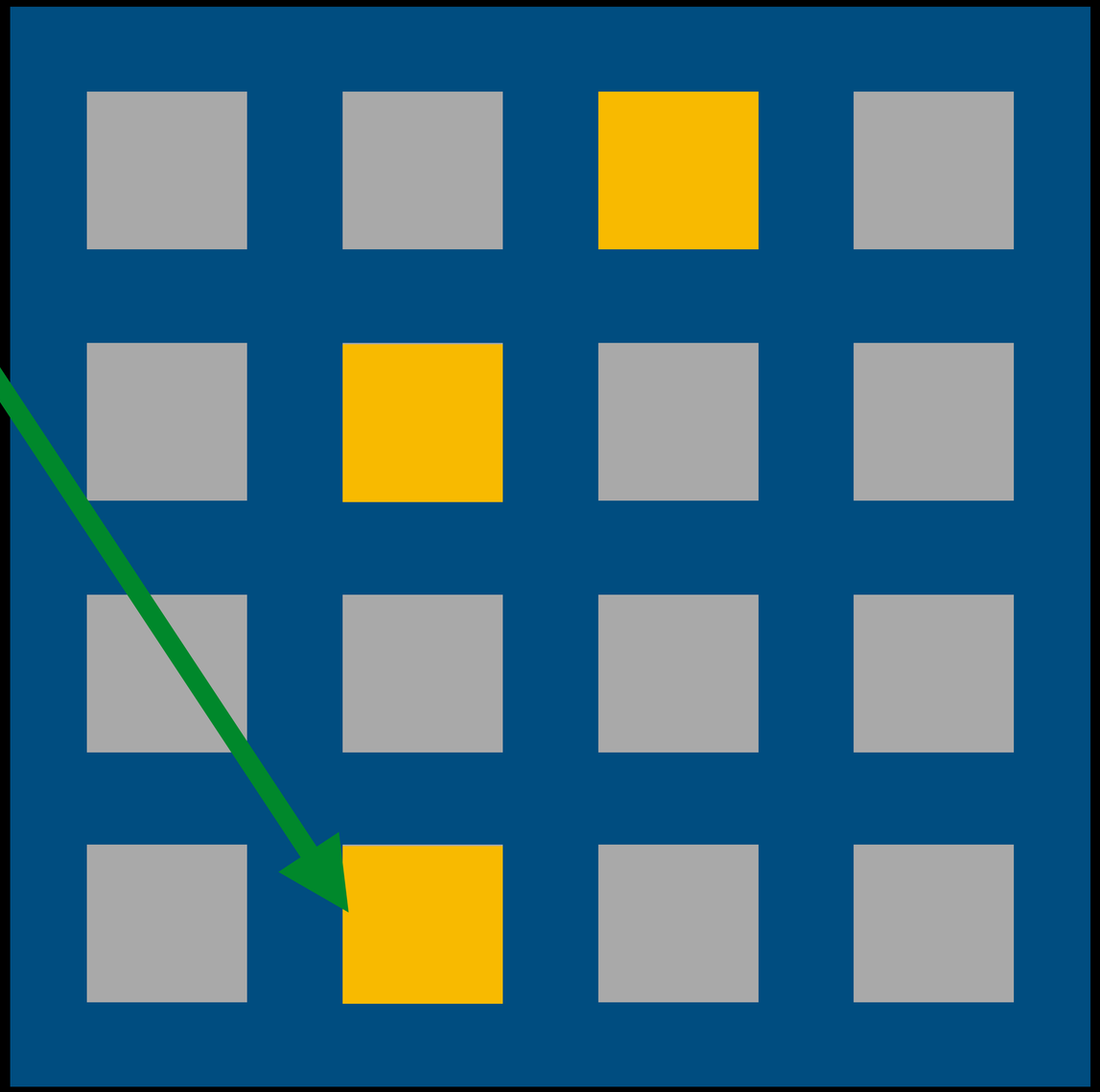
```
malloc(256)
```



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

```
malloc(256)
```

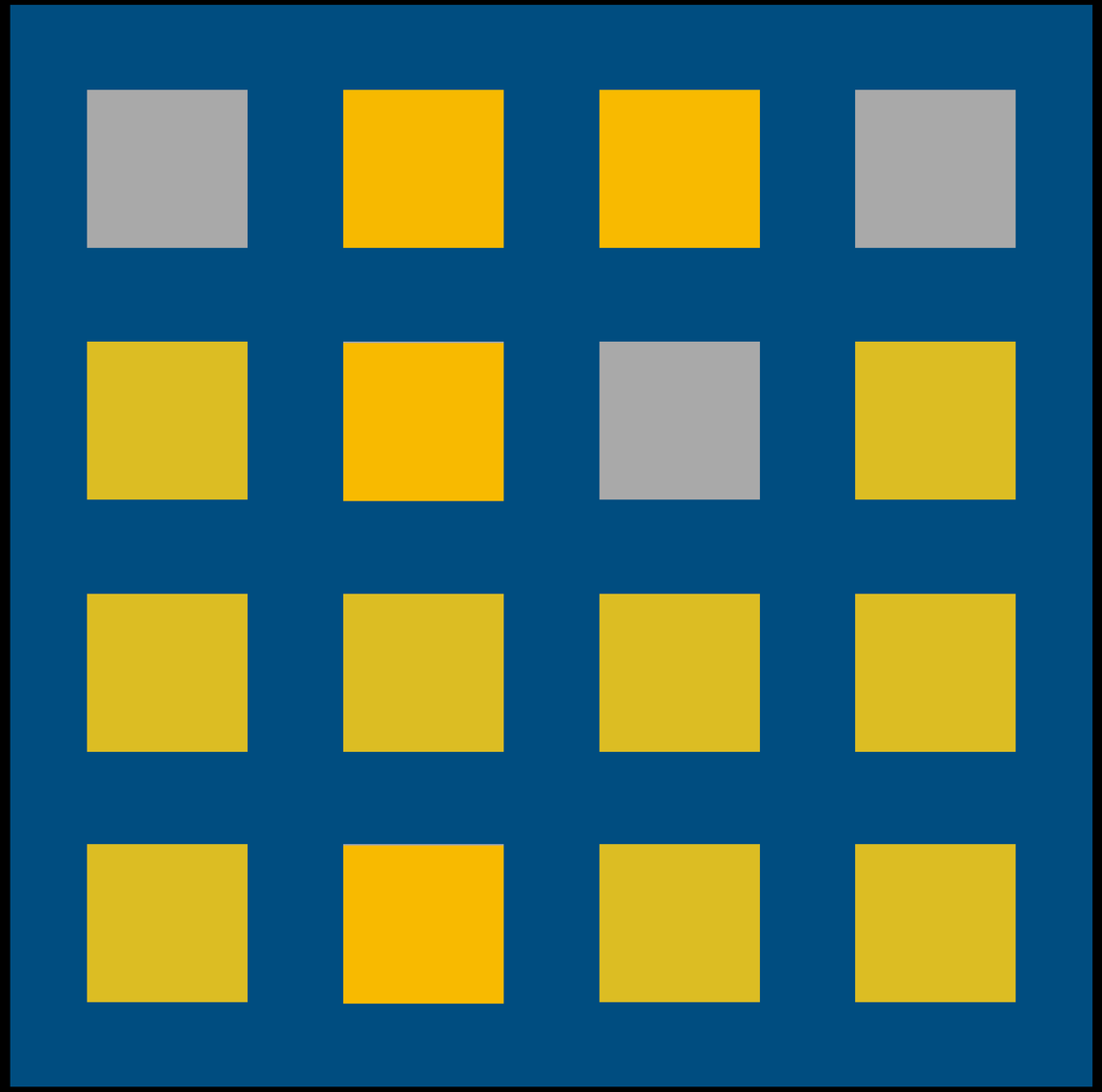


Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

(DieHard [Berger & Zorn 2006])

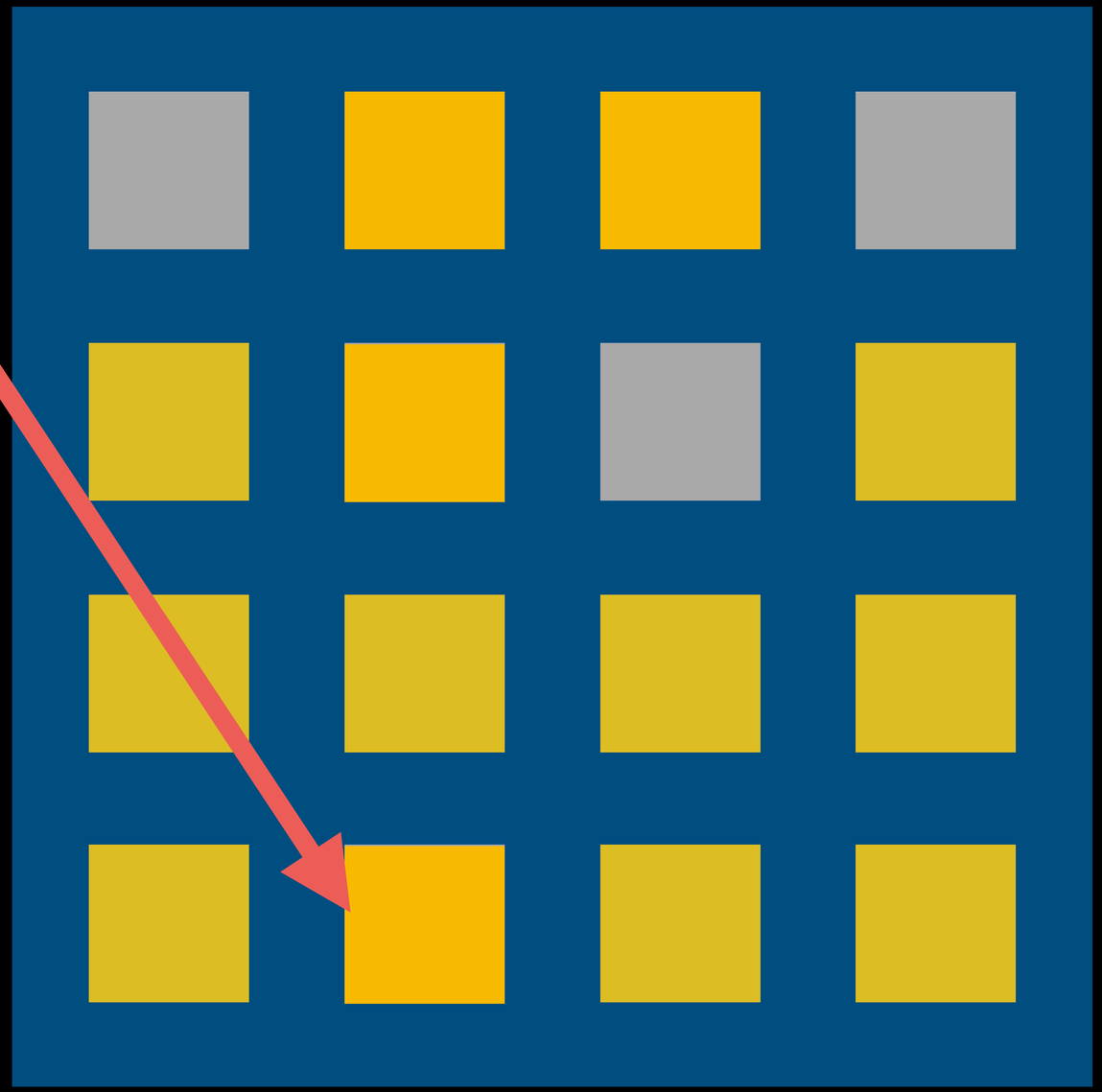
`malloc(256)`



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

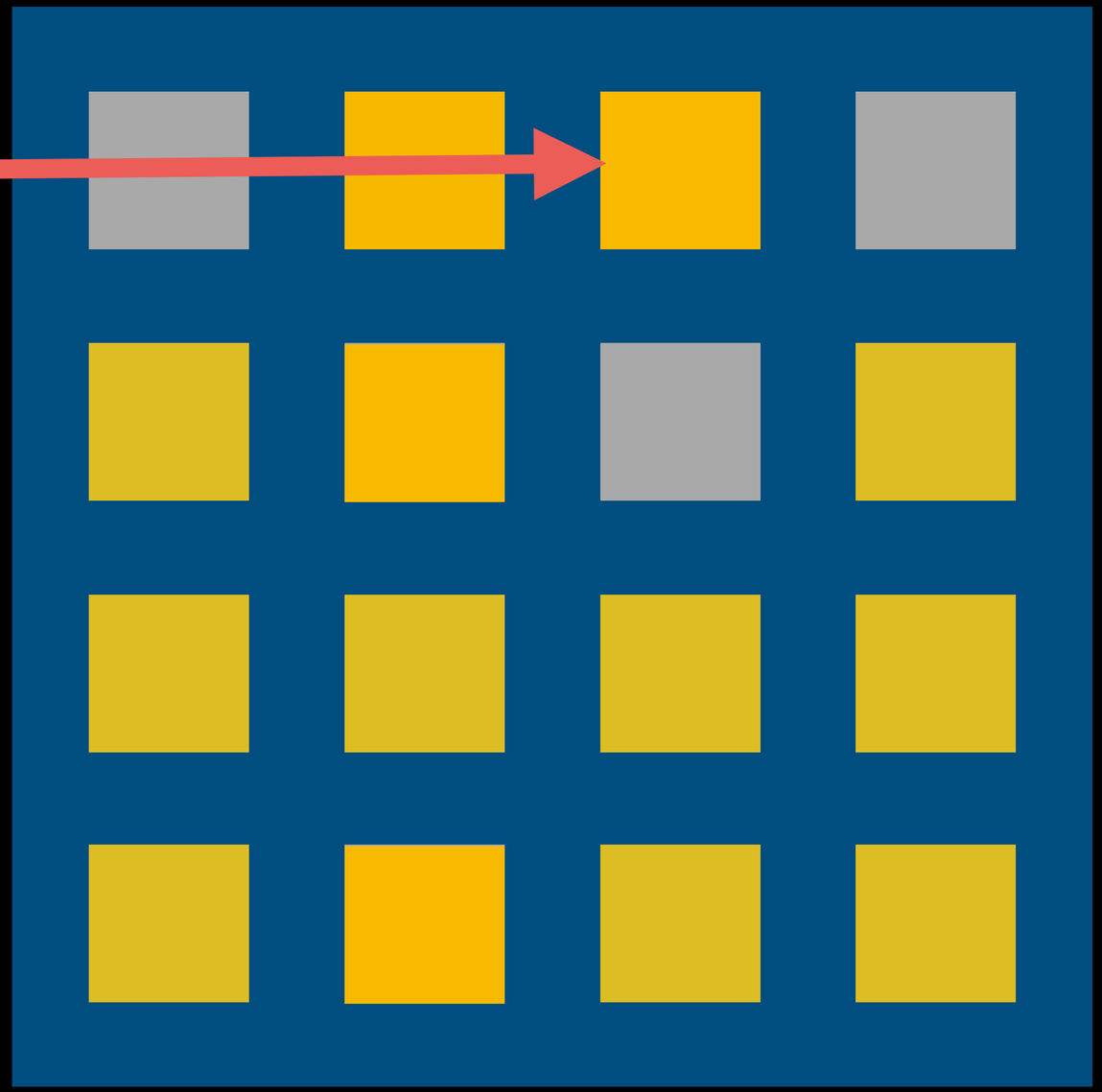
```
malloc(256)
```



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

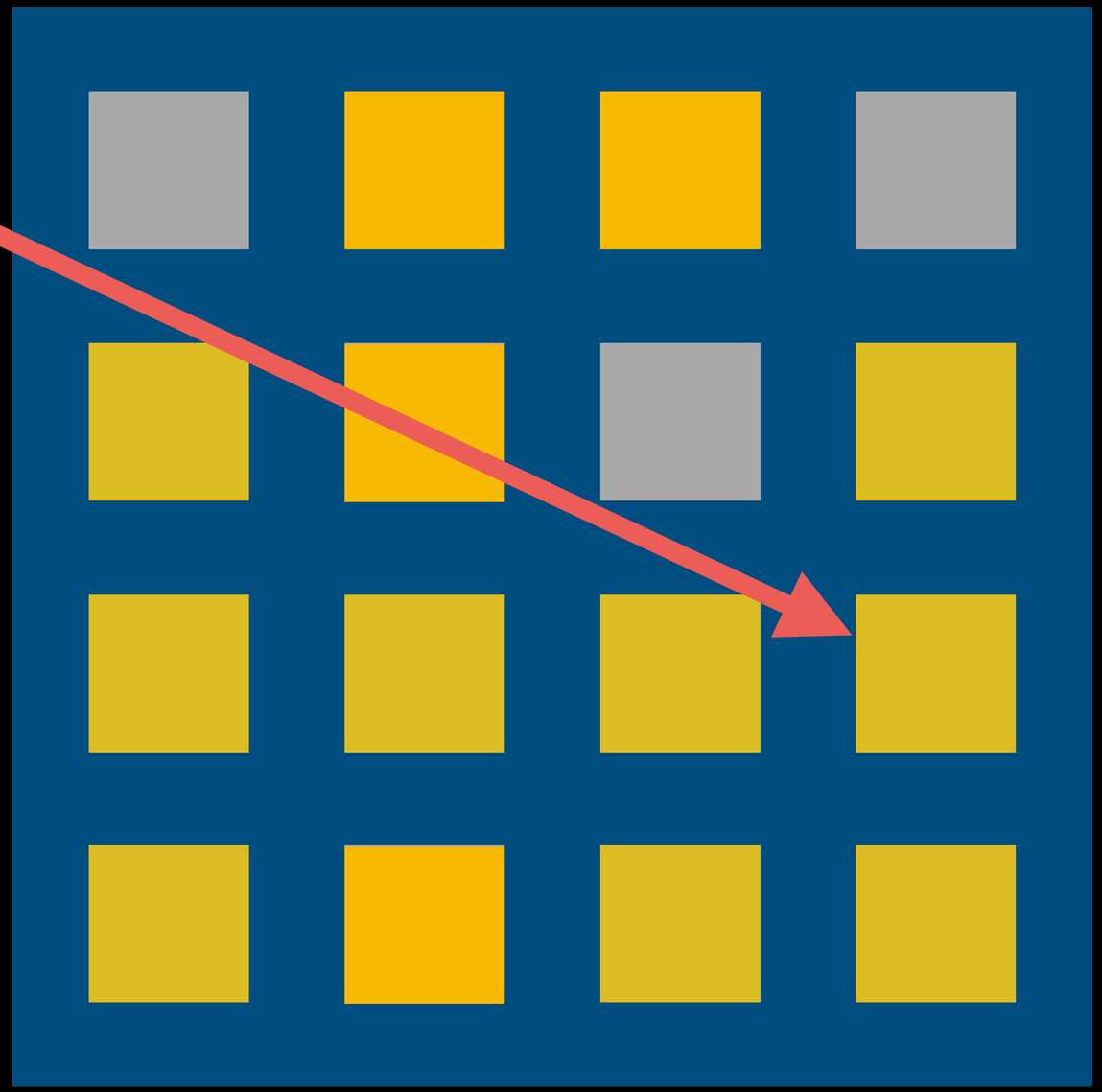
```
malloc(256)
```



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

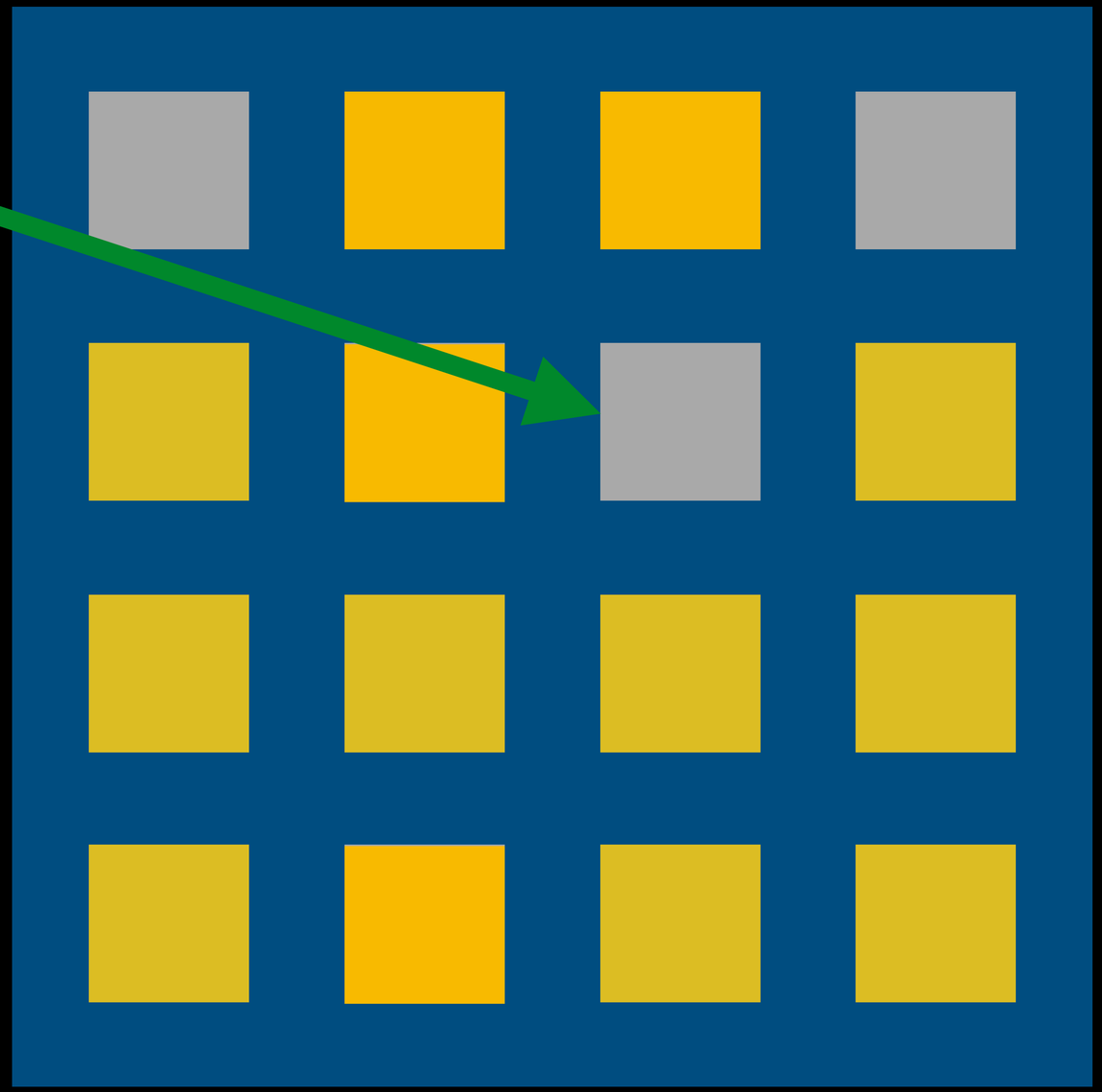
```
malloc(256)
```



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

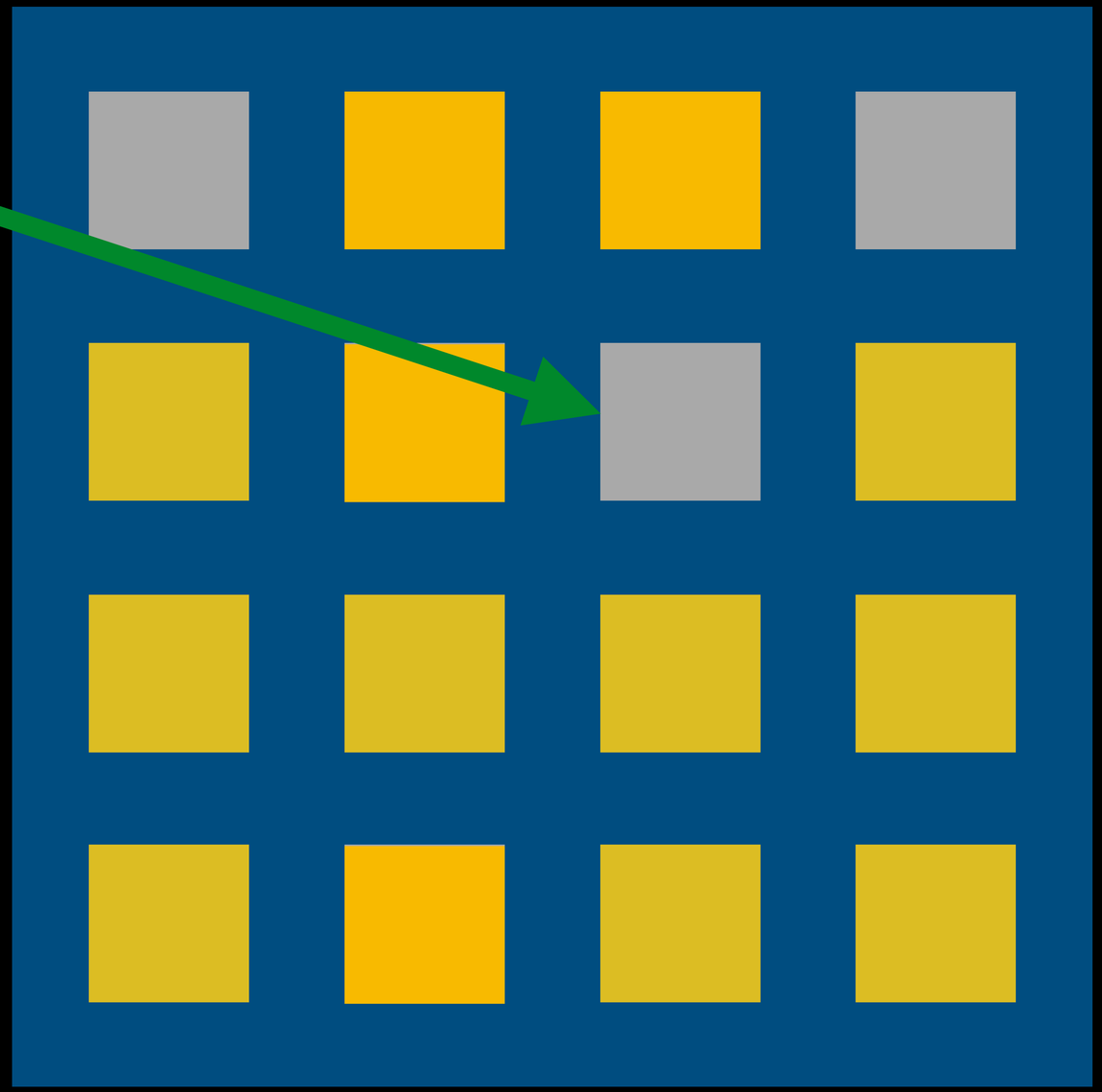
```
malloc(256)
```



Random probing:

```
while true:
   if rand_off().is_free:
      return rand_off
```

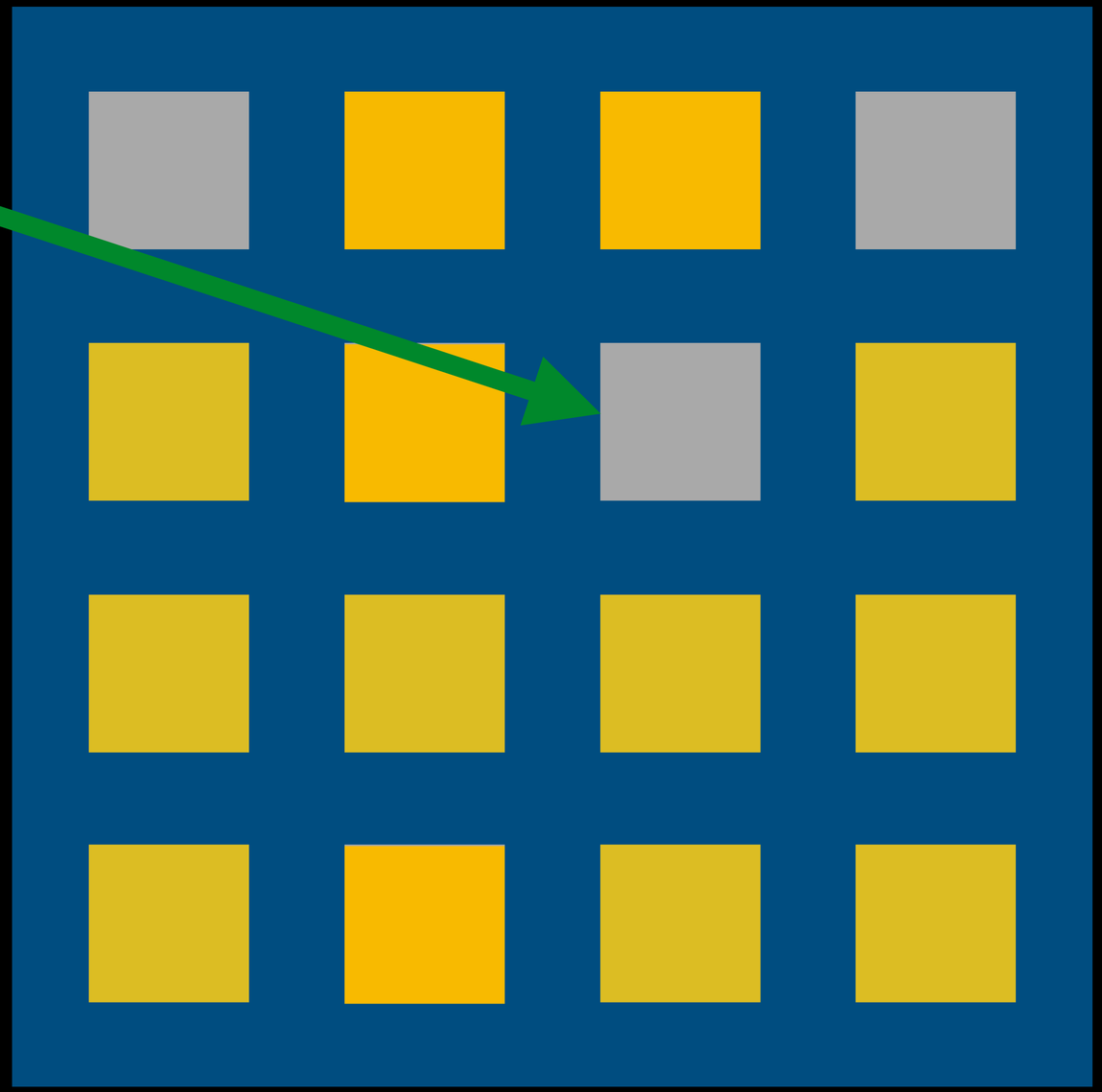
`malloc(256)`



Random probing:

Slow + space-efficient

`malloc(256)`



Random probing:

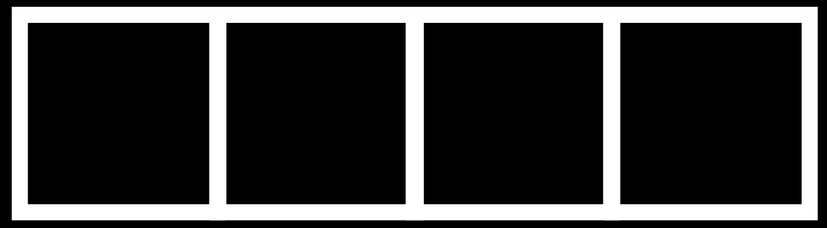
Slow + space-efficient
Fast + space-inefficient

Shuffle Vector:
Fast + space-efficient
random malloc

Shuffle Vector:
Fast + space-efficient
random malloc

Page

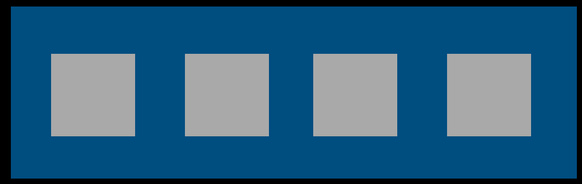

```
load
```



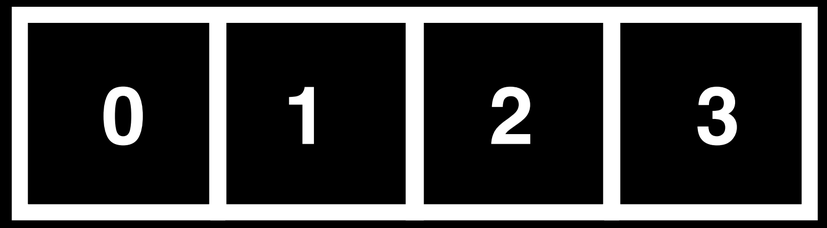
Thread-local shuffle vector

Shuffle Vector:
Fast + space-efficient random malloc

Page

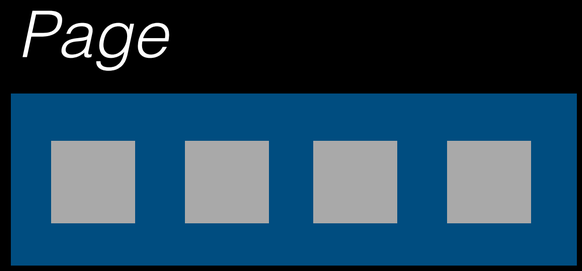


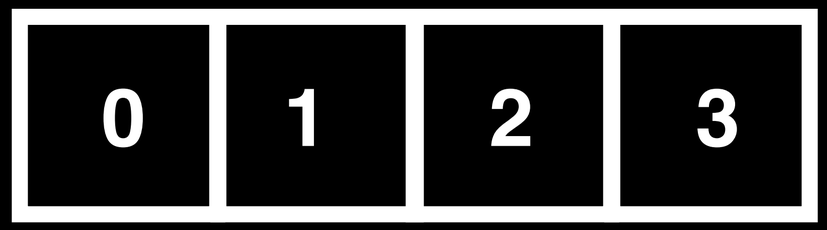
```
load
```



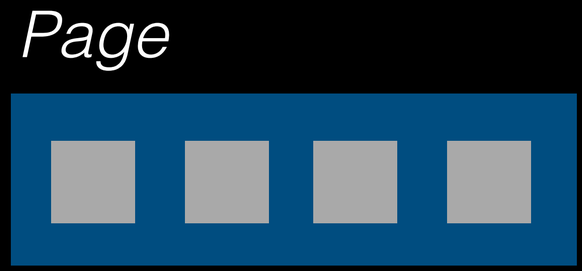
Thread-local shuffle vector

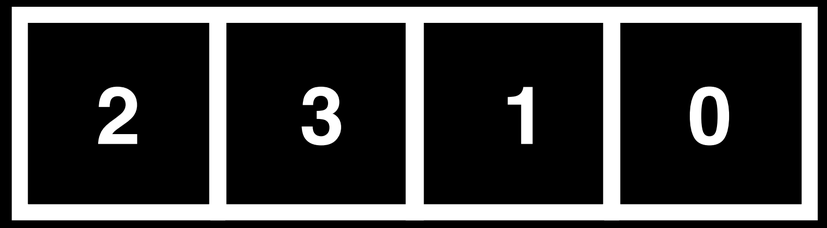
Shuffle Vector:
Fast + space-efficient random malloc





Shuffle Vector:
Fast + space-efficient random malloc

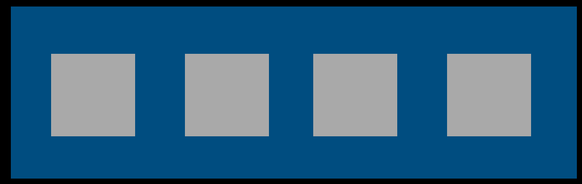




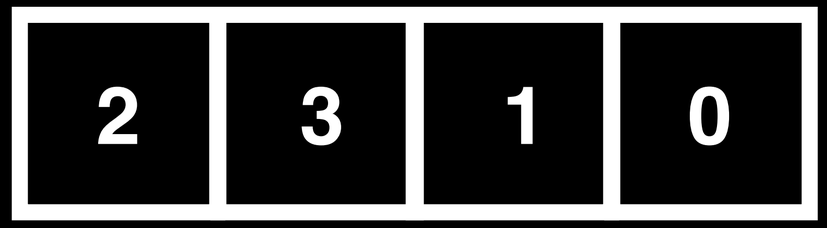
Thread-local shuffle vector

Shuffle Vector:
Fast + space-efficient random malloc

Page



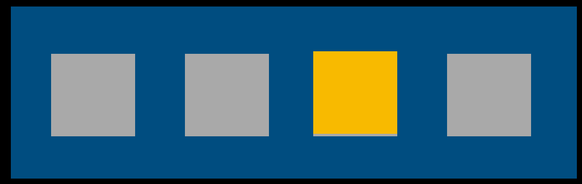
`malloc()`



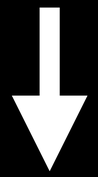
Thread-local shuffle vector

Shuffle Vector:
Fast + space-efficient random malloc

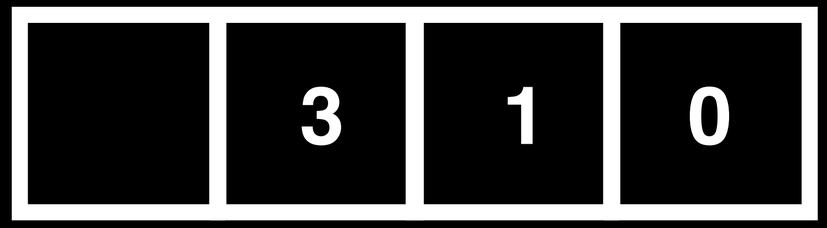
Page



```
malloc()
```



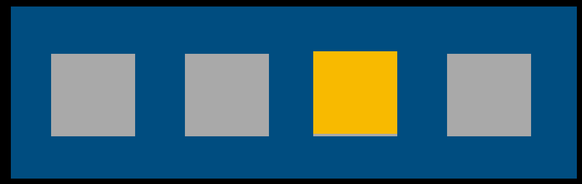
```
page_start +
2 * object_size
```



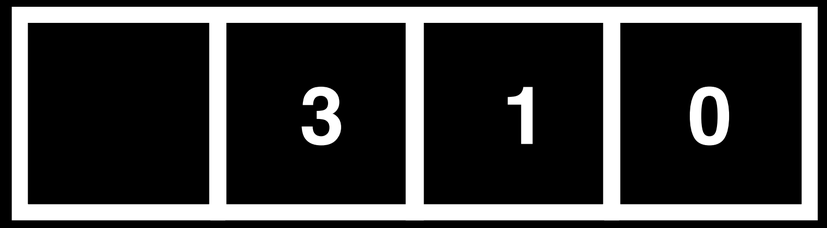
Thread-local shuffle vector

Shuffle Vector:
Fast + space-efficient random malloc

Page



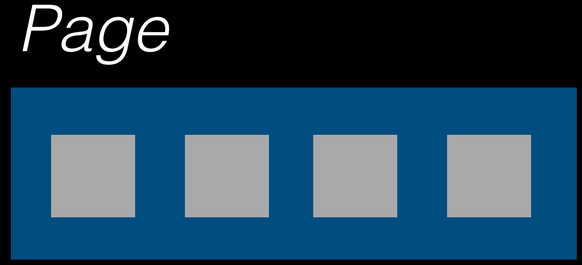
```
free(2)
```

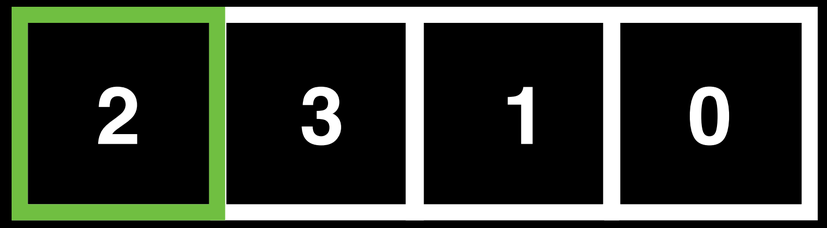


Thread-local shuffle vector

Shuffle Vector:
Fast + space-efficient random malloc

```
free( )
```

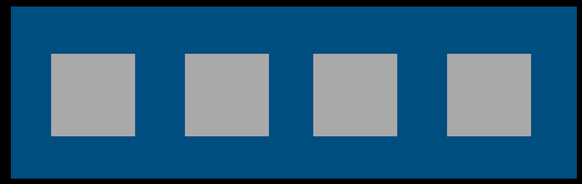




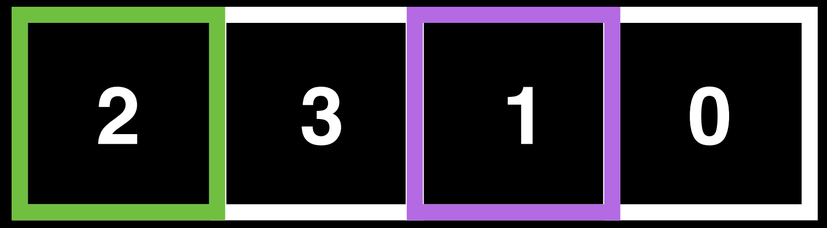
Thread-local shuffle vector

Shuffle Vector:
Fast + space-efficient random malloc

Page



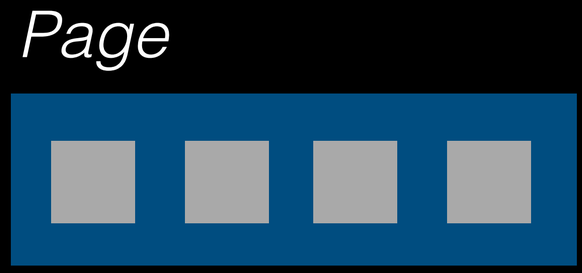
```
free( )

shuffle_one(
```




```
)
```

Thread-local shuffle vector

Shuffle Vector:
Fast + space-efficient random malloc

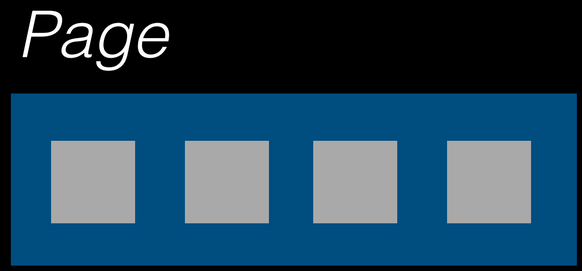


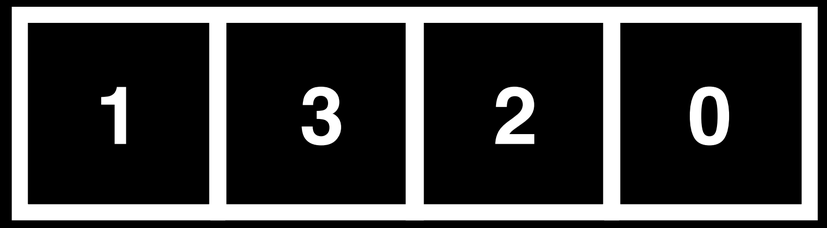

```
free( )

shuffle_one( 1 3 2 0 )
```

Thread-local shuffle vector

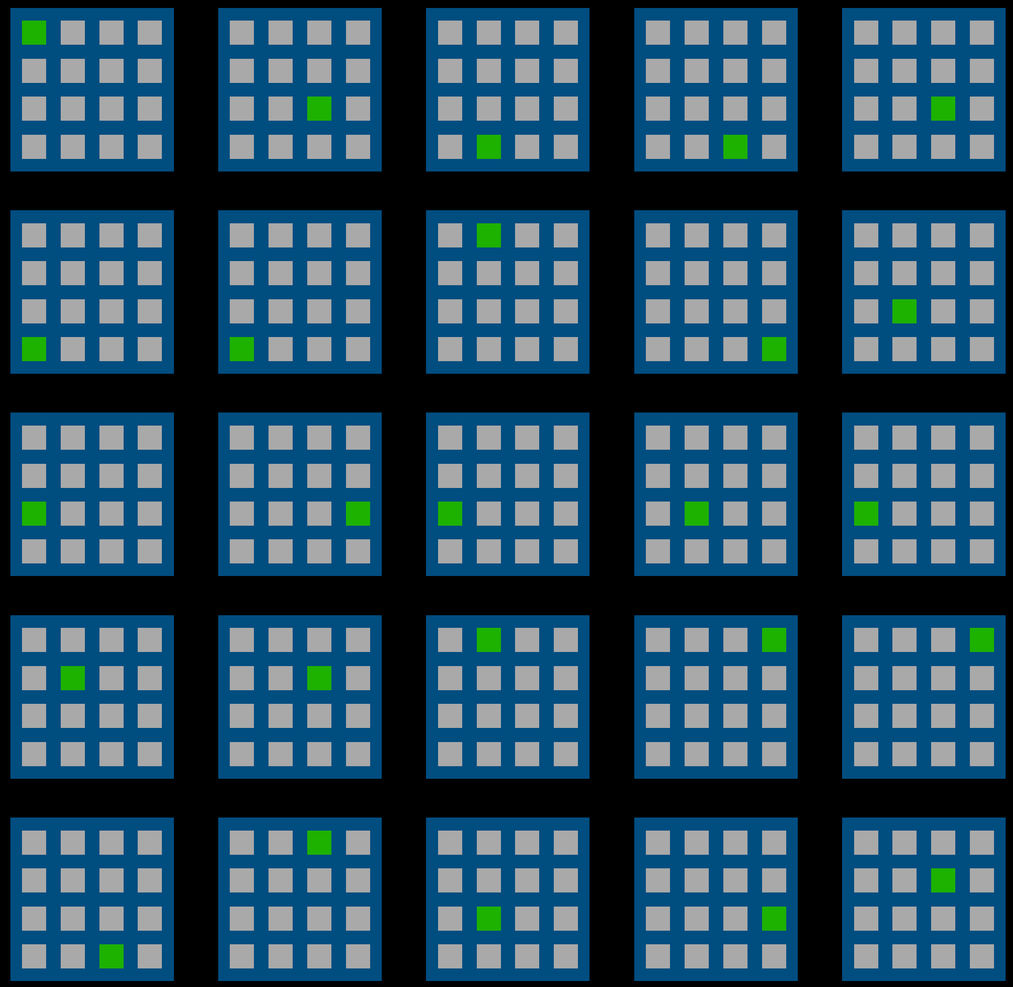
Shuffle Vector:
Fast + space-efficient random malloc

```
free( )
```

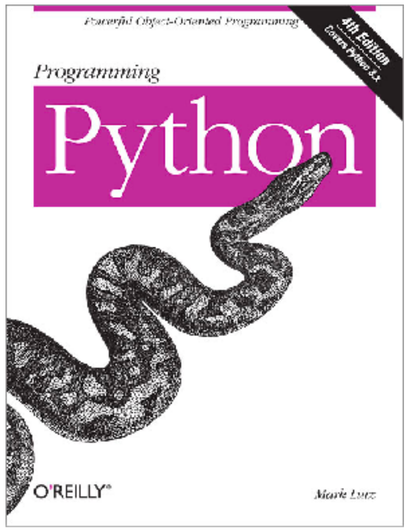


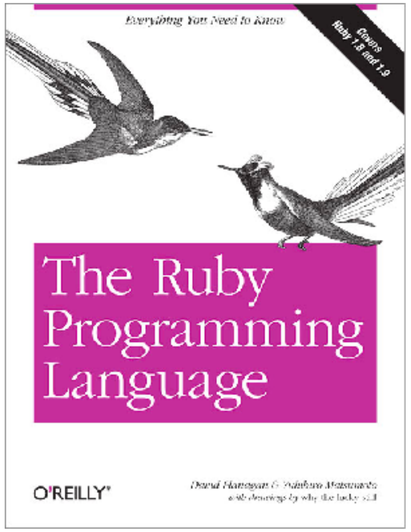


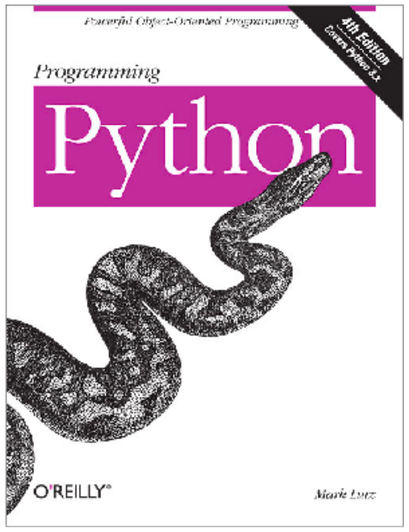
Thread-local shuffle vector

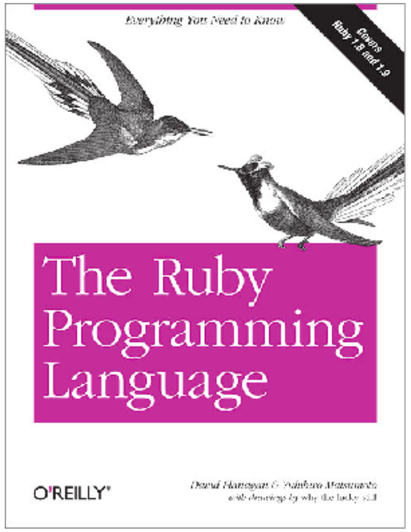


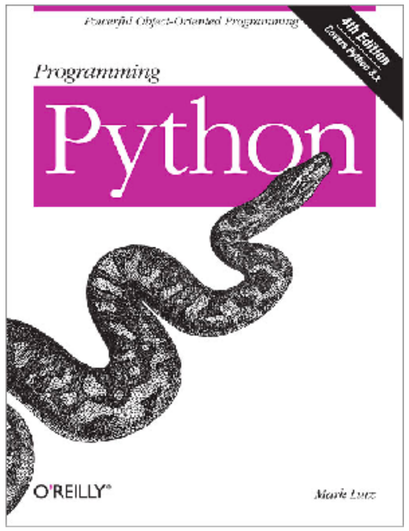
Many
Pages
Meshable

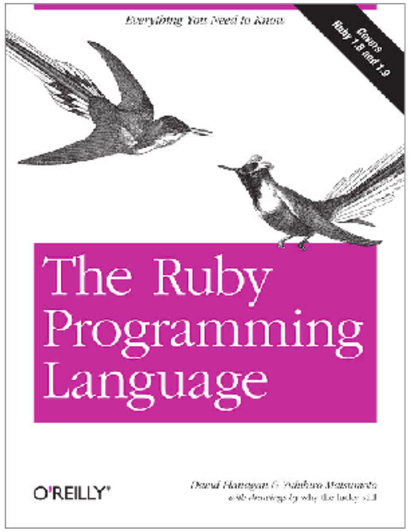

Programming
Python
O'REILLY
Mark Lutz


The Ruby
Programming
Language
O'REILLY

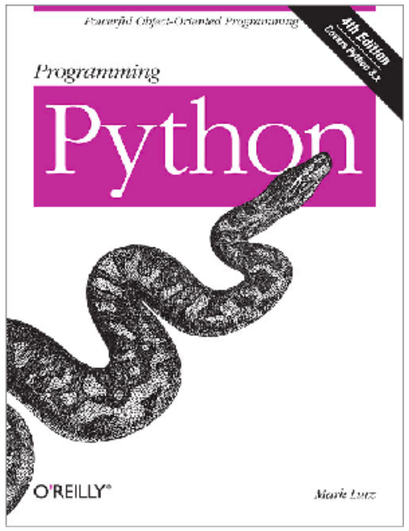

Programming
Python
O'REILLY
Mark Lutz

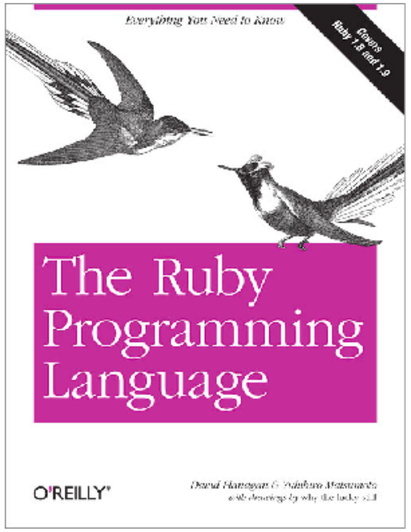

The Ruby
Programming
Language
O'REILLY





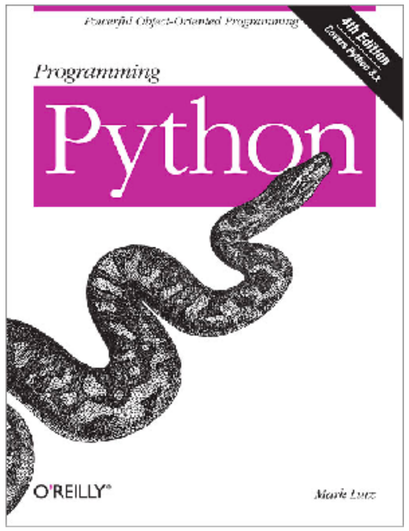
Q: Python: bytes for an int?

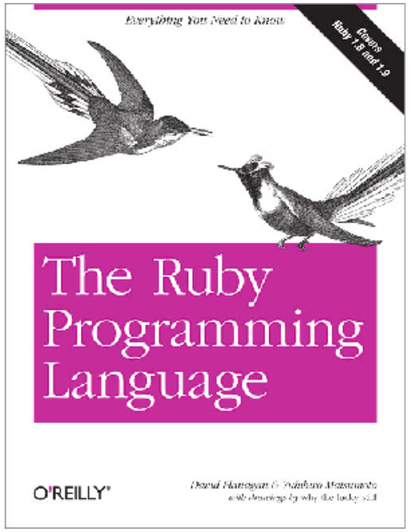




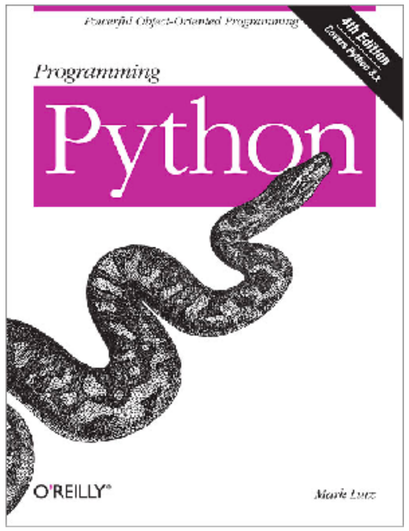
Q: Python: bytes for an int?

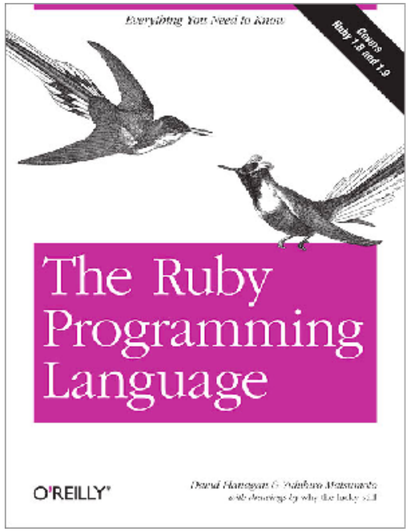
A: 28





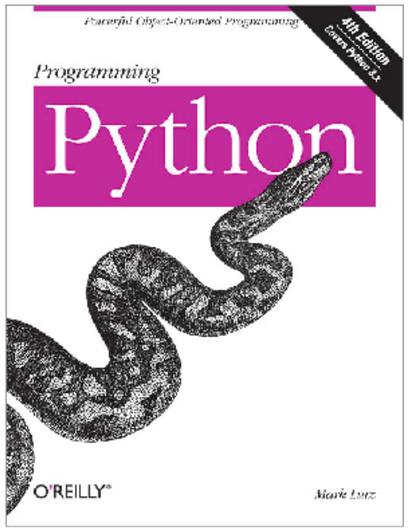
Q: Python: bytes in *empty* dict?

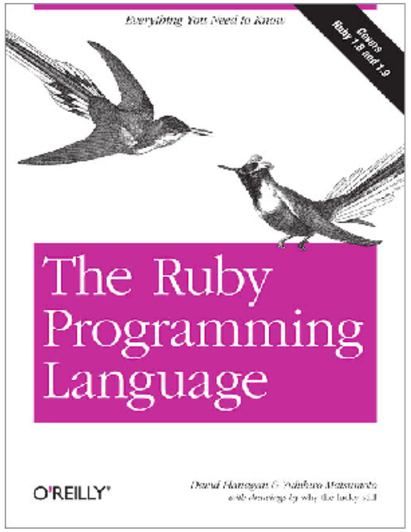




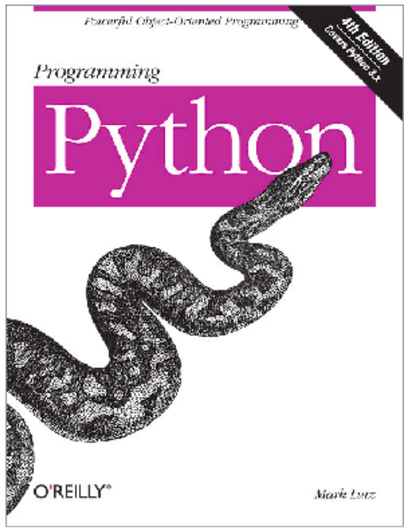
Q: Python: bytes in *empty* dict?

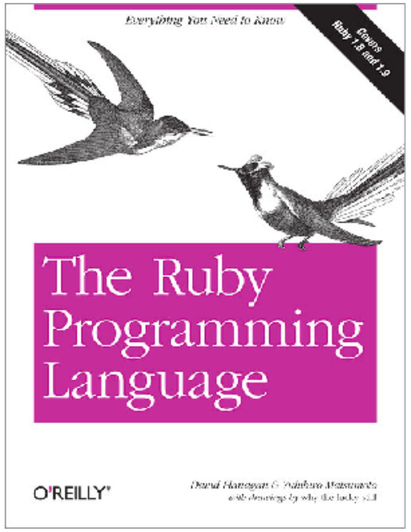
A: 240





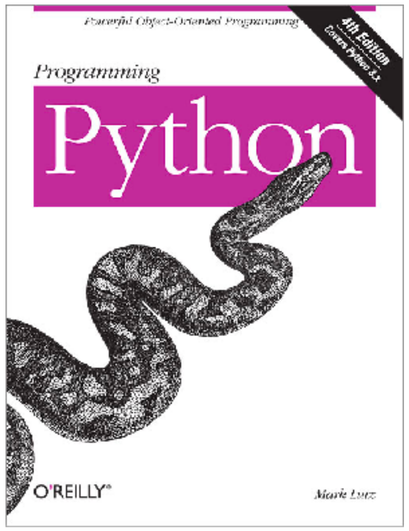
GC, but no / partial compaction

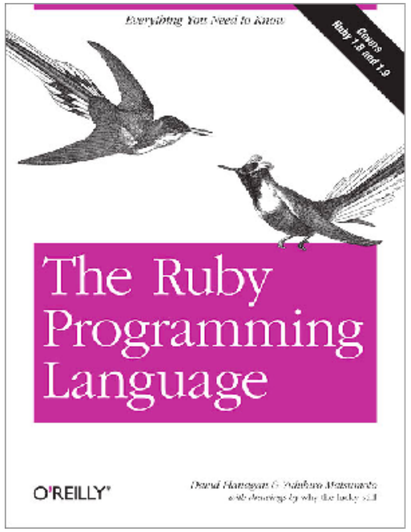




GC, but no / partial compaction

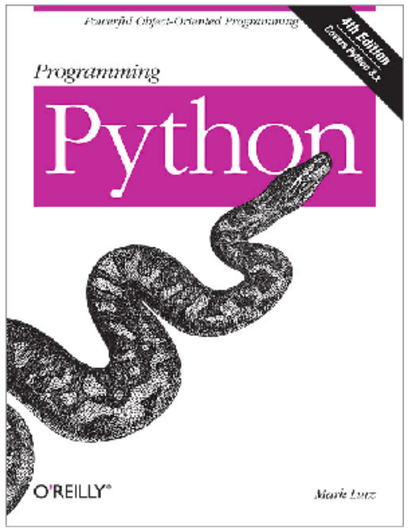
- Mesh = compaction for free!

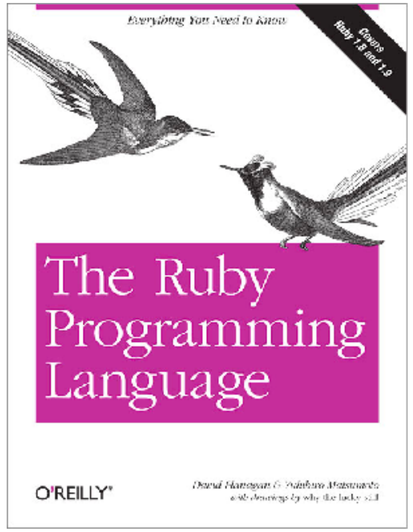




GC, but no / partial compaction

- **Mesh = compaction for free!**
- ***integrated* GC + Mesh**



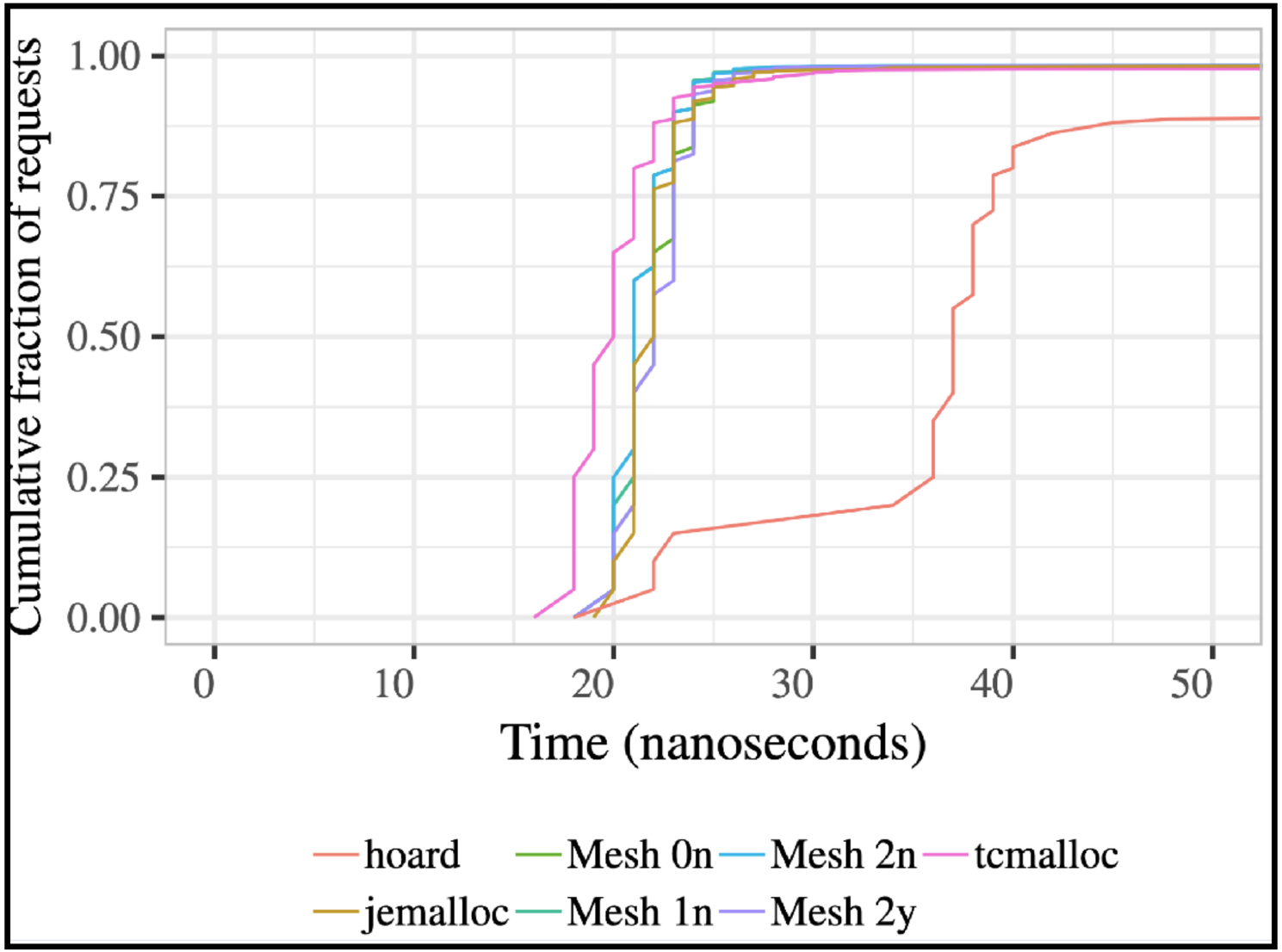


GC, but no / partial compaction

- Mesh = compaction for free!
- *integrated* GC + Mesh
- *vectorized* Mesh operations

http:// LIBMESH .org

Latency: malloc



Latency: free