Hauser Studio

Studio H

stdio.h

Mark KatzenMuenster
@katzenmuenster

Are all the shorts and videos for @cs50 filmed in a studio on @Harvard campus? If so it should be called Studio H, like <stdio.h>.

7:22 PM - 20 Sep 2015

problem set 3

$O(n^2)$

```
 1  pick up phone book
 2  open to middle of phone book
 3  look at names
 4  if "Smith" is among names
 5      call Mike
 6  else if "Smith" is earlier in book
 7      open to middle of left half of book
 8      go to line 3
 9  else if "Smith" is later in book
10      open to middle of right half of book
11      go to line 3
12  else
13      give up
```

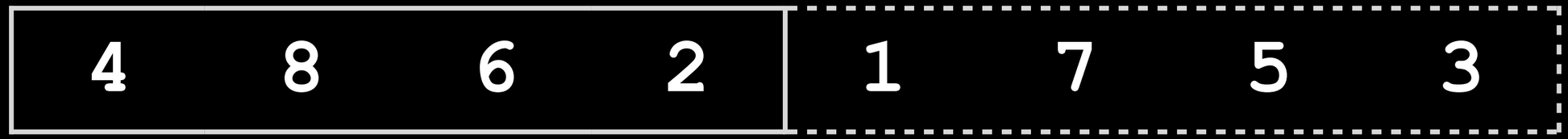
```
 1  pick up phone book
 2  open to middle of phone book
 3  look at names
 4  if "Smith" is among names
 5      call Mike
 6  else if "Smith" is earlier in book
 7      search for Mike in left half of book
 8
 9  else if "Smith" is later in book
10      search for Mike in right half of book
11
12  else
13      give up
```

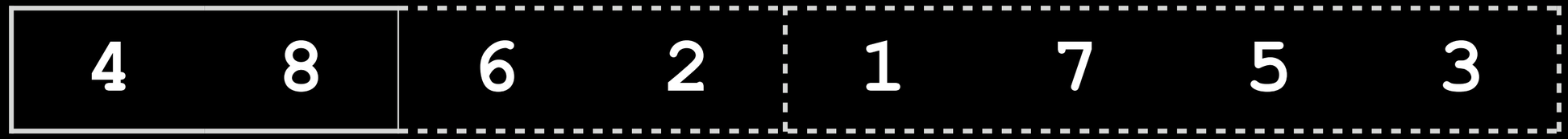
merge sort

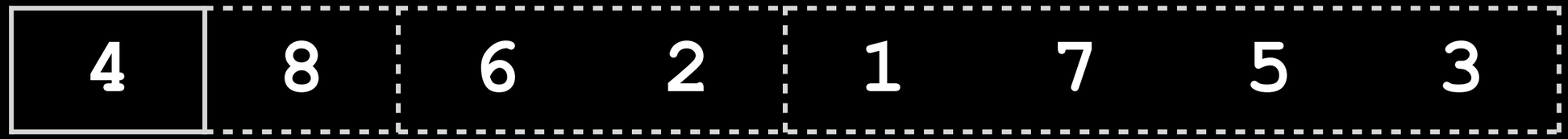
```
On input of n elements
    if n < 2
        return
    else
        sort left half of elements
        sort right half of elements
        merge sorted halves
```

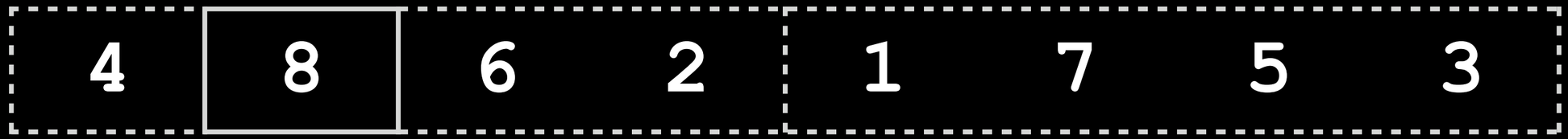
4 8 6 2 1 7 5 3

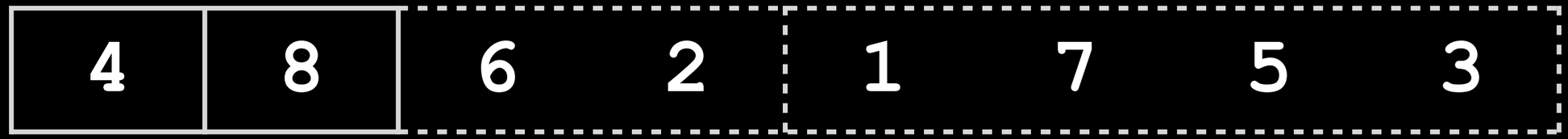

4 8 6 2 1 7 5 3

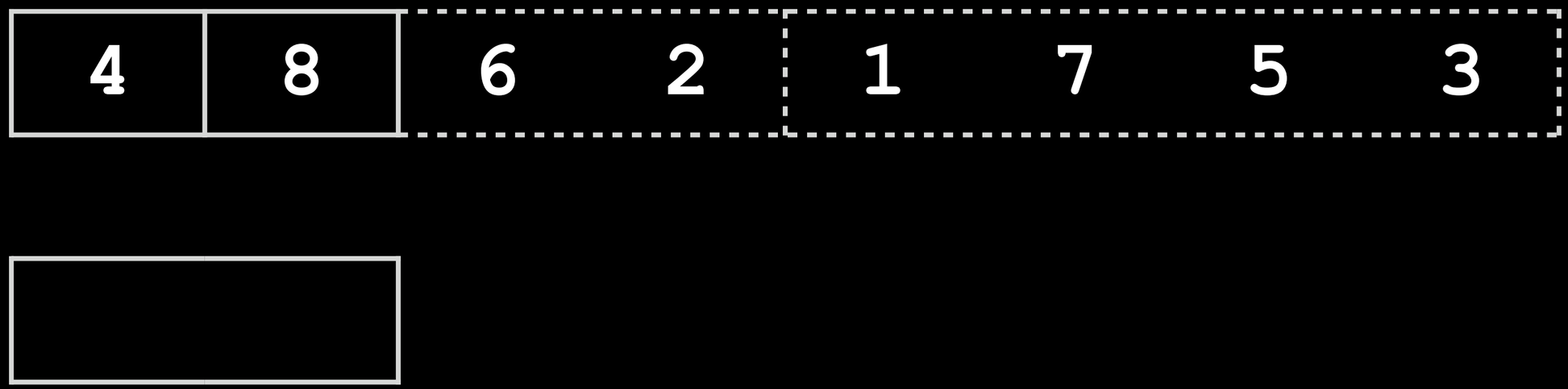

4 8 6 2
1 7 5 3

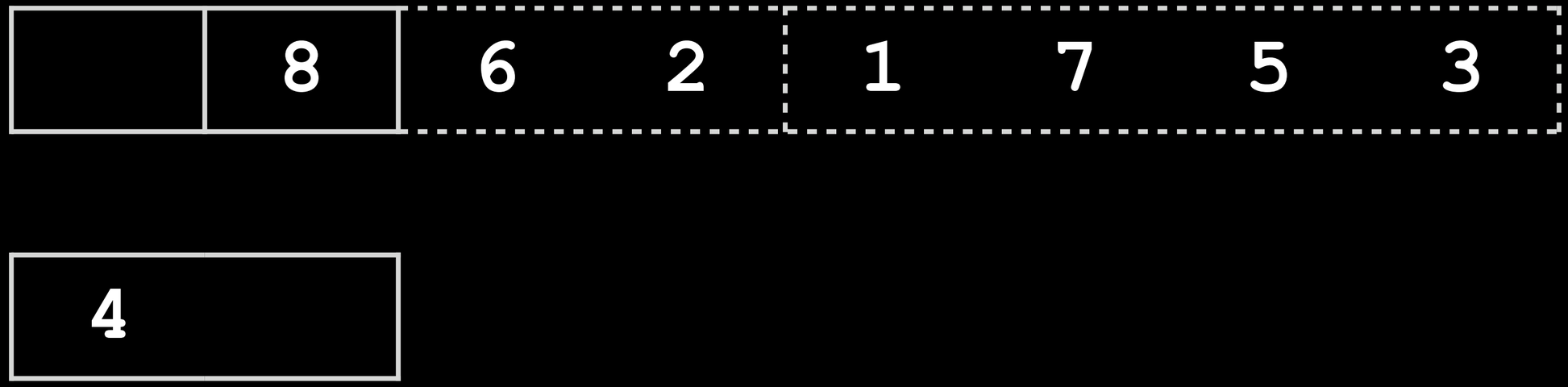

4 8 6 2 1 7 5 3

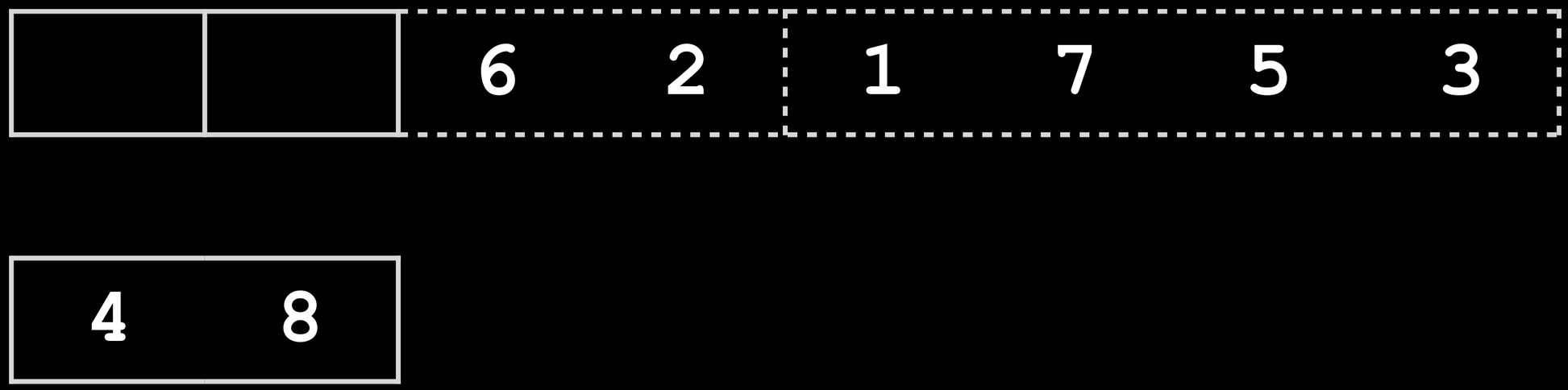

4
8
6
2
1
7
5
3

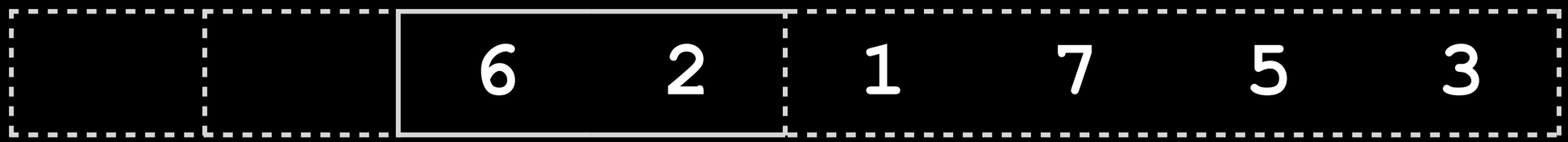

4
8
6
2
1
7
5
3


4
8
6
2
1
7
5
3

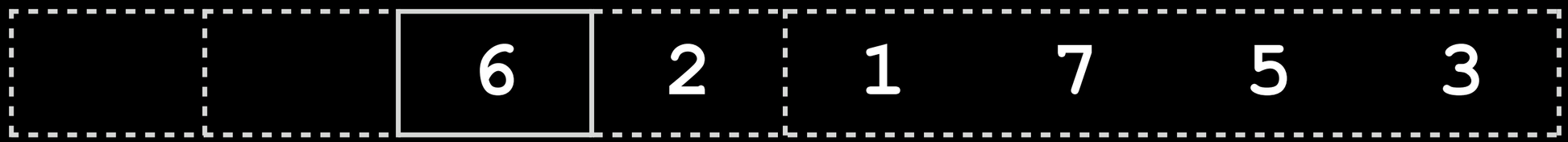

4
8
6
2
1
7
5
3


8
6
2
1
7
5
3
4

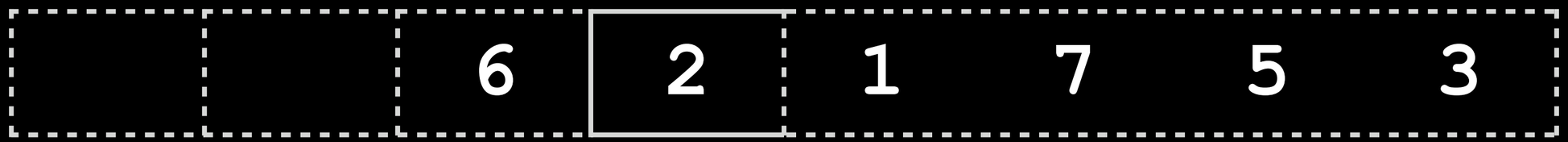

6
2
1
7
5
3
4
8


6 2 1 7 5 3

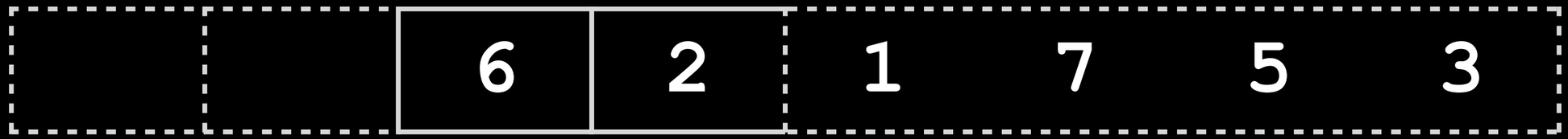

4 8


6
2
1
7
5
3

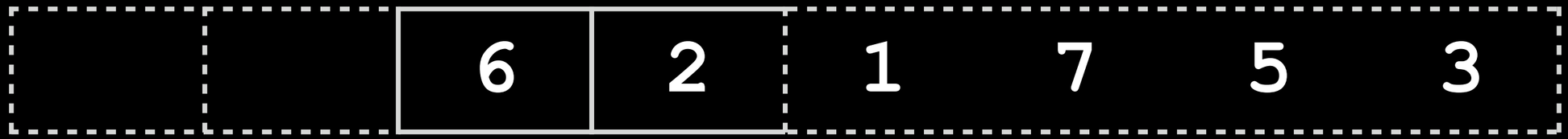

4
8

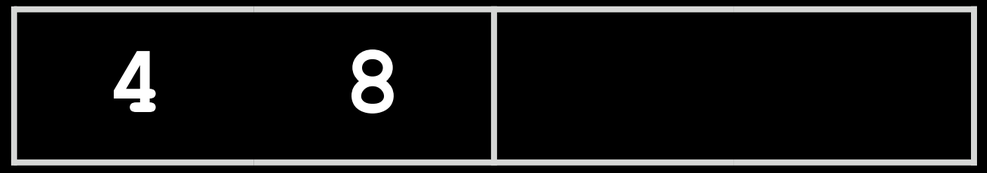

6
2
1
7
5
3

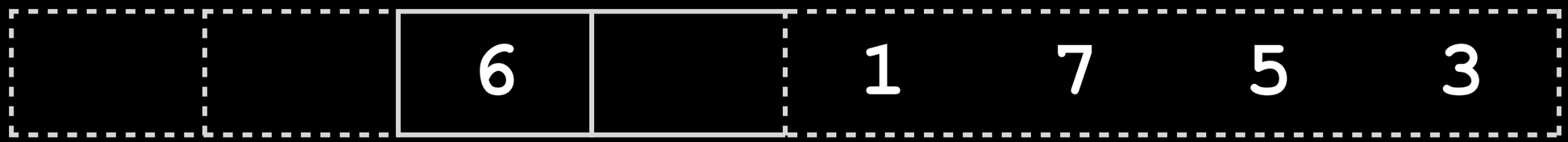

4
8

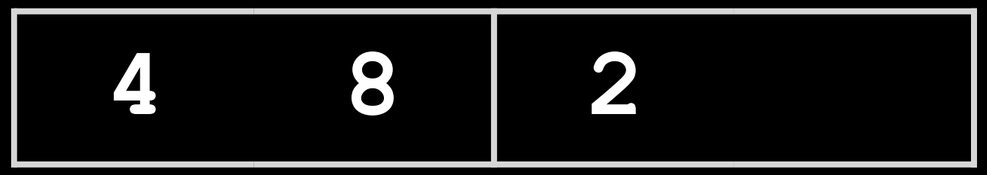

6
2
1
7
5
3

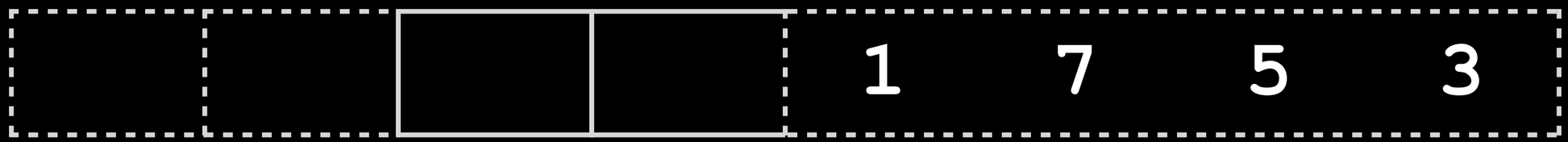

4
8

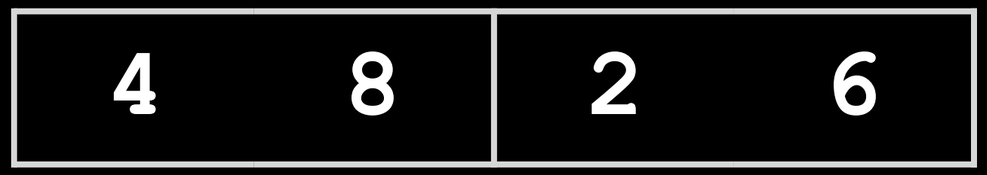

6
2
1
7
5
3

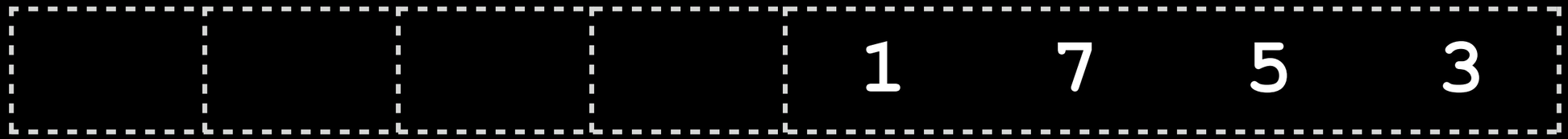

4
8

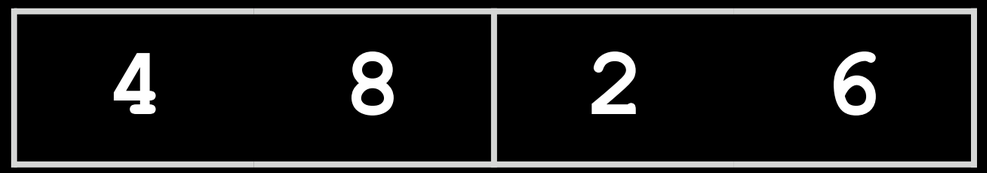

6
1
7
5
3

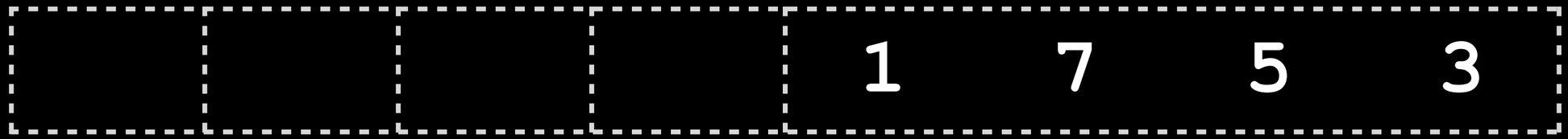

4
8
2

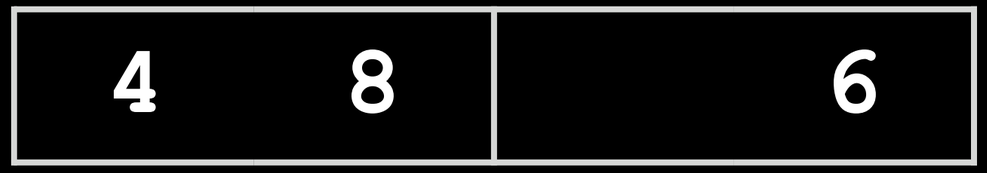

1
7
5
3


4
8
2
6

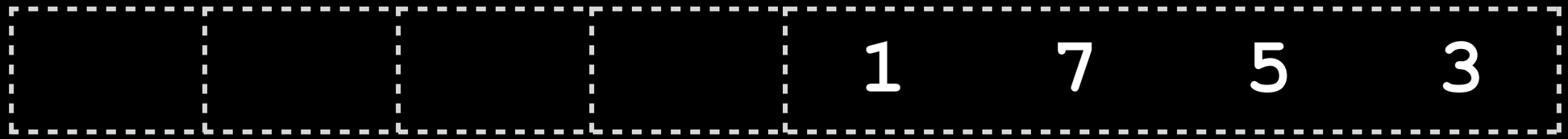

1
7
5
3

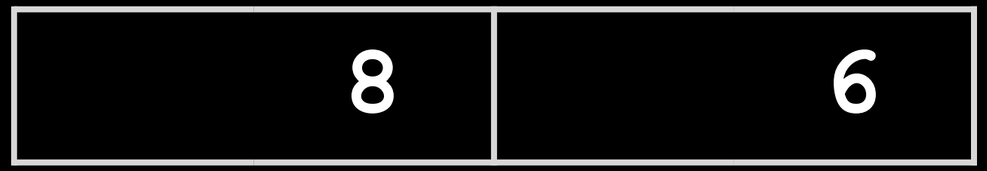

4
8
2
6


1
7
5
3

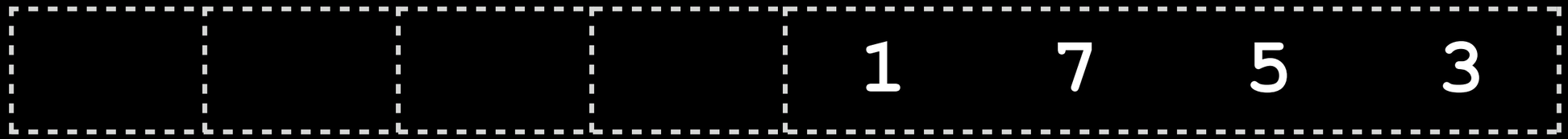

4
8
6

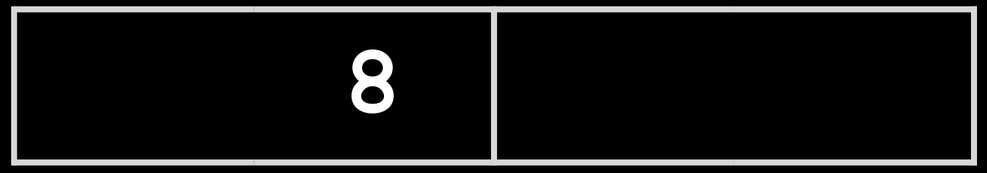

2


1
7
5
3

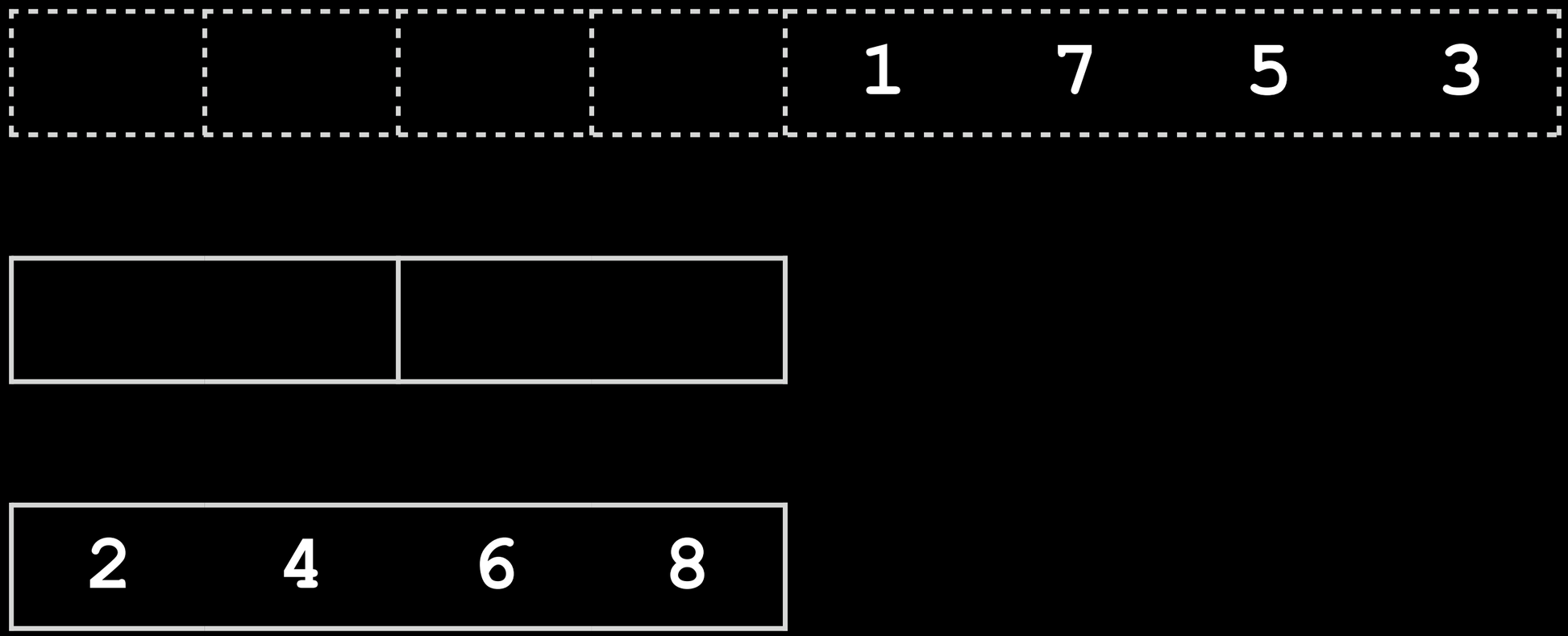

8
6

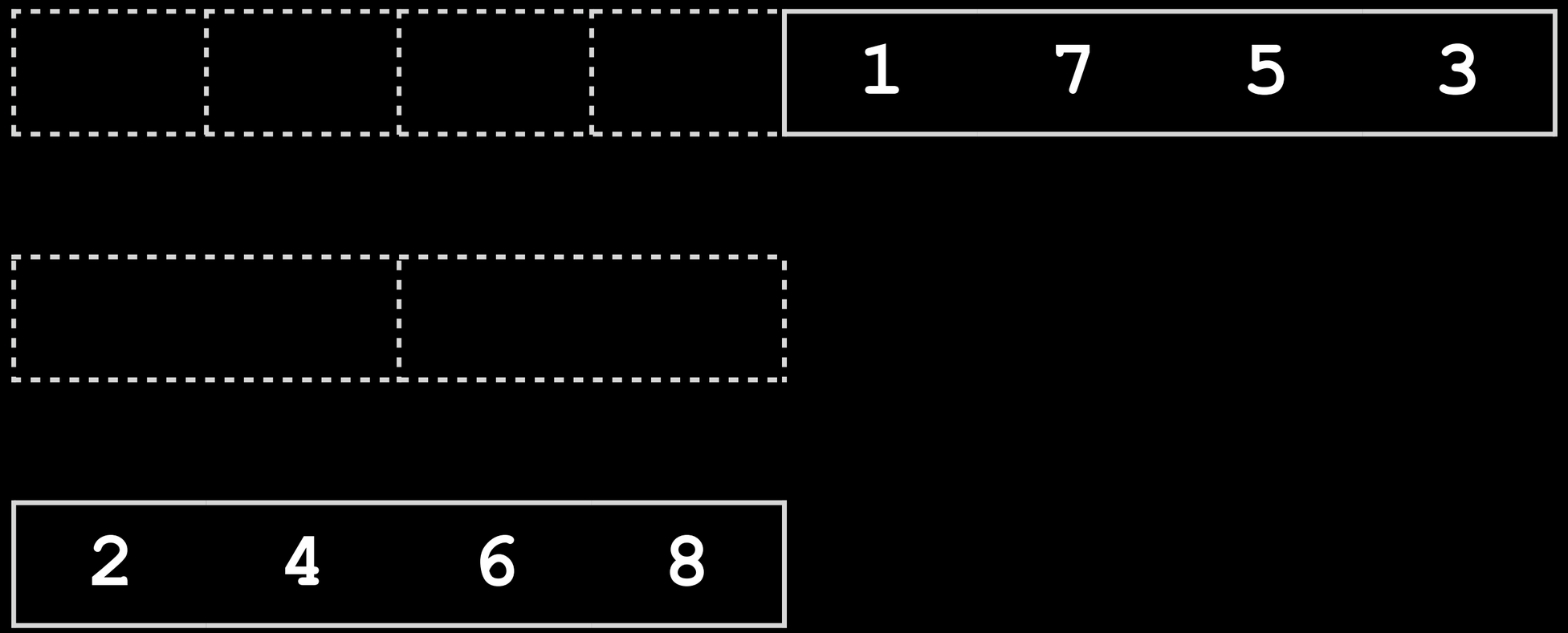

2
4

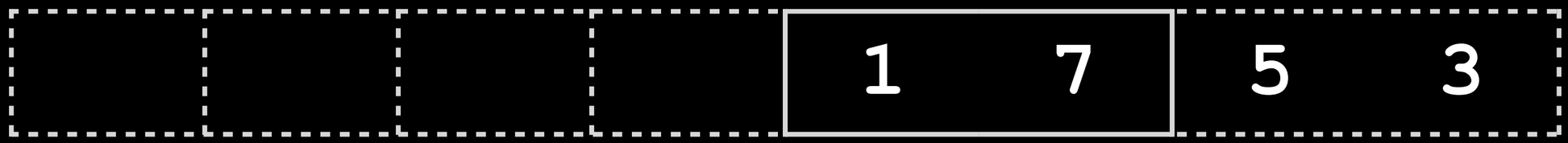

1
7
5
3

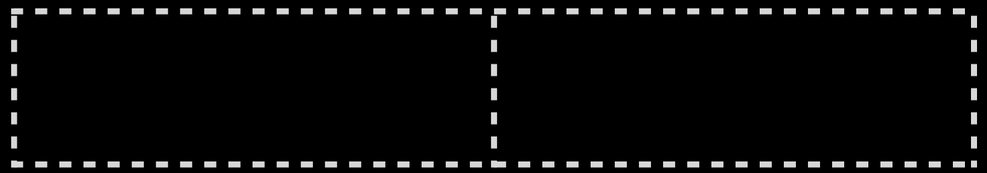

8


2
4
6

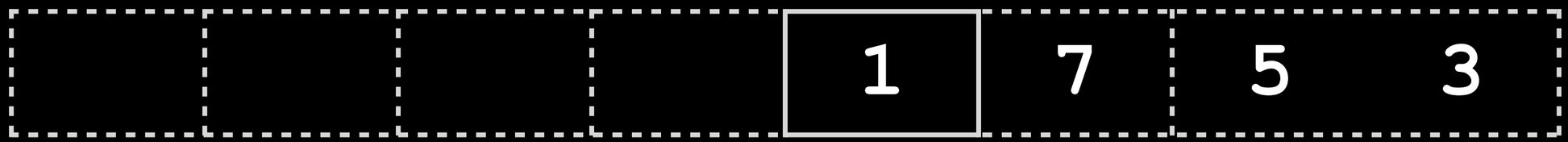

1 7 5 3
2 4 6 8

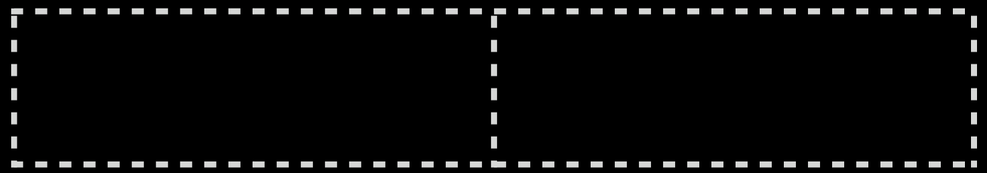

1 7 5 3
2 4 6 8


1
7
5
3

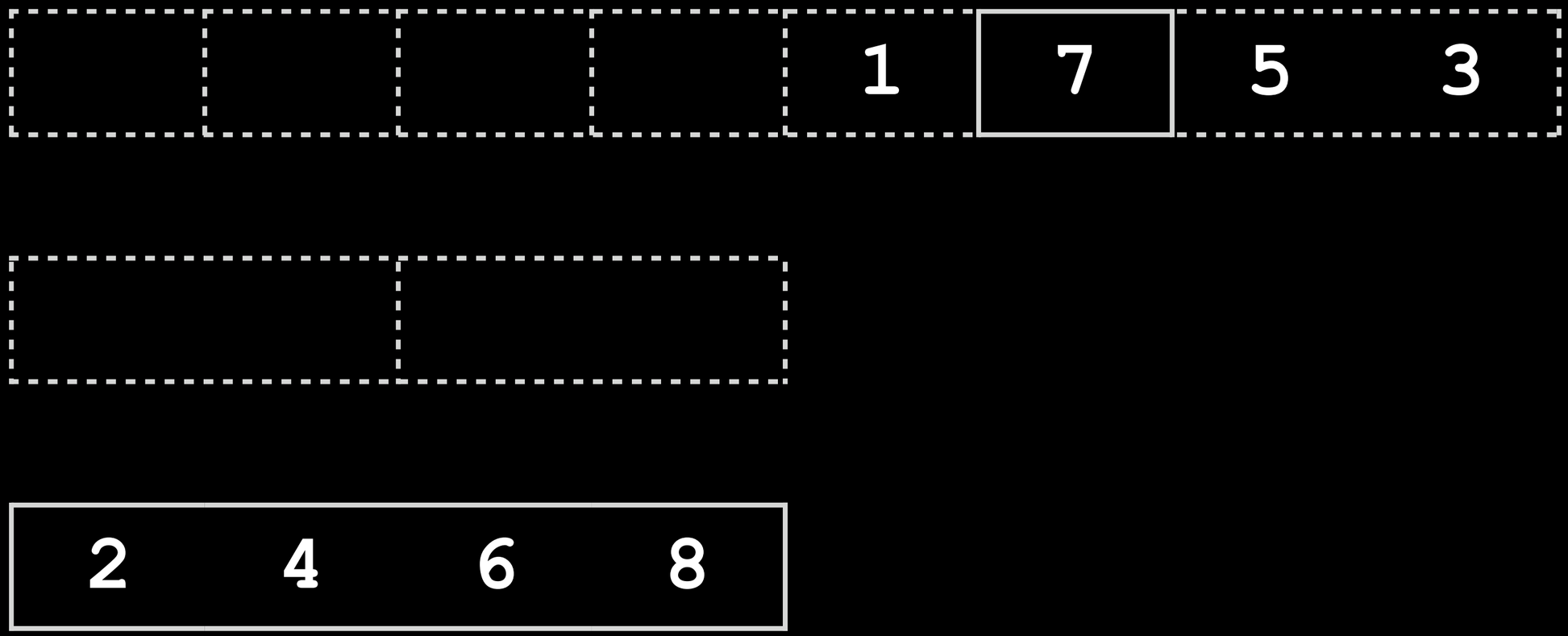


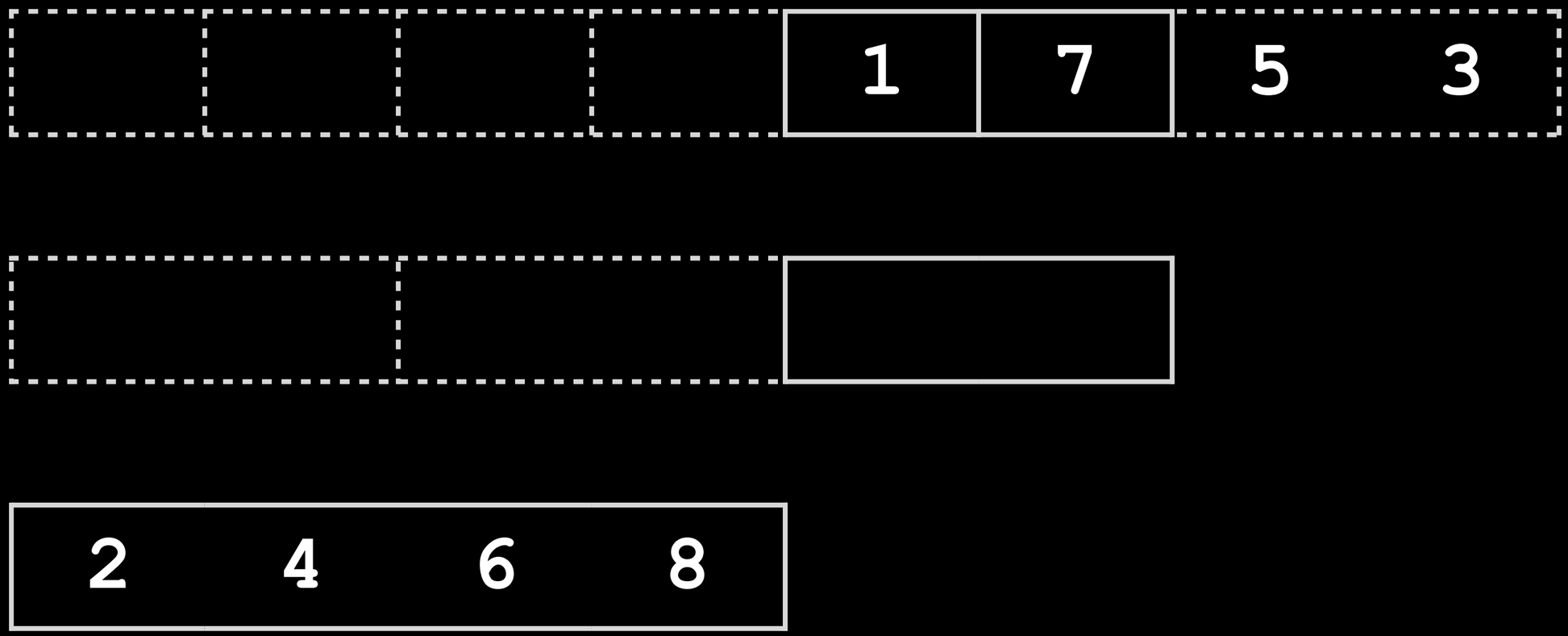

2
4
6
8

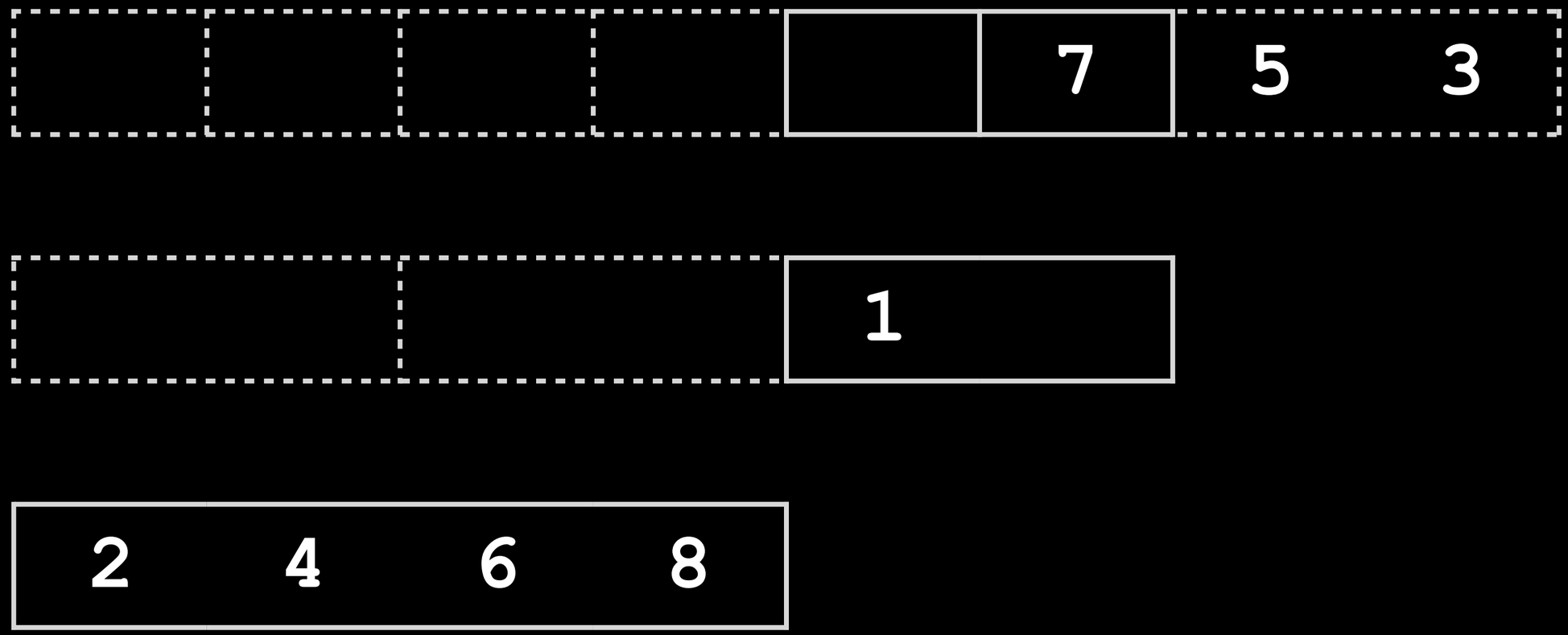

1
7
5
3

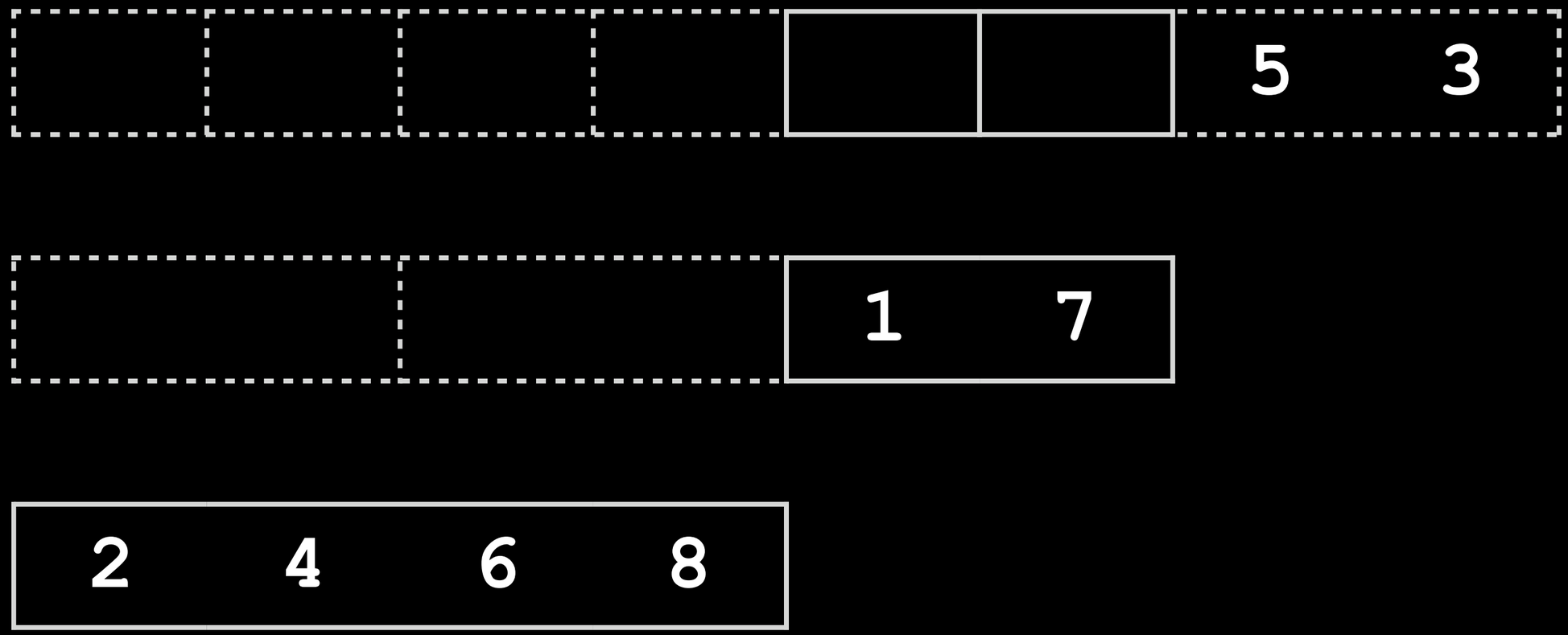


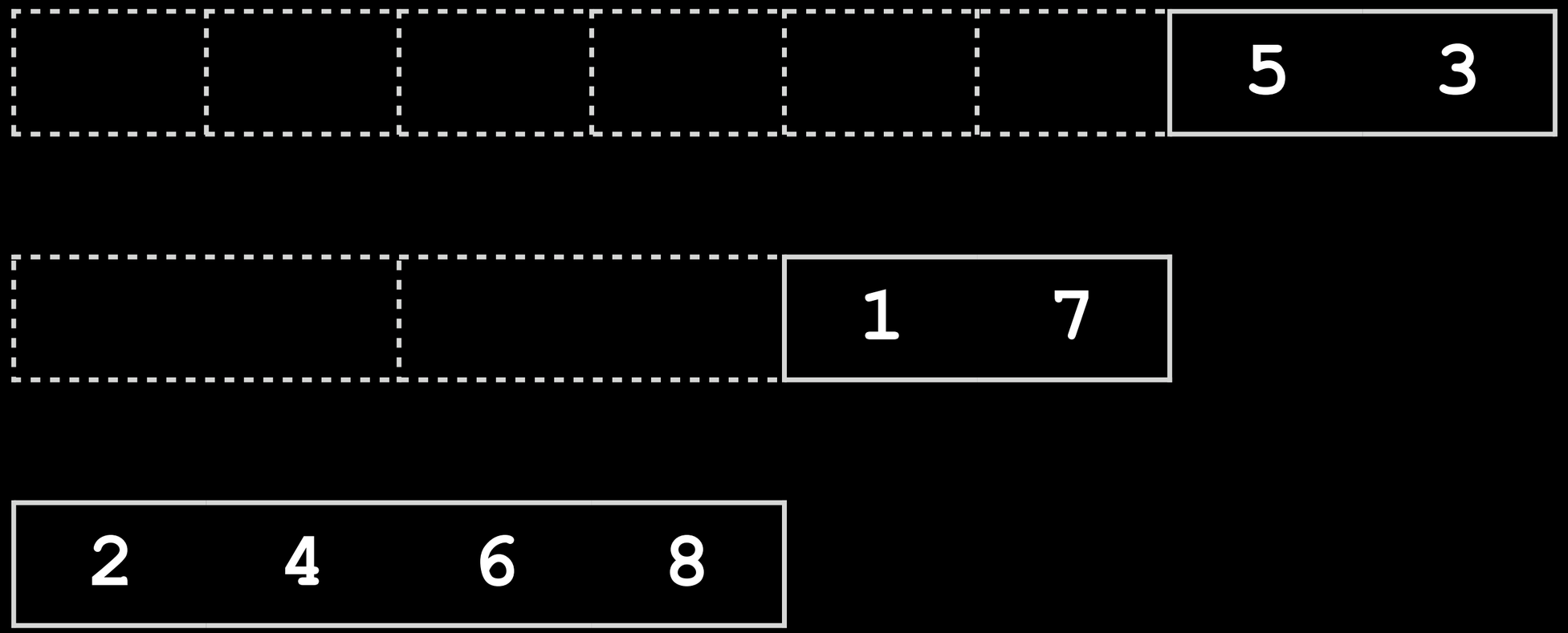

2
4
6
8

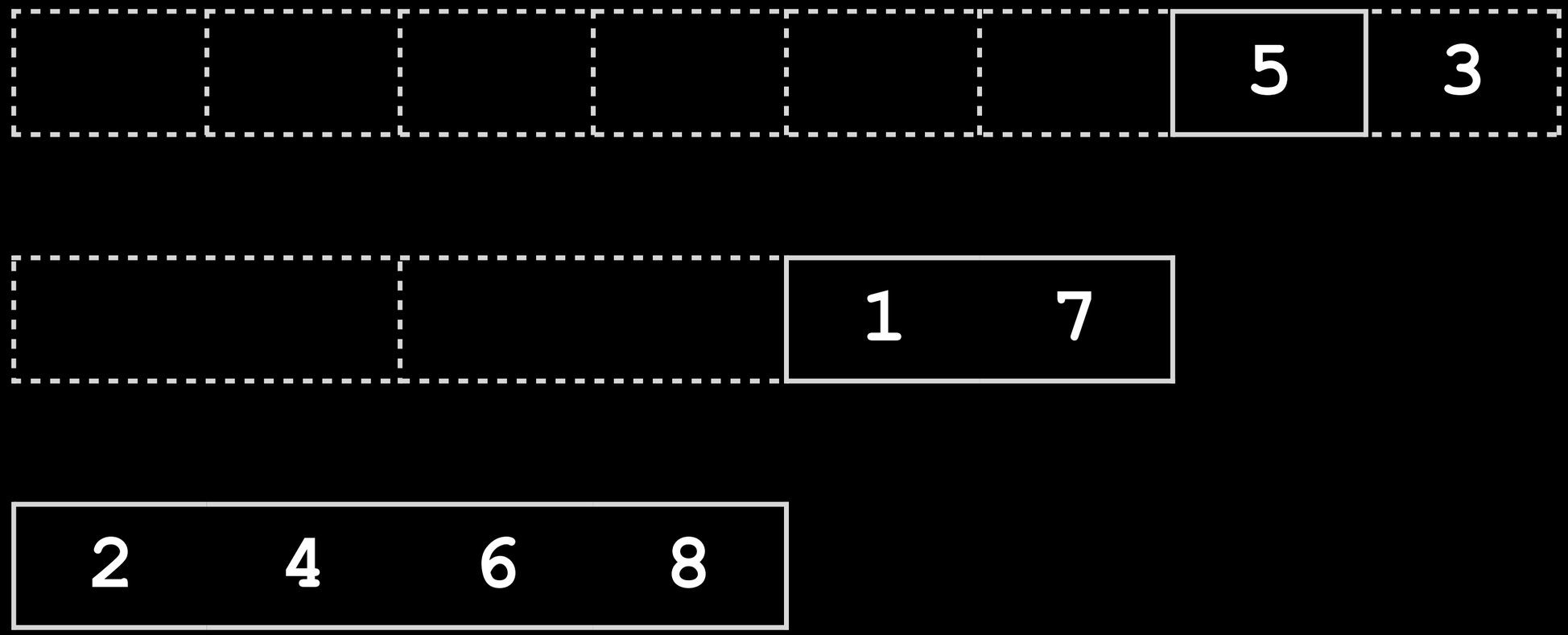

1
7
5
3
2
4
6
8

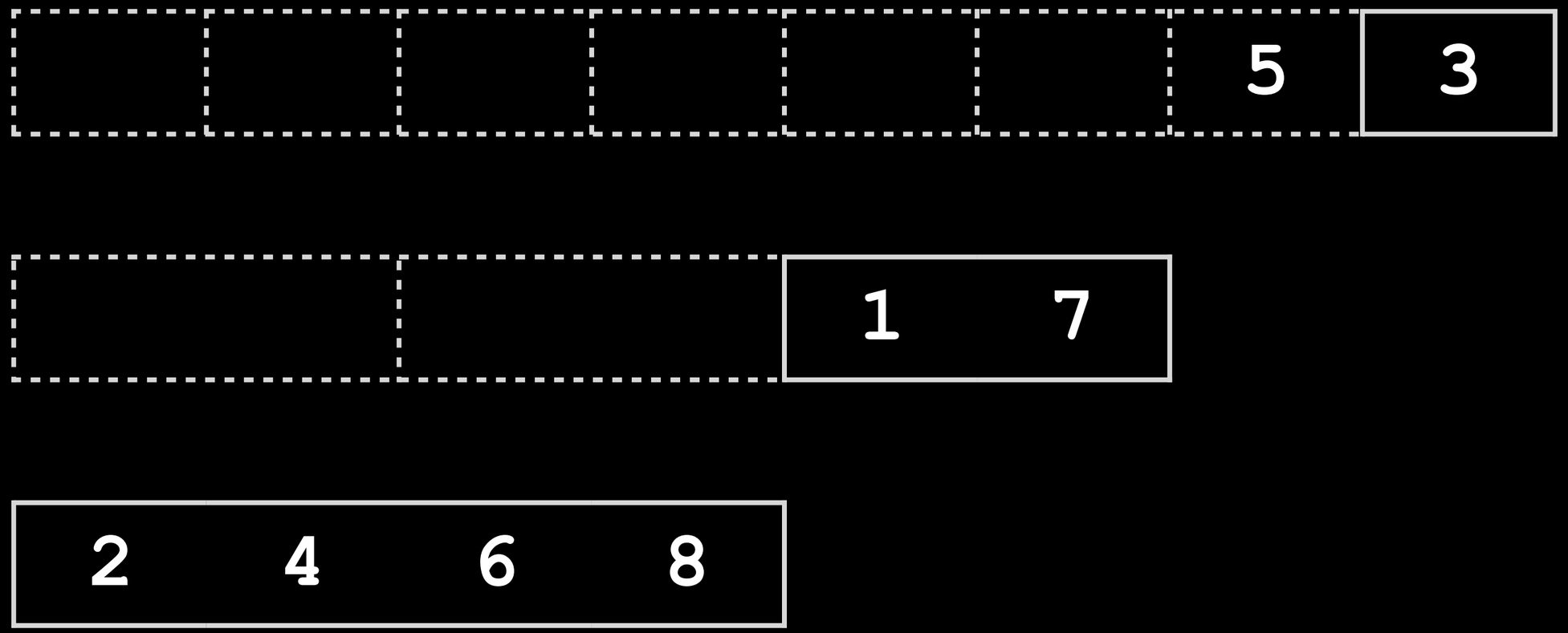

1
7
5
3
2
4
6
8

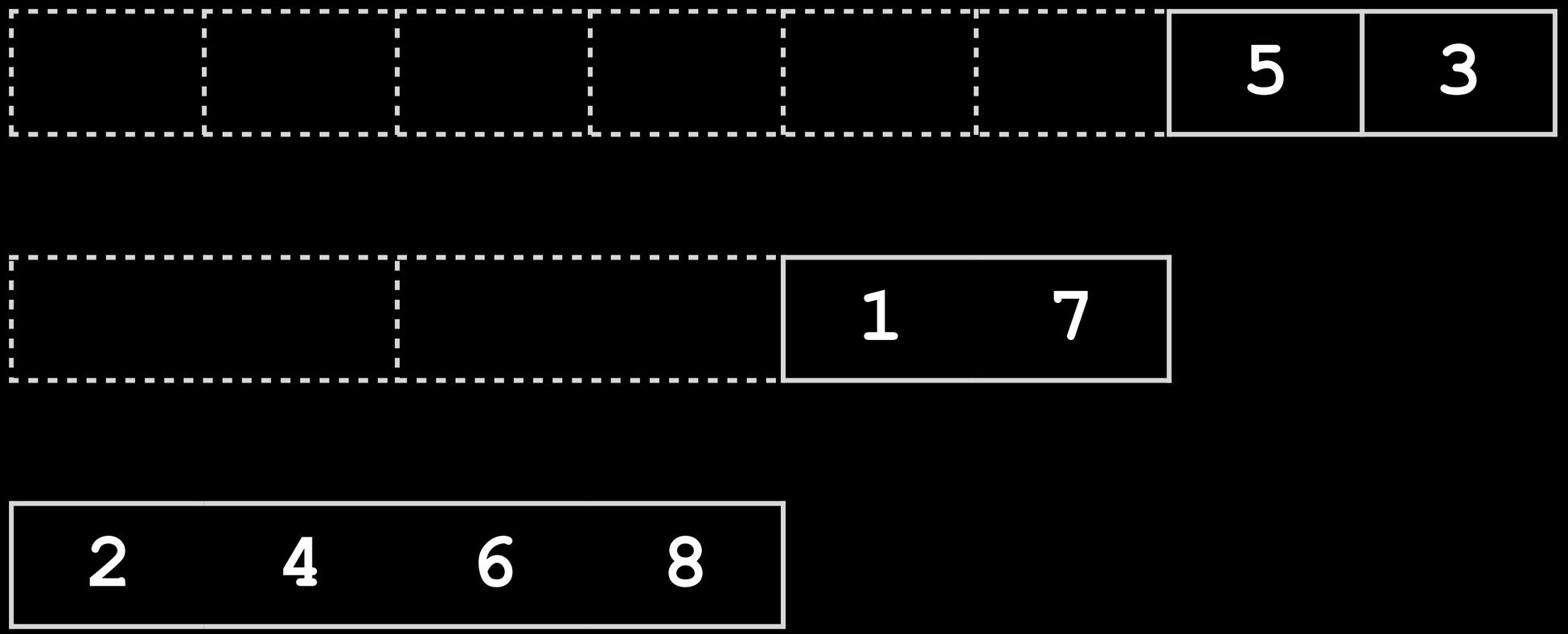

7
5
3
1
2
4
6
8

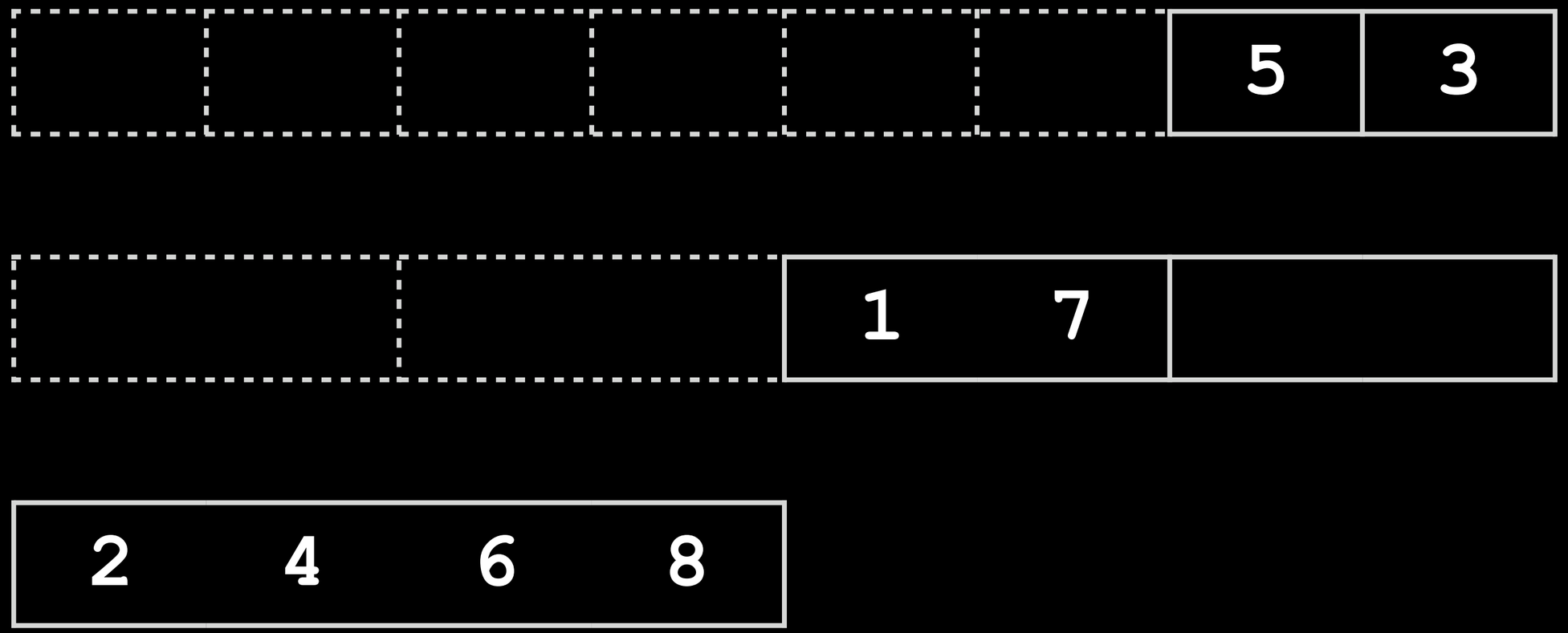

5
3
1
7
2
4
6
8

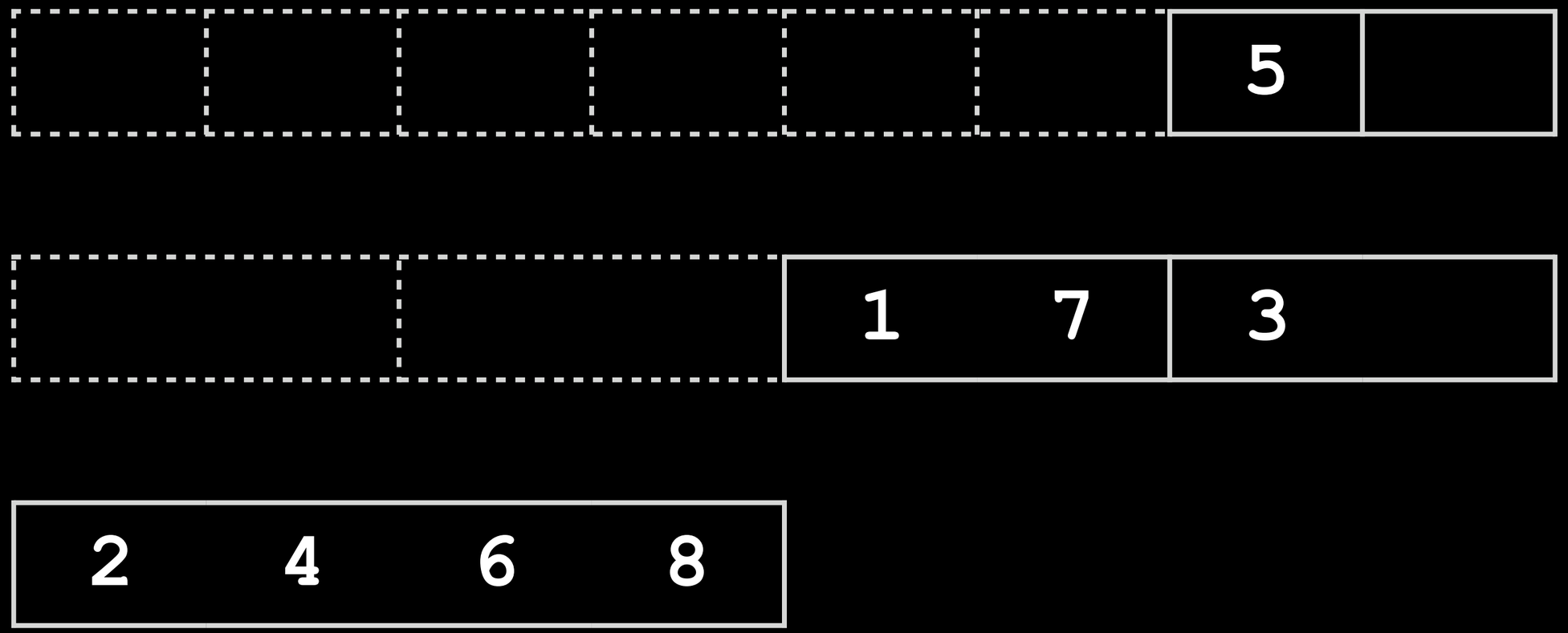

5 3
1 7
2 4 6 8

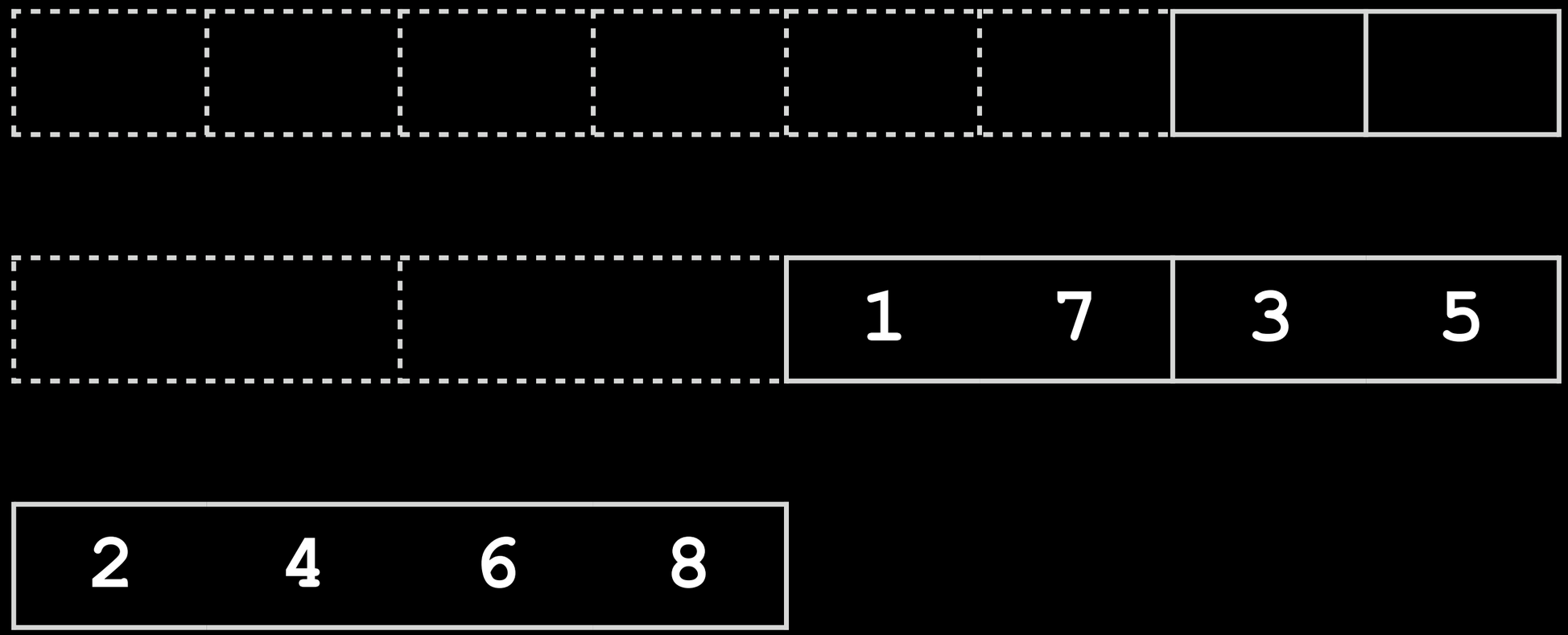

5
3
1
7
2
4
6
8

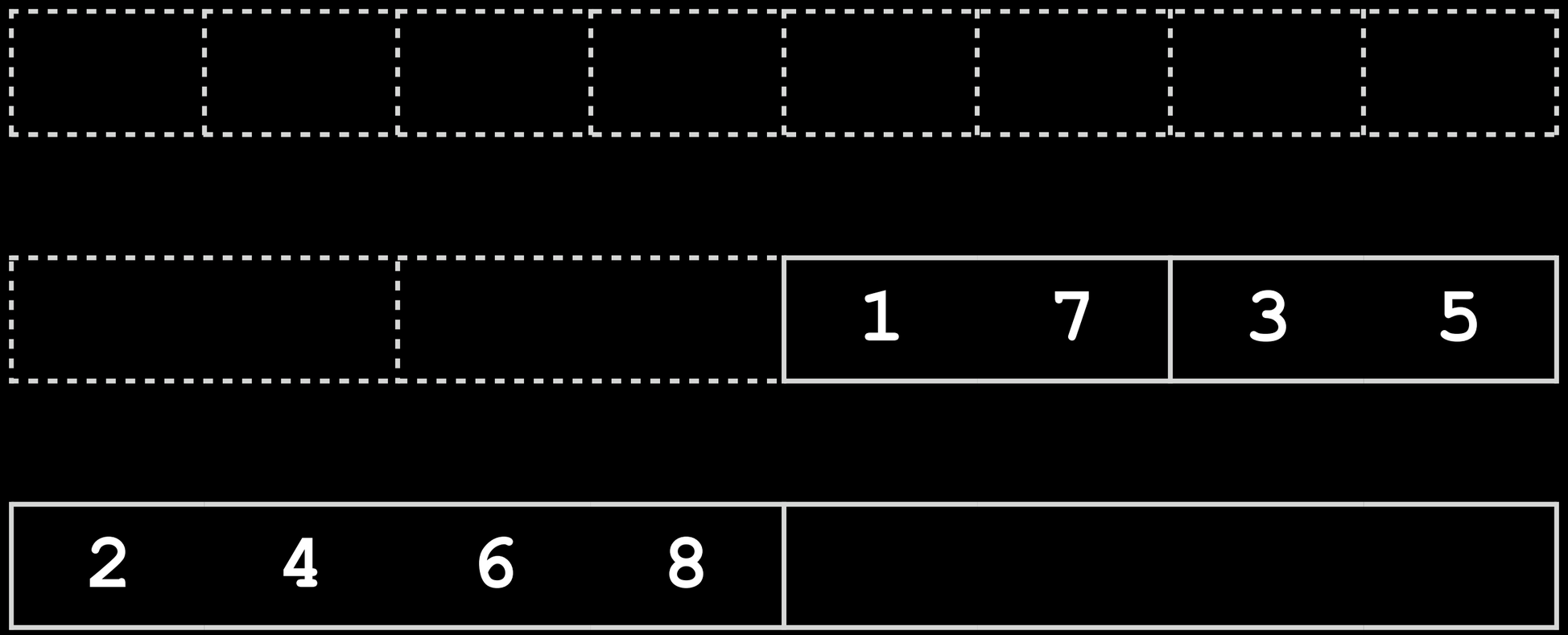

5
3
1
7
2
4
6
8

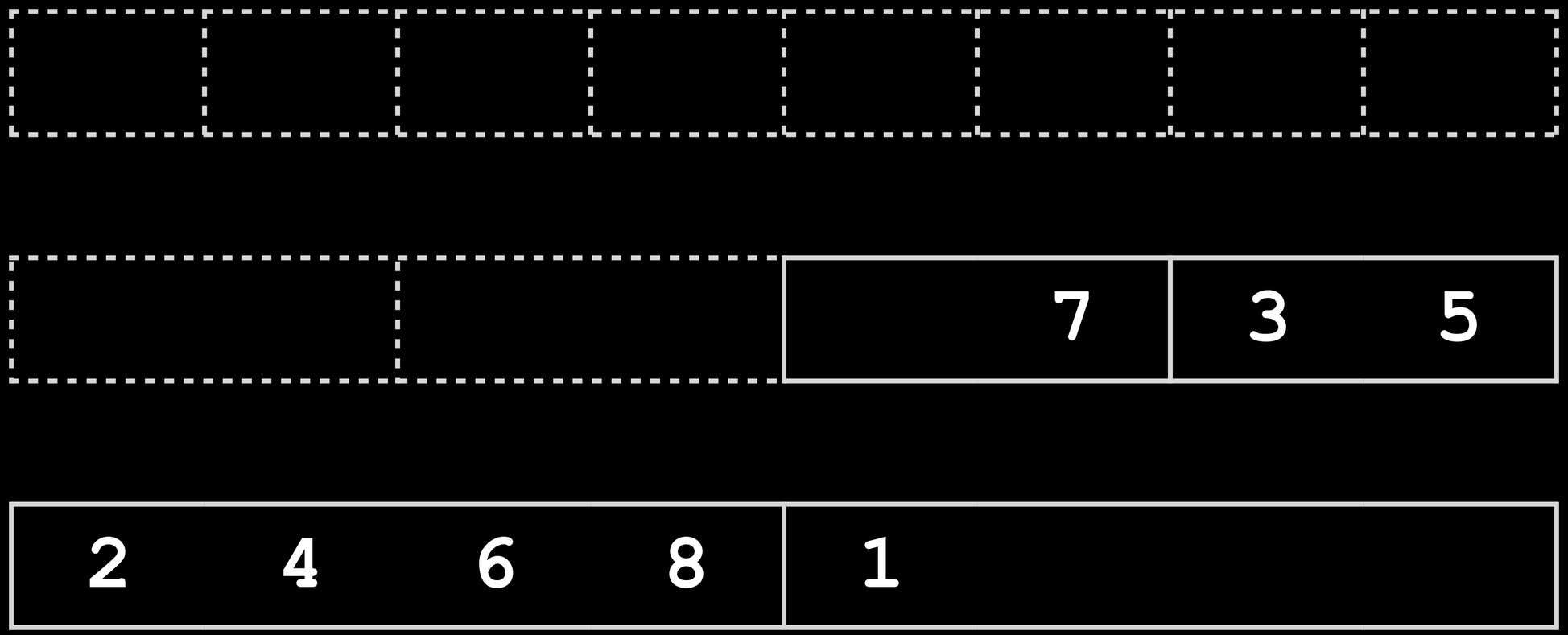

5
3
1
7
2
4
6
8

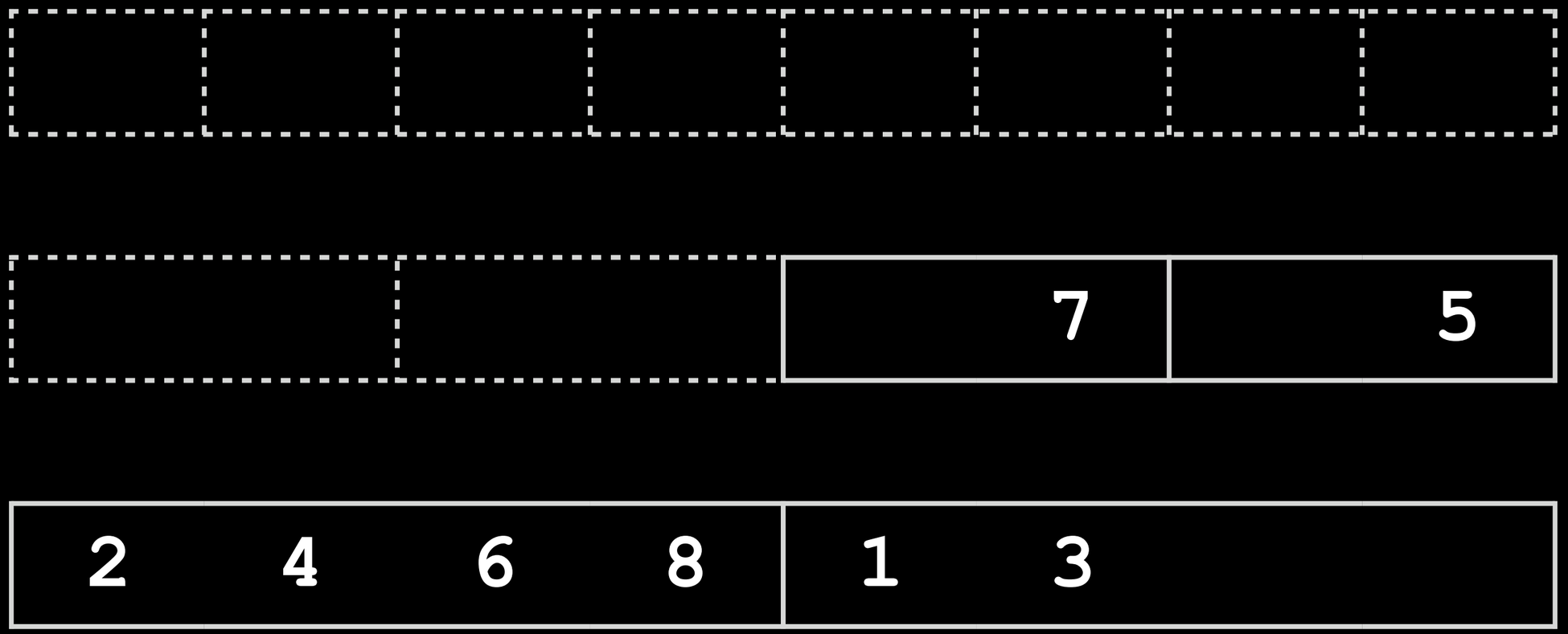

5
3
1
7
2
4
6
8

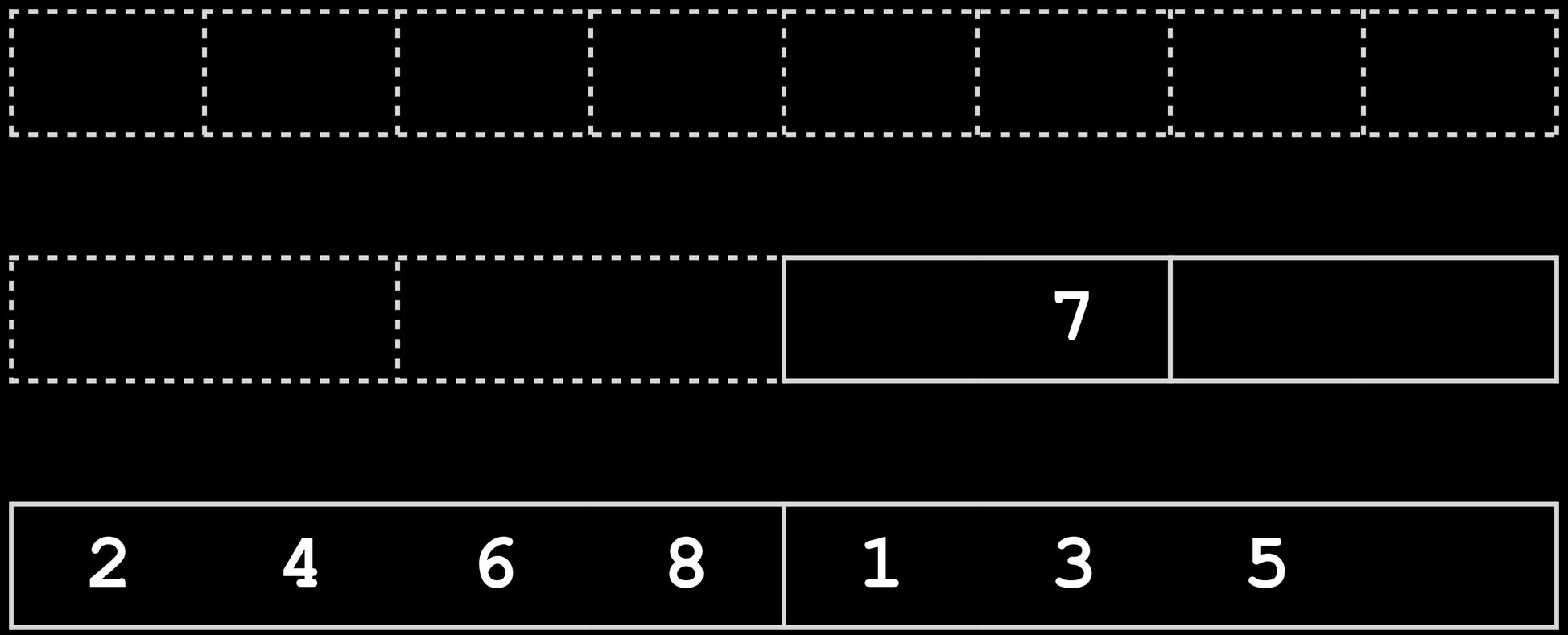

5
1
7
3
2
4
6
8

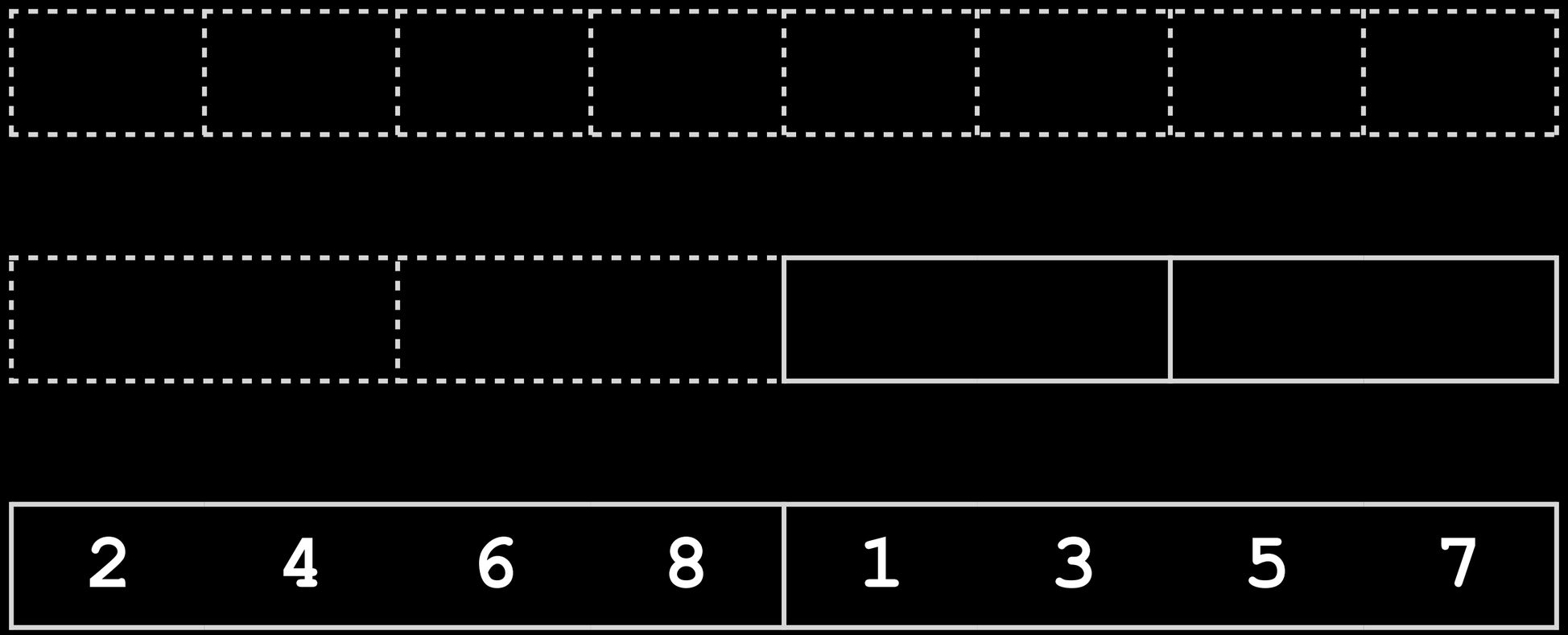

1
7
3
5
2
4
6
8

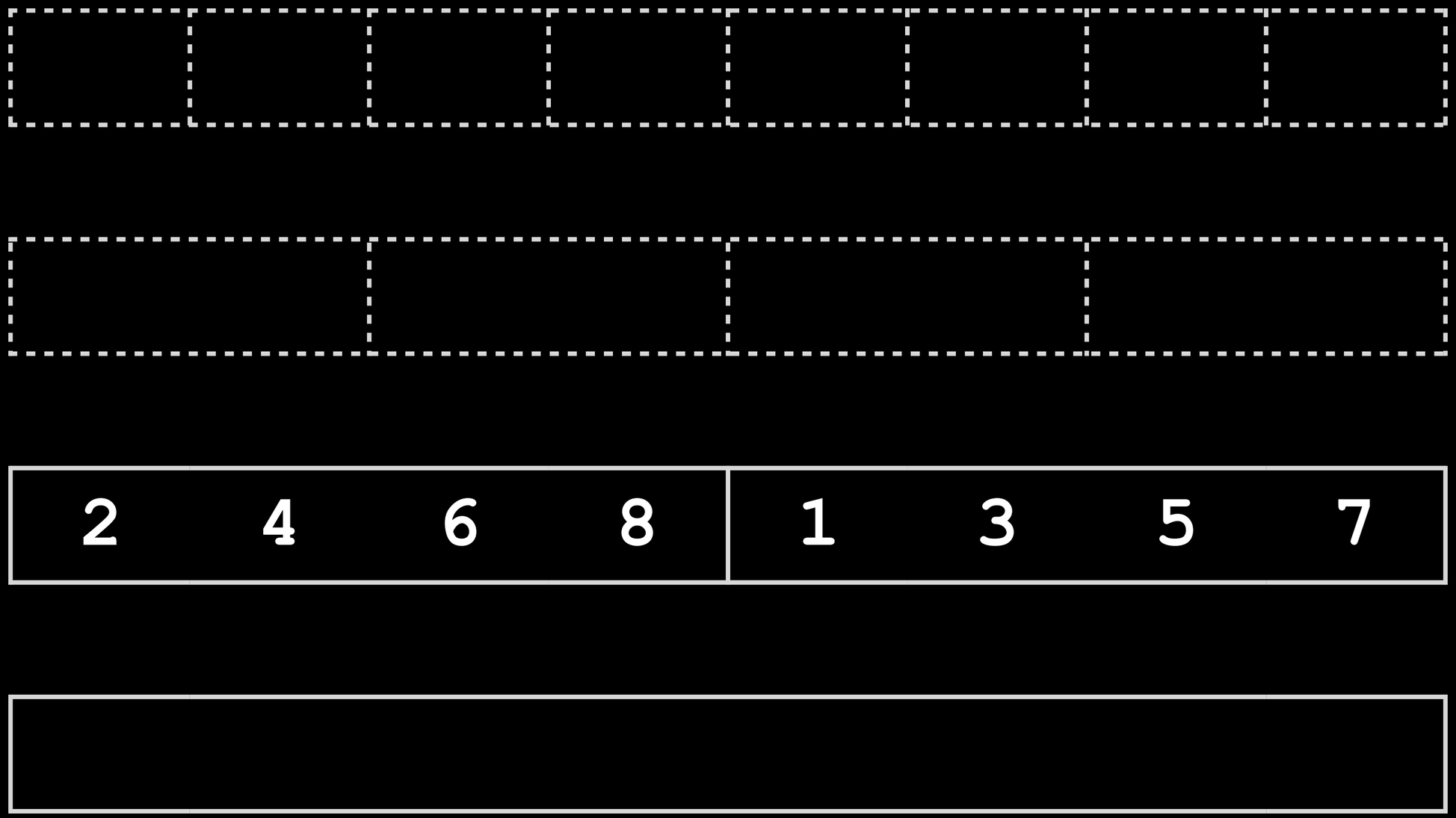

1
7
3
5
2
4
6
8

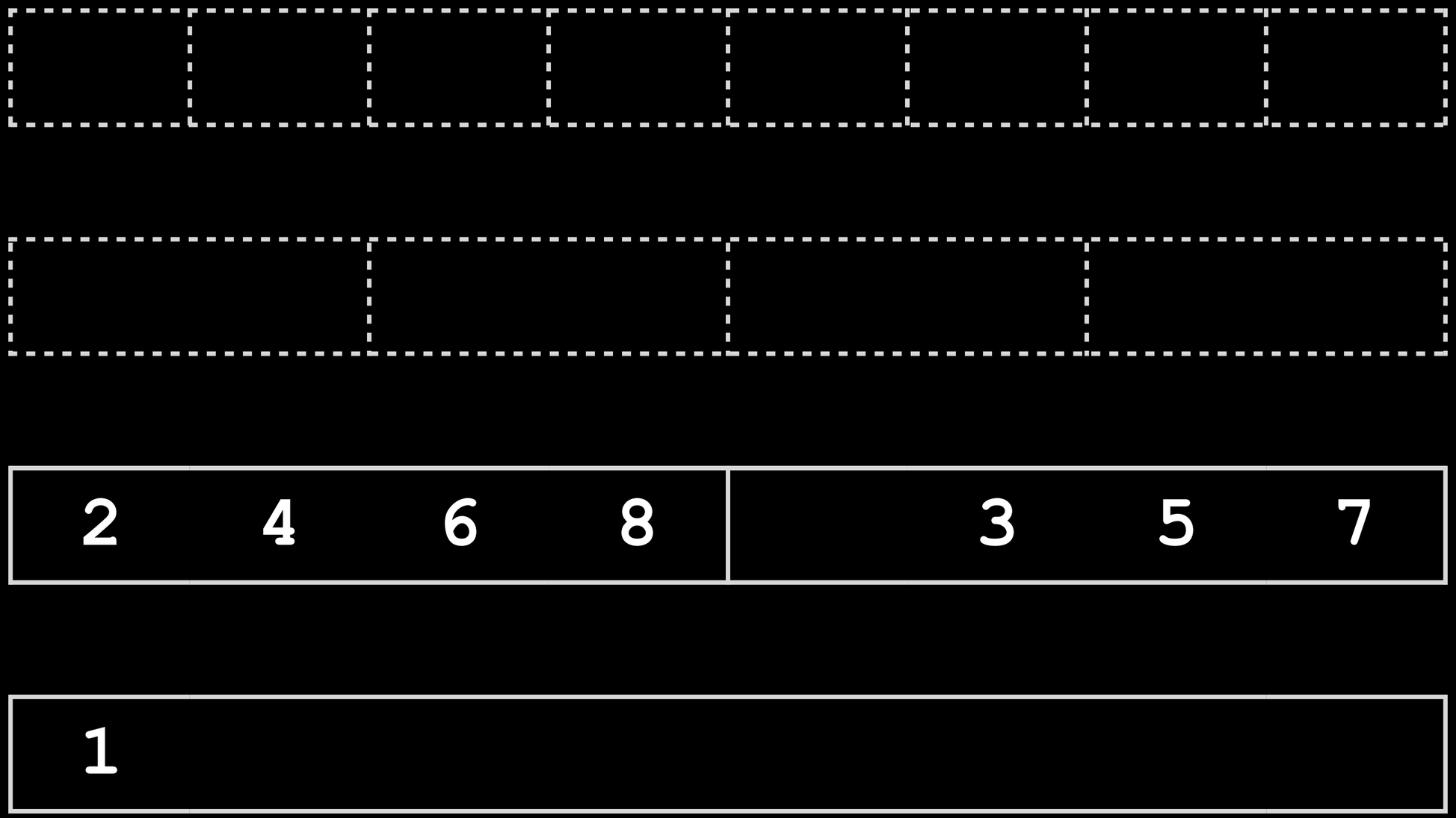

7
3
5
2
4
6
8
1

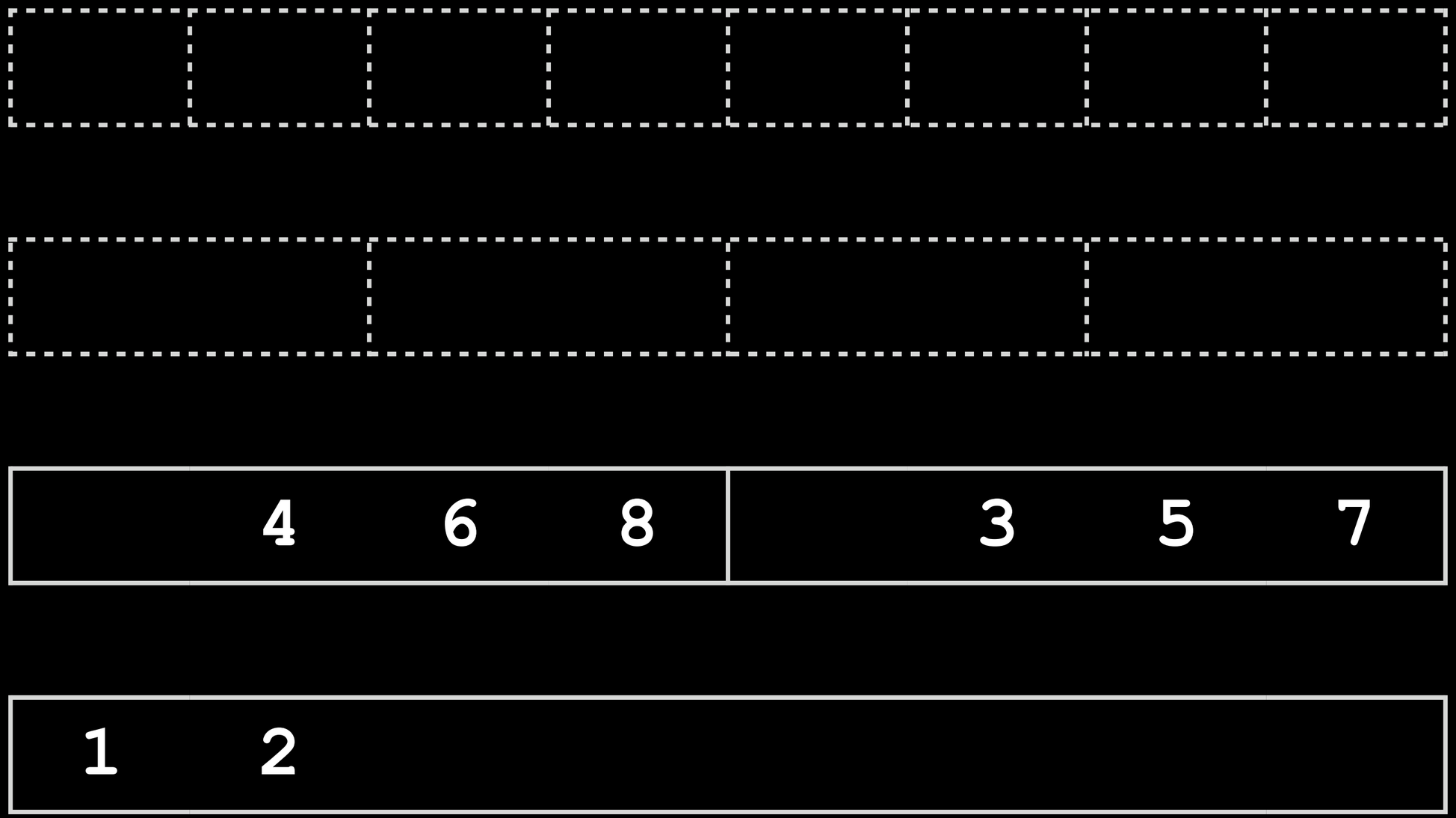

7
5
2
4
6
8
1
3

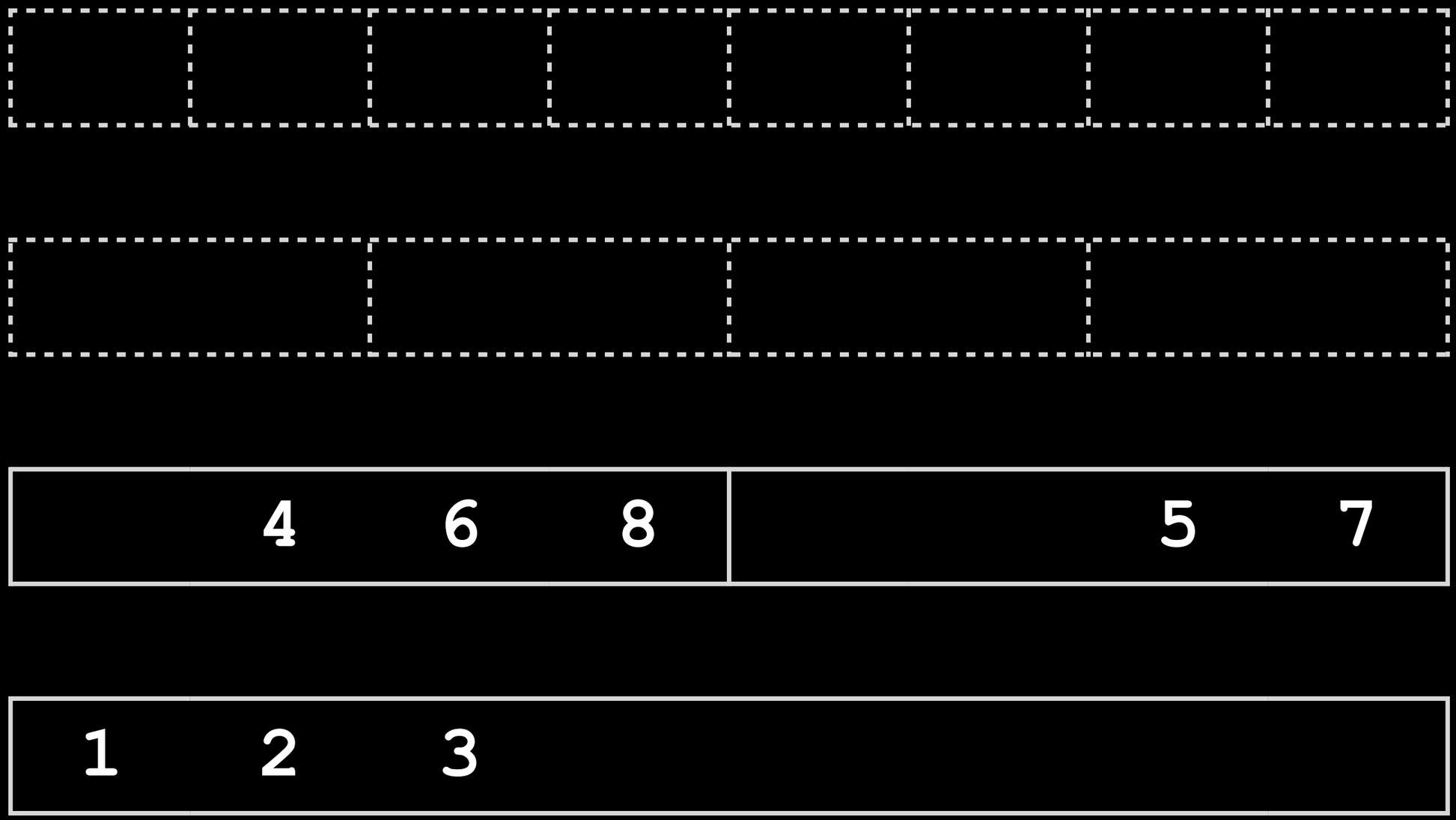

7
2
4
6
8
1
3
5

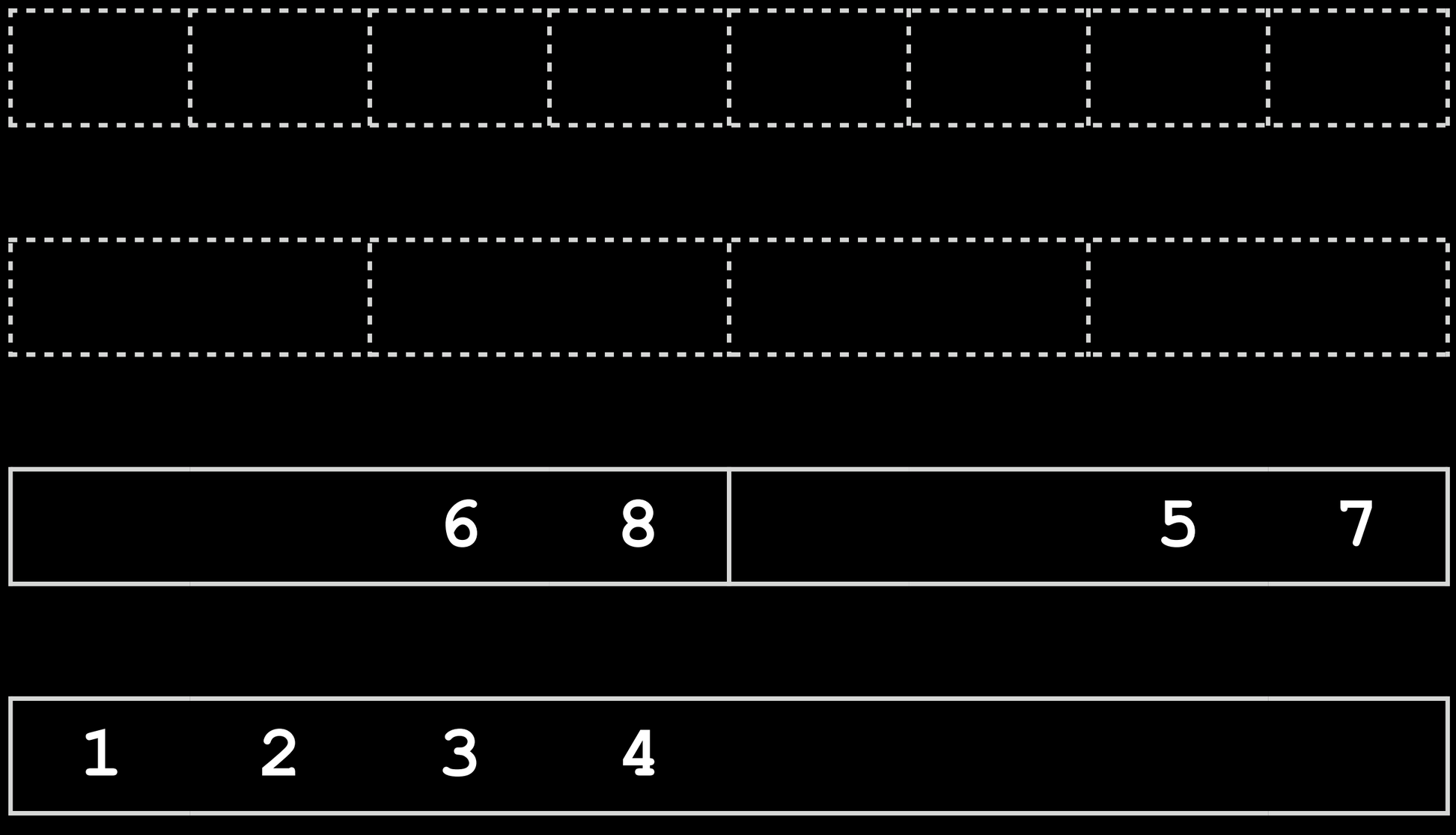

2 4 6 8 1 3 5 7

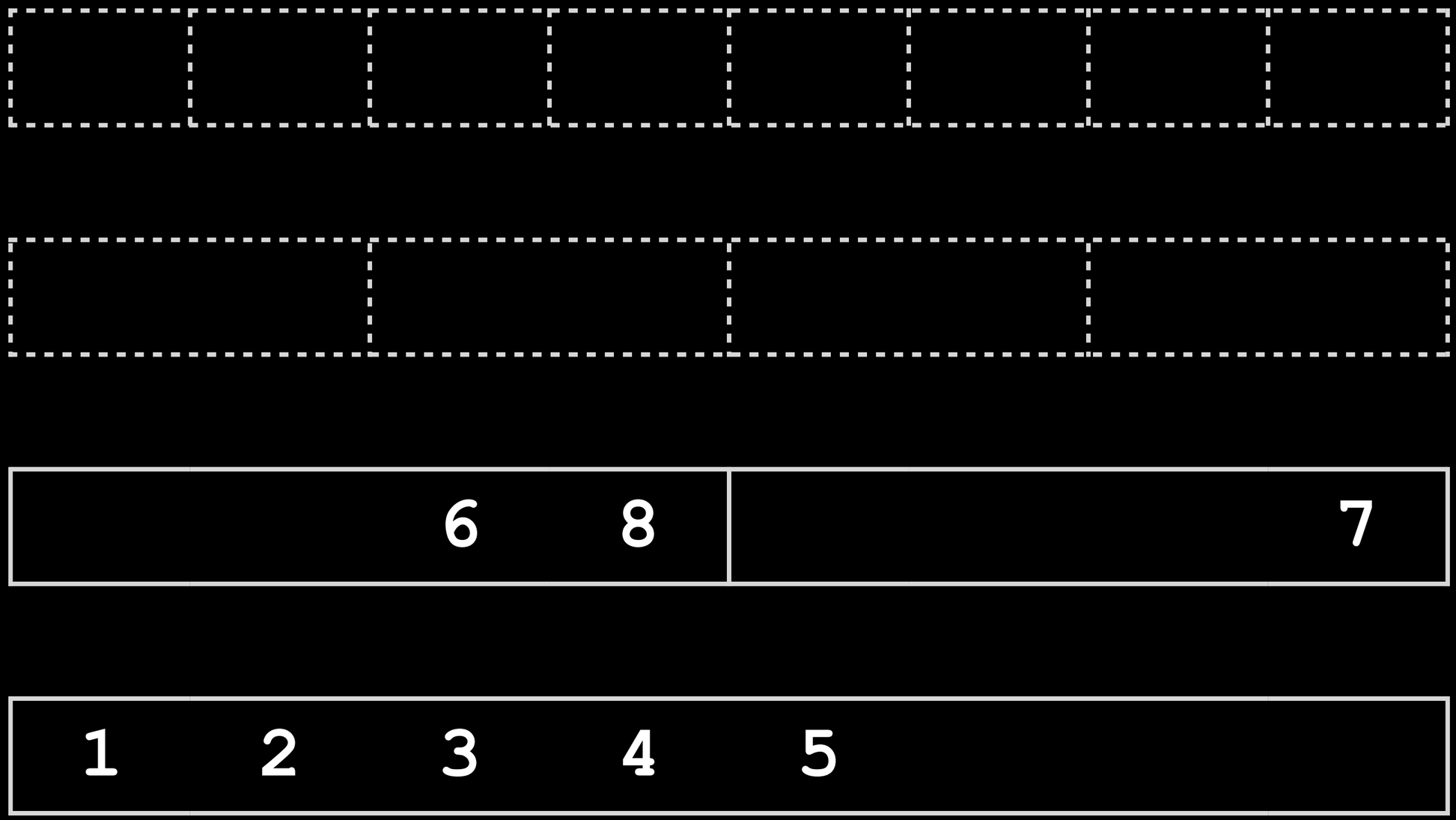

2 4 6 8
1 3 5 7

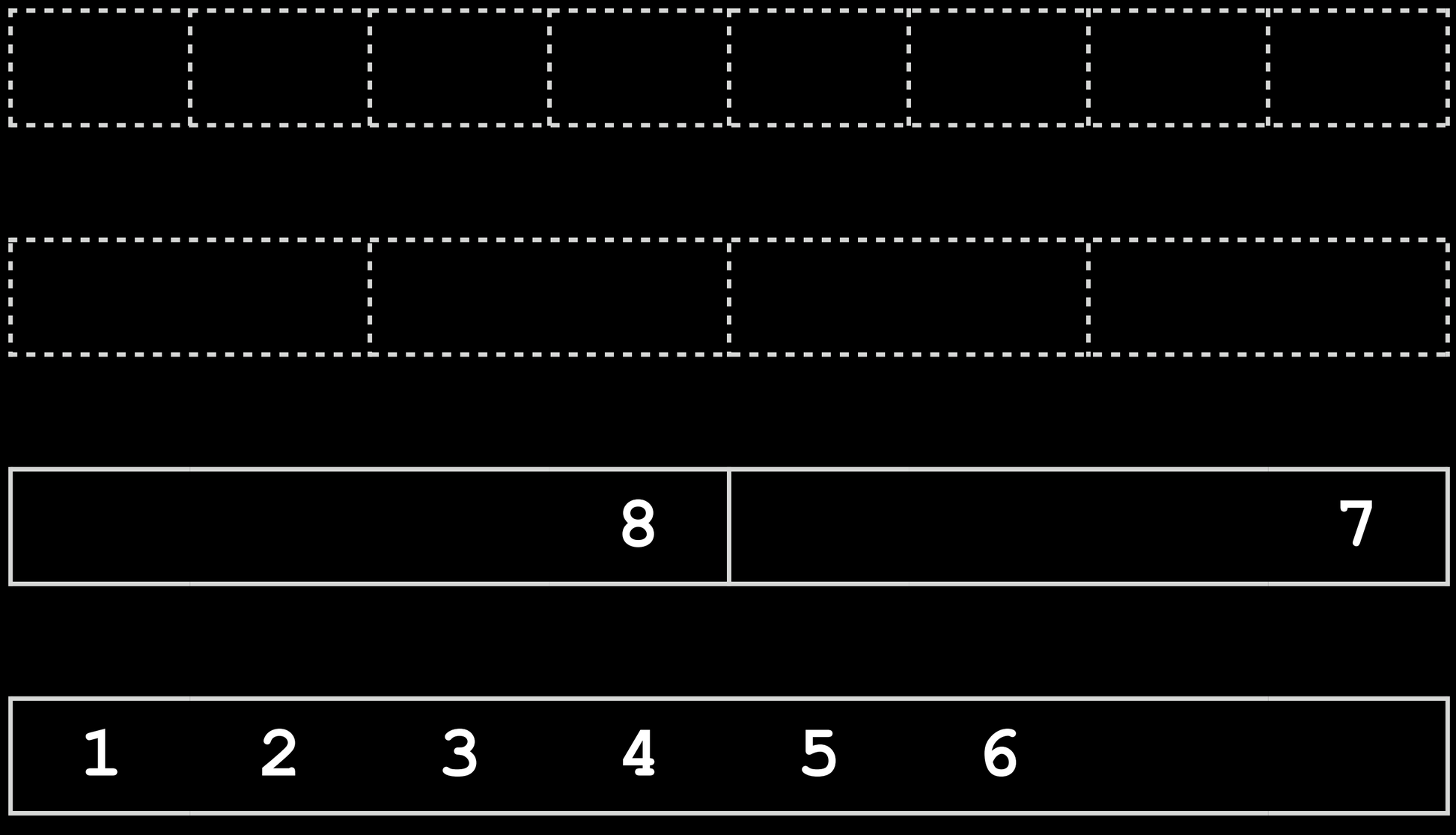

2 4 6 8
3 5 7
1

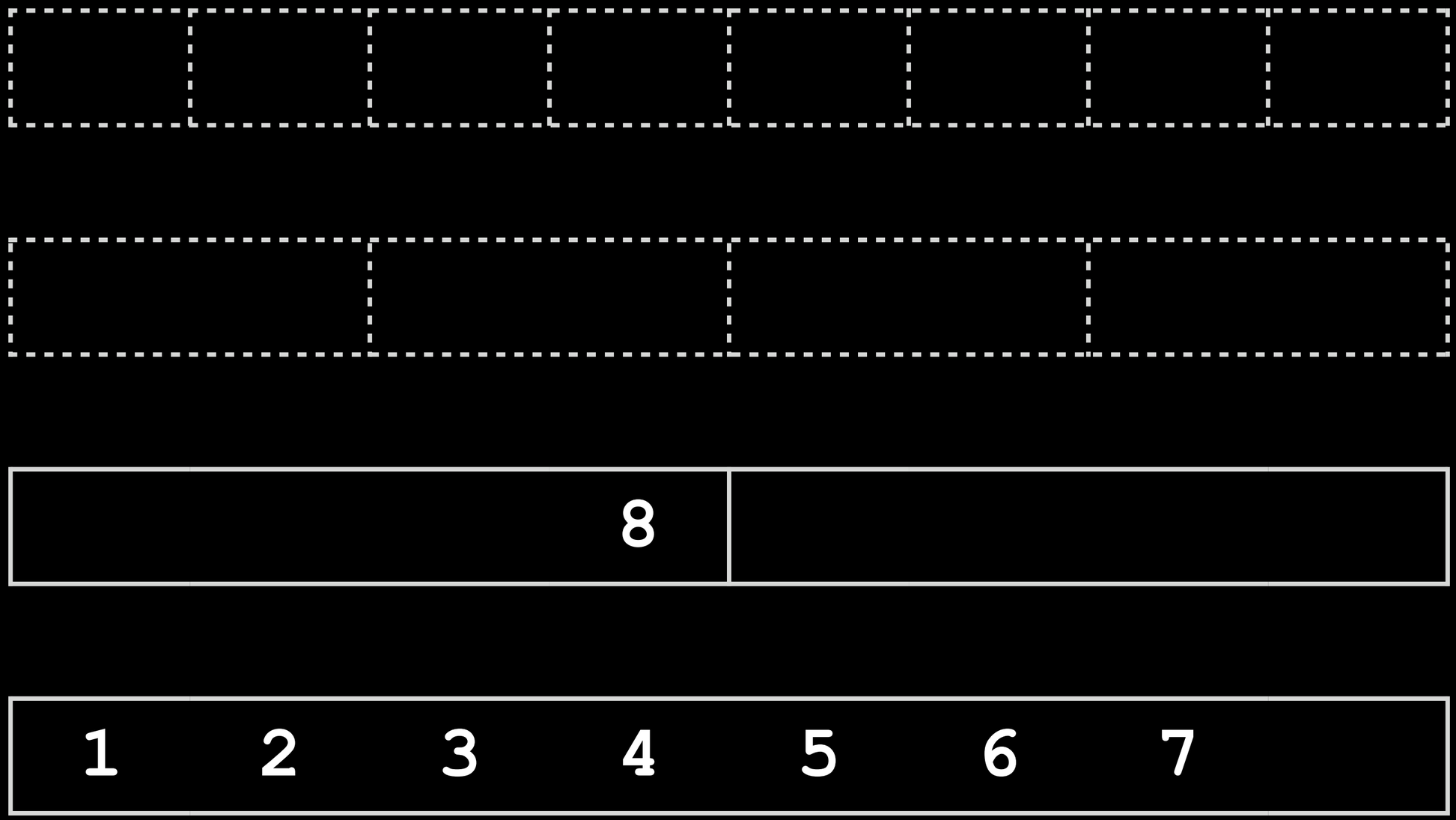

4 6 8
3 5 7
1 2

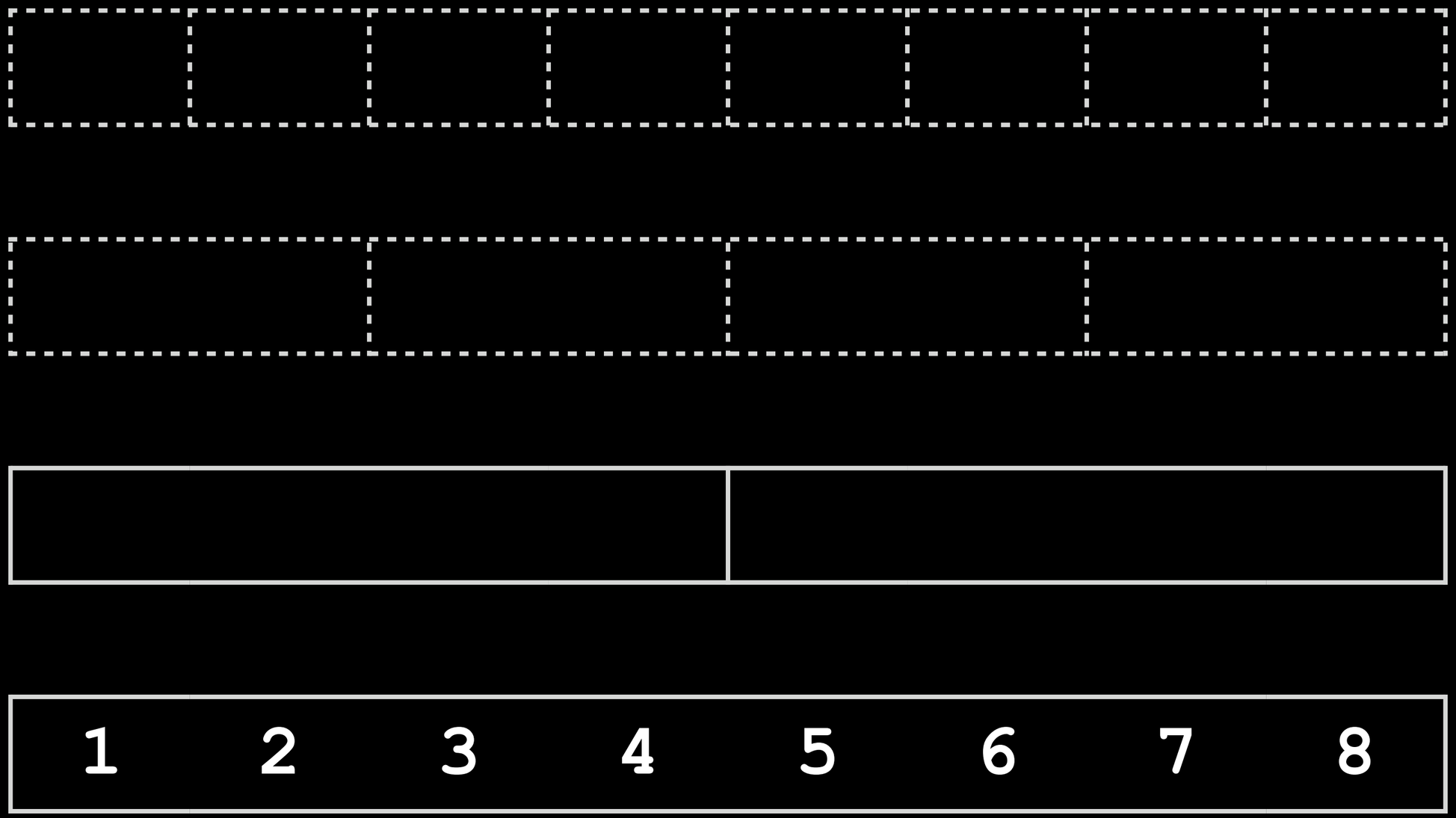

4 6 8
5 7
1 2 3

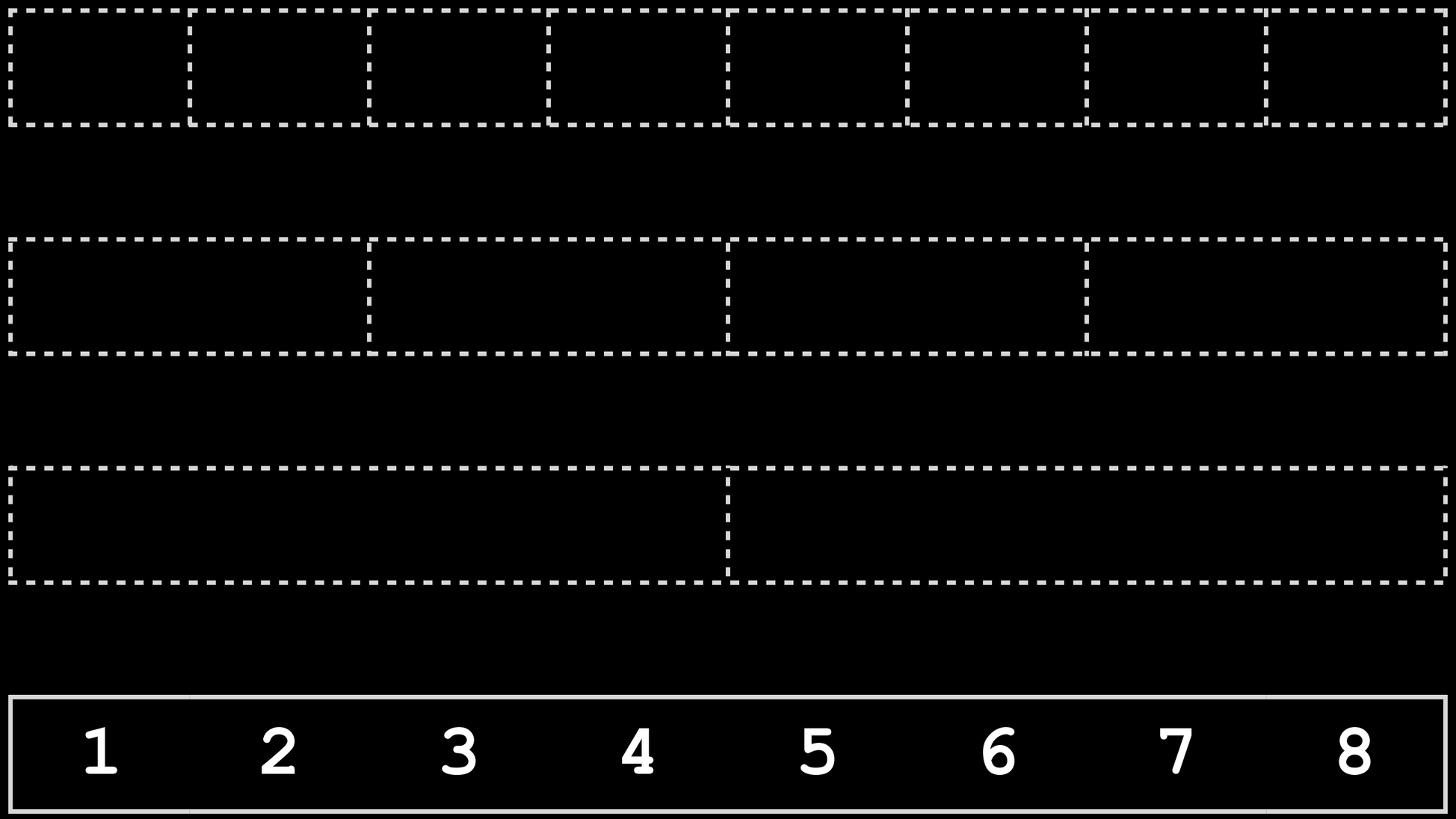

6 8
5 7
1 2 3 4


6 8
7
1 2 3 4 5


8
7
1
2
3
4
5
6


8
1 2 3 4 5 6 7


1 2 3 4 5 6 7 8

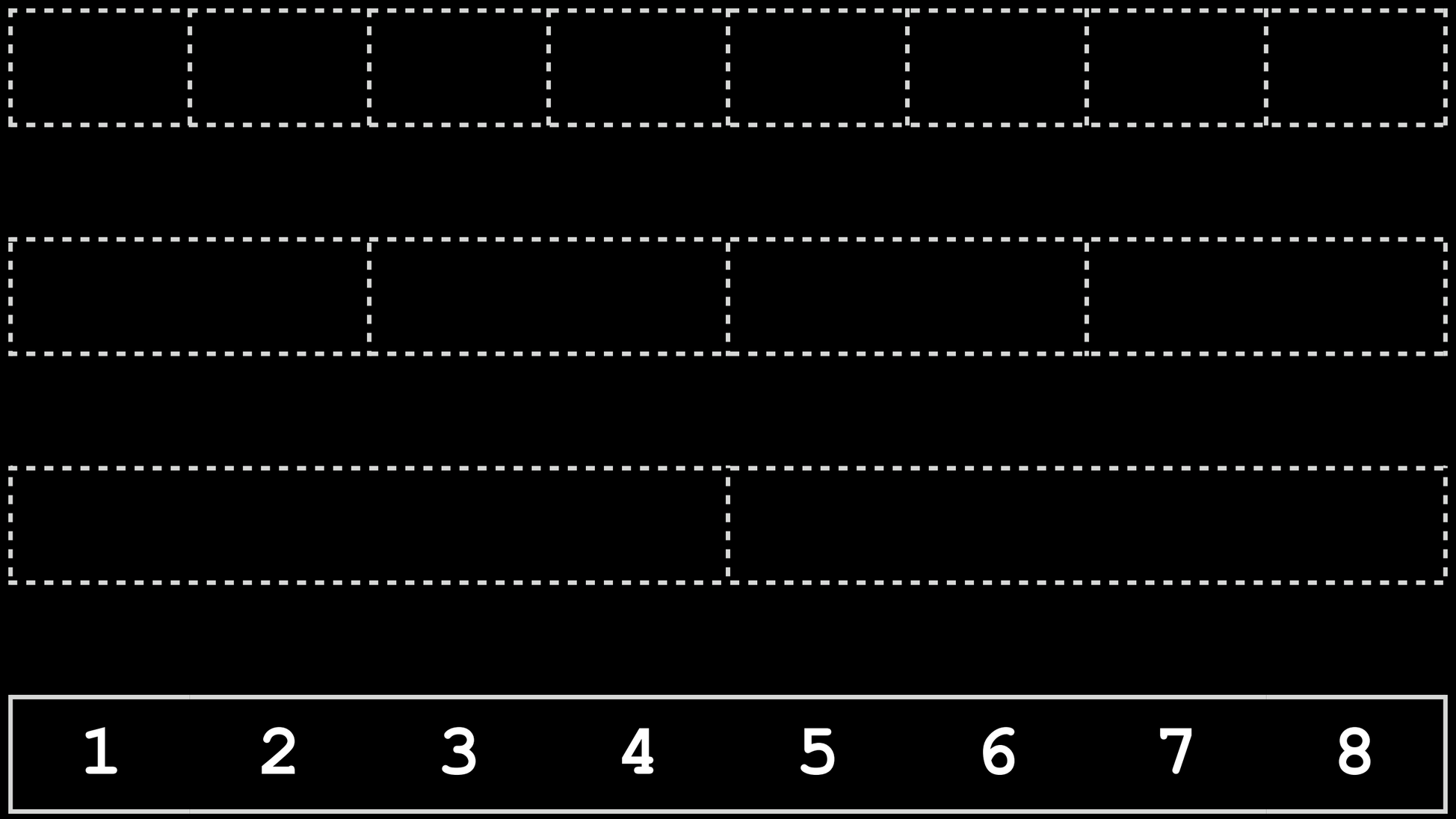

1 2 3 4 5 6 7 8

$\log n$

$$\log_2 n$$

$$\log_2 8$$

3


1 2 3 4 5 6 7 8

n

8

$n \log n$

$O(n \log n)$

```
On input of n elements
    if n < 2
        return
    else
        sort left half of elements
        sort right half of elements
        merge sorted halves
```

```
On input of n elements
    if n < 2
        return
    else
        sort left half of elements
        sort right half of elements
        merge sorted halves
```

$$T(n) = O(1)$$

if $n < 2$

```
On input of n elements
    if n < 2
        return
    else
        sort left half of elements
        sort right half of elements
        merge sorted halves
```

```
On input of n elements
    if n < 2
        return
    else
        sort left half of elements
        sort right half of elements
        merge sorted halves
```

```
On input of n elements
     if n < 2
          return
     else
          sort left half of elements
          sort right half of elements
          merge sorted halves
```

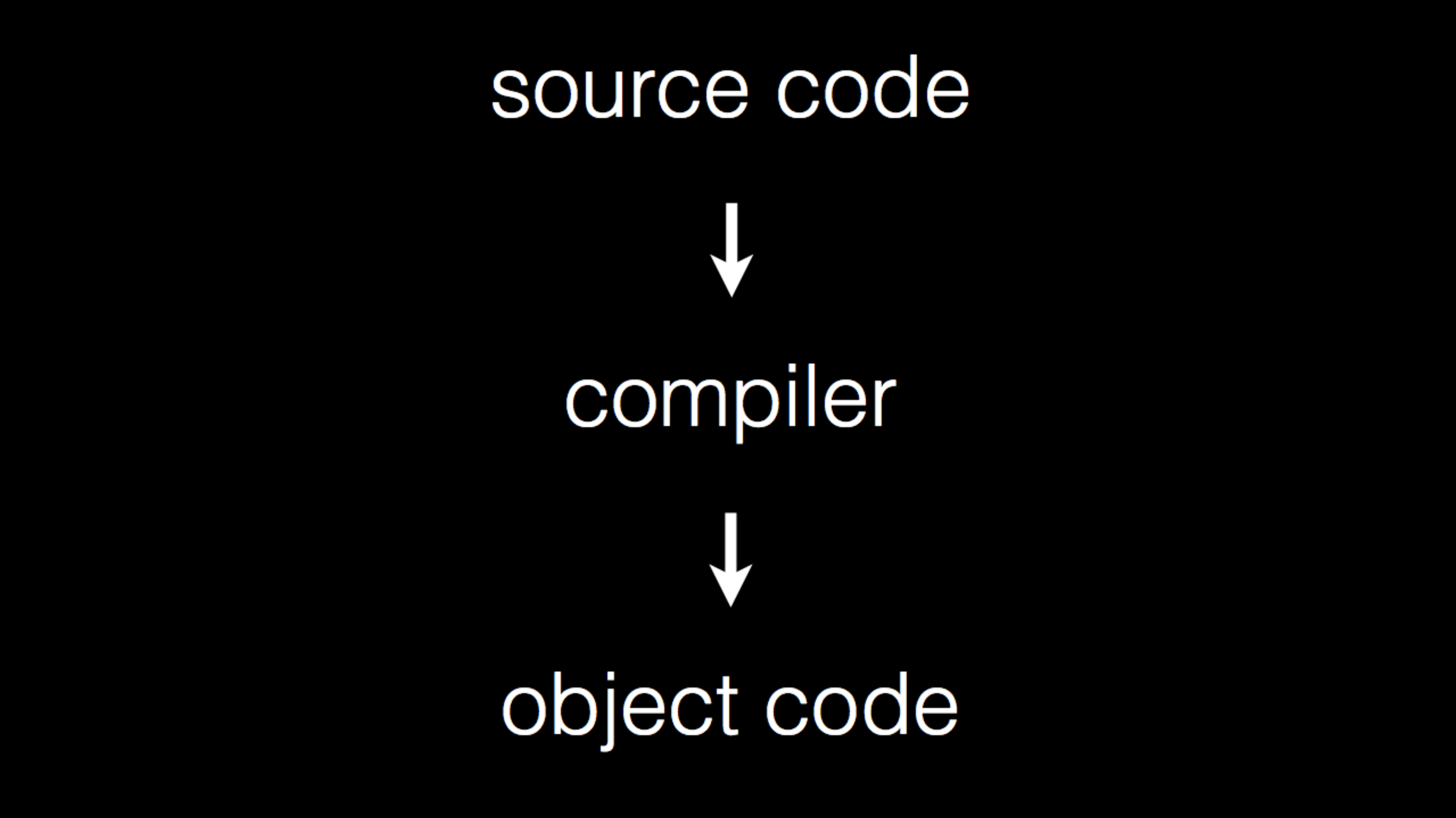
$$T(n) = T(n/2) + T(n/2) + O(n)$$

if $n \geq 2$

$O(n \log n)$

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```


source code
compiler
object code

```
01111111 01000101 01001100 01000110 00000010 00000001 00000001 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001 00000000 00000000 00000000
01000000 00000100 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11101000 00010001 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000 00000000 00111000 00000000
00001001 00000000 01000000 00000000 00011110 00000000 00011011 00000000
00000110 00000000 00000000 00000000 00000101 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000011 00000000 00000000 00000000 00000100 00000000 00000000 00000000
00111000 00000010 00000000 00000000 00000000 00000000 00000000 00000000
00111000 00000010 01000000 00000000 00000000 00000000 00000000 00000000
00111000 00000010 01000000 00000000 00000000 00000000 00000000 00000000
00011100 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00011100 00000000 00000000 00000000 00000000 00000000 00000000 00000000
...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq   %rbp
.Ltmp2:
    .cfi_def_cfa_offset 16
.Ltmp3:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp4:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    leaq    .L.str, %rdi
    movb    $0, %al
    callq   printf
    movl    $0, %ecx
    movl    %eax, -4(%rbp)          # 4-byte Spill
    movl    %ecx, %eax
    addq    $16, %rsp
    popq    %rbp
    ret
.Ltmp5:
    .size   main, .Ltmp5-main
    .cfi_endproc

    .type   .L.str,@object          # @.str
    .section    .rodata.str1.1,"aMS",@progbits,1
.L.str:
    .asciz  "hello, world\n"
    .size   .L.str, 14
...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq   %rbp
.Ltmp2:
    .cfi_def_cfa_offset 16
.Ltmp3:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp4:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    leaq    .L.str, %rdi
    movb    $0, %al
    callq   printf
    movl    $0, %ecx
    movl    %eax, -4(%rbp)          # 4-byte Spill
    movl    %ecx, %eax
    addq    $16, %rsp
    popq    %rbp
    ret
.Ltmp5:
    .size   main, .Ltmp5-main
    .cfi_endproc

    .type   .L.str,@object          # @.str
    .section    .rodata.str1.1,"aMS",@progbits,1
.L.str:
    .asciz  "hello, world\n"
    .size   .L.str, 14
...
```

```
01111111 01000101 01001100 01000110 00000010 00000001 00000001 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001 00000000 00000000 00000000
01000000 00000100 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11101000 00010001 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000 00000000 00111000 00000000
00001001 00000000 01000000 00000000 00011110 00000000 00011011 00000000
00000110 00000000 00000000 00000000 00000101 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000011 00000000 00000000 00000000 00000100 00000000 00000000 00000000
00111000 00000010 00000000 00000000 00000000 00000000 00000000 00000000
00111000 00000010 01000000 00000000 00000000 00000000 00000000 00000000
00111000 00000010 01000000 00000000 00000000 00000000 00000000 00000000
00011100 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00011100 00000000 00000000 00000000 00000000 00000000 00000000 00000000
...
```

source code

assembly code

object code

source code

compile

assembly code

assemble

object code

hello.c

compile

hello.s

assemble

hello.o

hello.c

compile

hello.s

assemble

hello.o

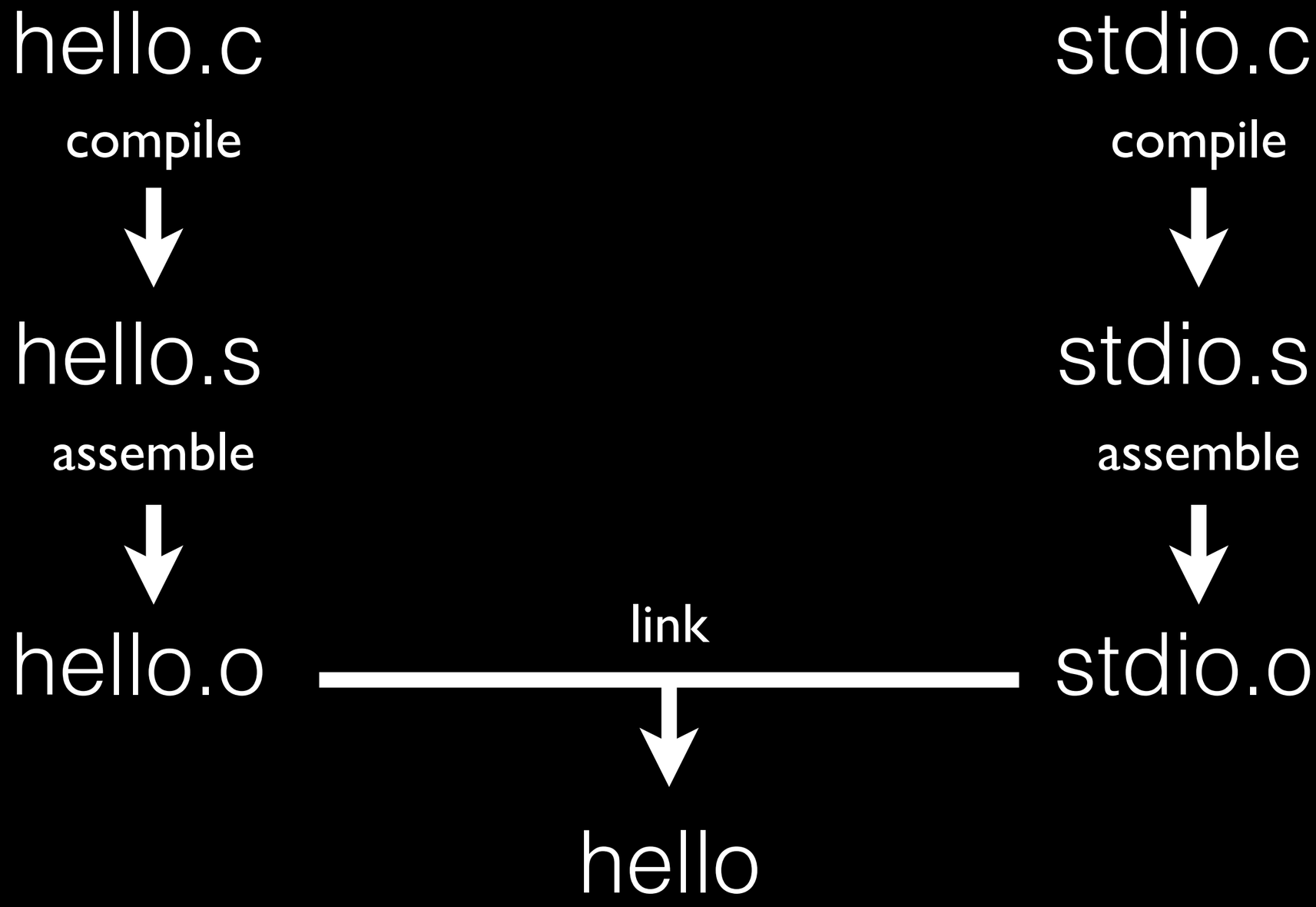
stdio.c

compile

stdio.s

assemble

stdio.o


hello.c
compile
hello.s
assemble
hello.o
stdio.c
compile
stdio.s
assemble
stdio.o
link
hello

bitwise operators

`&` `|` `^` `~` `<<` `>>`

9/9

0800 Arctan started
1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 795 correct
13" wc (032) MP - MC ~~1.9826[illegible]000~~ ~~2.13047641[illegible]~~) 4.615925059(-2)
(033) PRO 2 2.13047641[illegible]
correct 2.13067641[illegible]
Relays 6-2 in 033 failed special speed test
in relay " 10.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.

1545 
Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

1700 1630 Arctangent started.
closed down.

Relay 2145
Relay 3370

gdb

This is CS50.