This is CS50

data structures

abstract data types

queues

FIFO

enqueue

dequeue

```c
const int CAPACITY = 50;

typedef struct
{
    person people[CAPACITY];
    int size;
} queue;
```

stacks

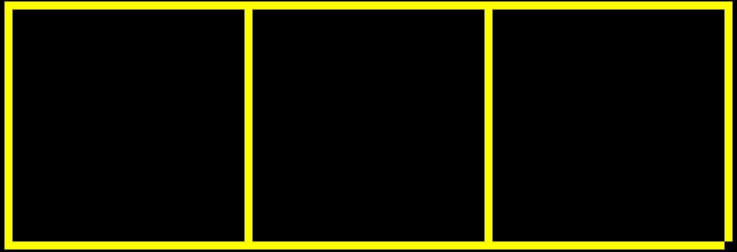
LIFO

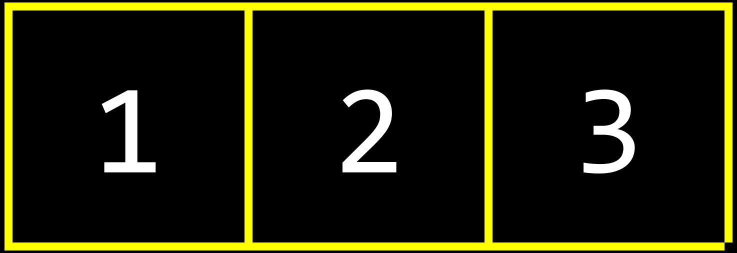
push

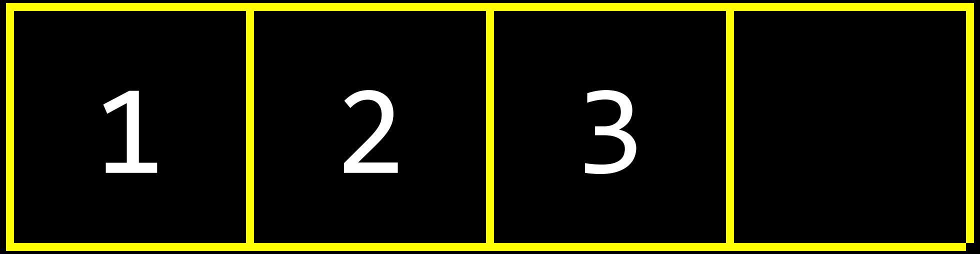
pop

```
const int CAPACITY = 50;

typedef struct
{
    person people[CAPACITY];
    int size;
} stack;
```

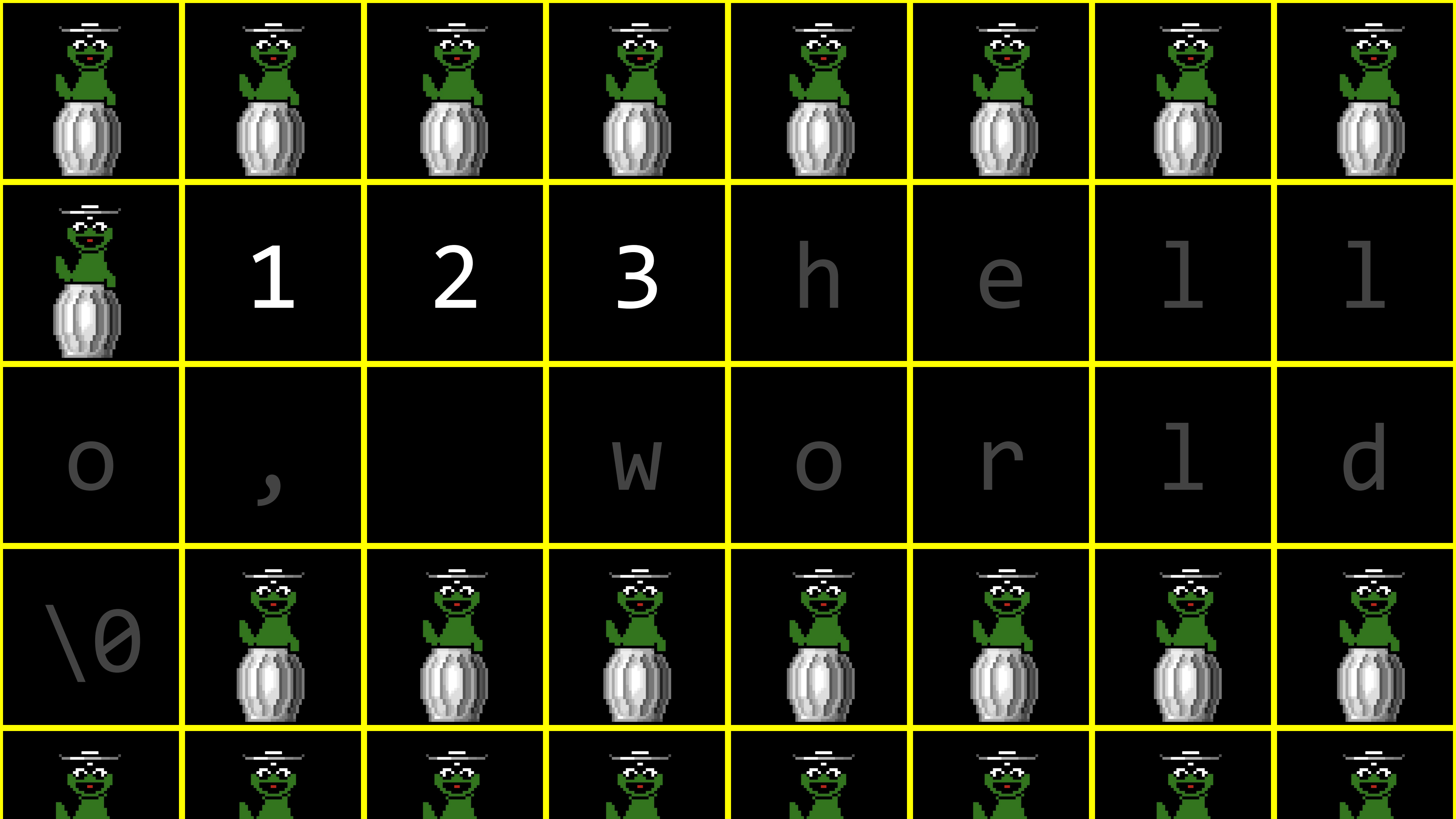
arrays



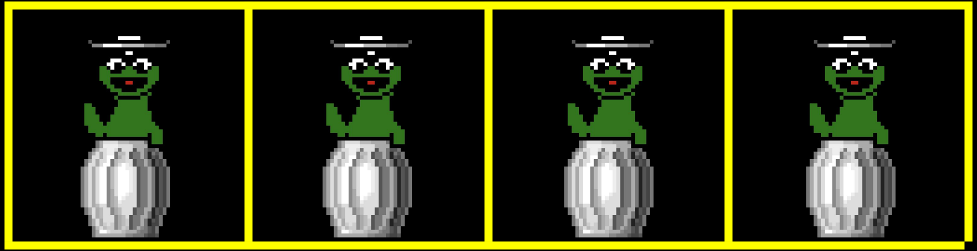

1
2
3

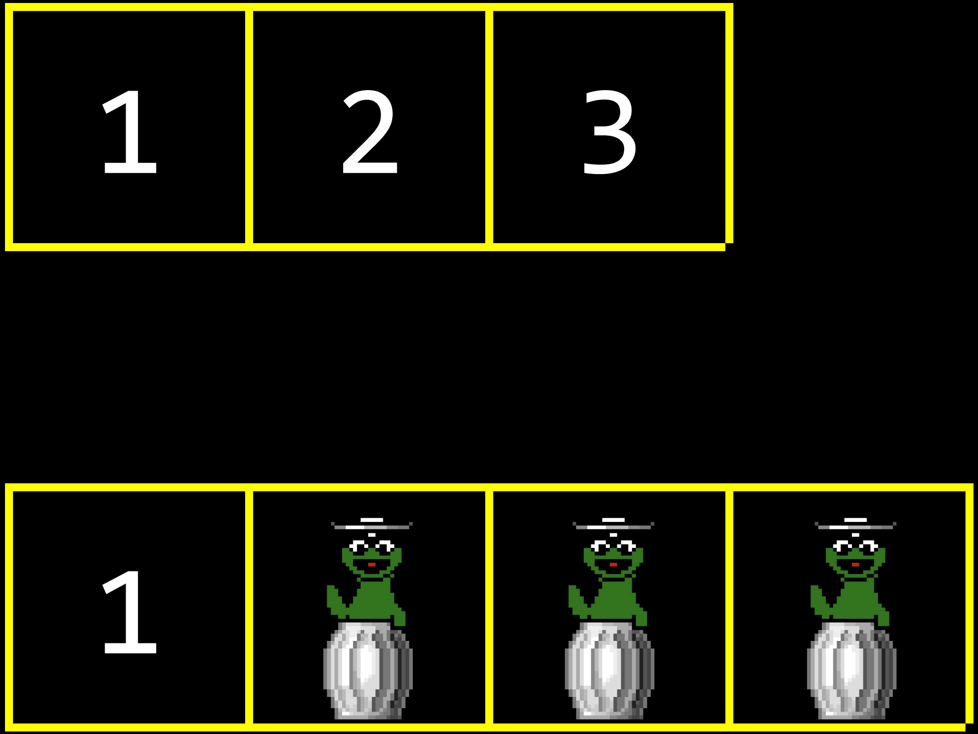

1
2
3

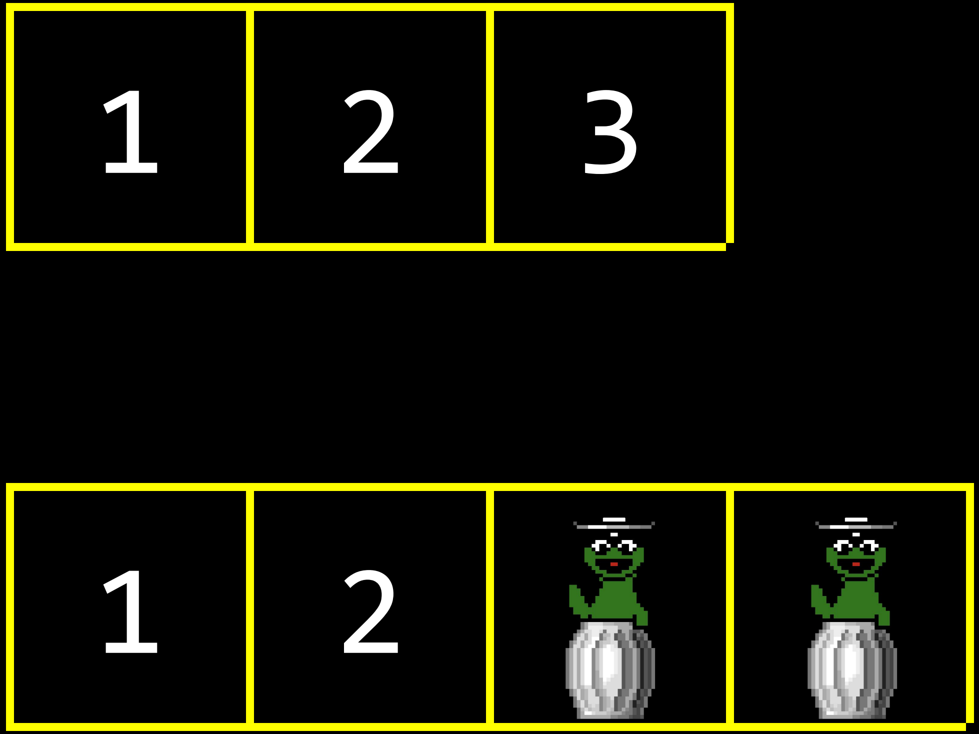
	1	2	3				


1
2
3
h
e
l
l
o
,
w
o
r
l
d
\0

1	2	3

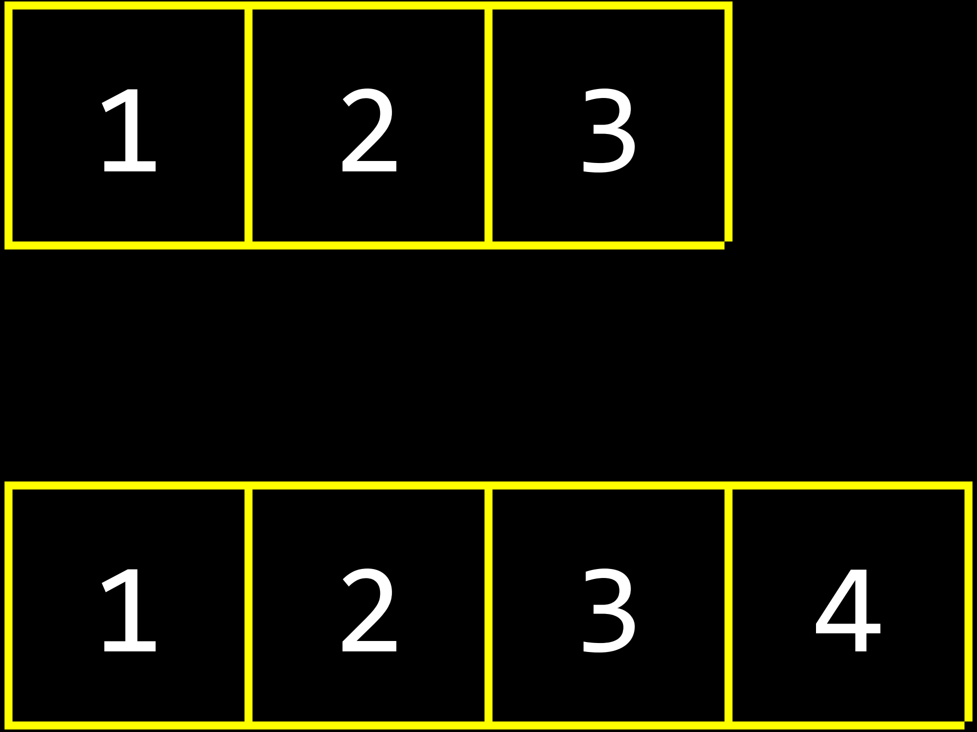


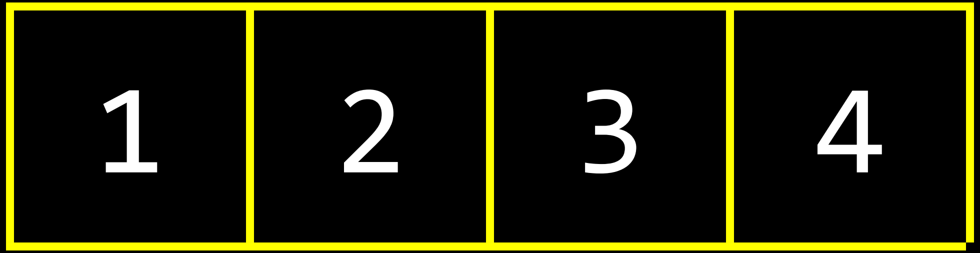

1
2
3
1


1
2
3
1
2

1	2	3

1	2	3	


1 2 3
1 2 3 4

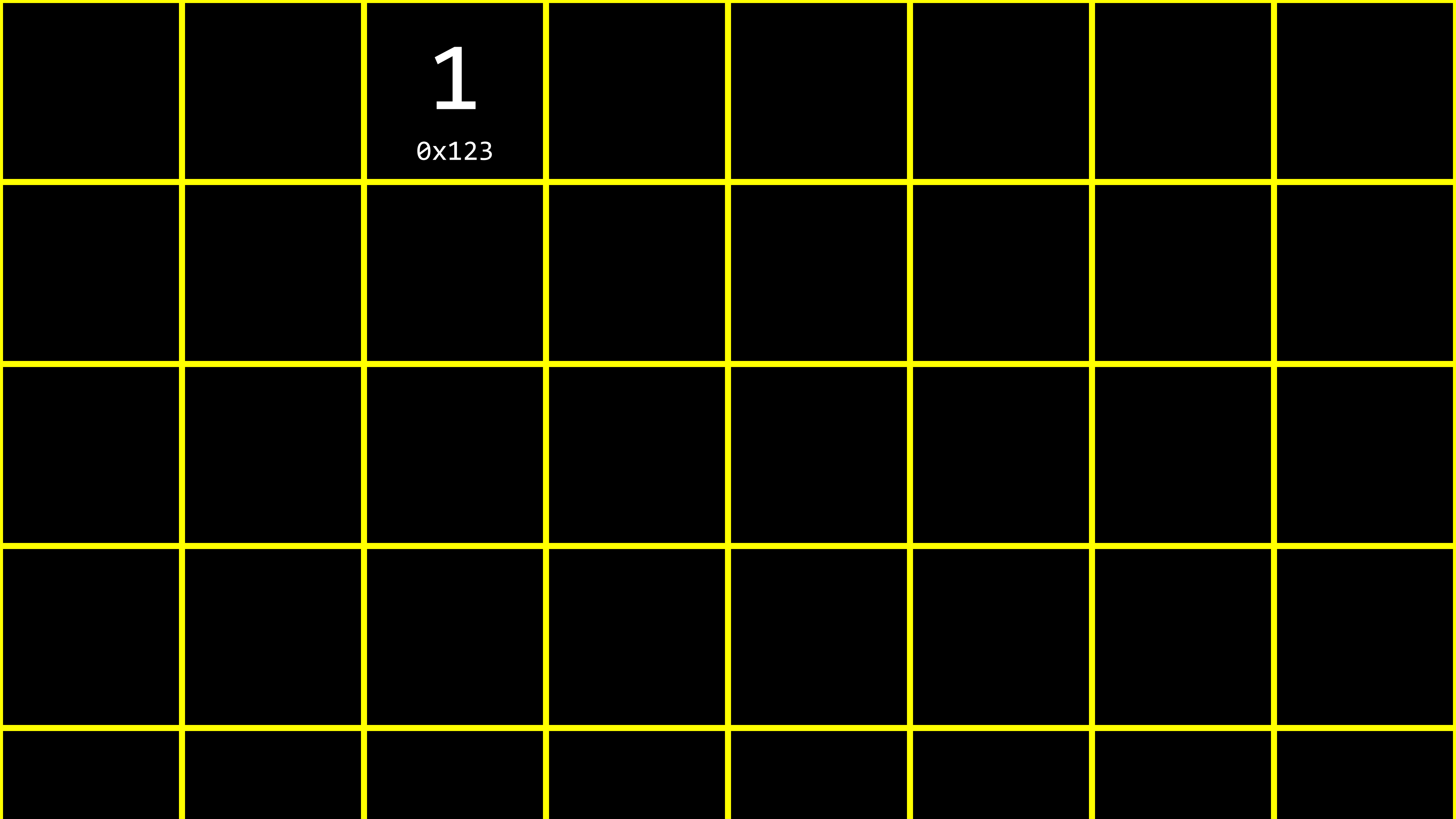

1
2
3
4

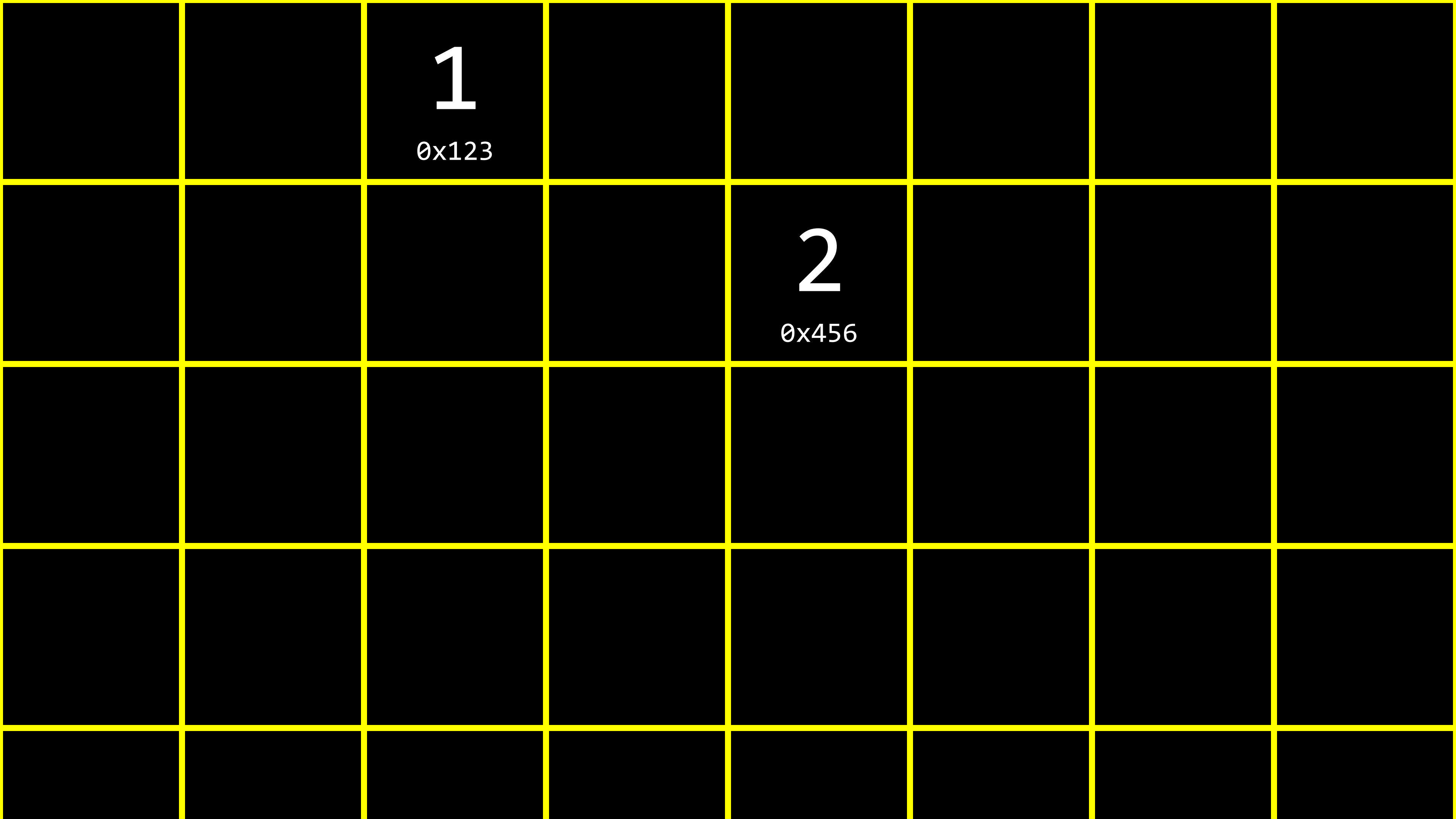
data structures

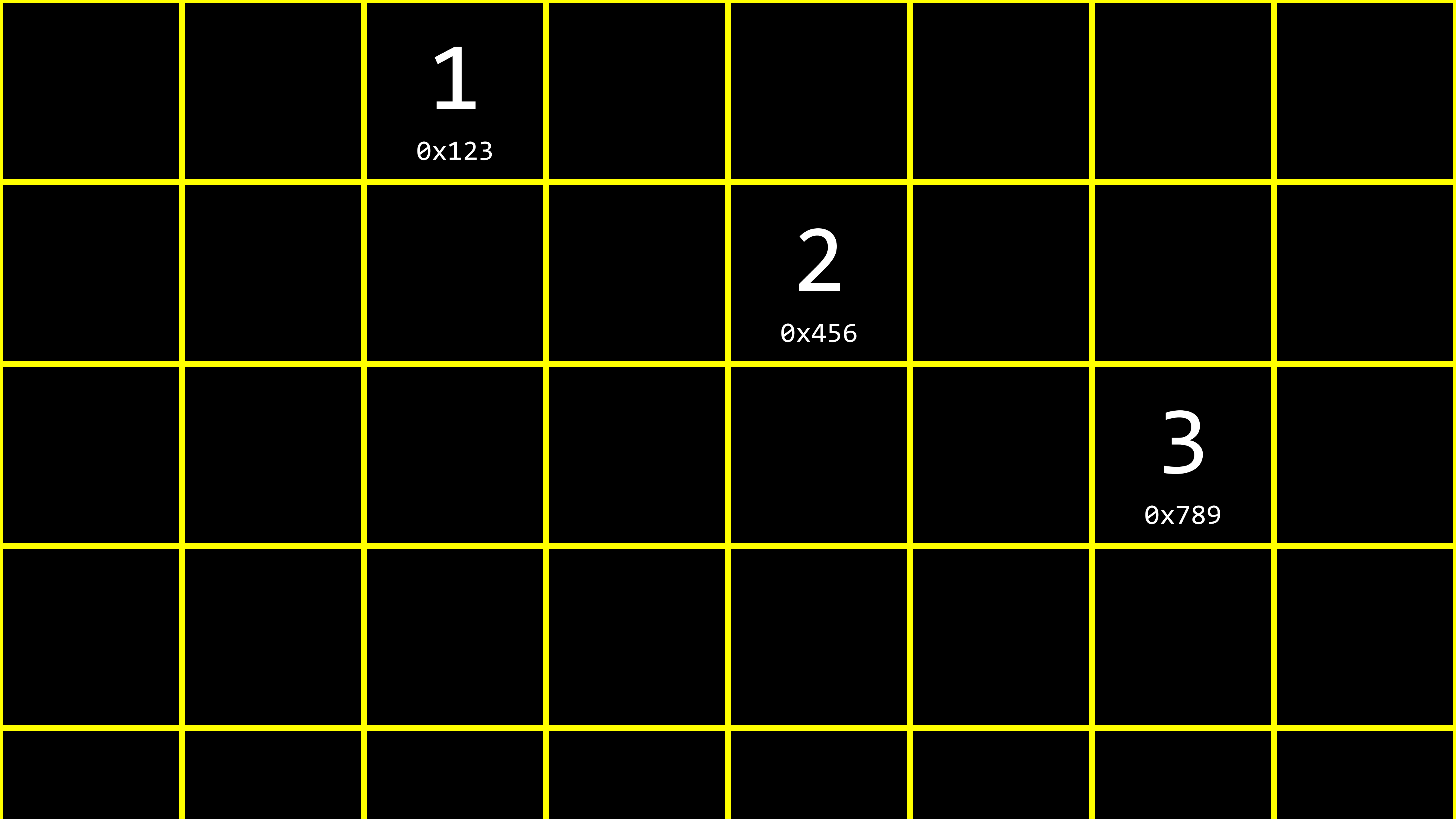
```
struct

.

*
```

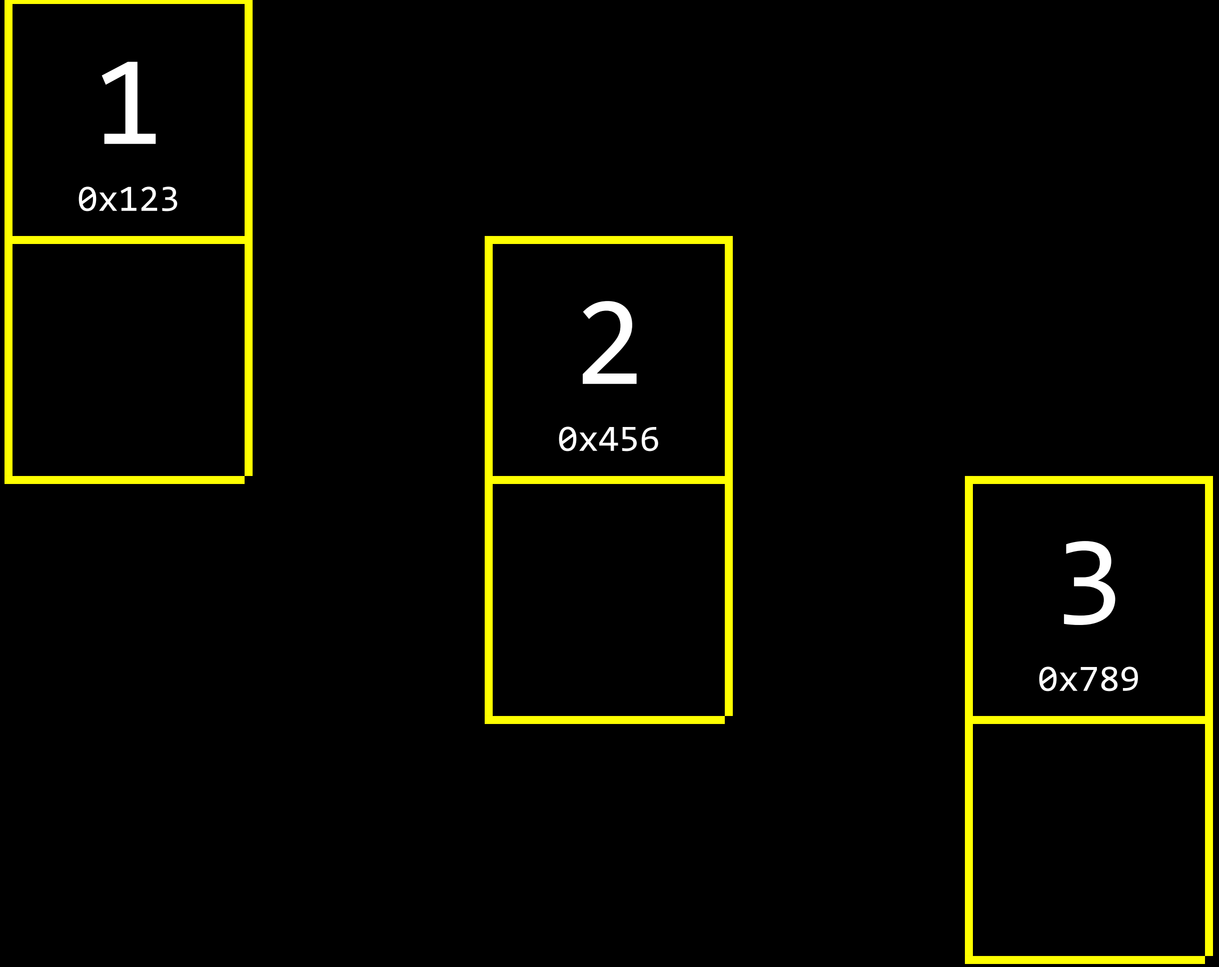
```
struct

->
```

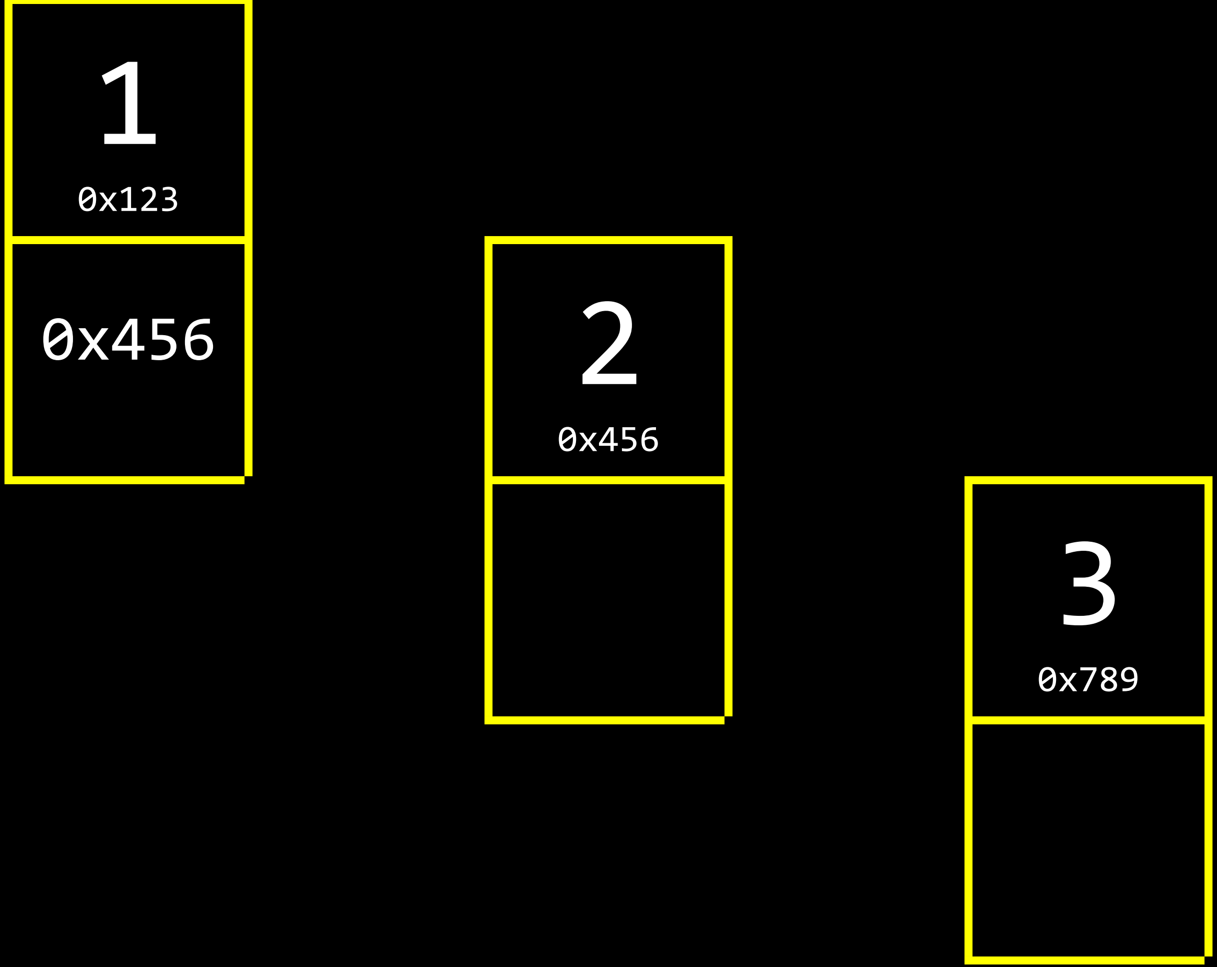
linked lists

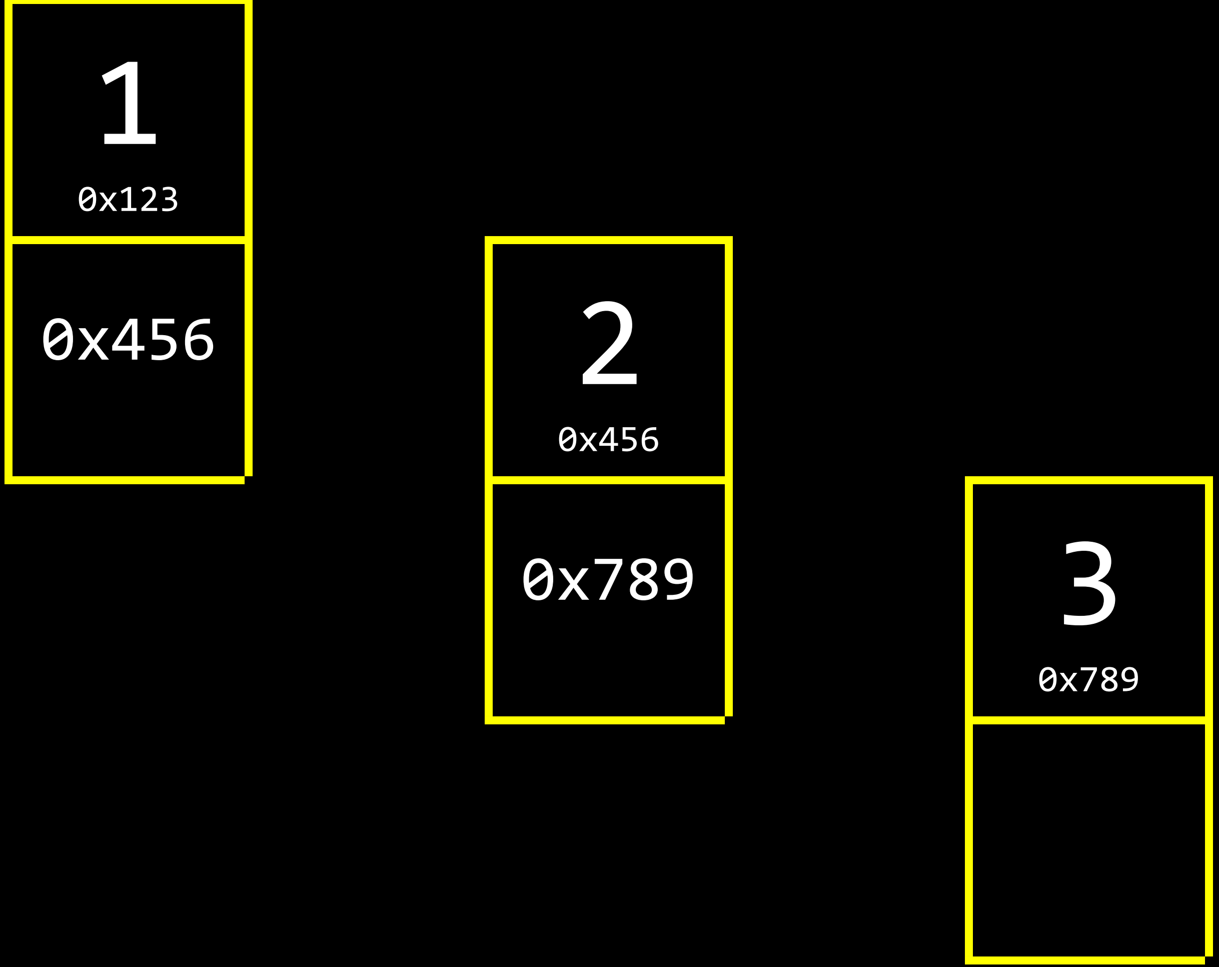

1
0x123

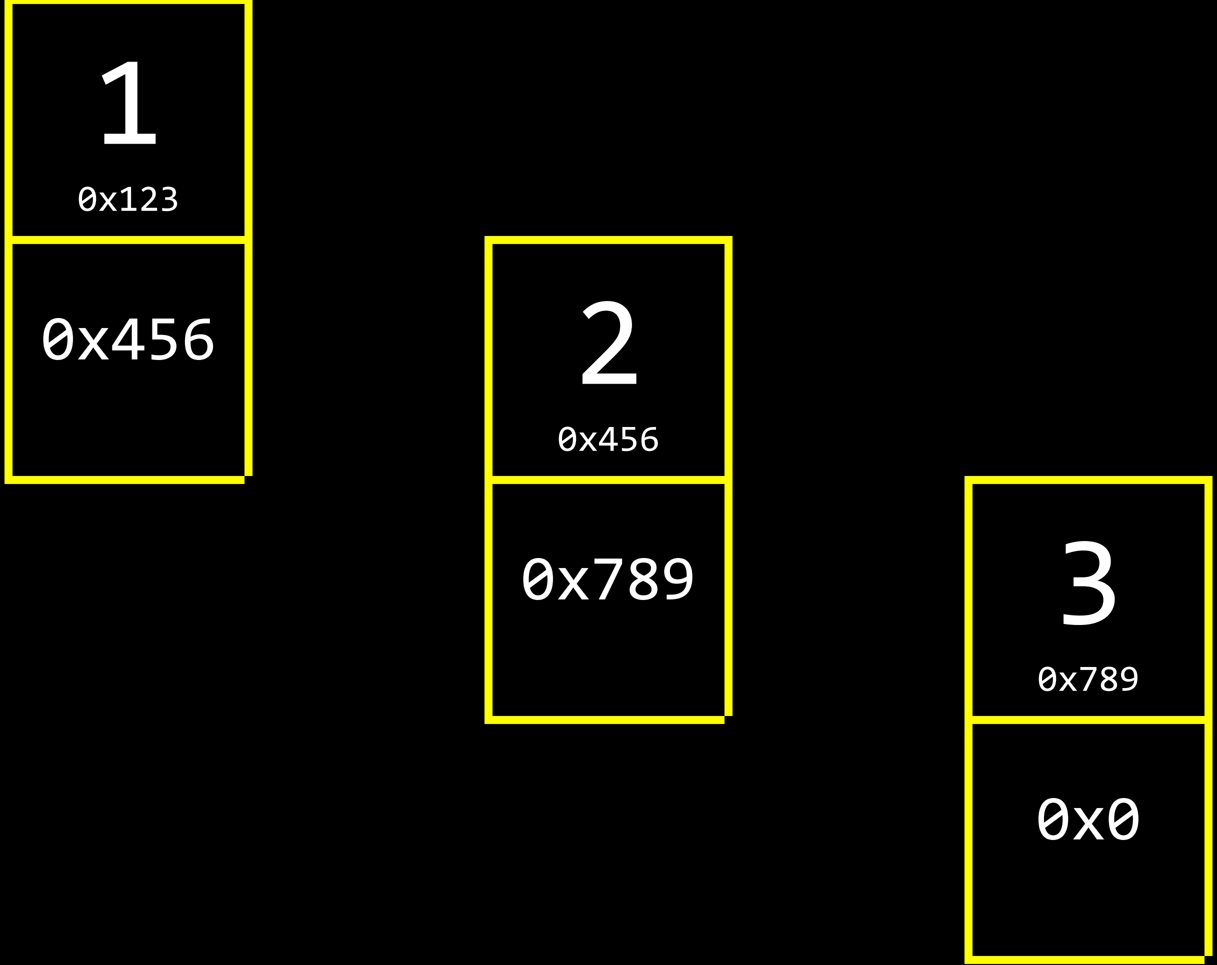

1
0x123
2
0x456

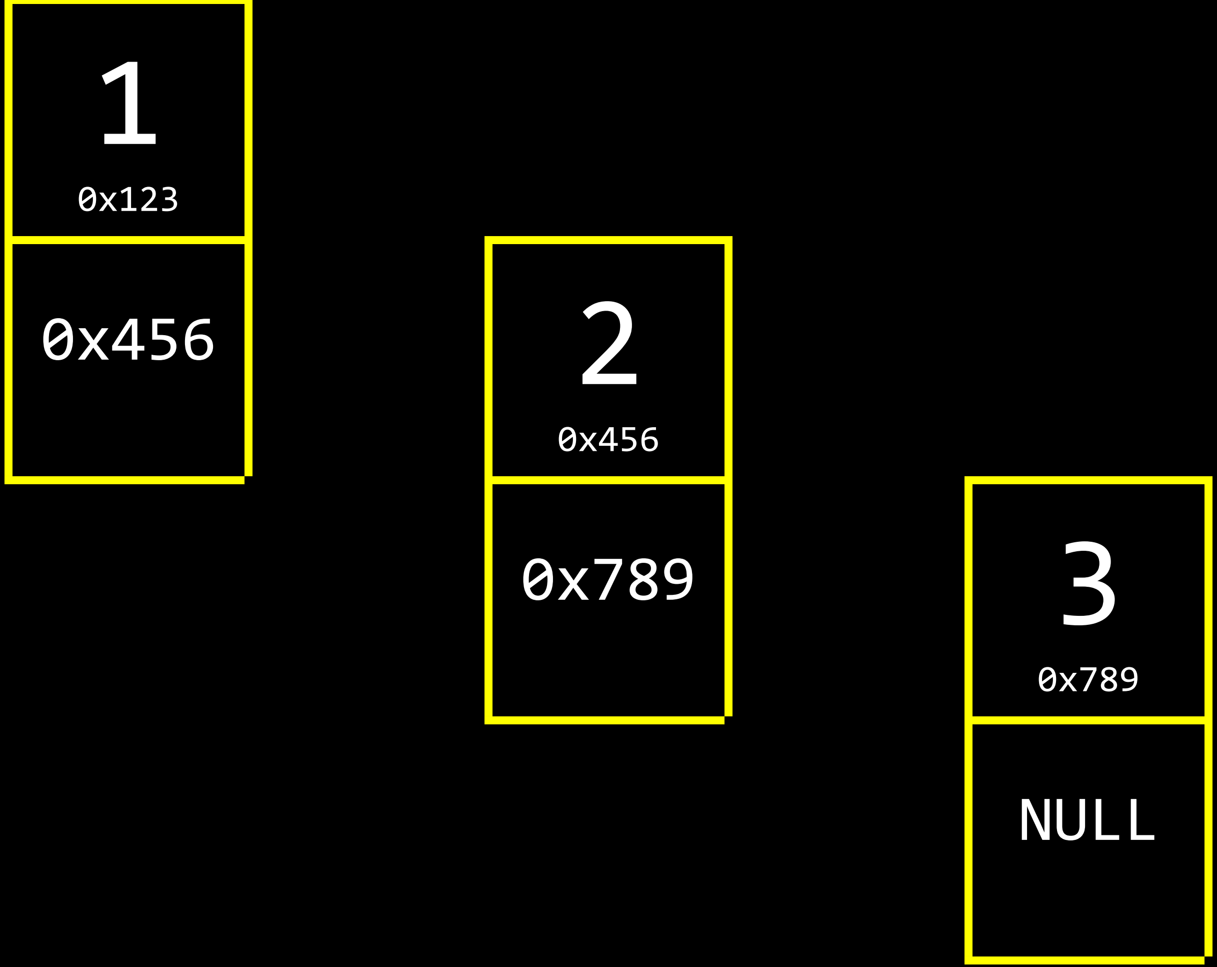

1
0x123
2
0x456
3
0x789

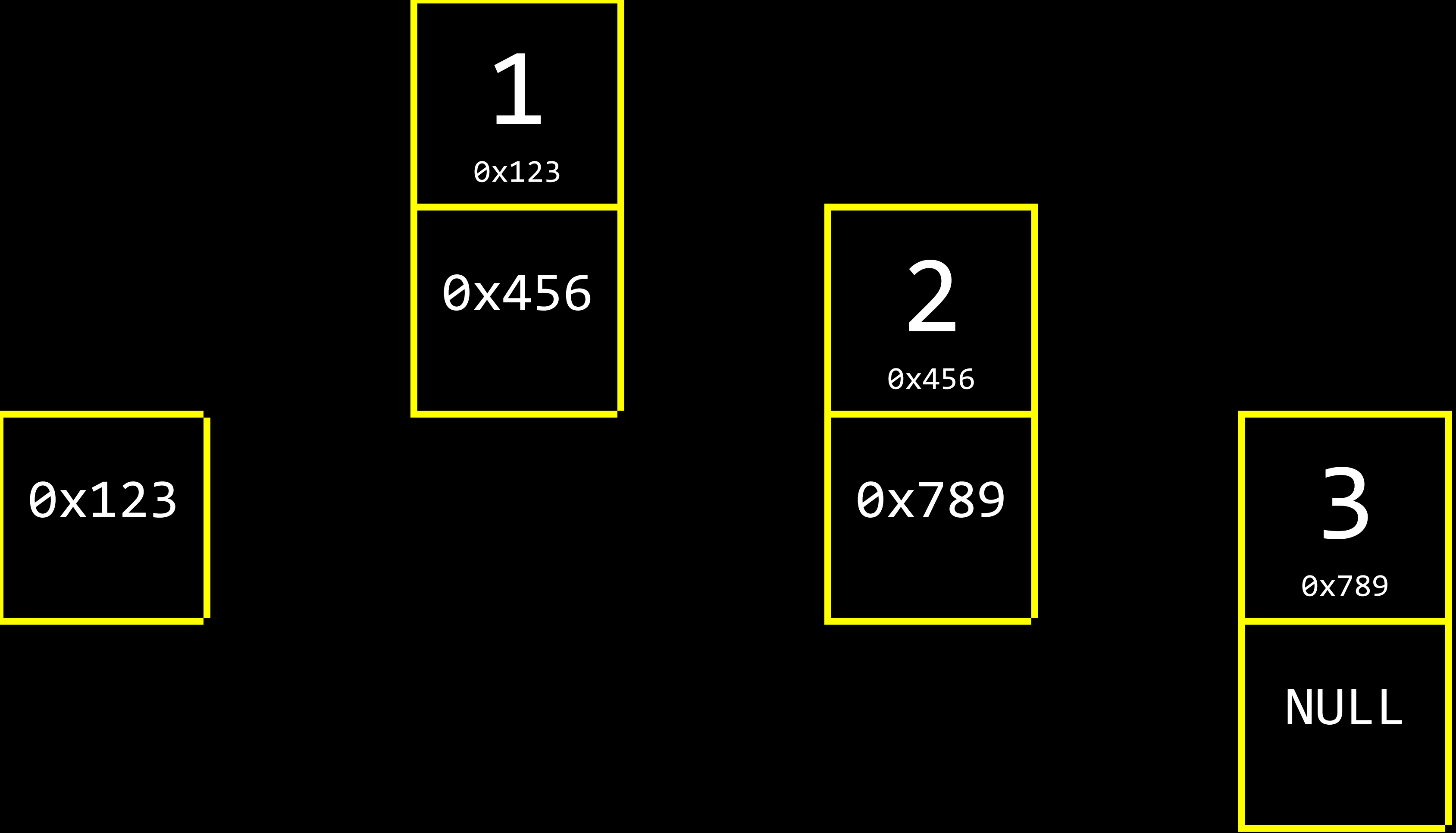

1
0x123
2
0x456
3
0x789

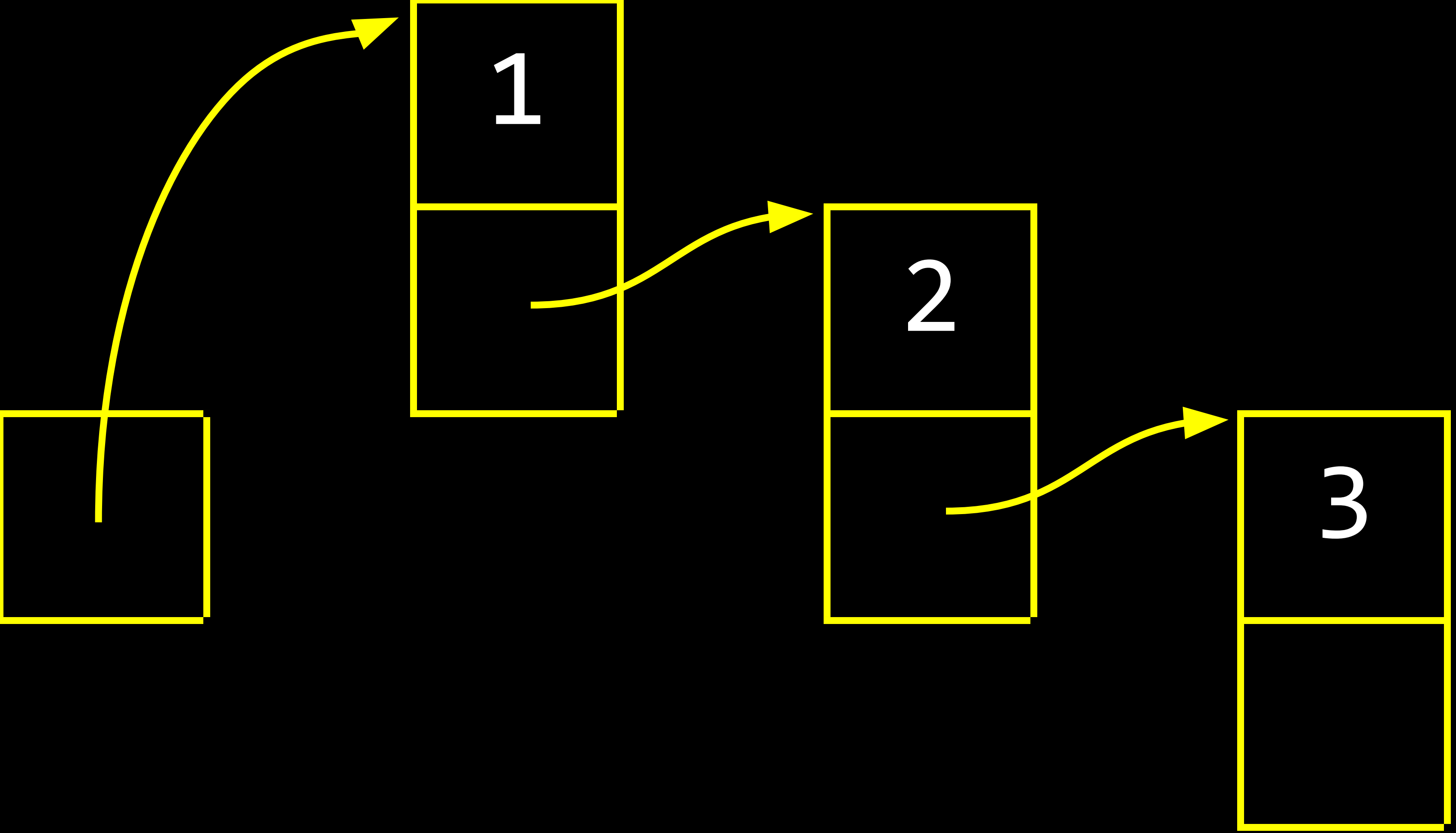

1
0x123
0x456
2
0x456
3
0x789


1
0x123
0x456
2
0x456
0x789
3
0x789


1
0x123
0x456
2
0x456
0x789
3
0x789
0x0


1
0x123
0x456
2
0x456
0x789
3
0x789
NULL


0x123
1
0x123
0x456
2
0x456
0x789
3
0x789
NULL


1
2
3

```
typedef struct
{
    char *name;
    char *number;
} person;
```

```
typedef struct
{



} node;
```

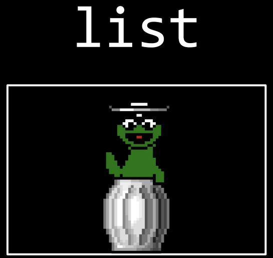
```
typedef struct
{
    int number;

} node;
```

```
typedef struct
{
    int number;
    node *next;
} node;
```

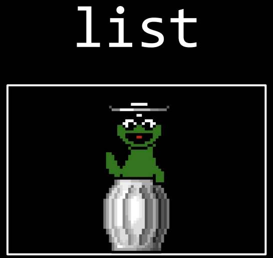
```
typedef struct node
{
    int number;
    node *next;
} node;
```

```
typedef struct node
{
    int number;
    struct node *next;
} node;
```

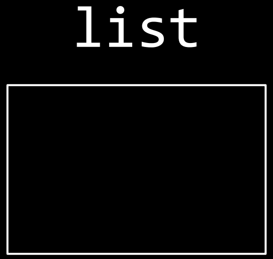
```
node *list;
node *list;
```



```
node *list = NULL;
```

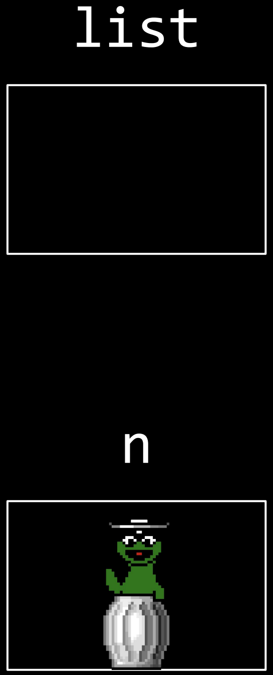


```
node *list = NULL;
```

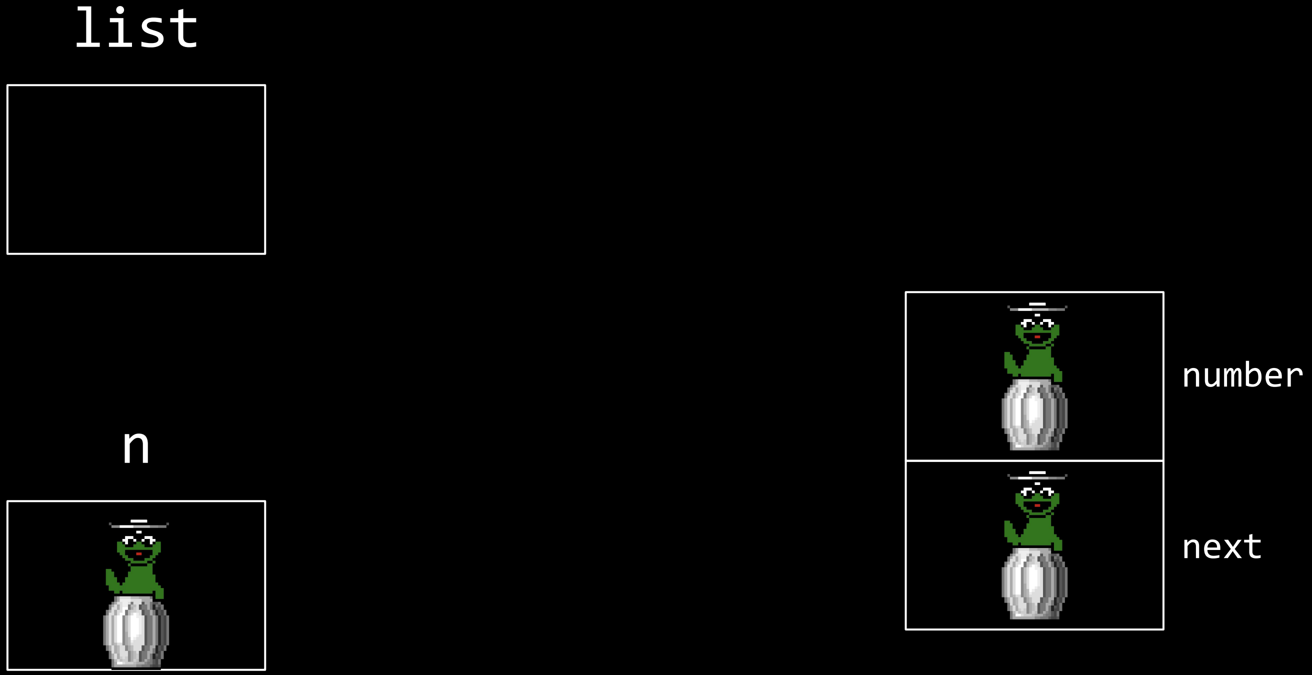


```
node *n = malloc(sizeof(node));
```

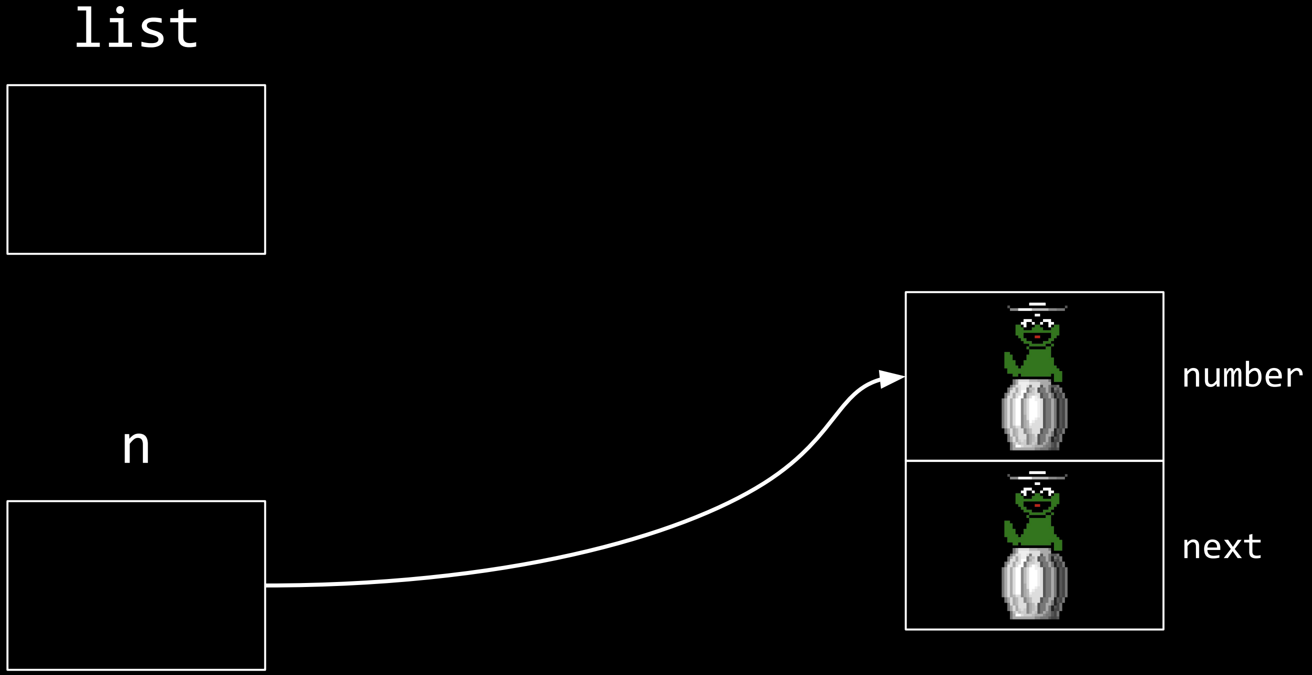
```
node *n = malloc(sizeof(node));
```



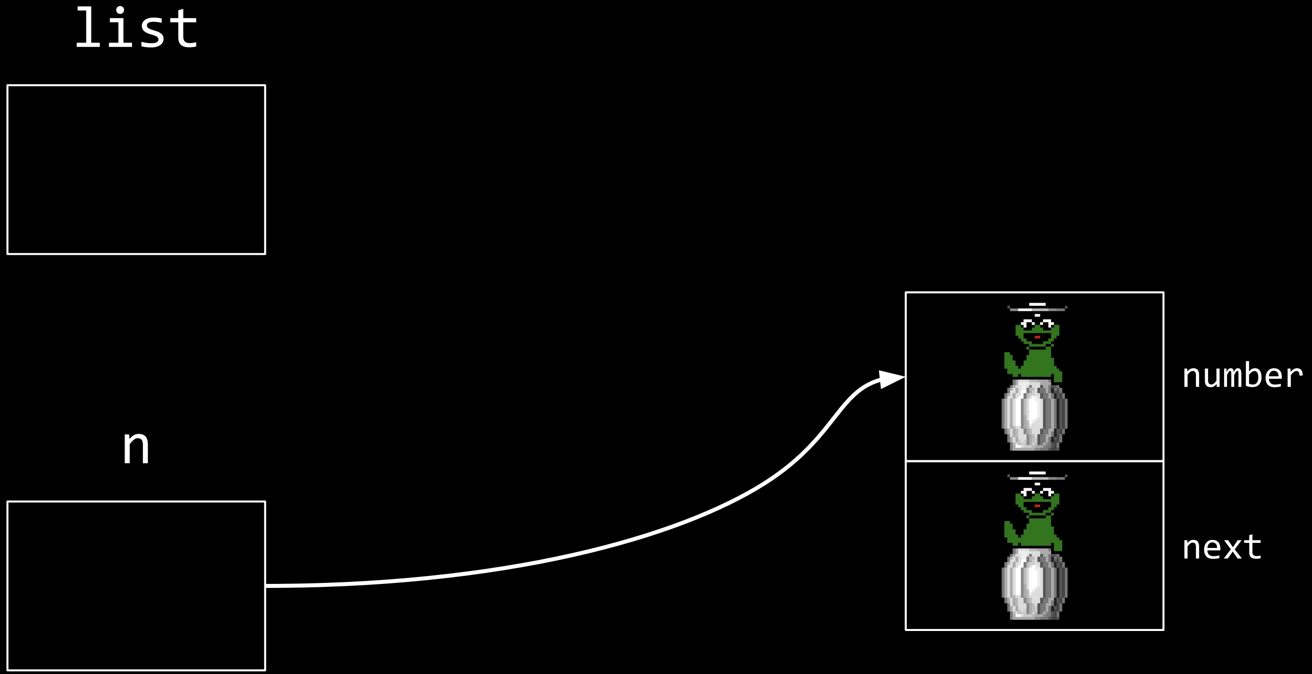
```
node *n = malloc(sizeof(node));
```



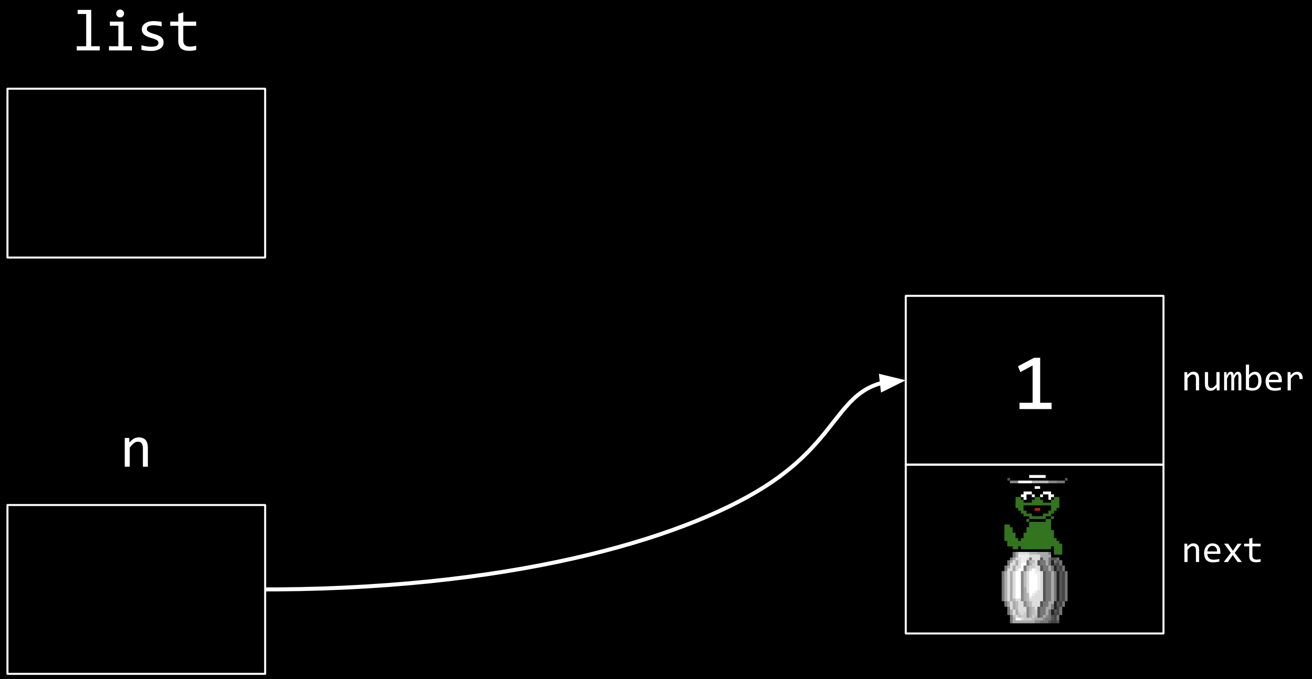
```
node *n = malloc(sizeof(node));
```



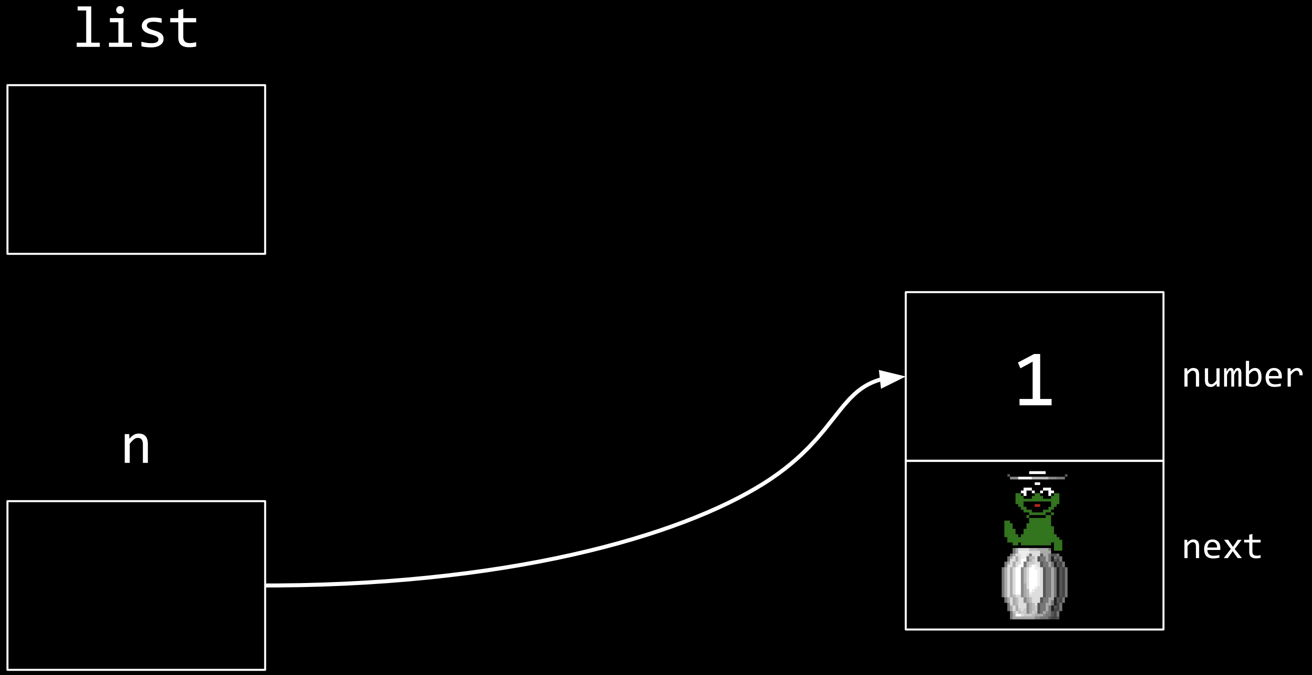
```
(*n).number = 1;
```



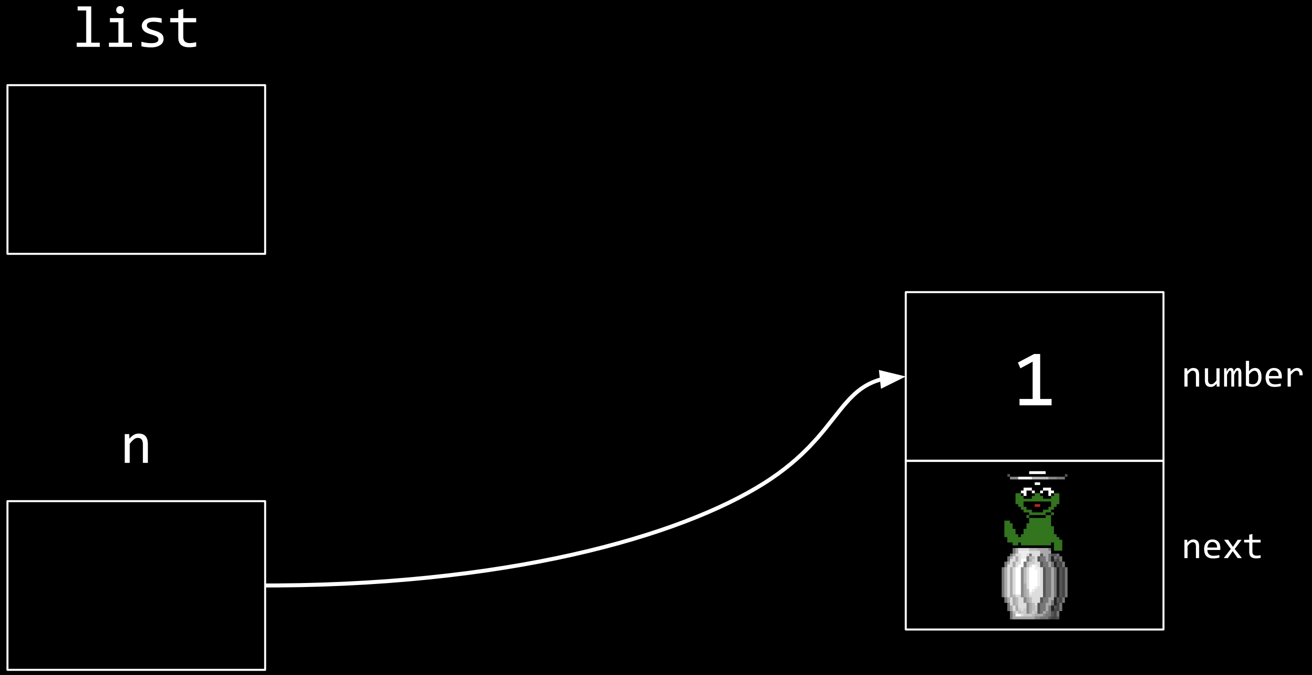
```
(*n).number = 1;
```



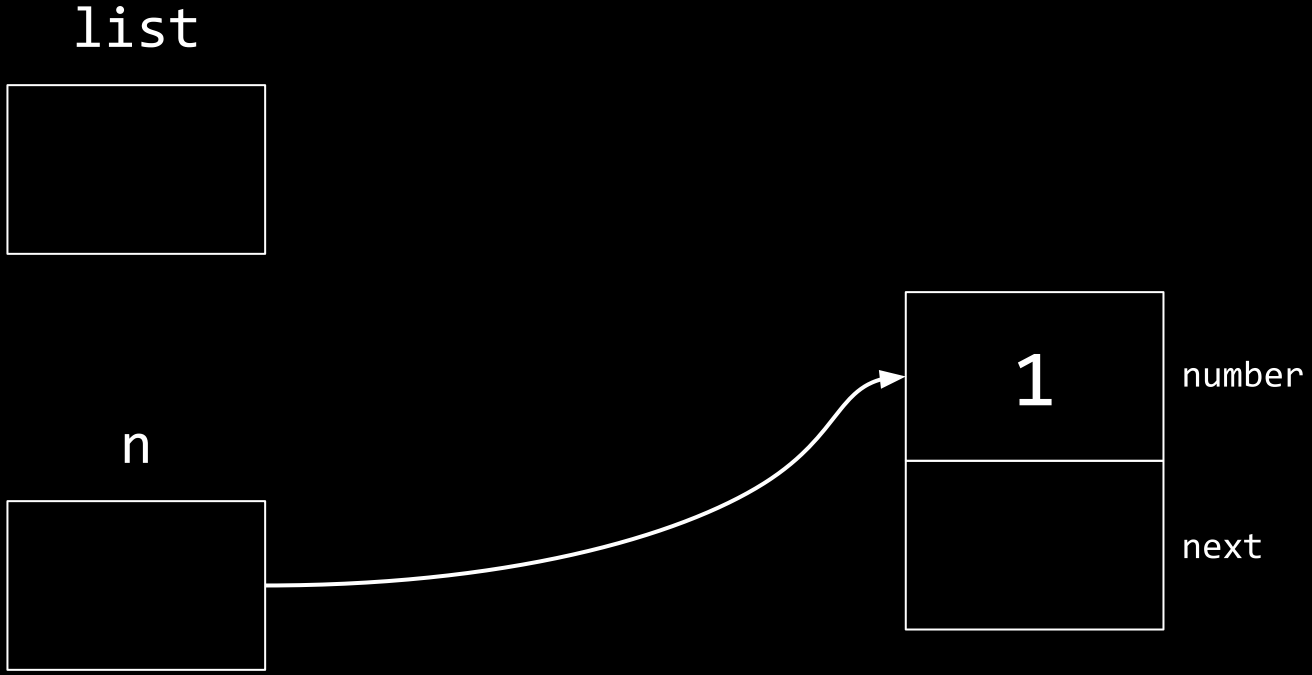
```
n->number = 1;
```



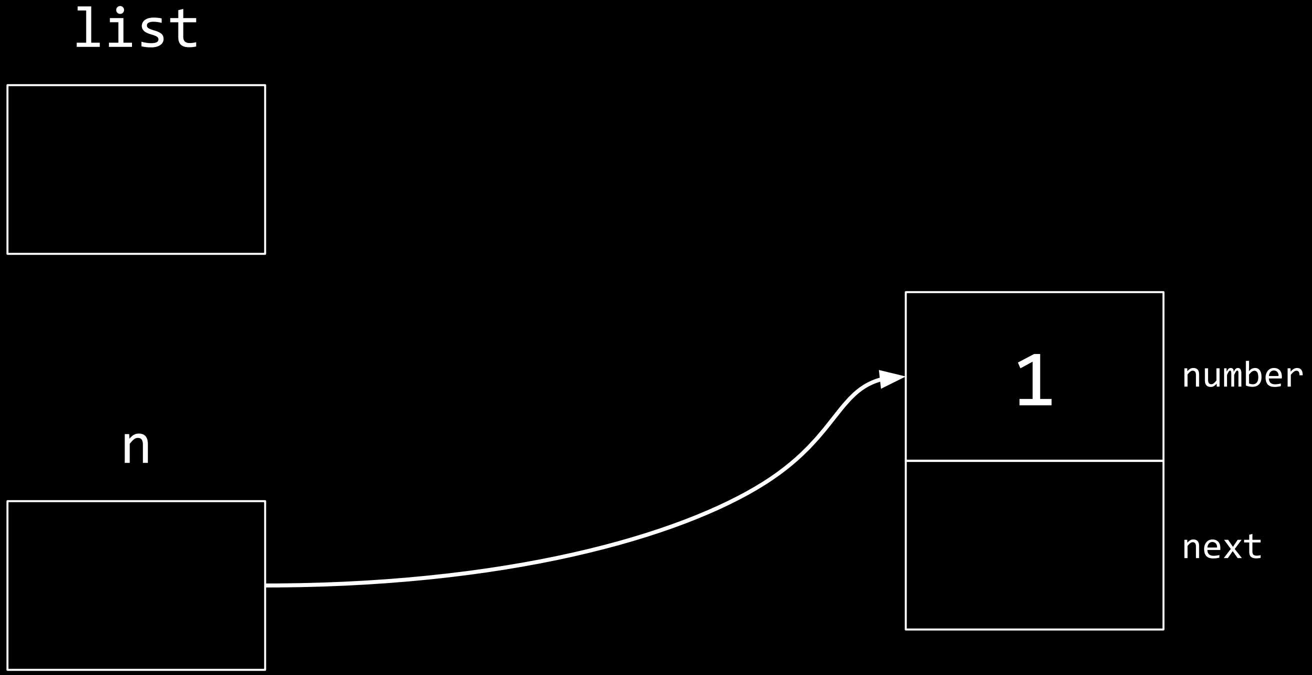
```
n->next = NULL;
```



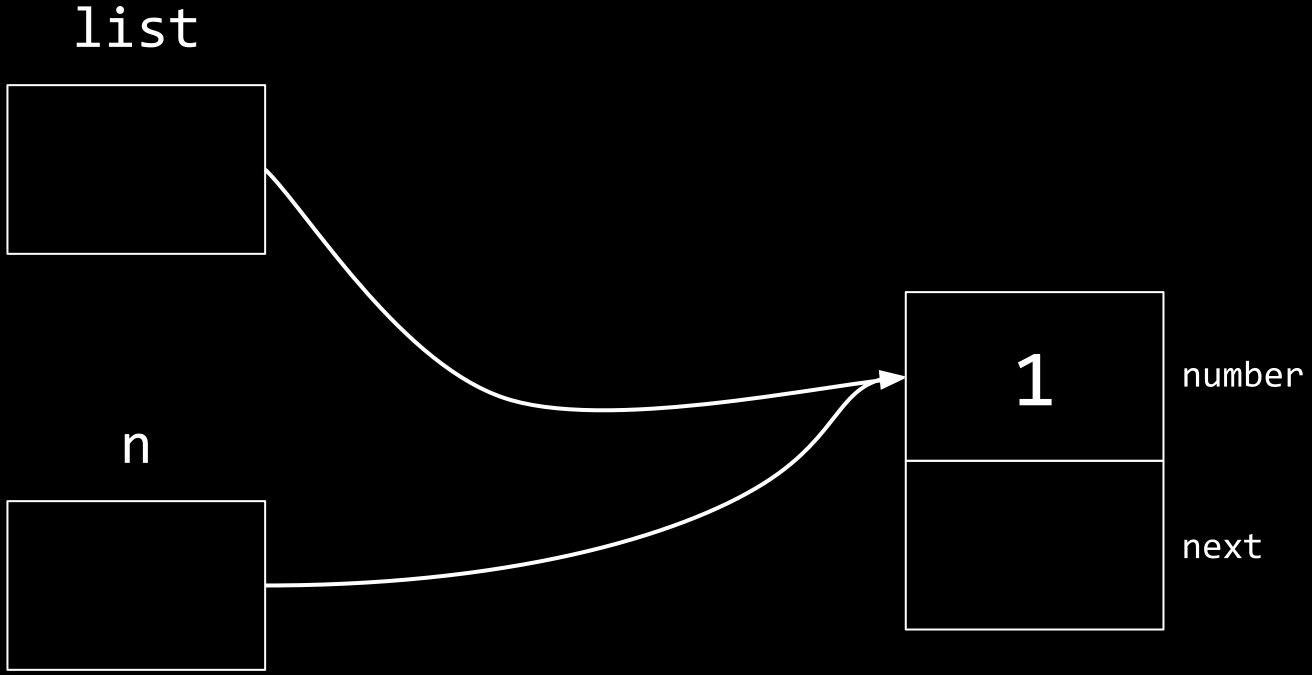
```
n->next = NULL;
```

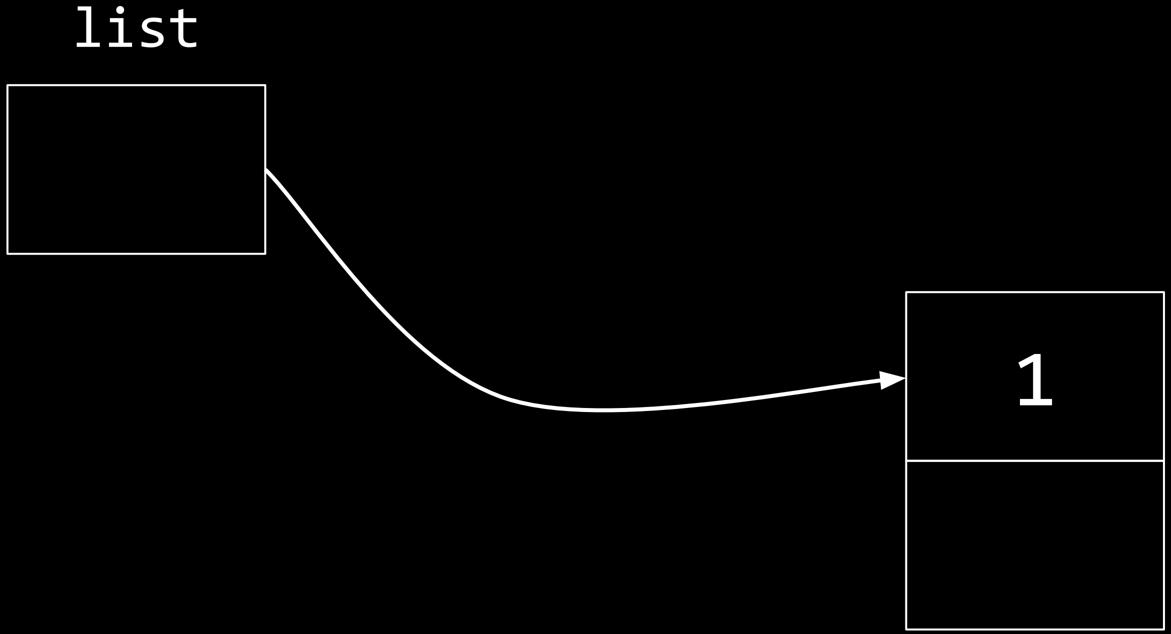


```
list = n;
```

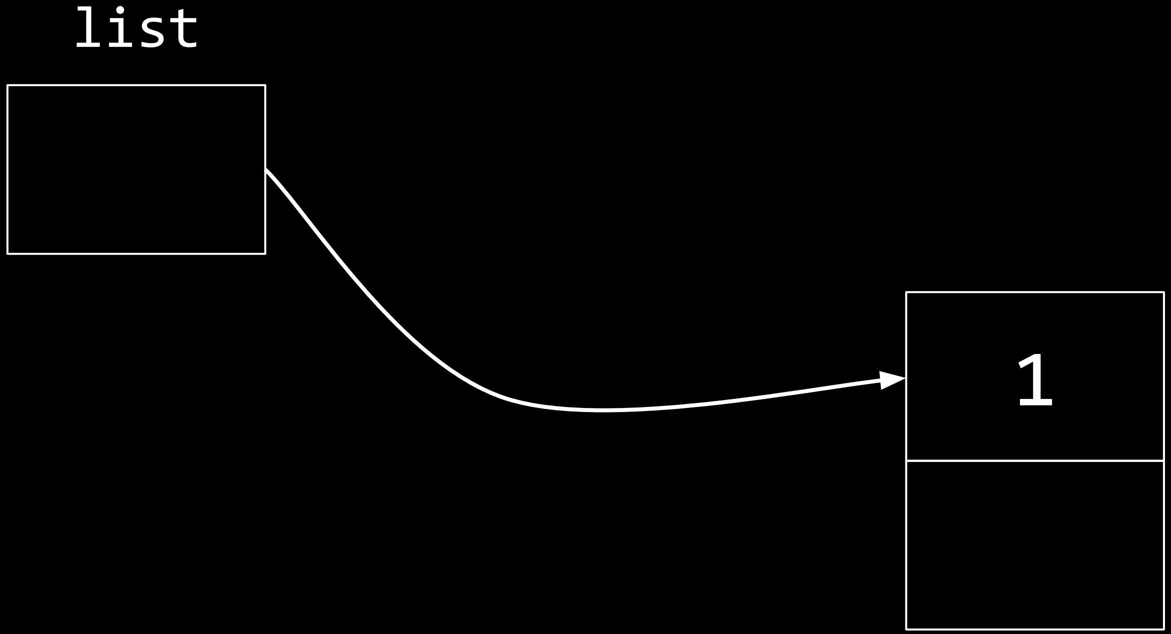


```
list = n;
```

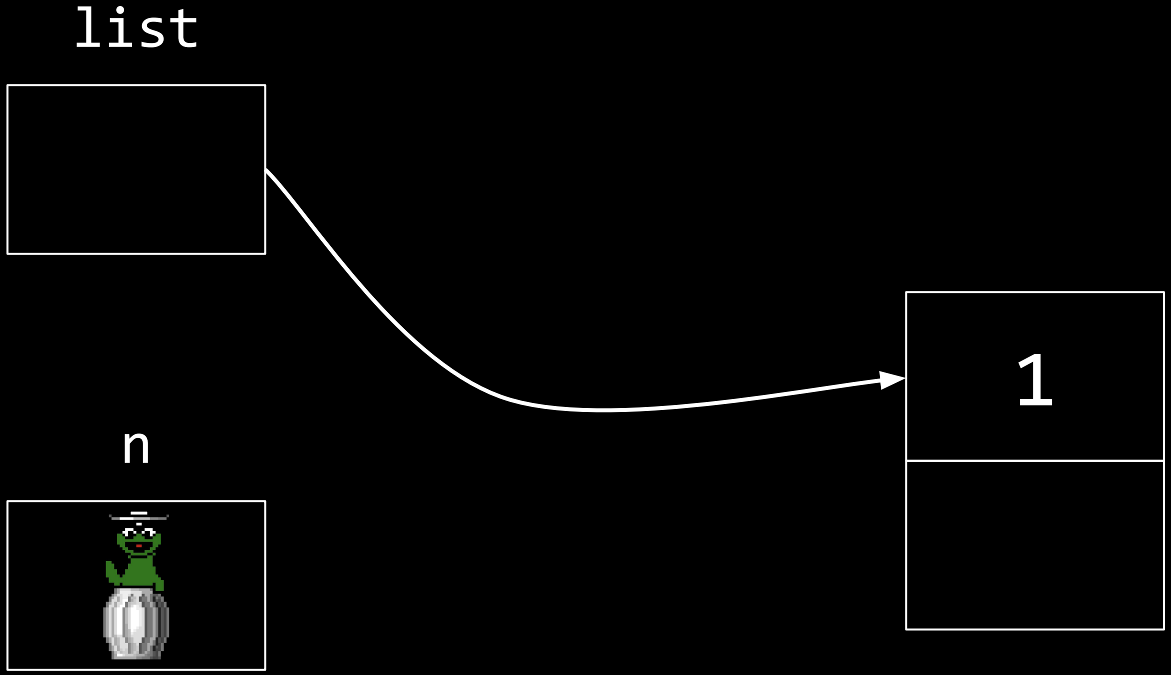



list
1

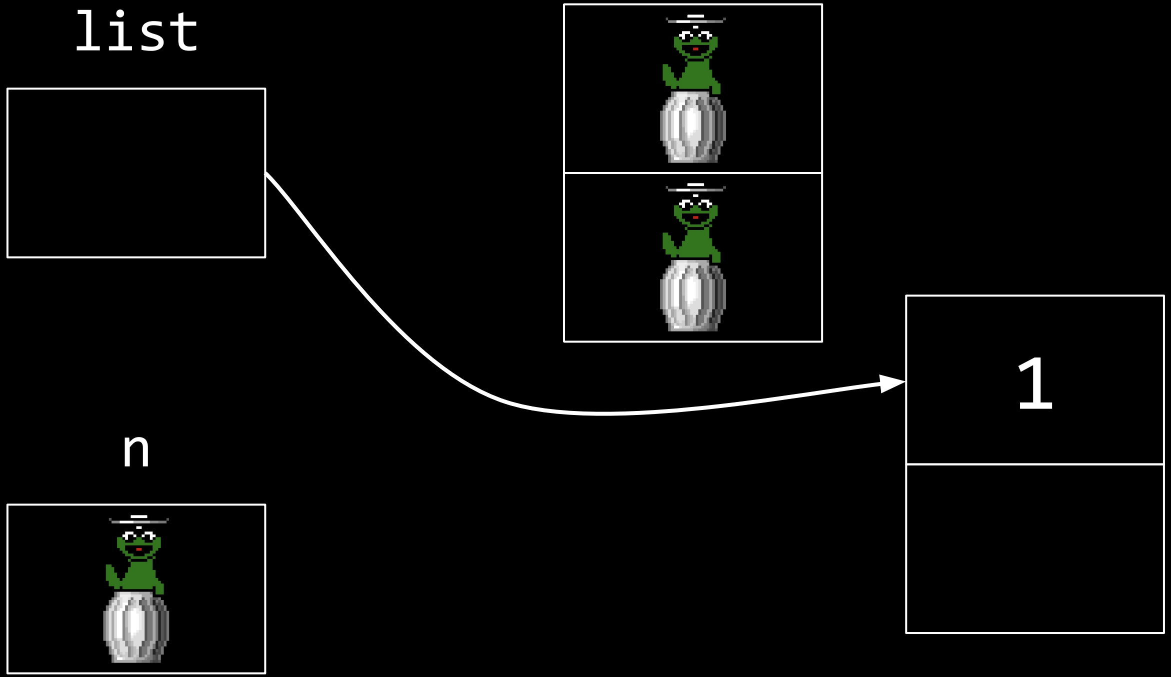
```
node *n = malloc(sizeof(node));
```



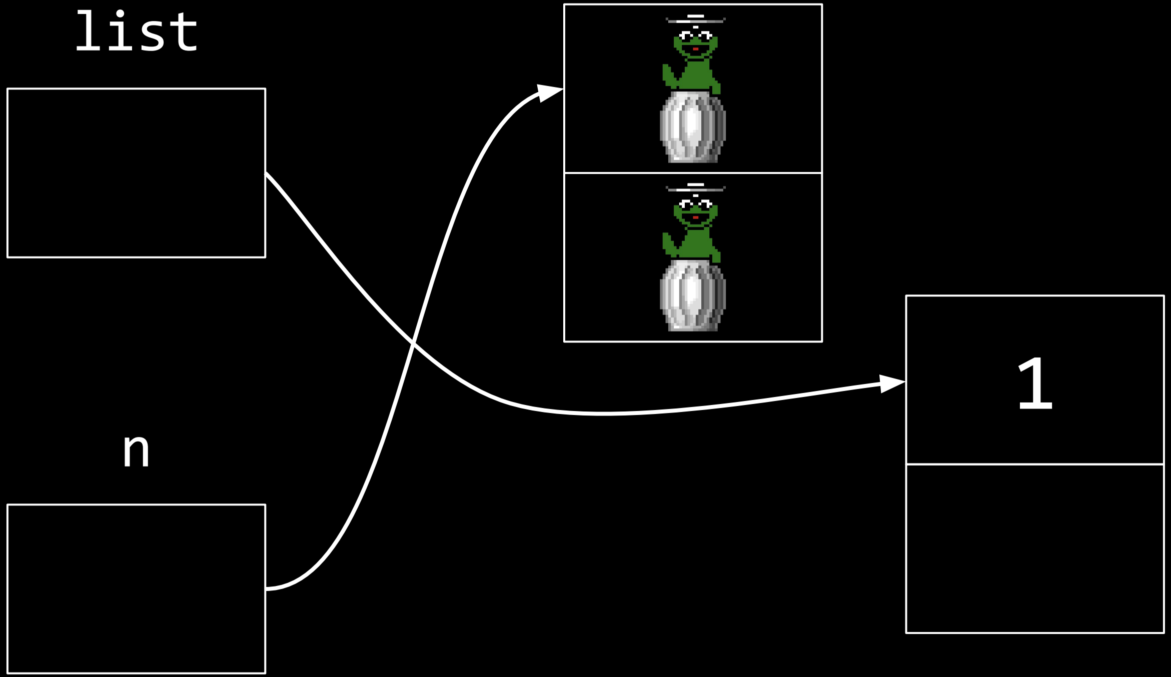
```
node *n = malloc(sizeof(node));
```



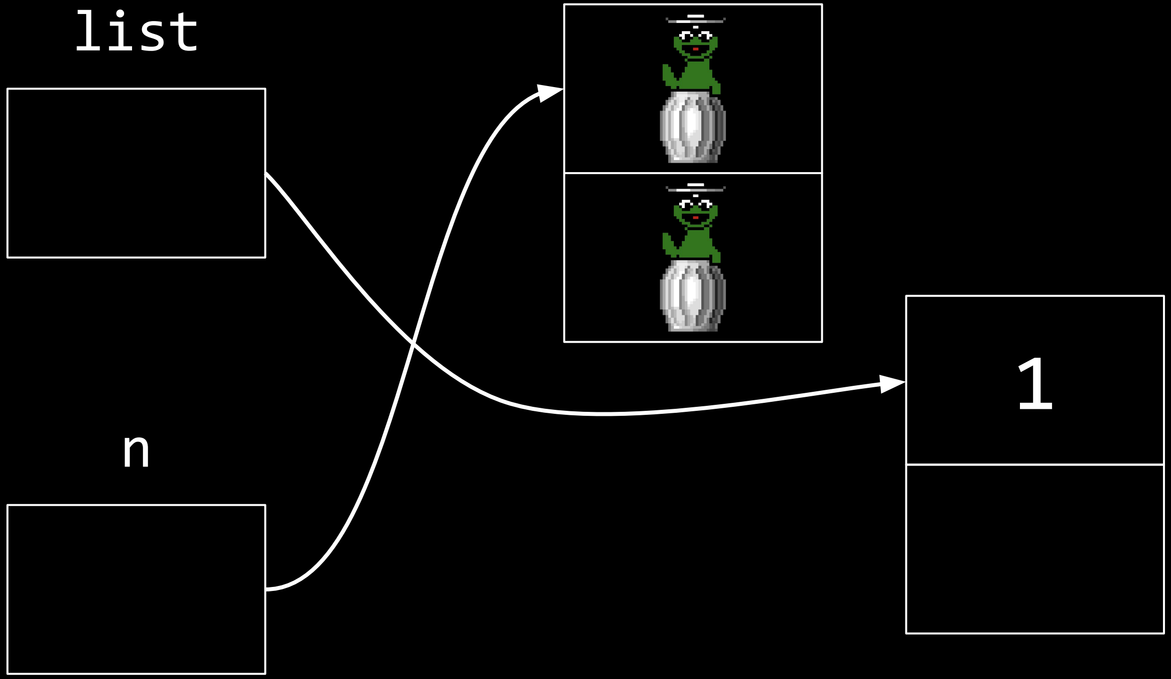
```
node *n = malloc(sizeof(node));
```



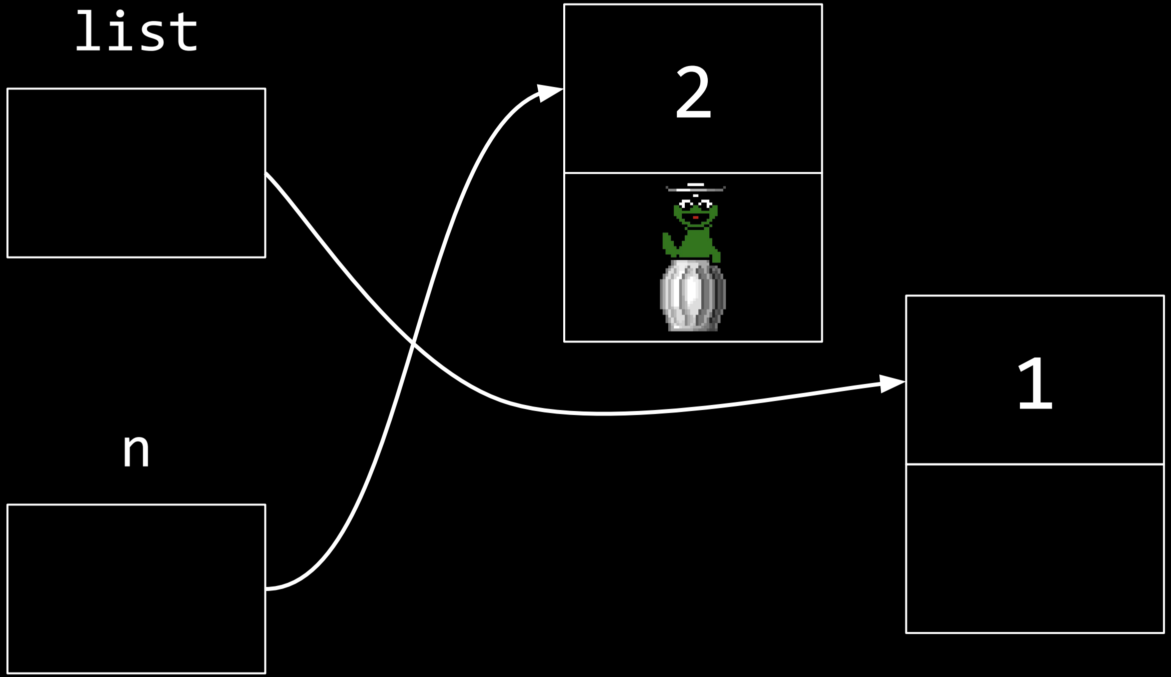
```
node *n = malloc(sizeof(node));
```



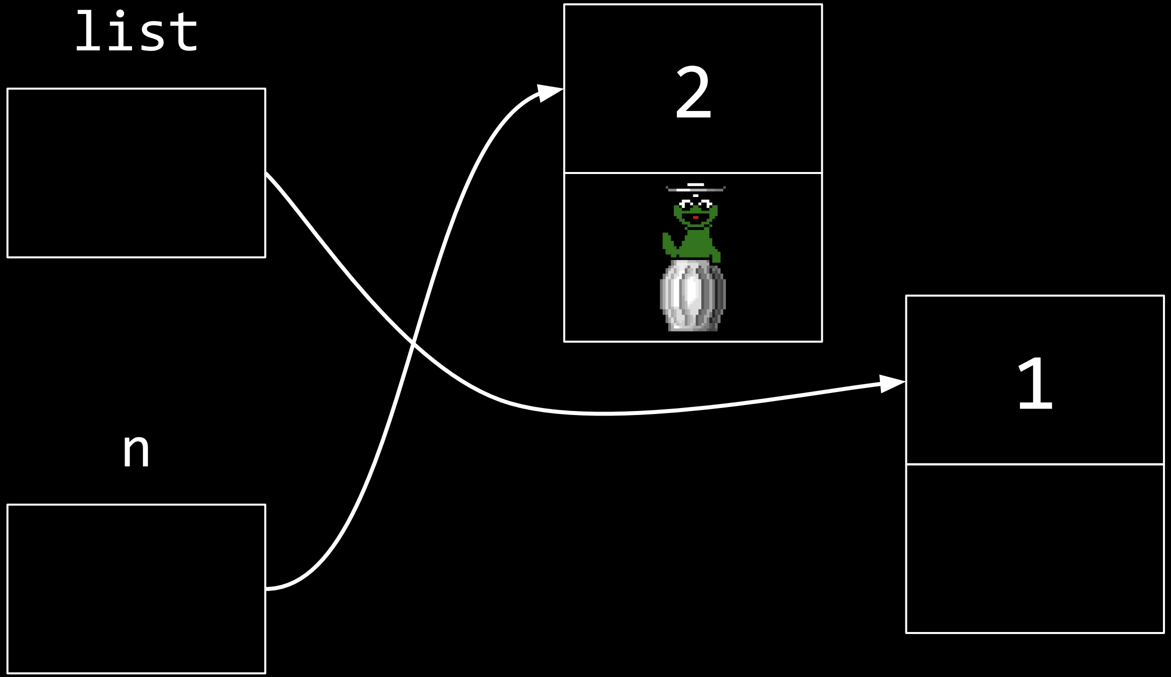
```
n->number = 2;
```



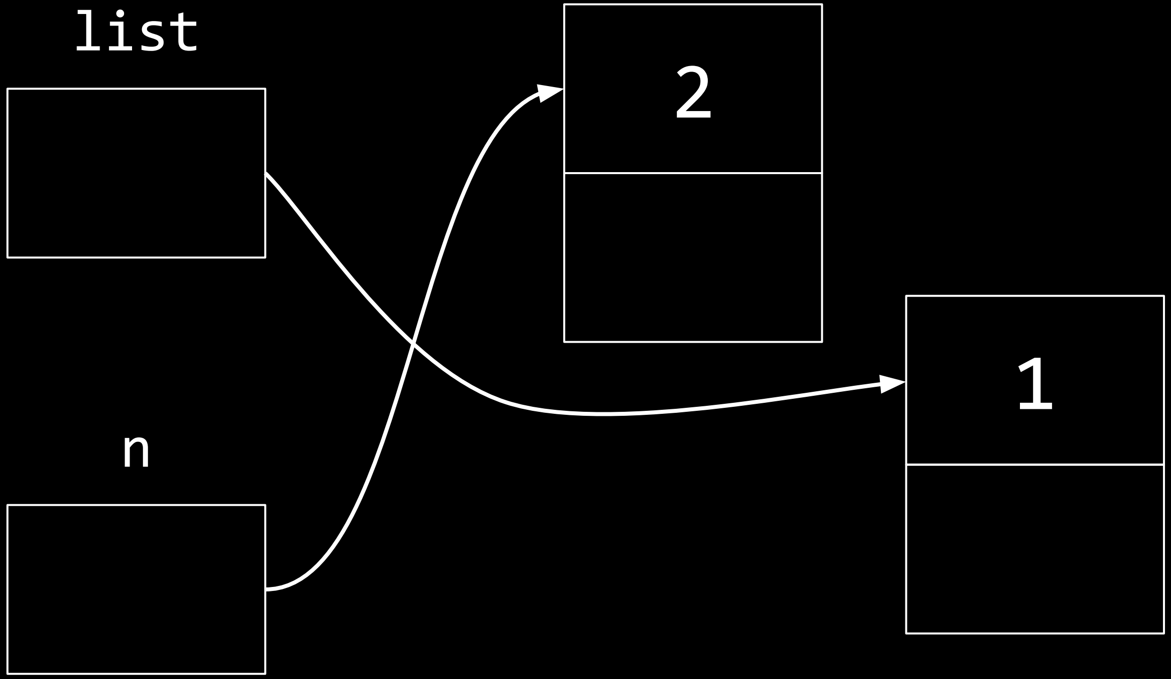
```
n->number = 2;
```

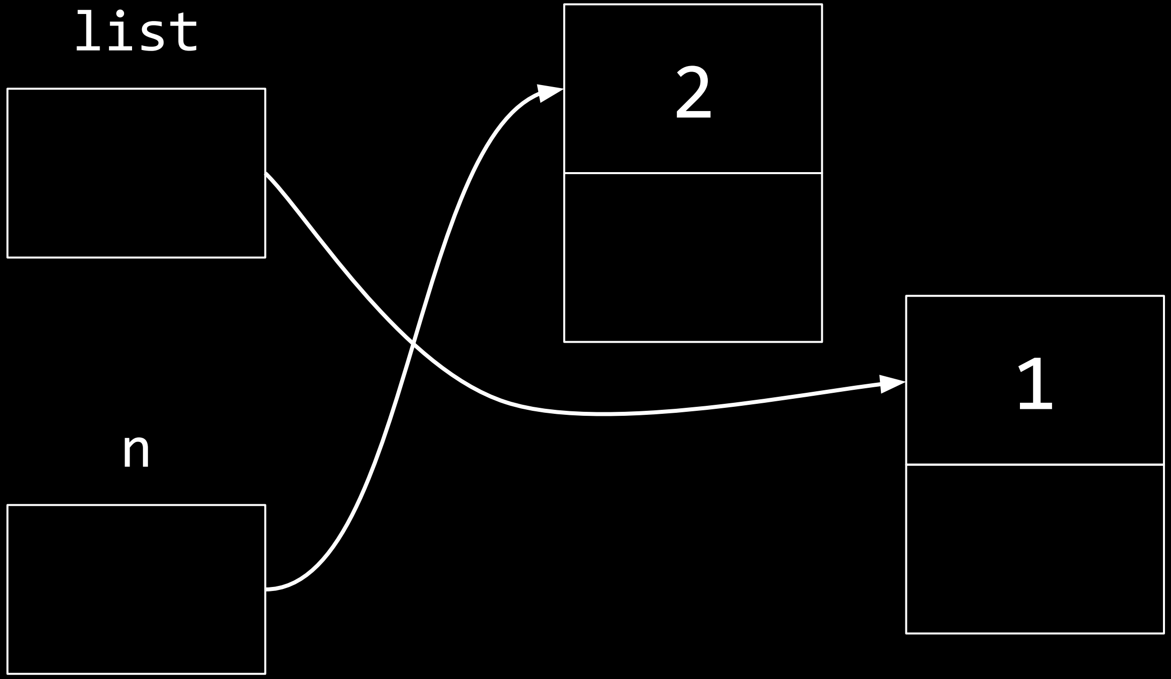


```
n->next = NULL;
```

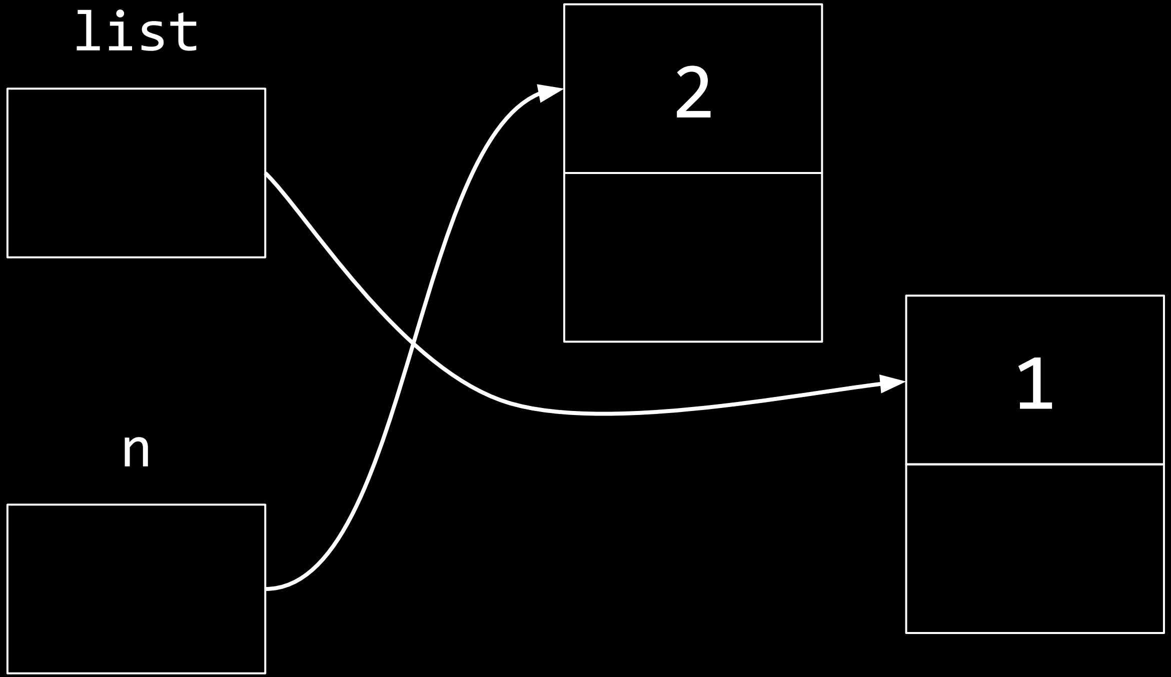


```
n->next = NULL;
```

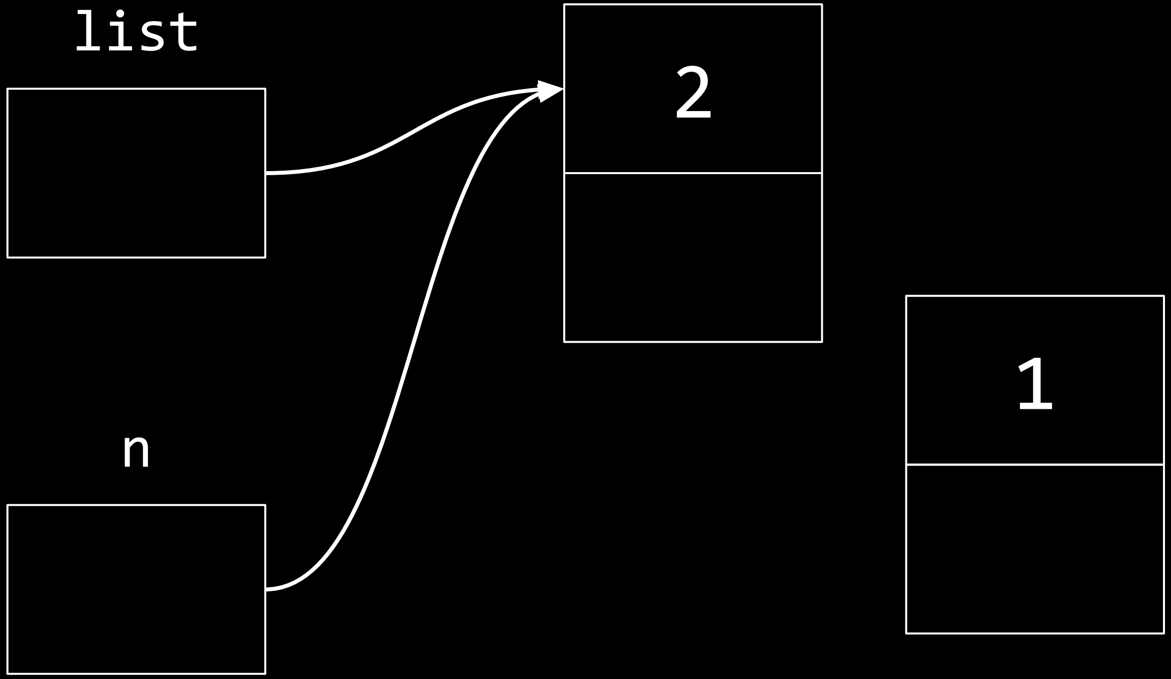



list
n
2
1

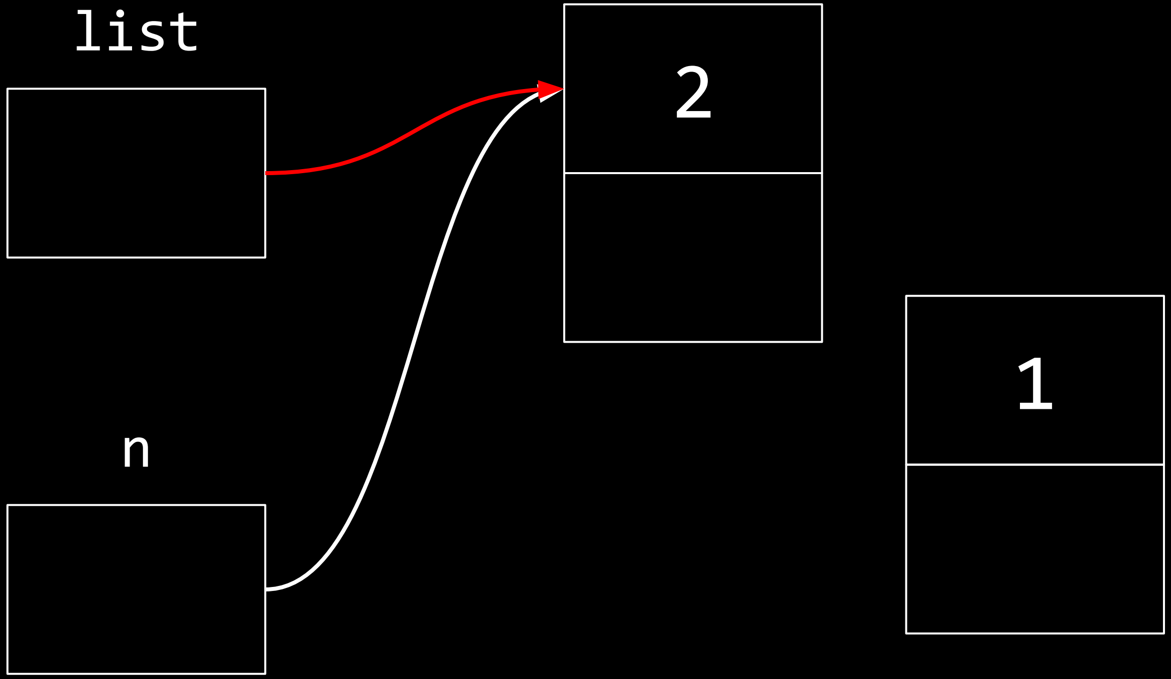
```
list = n;
```

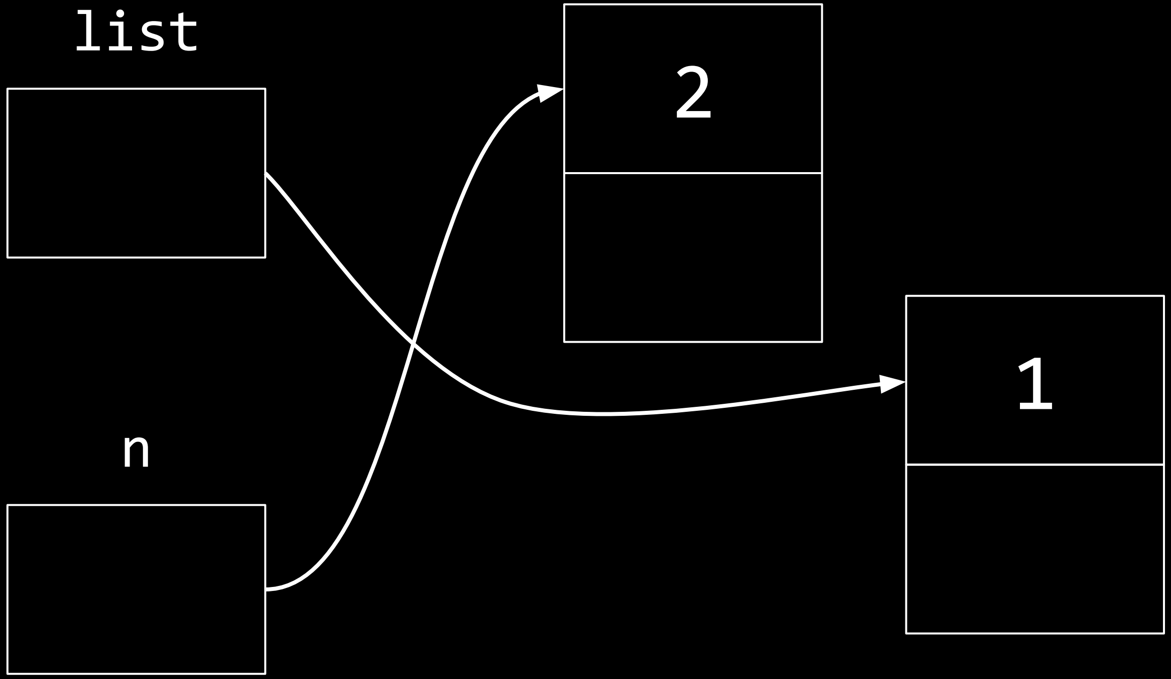


```
list = n;
```

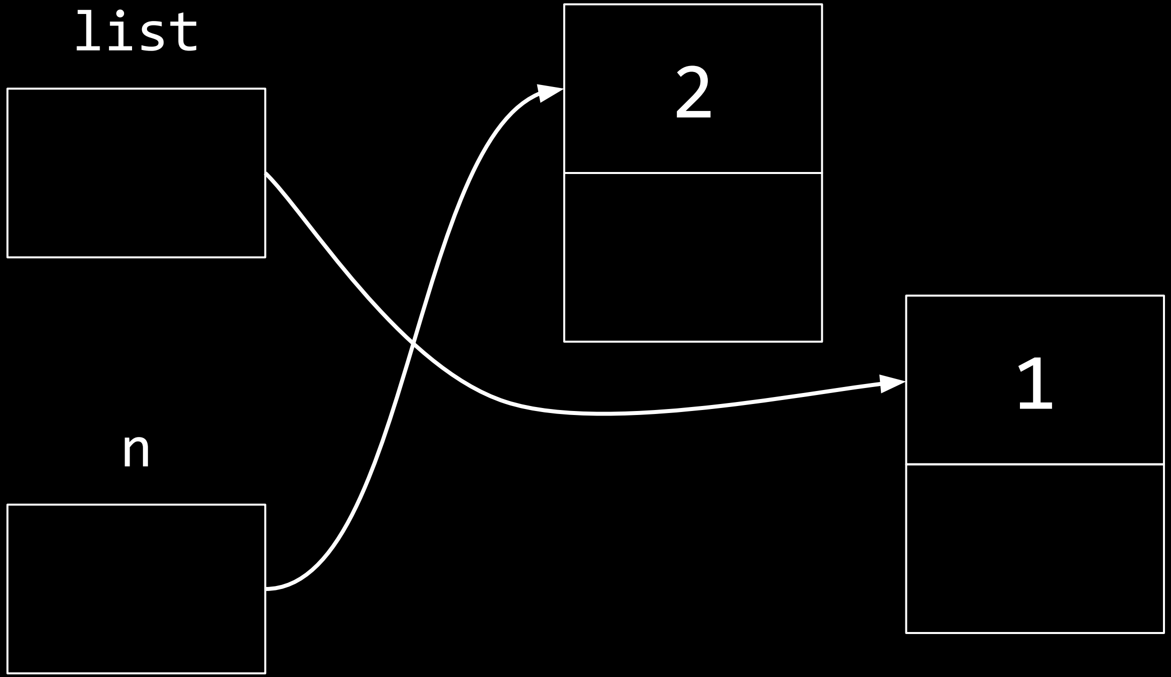


```
list = n;
```

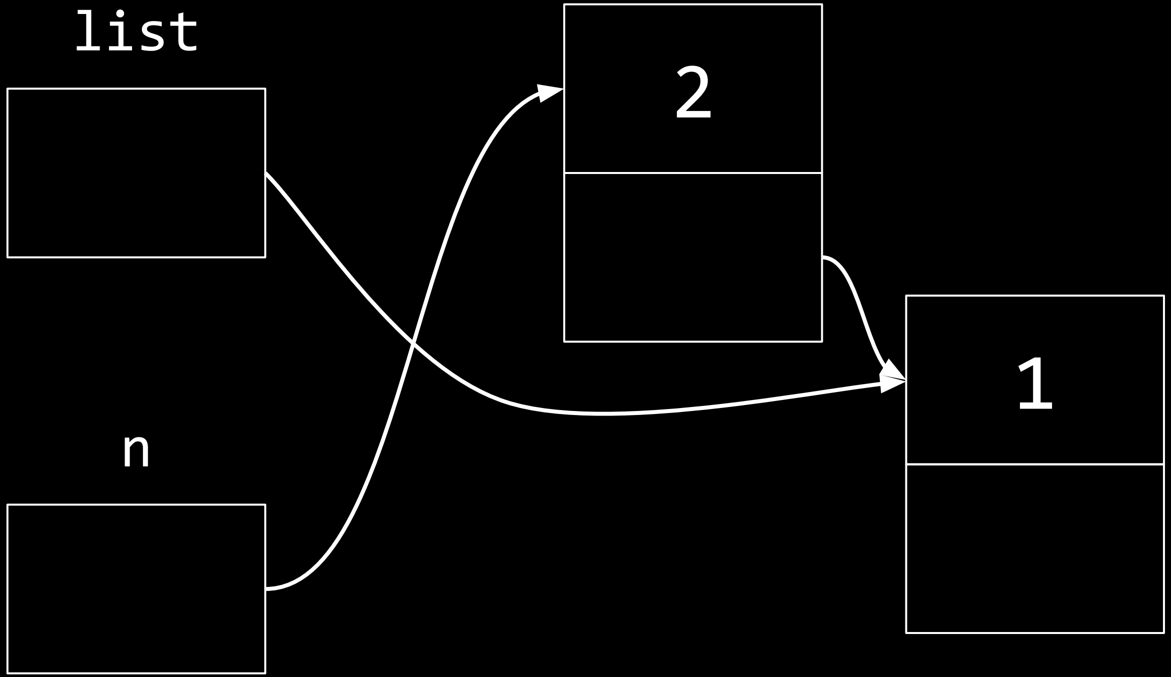



list
n
2
1

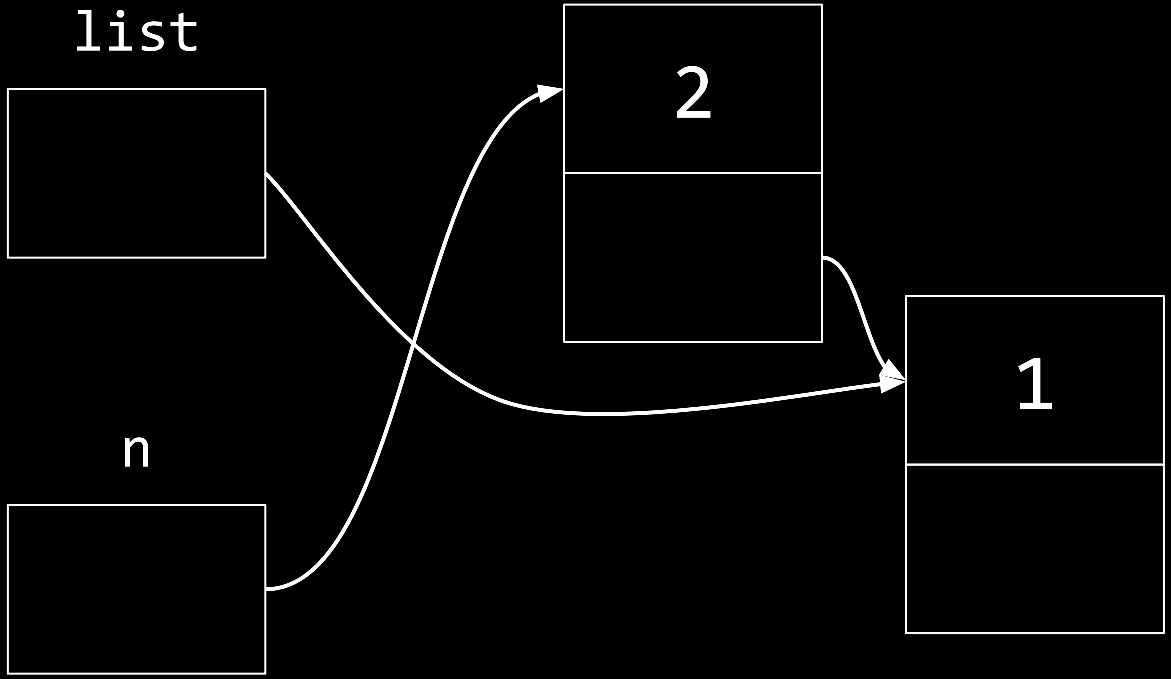
```
n->next = list;
```



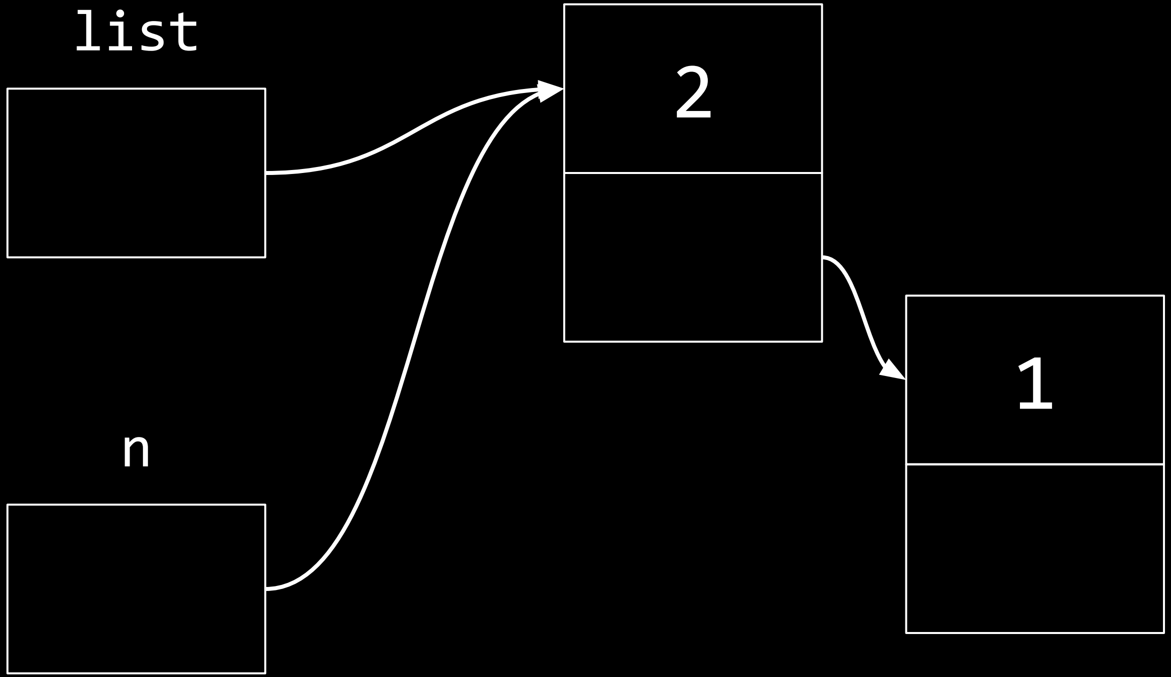
```
n->next = list;
```

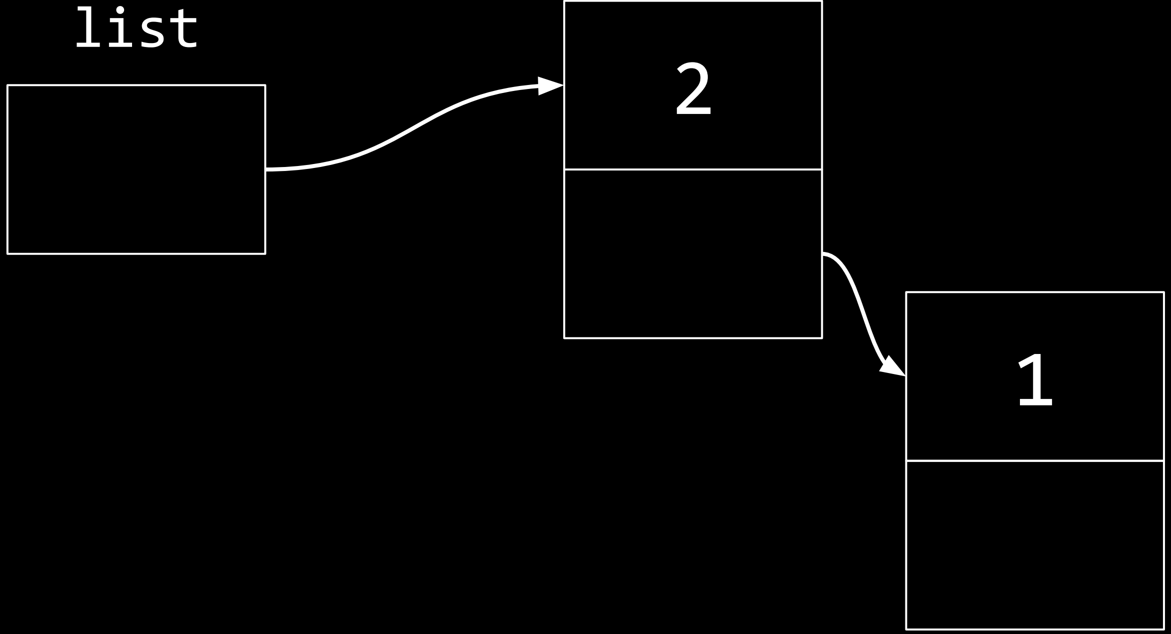


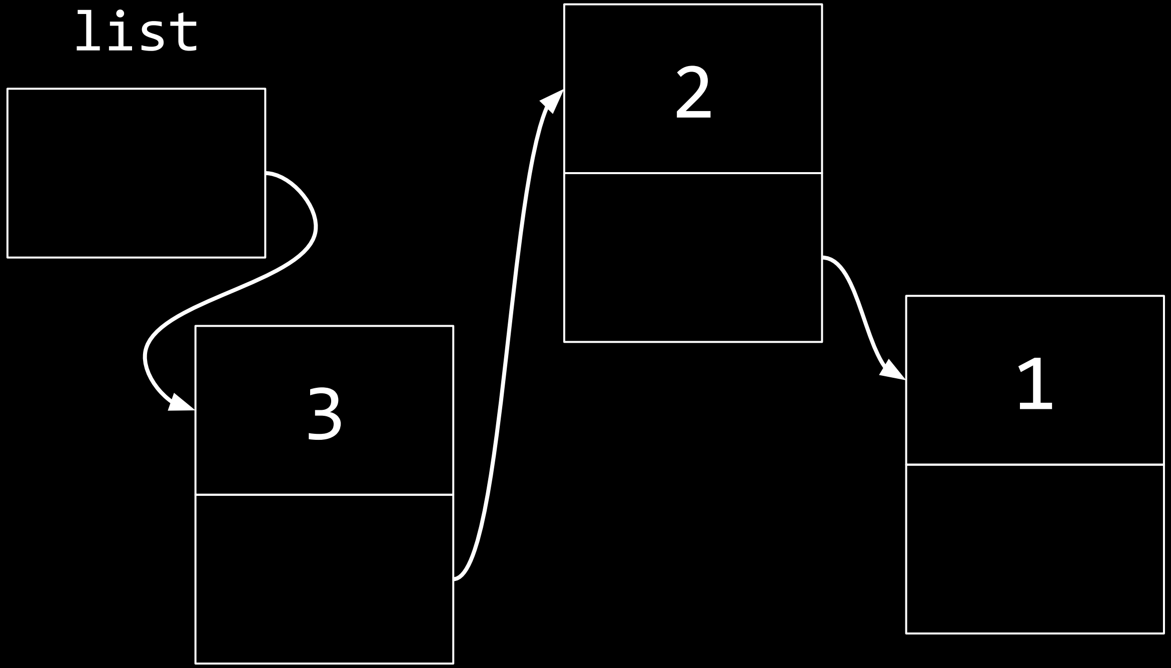
```
list = n;
```

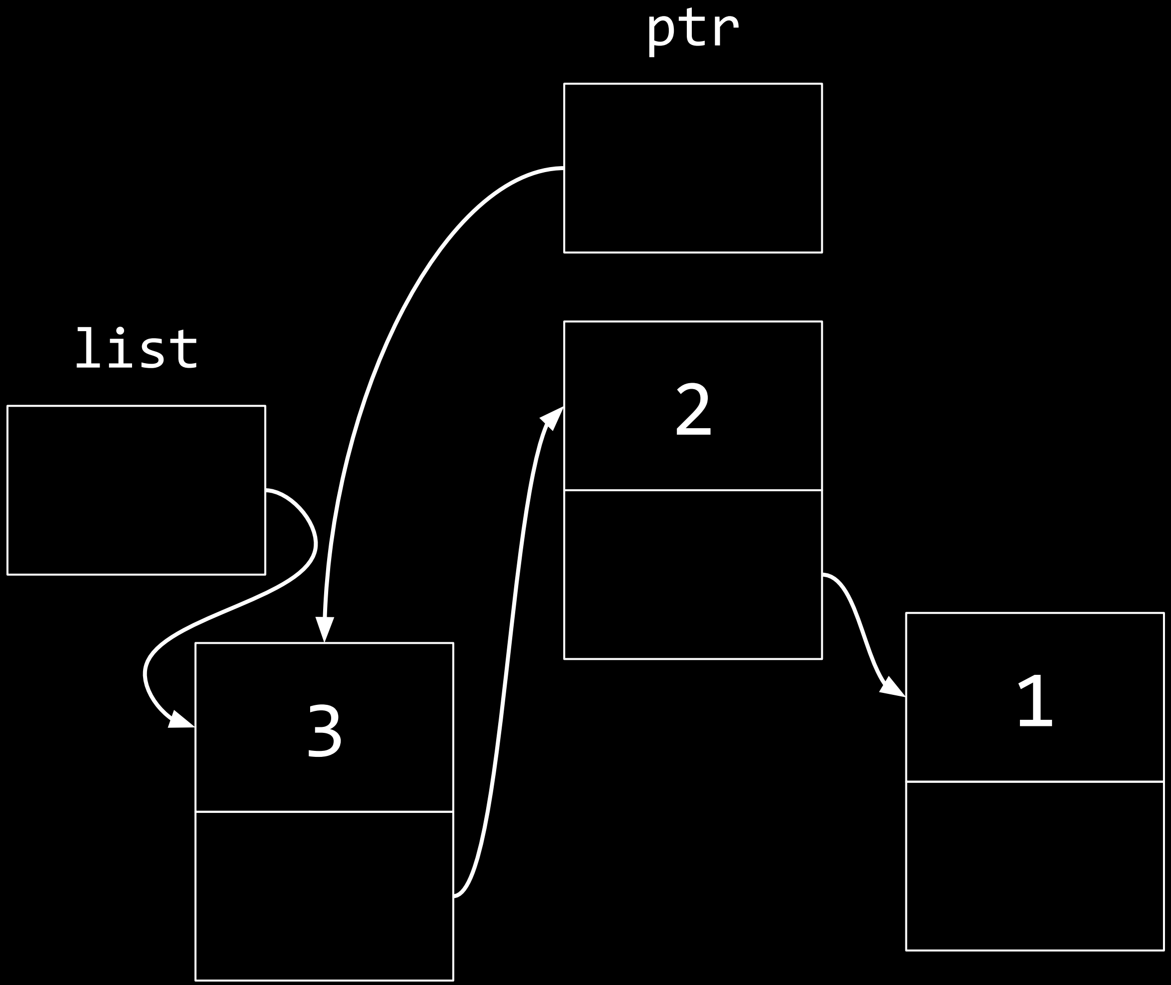


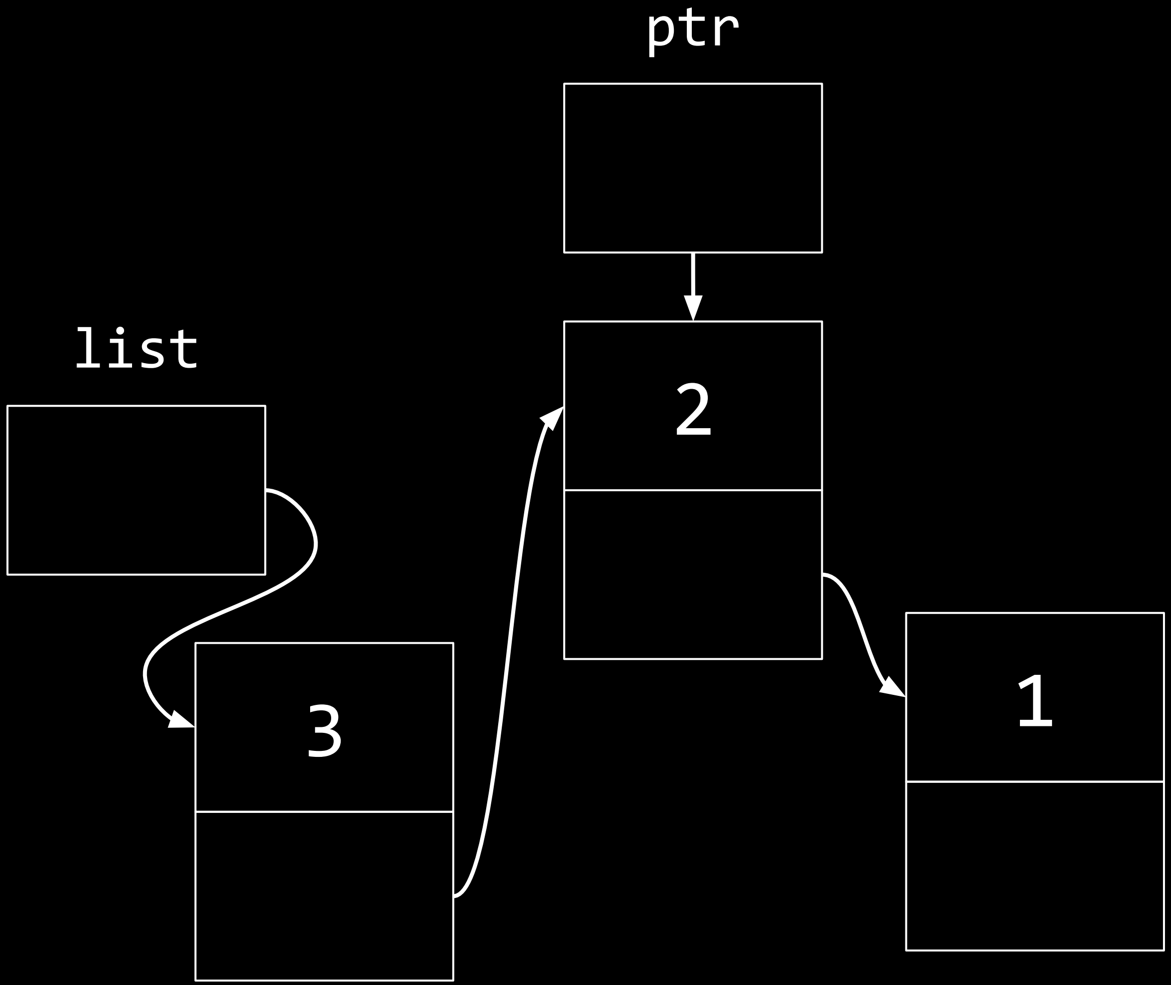
```
list = n;
```

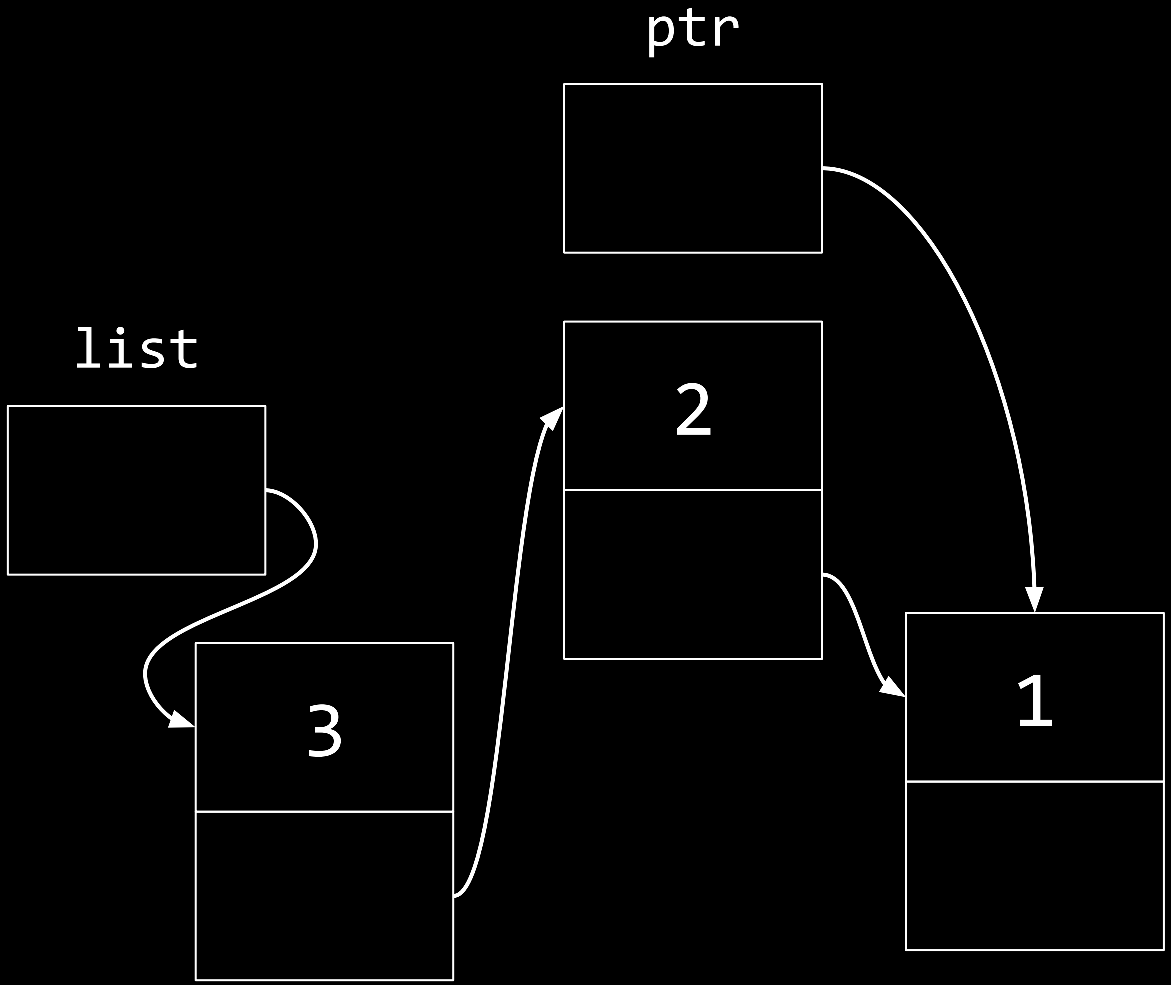


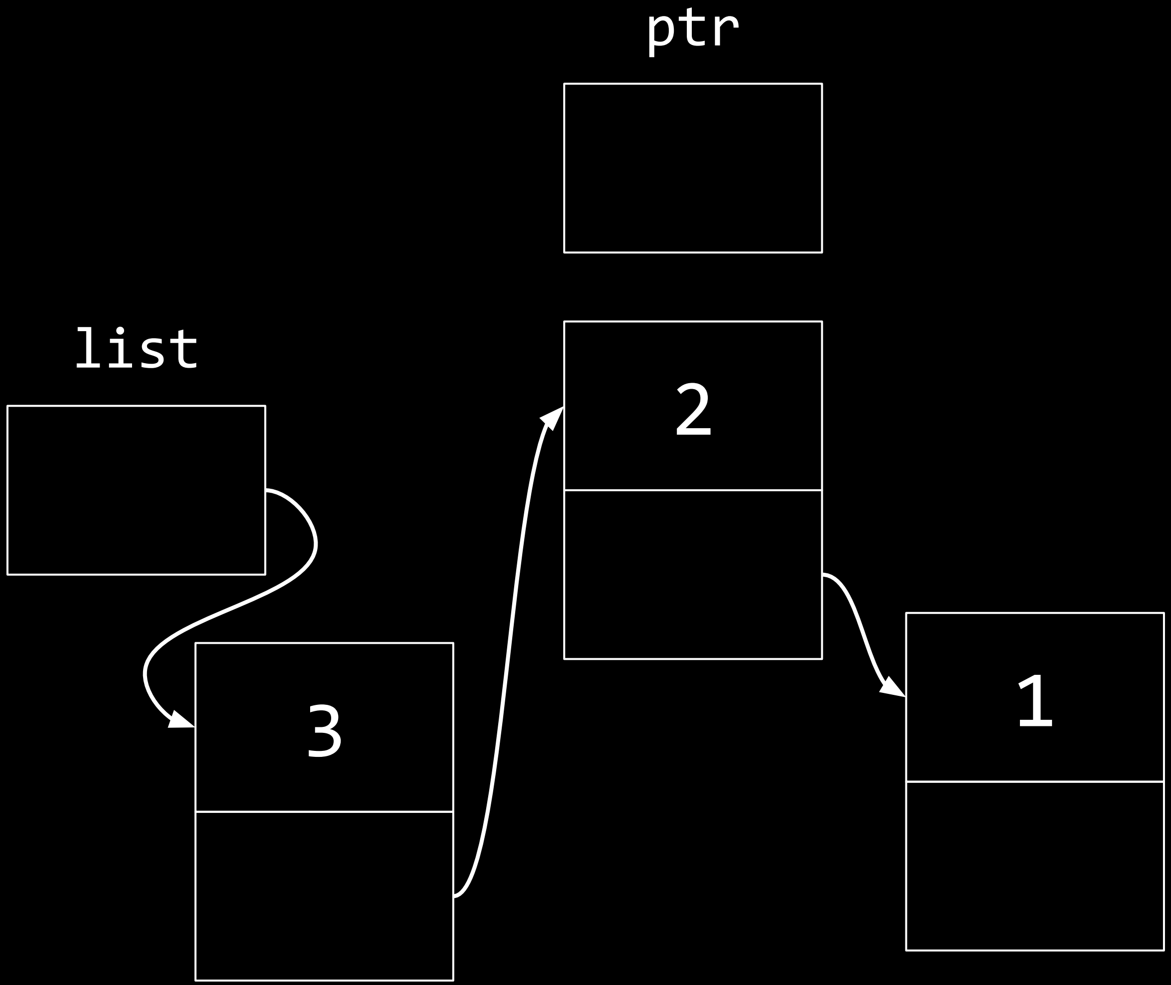

list
2
1

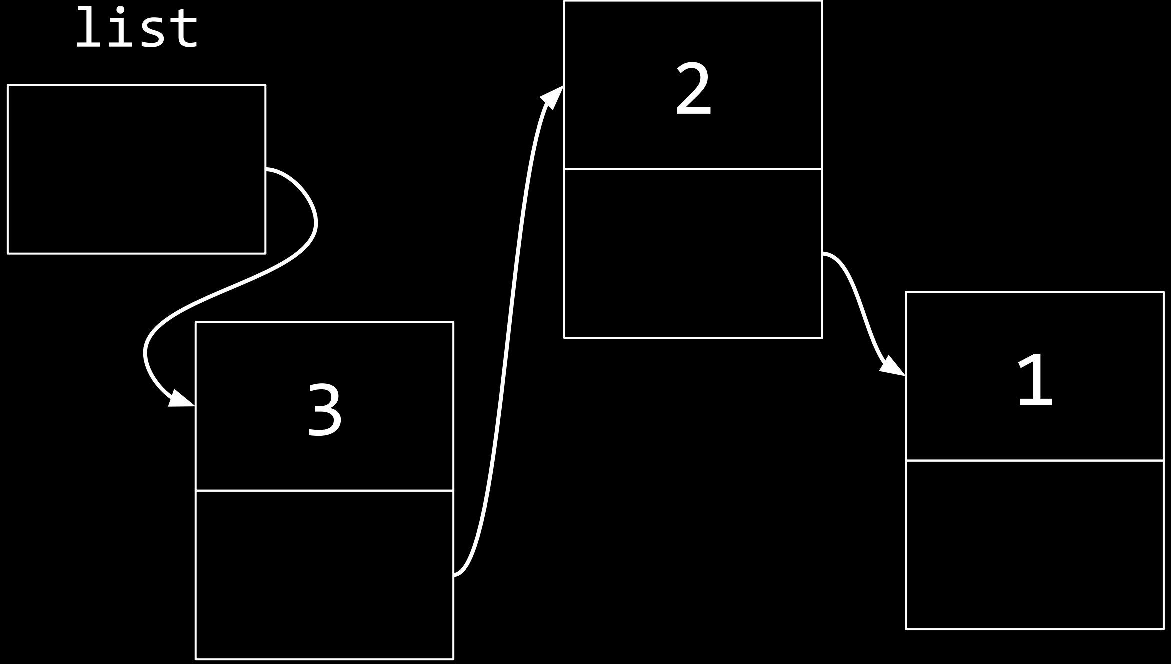

list
3
2
1


ptr
list
2
3
1


ptr
list
2
3
1


ptr
list
2
3
1


ptr
list
2
3
1


list
3
2
1

$O(n^2)$

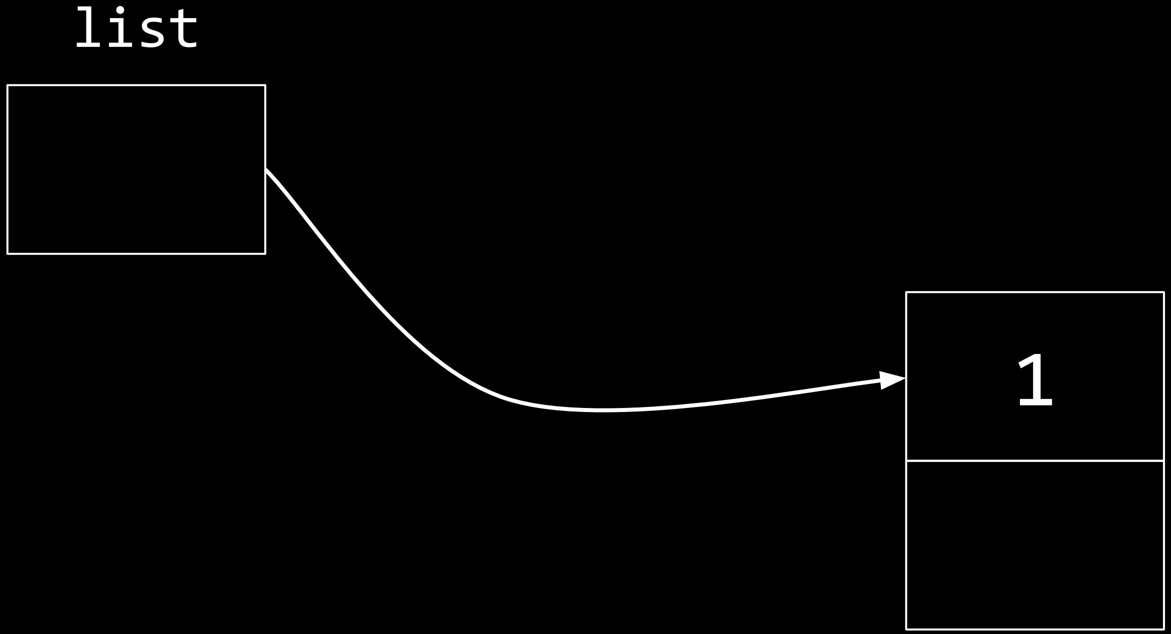
$O(n \log n)$

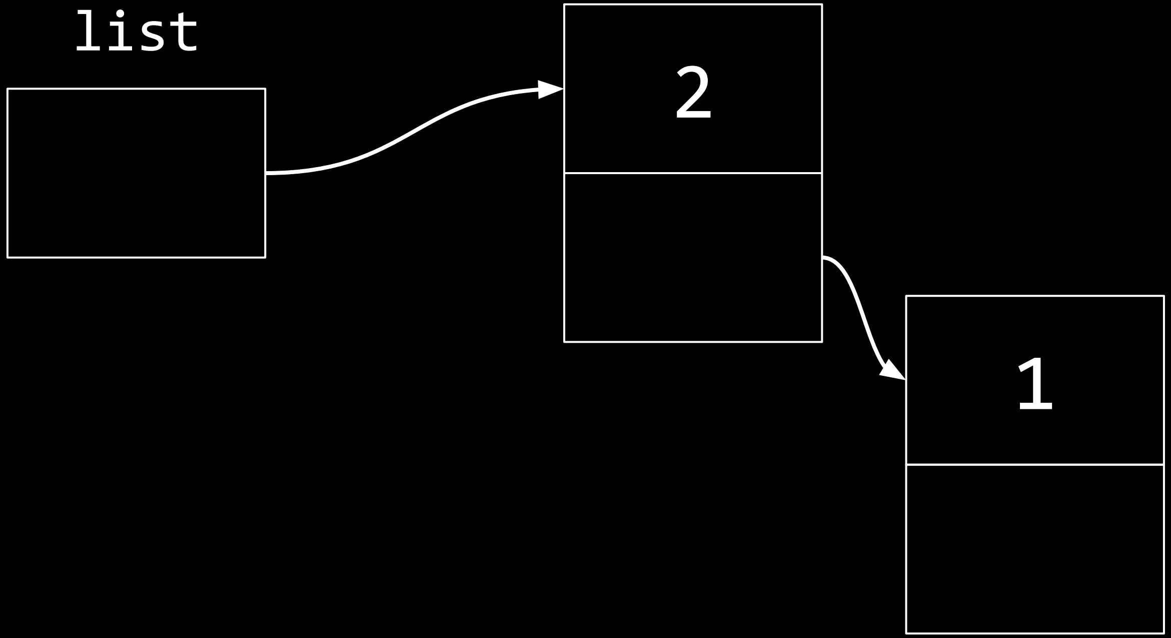
$O(n)$

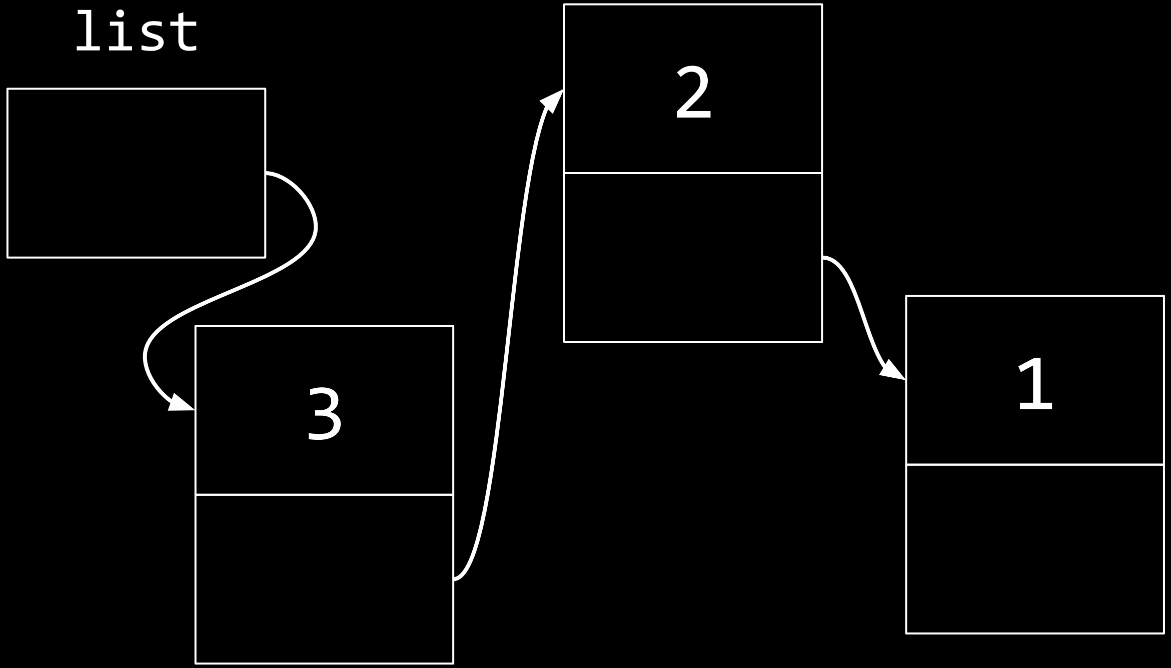
$O(\log n)$

$O(1)$

list


list
1


list
2
1


list
3
2
1

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

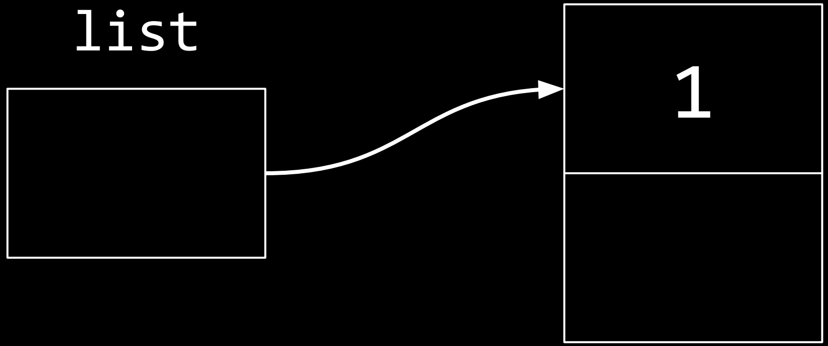
$O(n \log n)$

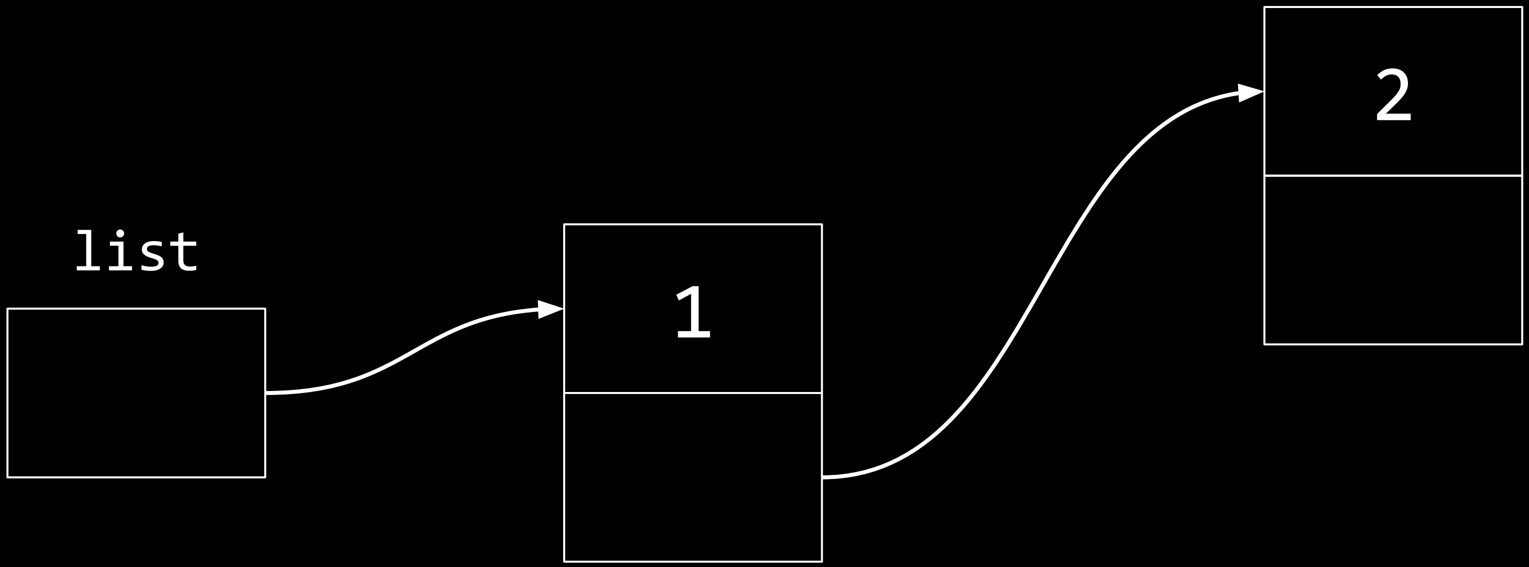
$O(n)$

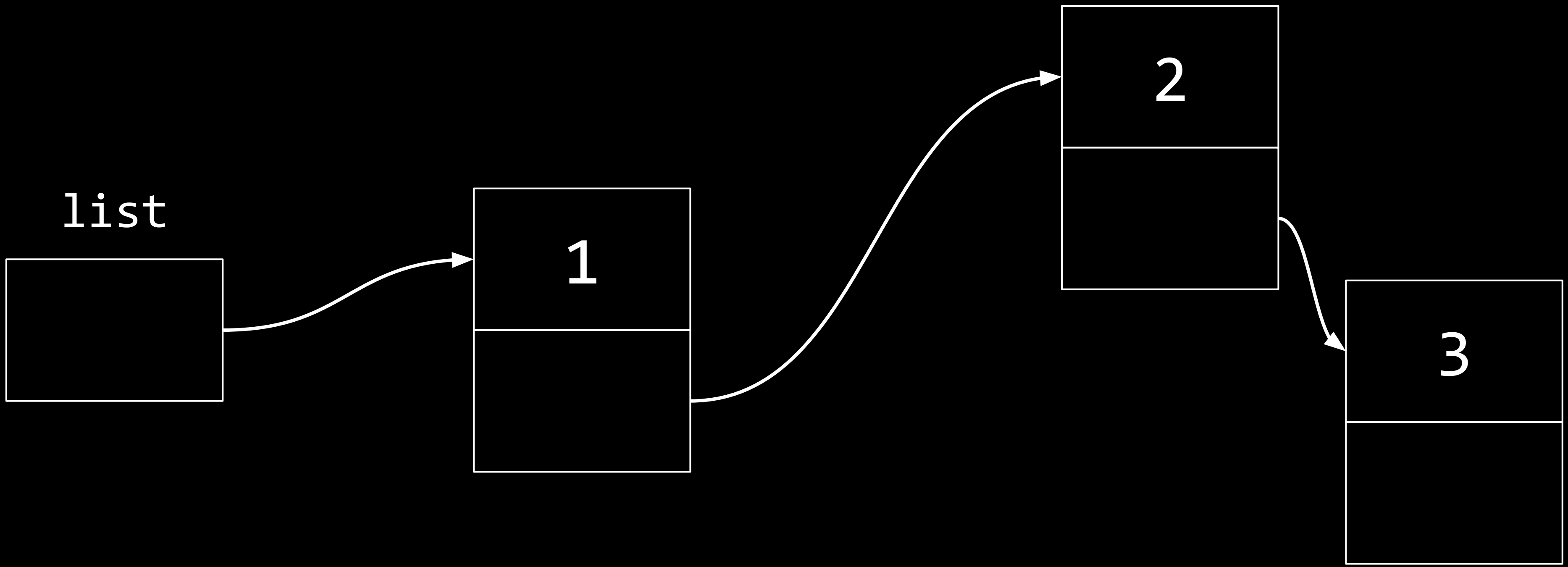
$O(\log n)$

$O(1)$

list


list
1


list
1
2


list
1
2
3

$O(n^2)$

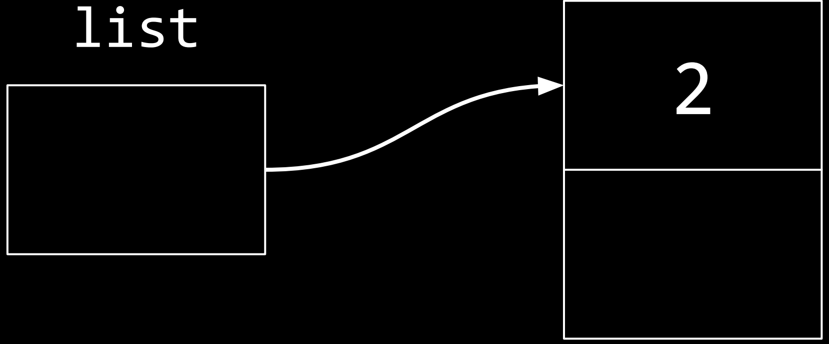
$O(n \log n)$

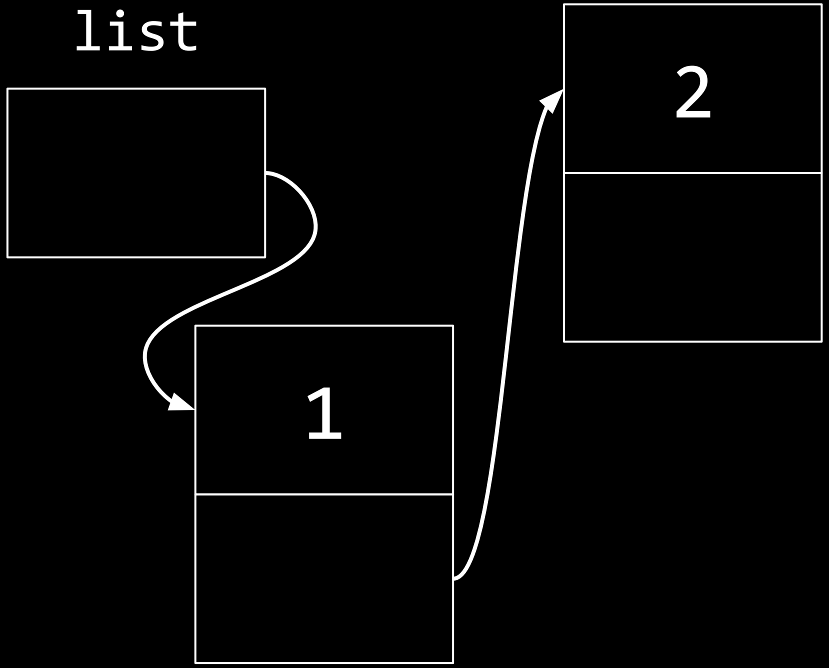
$O(n)$

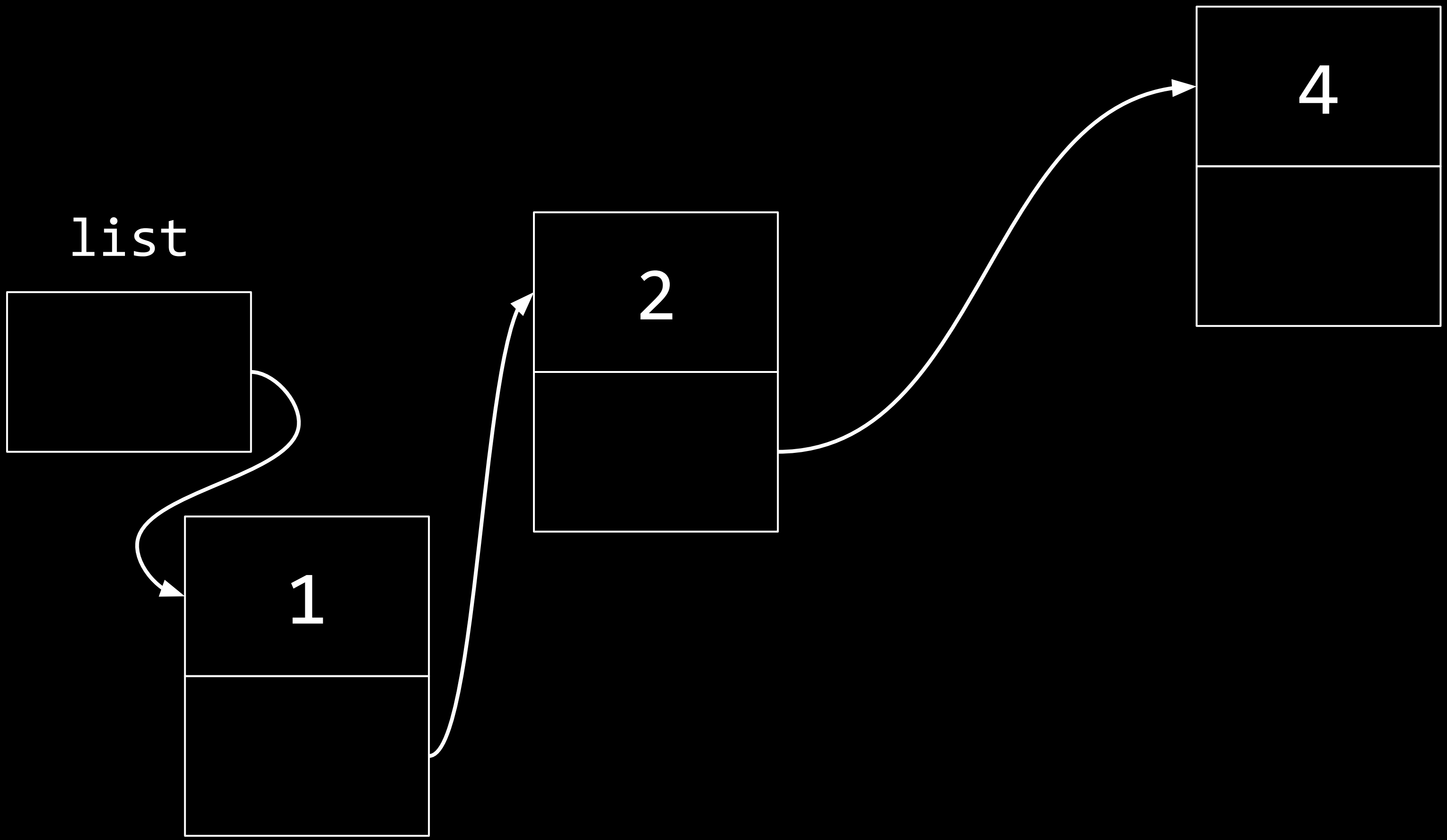
$O(\log n)$

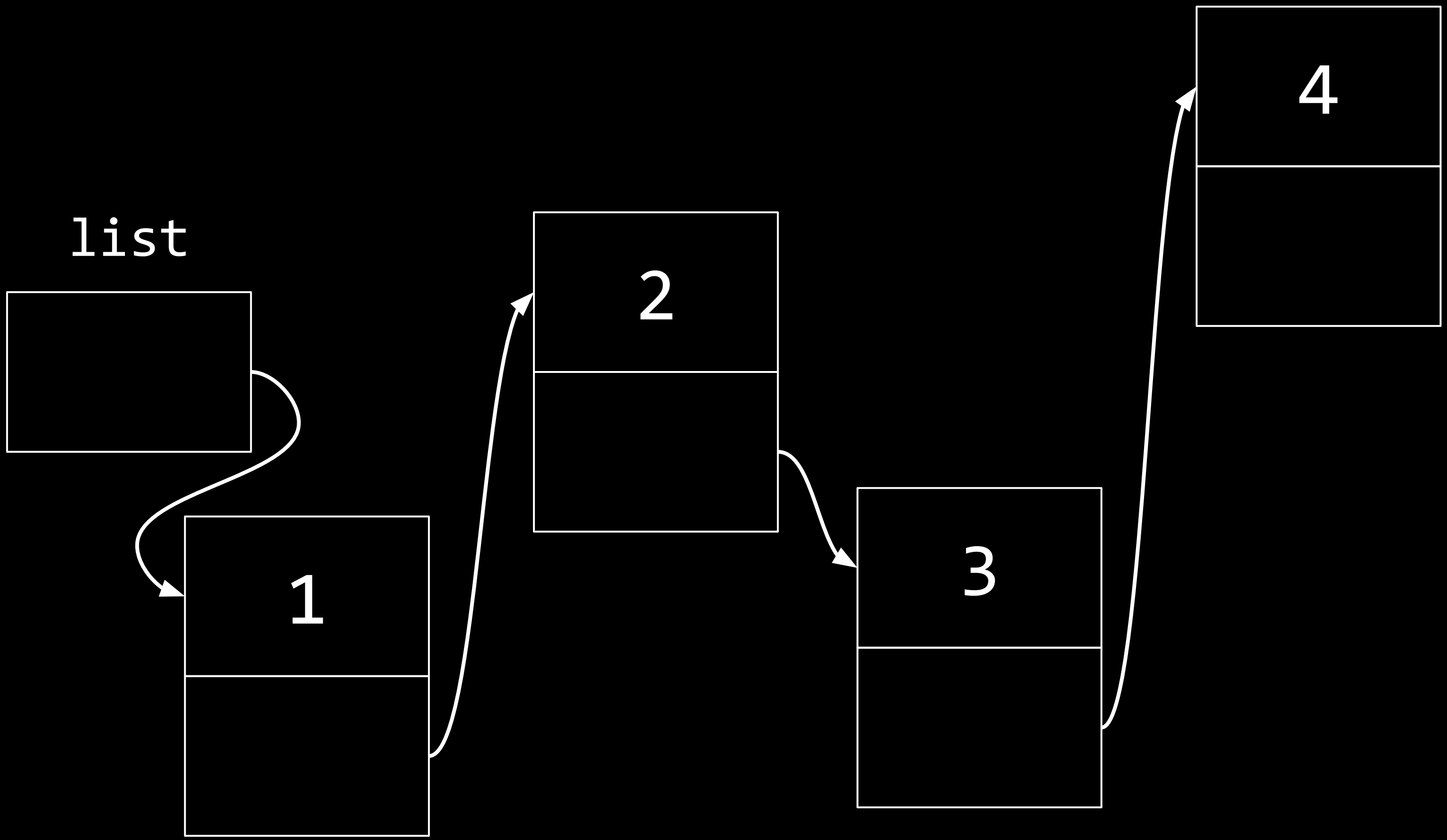
$O(1)$

list


list
2


list
1
2


list
1
2
4


list
1
2
3
4

$O(n^2)$

$O(n \log n)$

$O(n)$

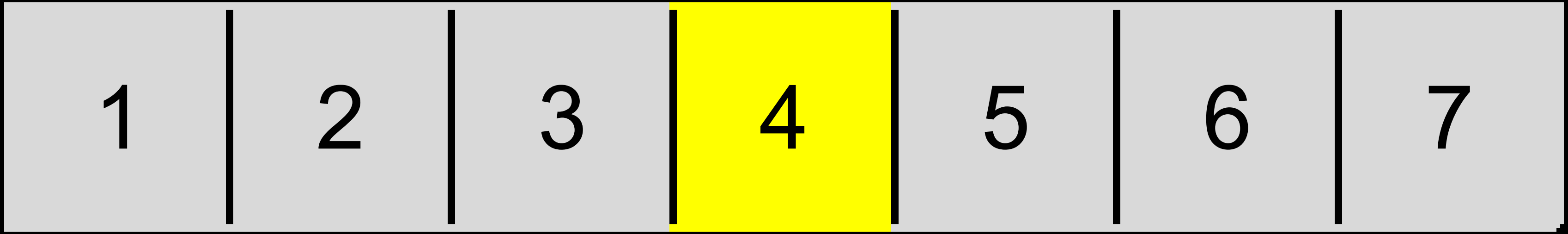
$O(\log n)$

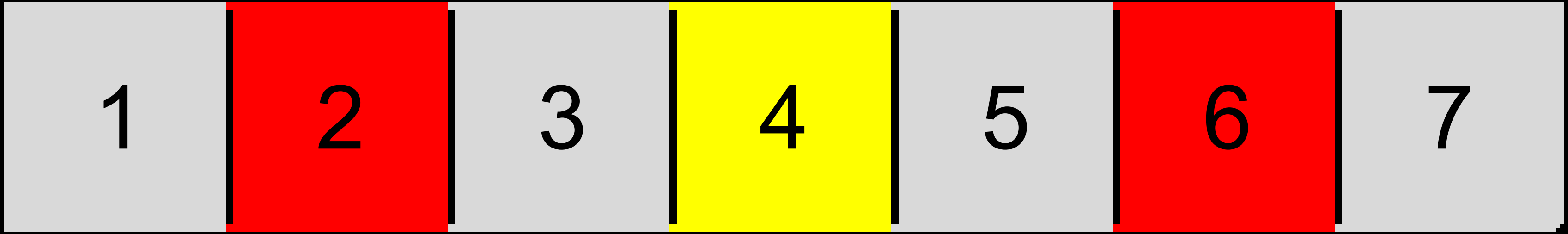
$O(1)$

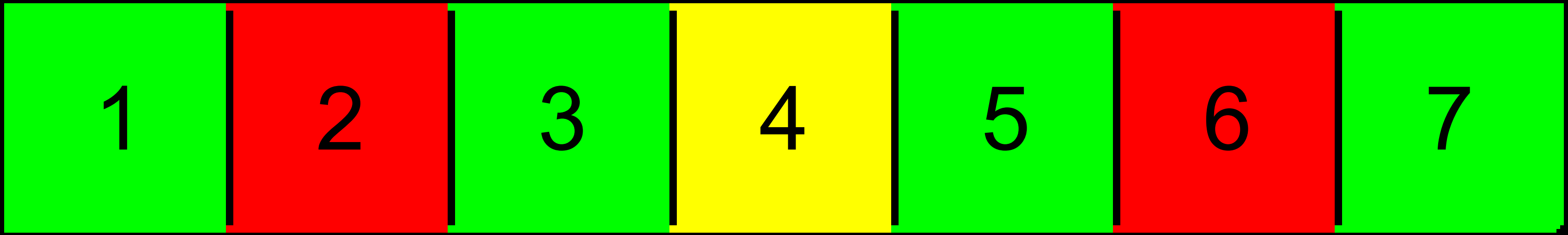
trees

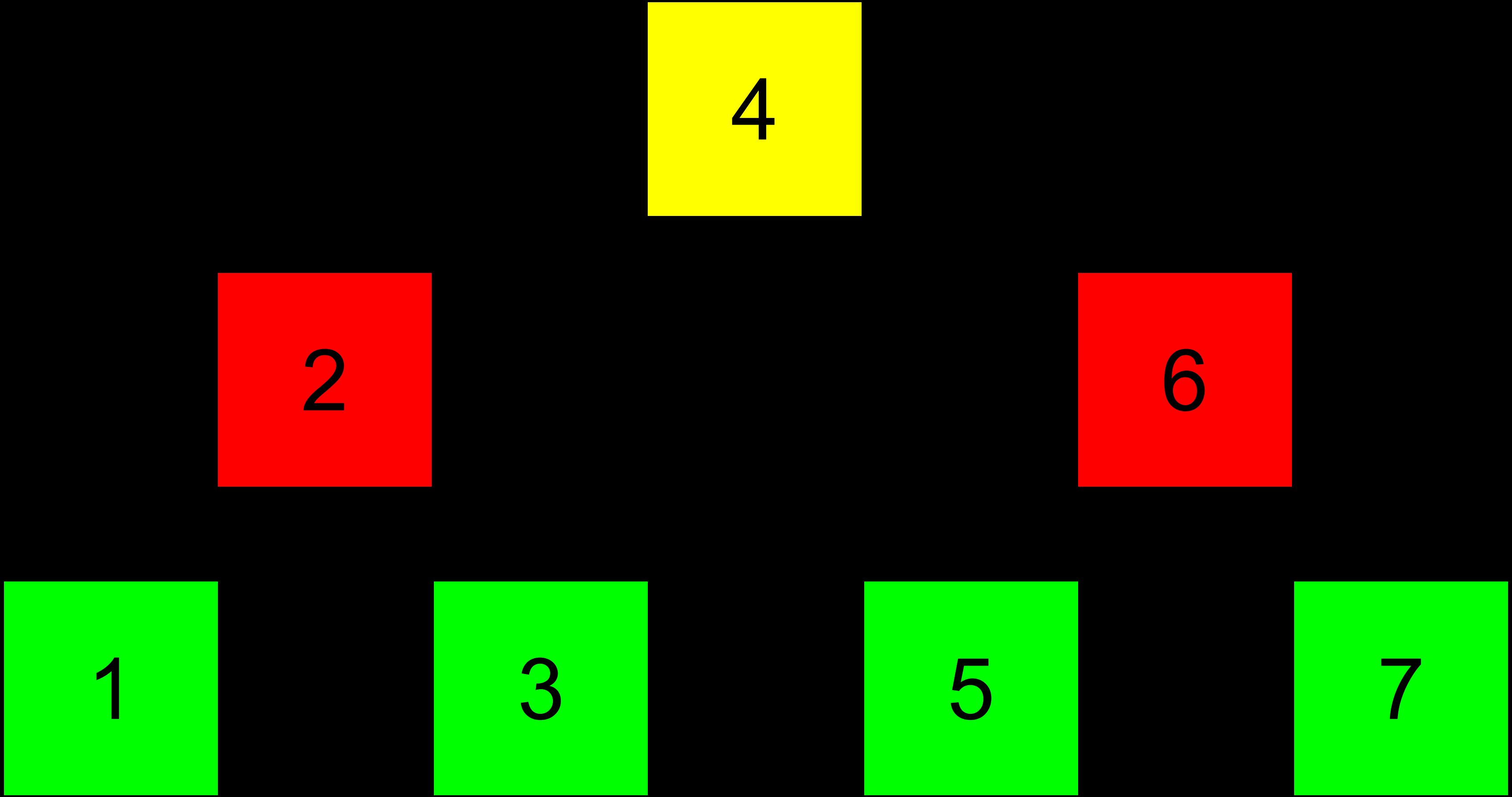
binary search trees

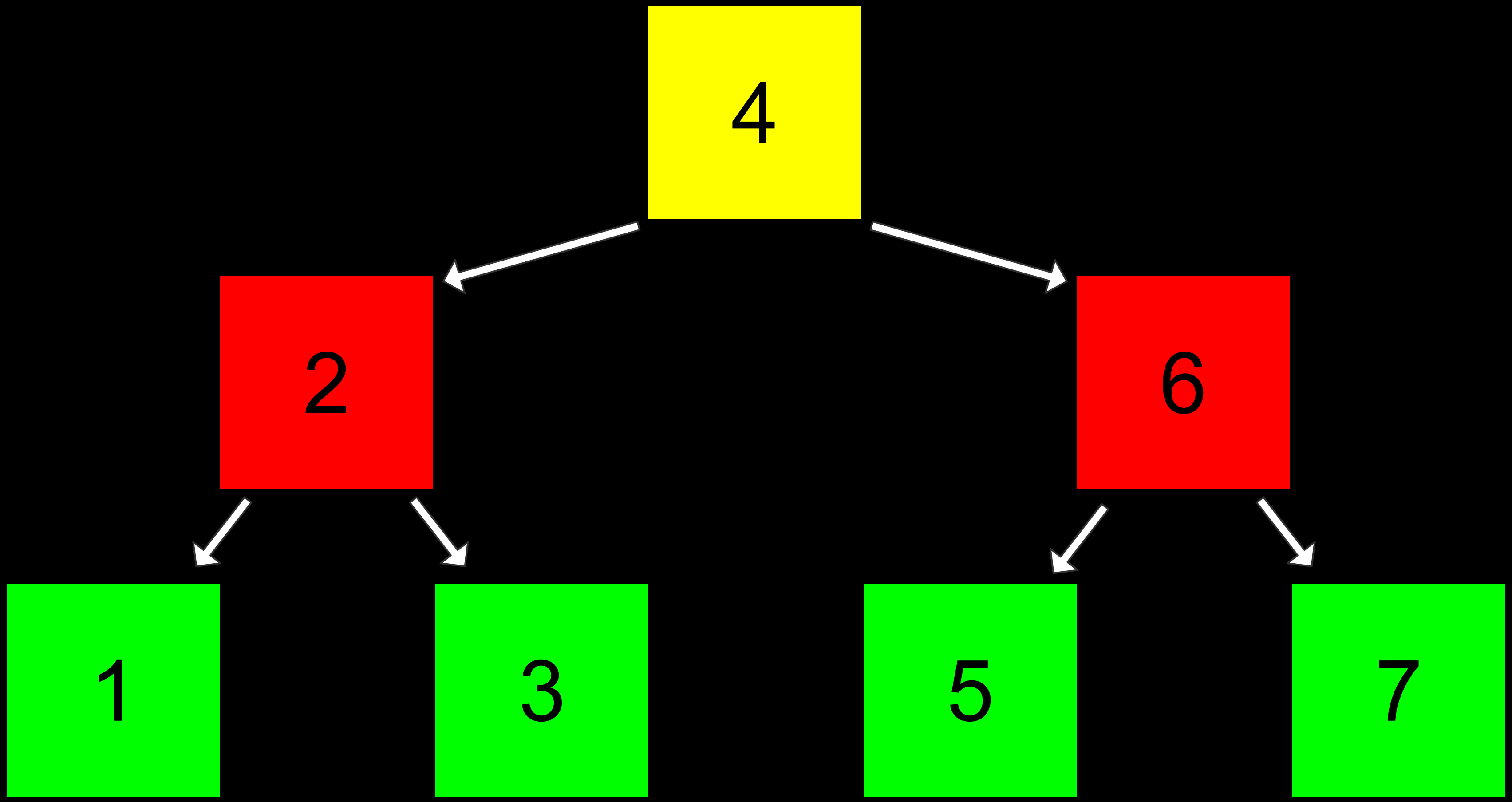
1	2	3	4	5	6	7


1
2
3
4
5
6
7


1
2
3
4
5
6
7


1
2
3
4
5
6
7


4
2
6
1
3
5
7

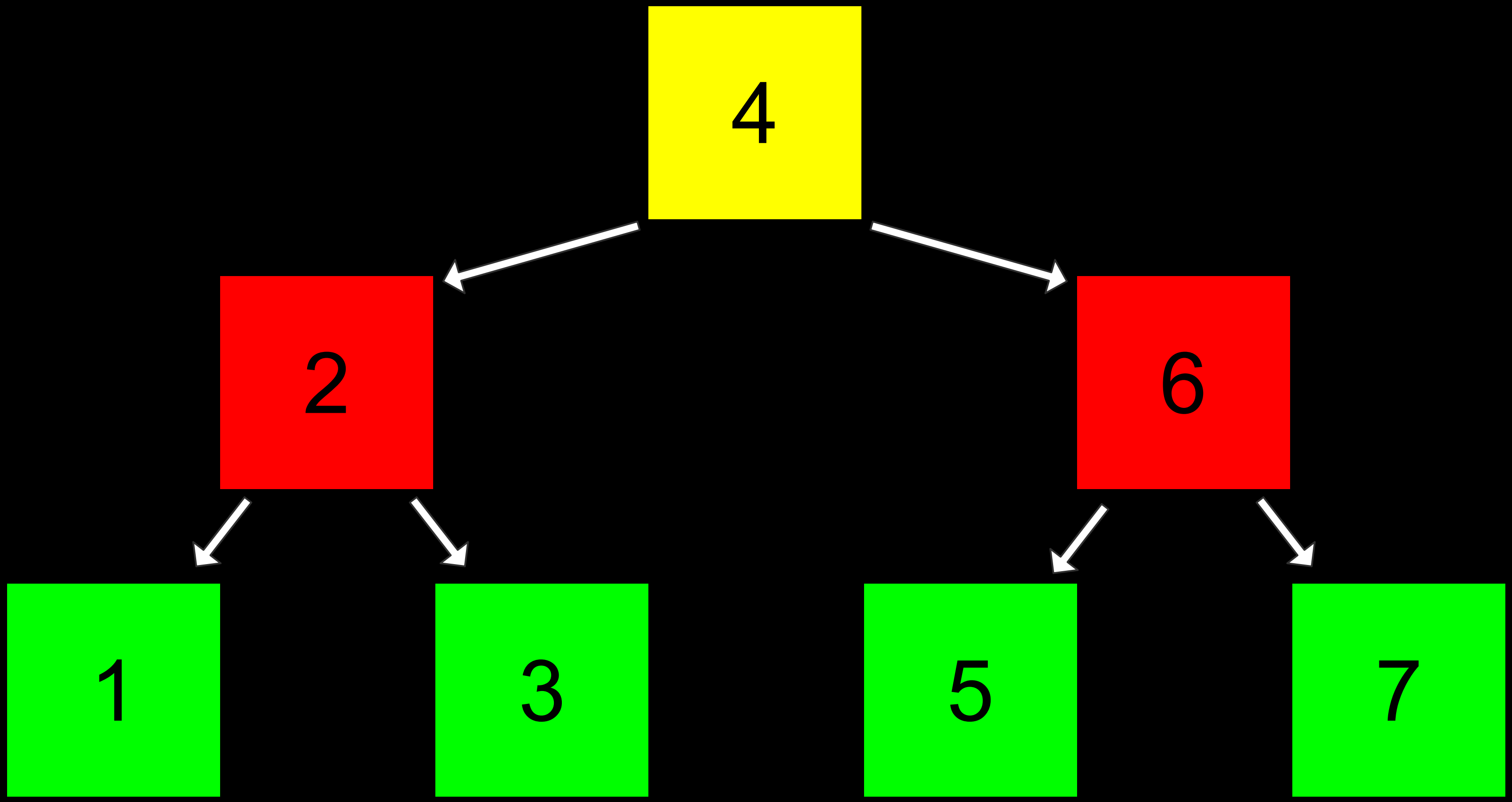

4
2
6
1
3
5
7

```
typedef struct node
{
    int number;
    struct node *next;
} node;
```

```
typedef struct node
{
    int number;

} node;
```

```
typedef struct node
{
    int number;


} node;
```

```
typedef struct node
{
    int number;
    struct node *left;
    struct node *right;
} node;
```


4
2
6
1
3
5
7

```
bool search(node *tree, int number)
{
















}
```

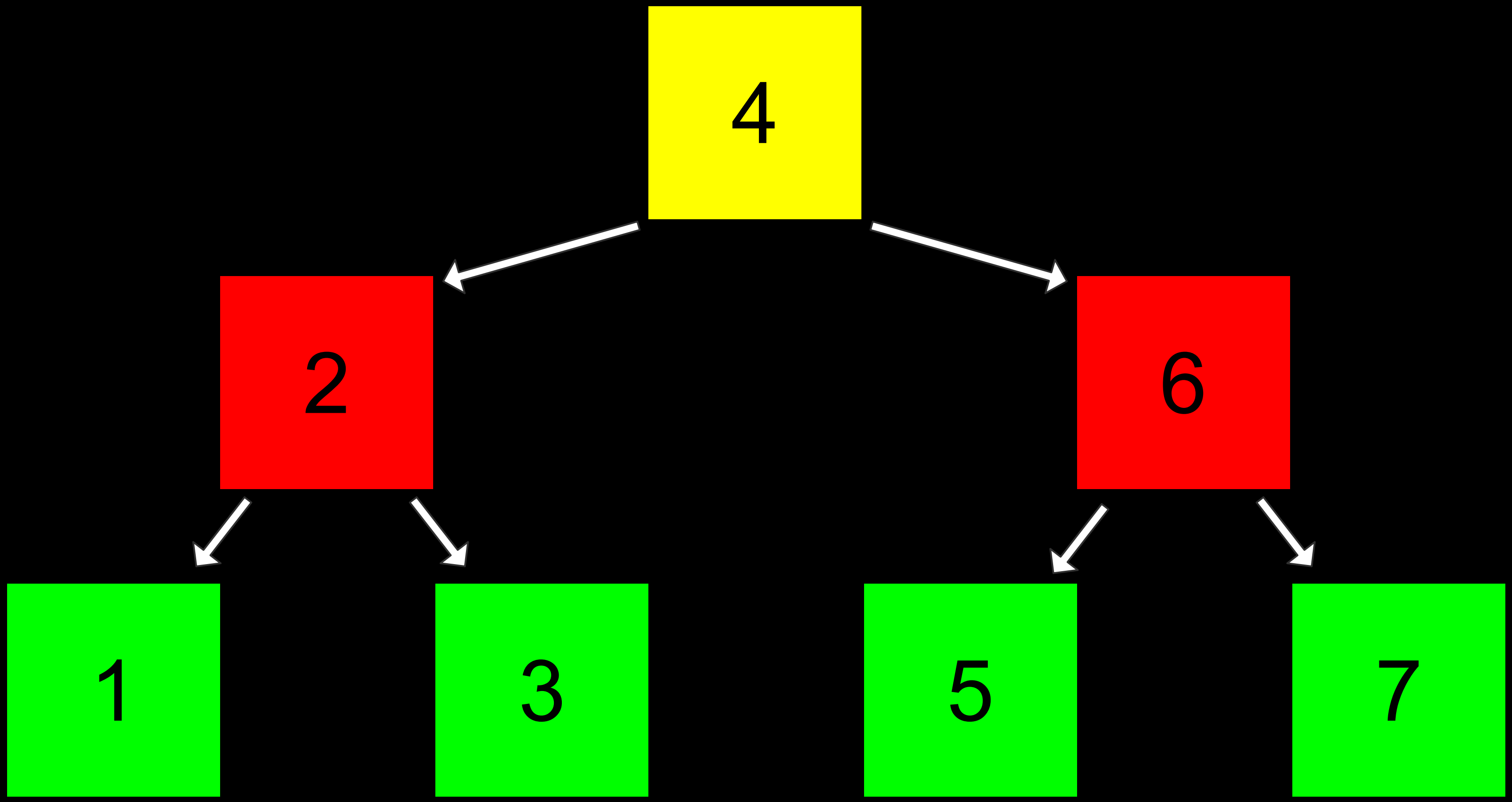
```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }











}
```

```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }








}
```

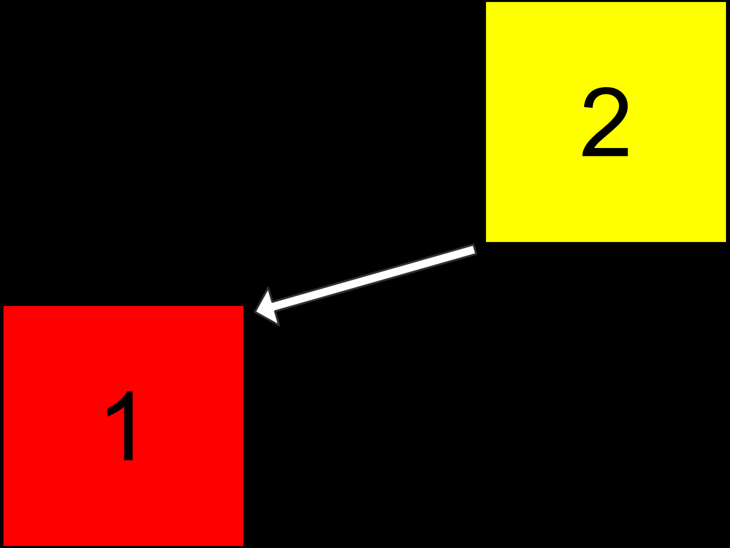
```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }
    else if (number > tree->number)
    {
        return search(tree->right, number);
    }




}
```

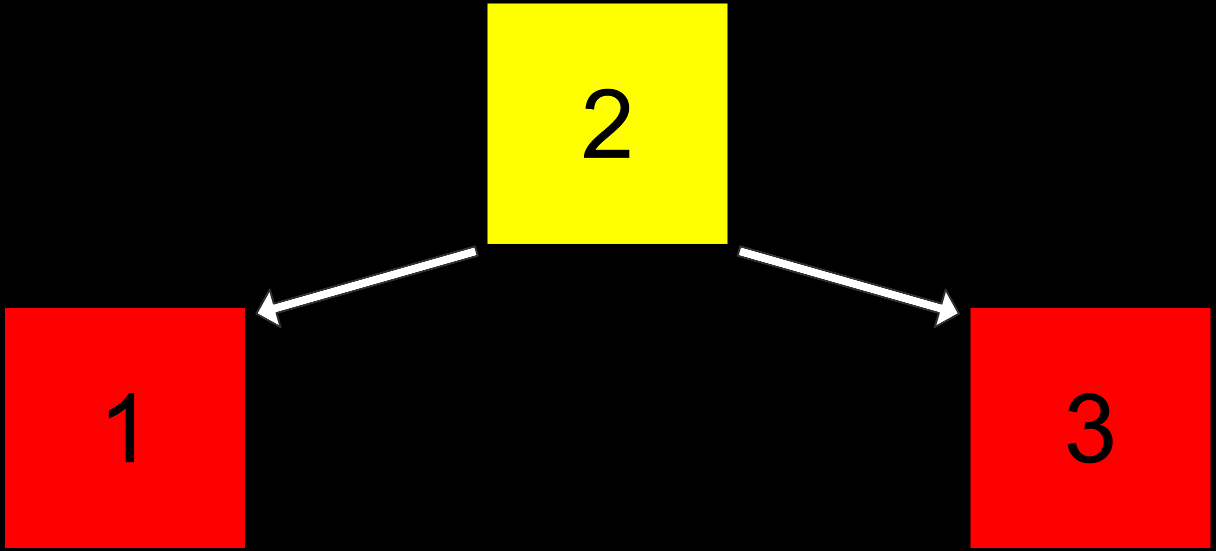
```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }
    else if (number > tree->number)
    {
        return search(tree->right, number);
    }
    else if (number == tree->number)
    {
        return true;
    }
}
```

```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }
    else if (number > tree->number)
    {
        return search(tree->right, number);
    }
    else
    {
        return true;
    }
}
```

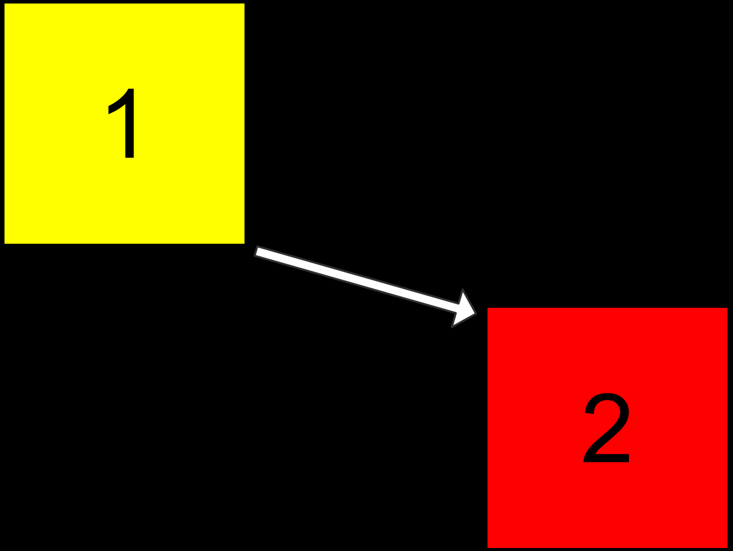

4
2
6
1
3
5
7

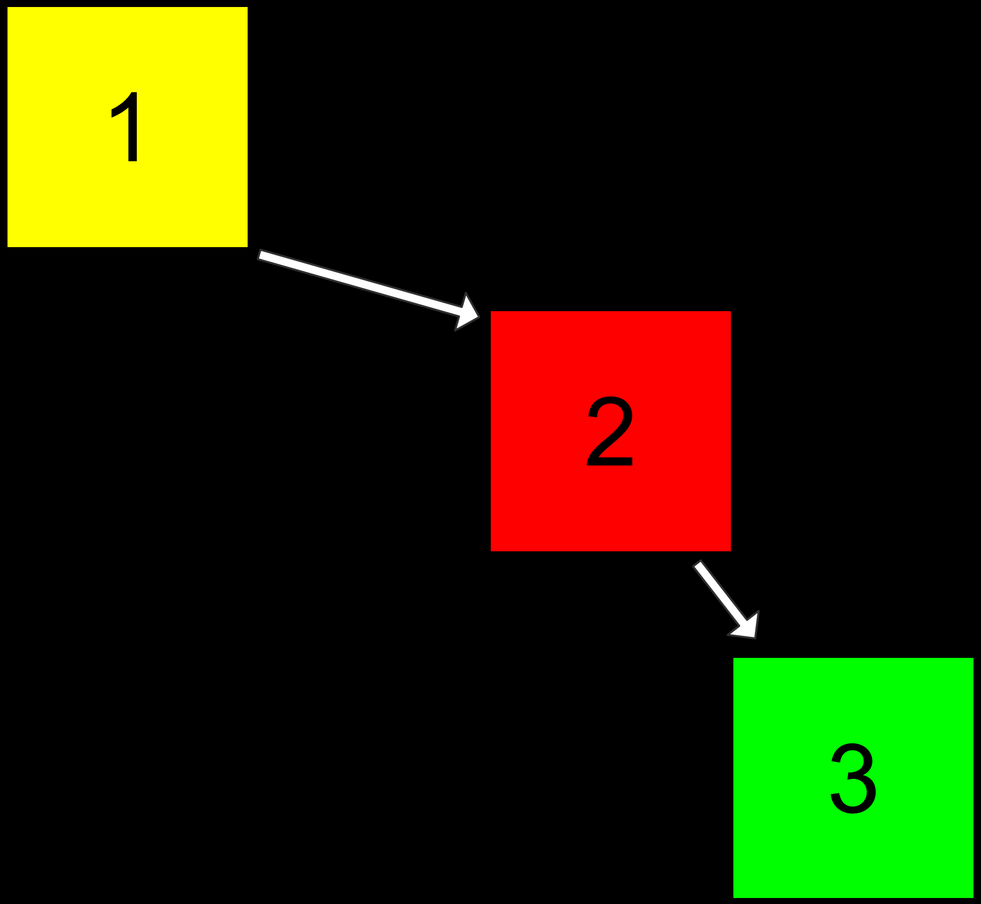
2


2
1


2
1
3

1


1
2


1
2
3

$O(n^2)$

$O(n \log n)$

$O(n)$

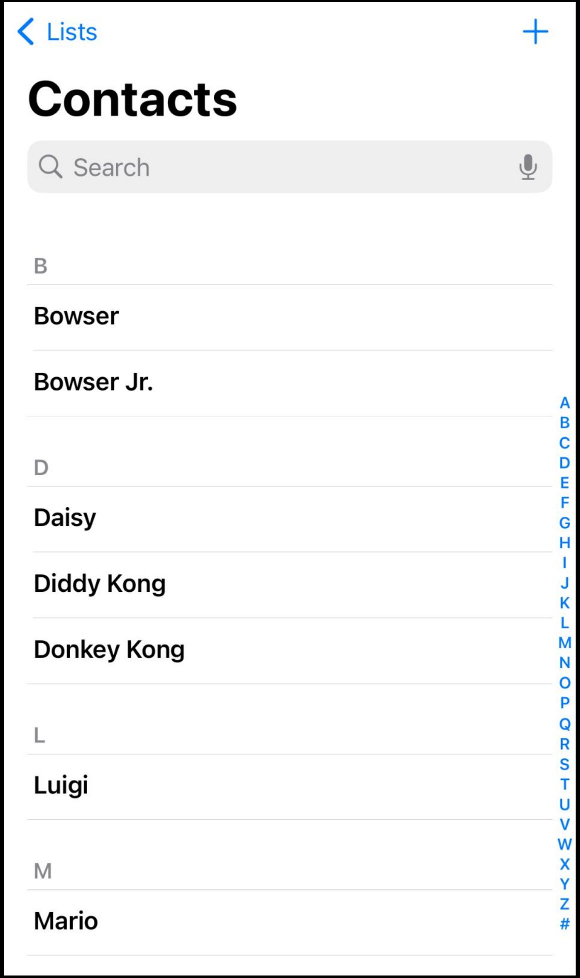
$O(\log n)$

$O(1)$

dictionaries

word	definition

key	value



B

Bowser

Bowser Jr.

D

Daisy

Diddy Kong

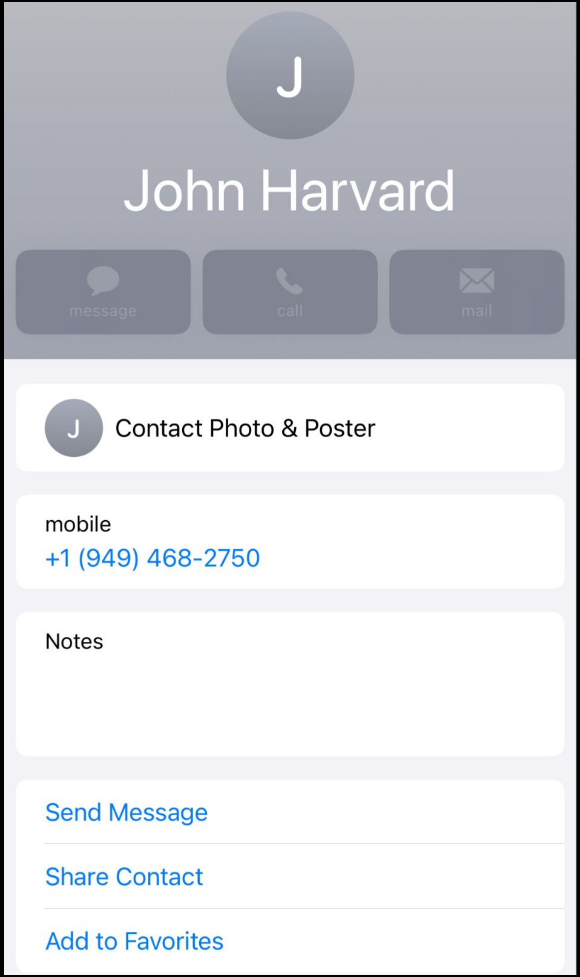
Donkey Kong

L

Luigi

M

Mario



John Harvard

Contact Photo & Poster

mobile
+1 (949) 468-2750

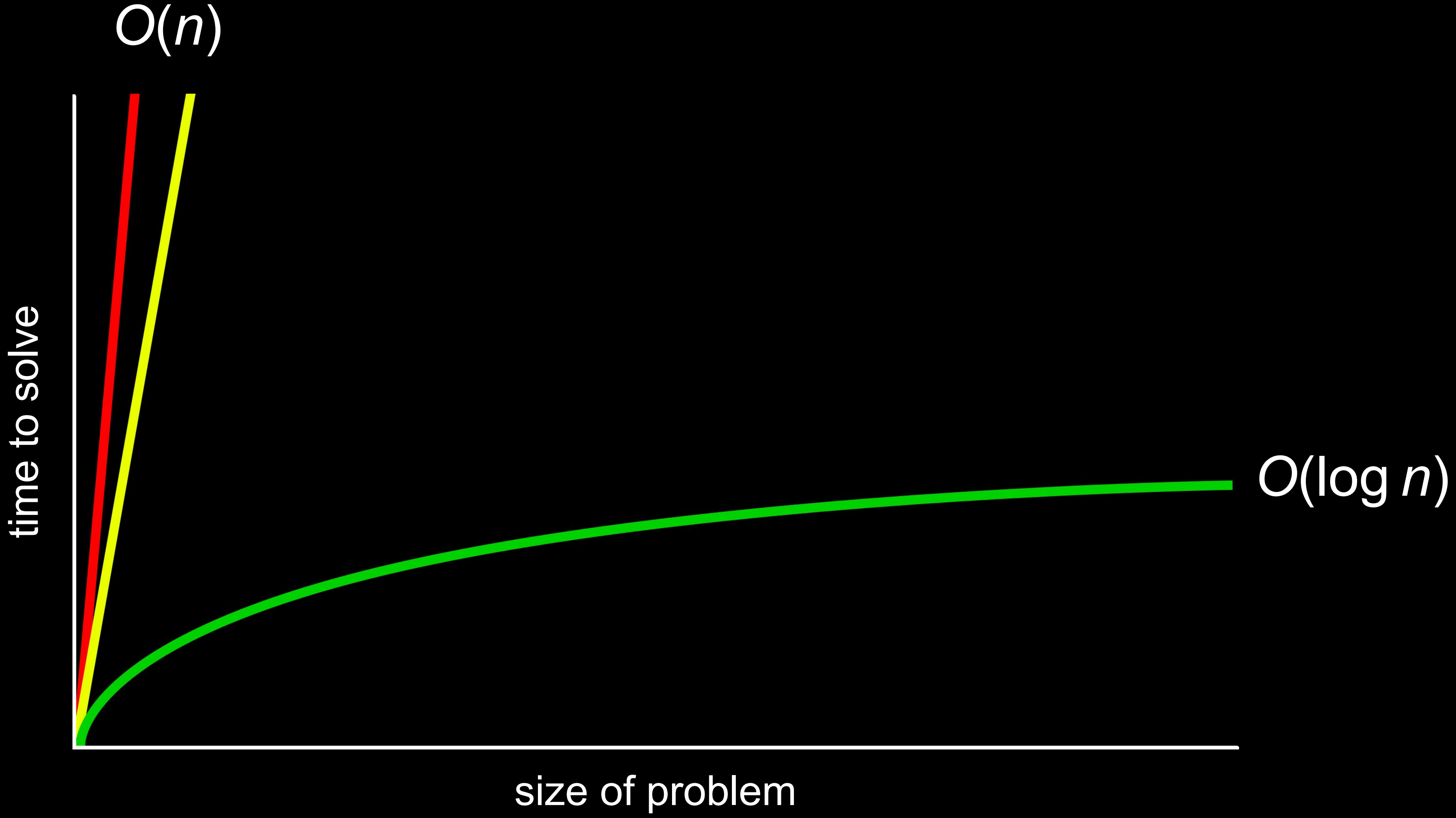
Notes

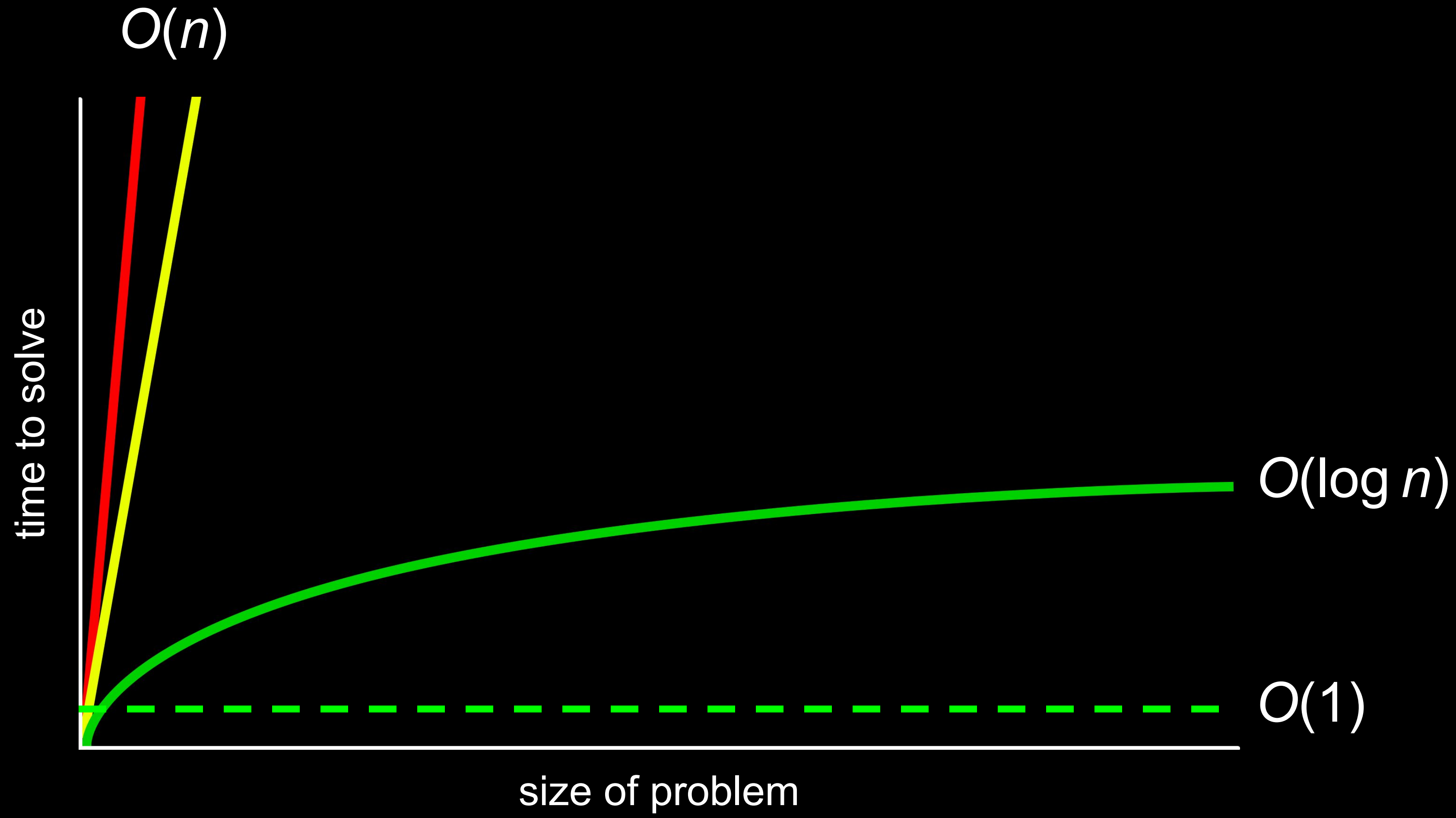
Send Message

Share Contact

Add to Favorites

name	number

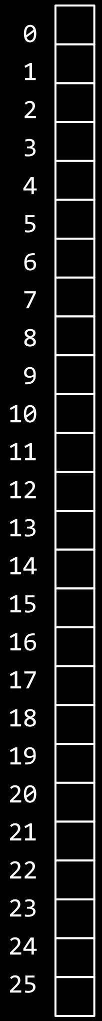

$O(n)$
$O(\log n)$
time to solve
size of problem


$O(n)$
time to solve
$O(\log n)$
$O(1)$
size of problem

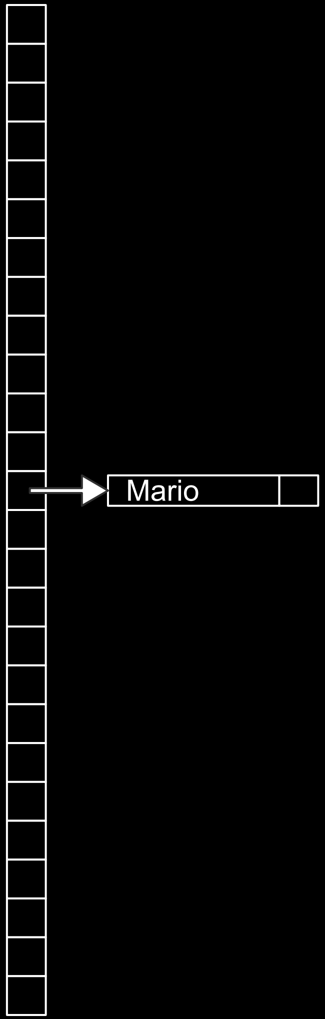
hashing

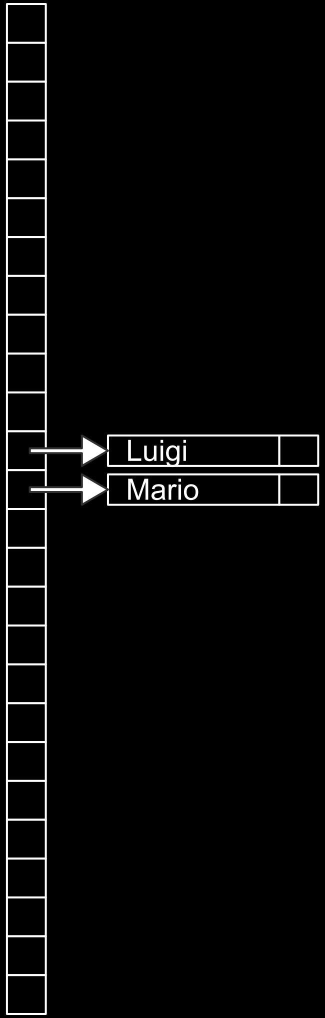
hash function

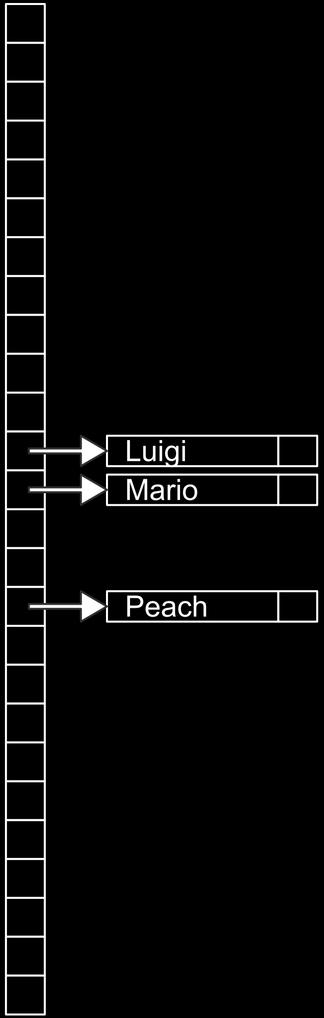
hash tables

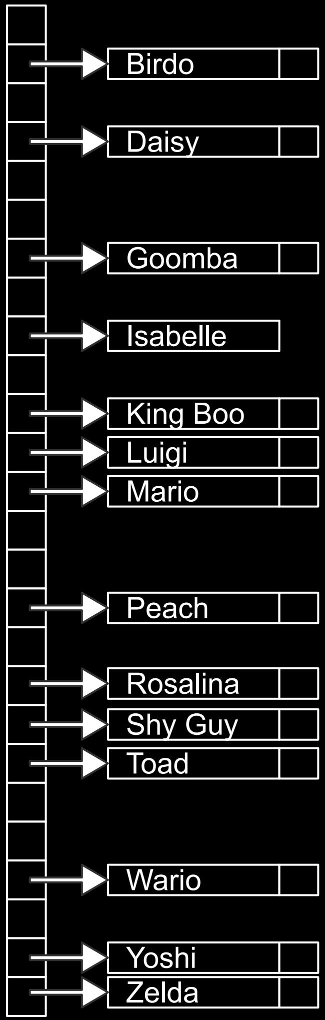

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

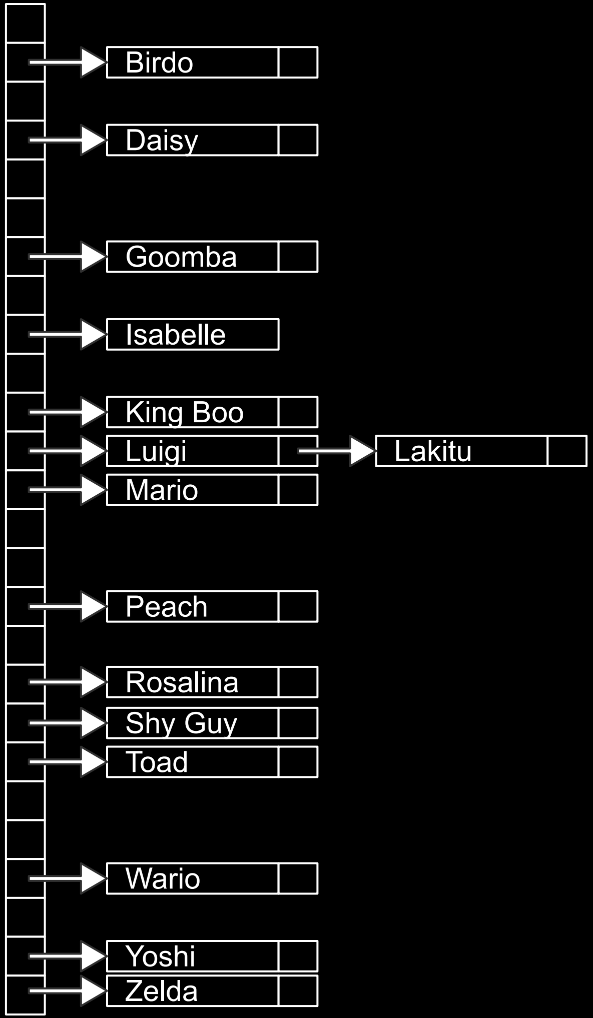
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	

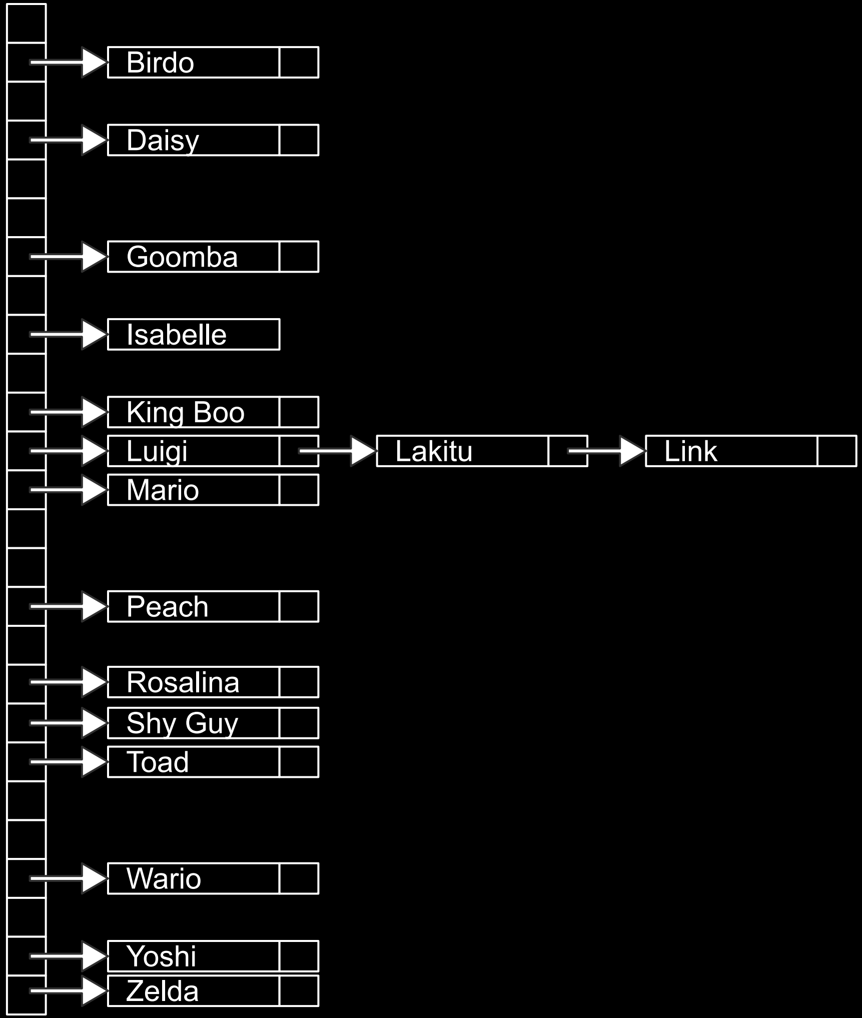

Mario

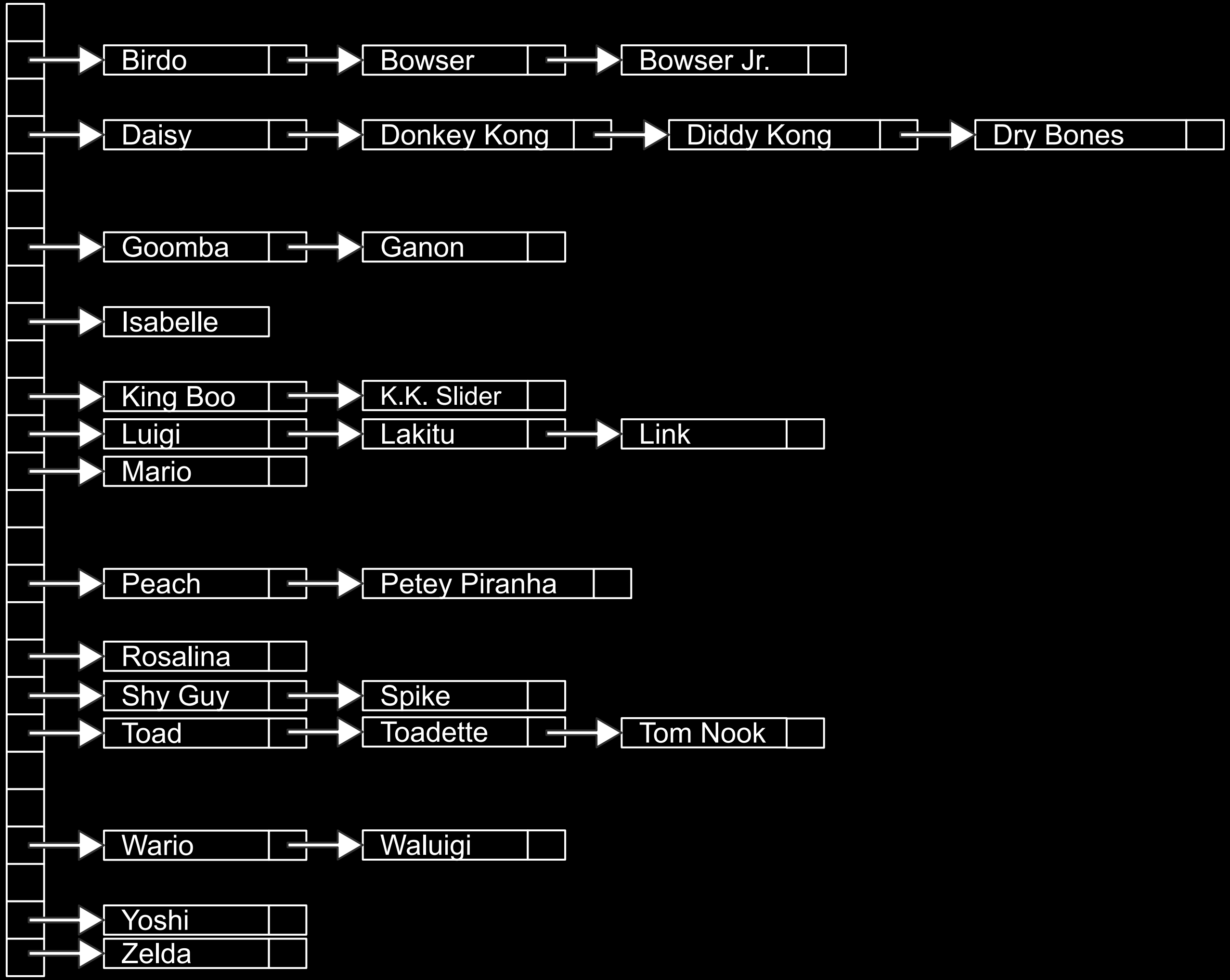

Luigi
Mario


Luigi
Mario
Peach


Birdo
Daisy
Goomba
Isabelle
King Boo
Luigi
Mario
Peach
Rosalina
Shy Guy
Toad
Wario
Yoshi
Zelda


Birdo
Daisy
Goomba
Isabelle
King Boo
Luigi
Lakitu
Mario
Peach
Rosalina
Shy Guy
Toad
Wario
Yoshi
Zelda


Birdo
Daisy
Goomba
Isabelle
King Boo
Luigi
Lakitu
Link
Mario
Peach
Rosalina
Shy Guy
Toad
Wario
Yoshi
Zelda


Birdo
Bowser
Bowser Jr.
Daisy
Donkey Kong
Diddy Kong
Dry Bones
Goomba
Ganon
Isabelle
King Boo
K.K. Slider
Luigi
Lakitu
Link
Mario
Peach
Petey Piranha
Rosalina
Shy Guy
Spike
Toad
Toadette
Tom Nook
Wario
Waluigi
Yoshi
Zelda

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$

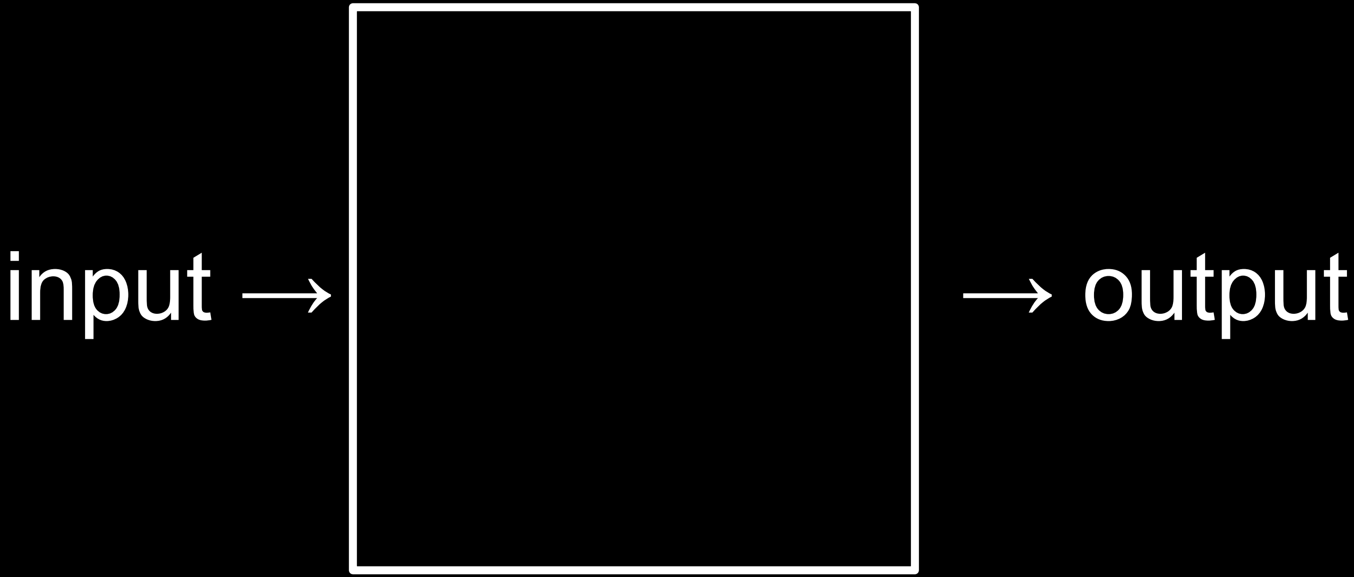
$O(\log n)$

$O(1)$

```
typedef struct
{
    char *name;
    char *number;
} person;
```

```
typedef struct node
{
    char *name;
    char *number;
    struct node *next;
} node;
```

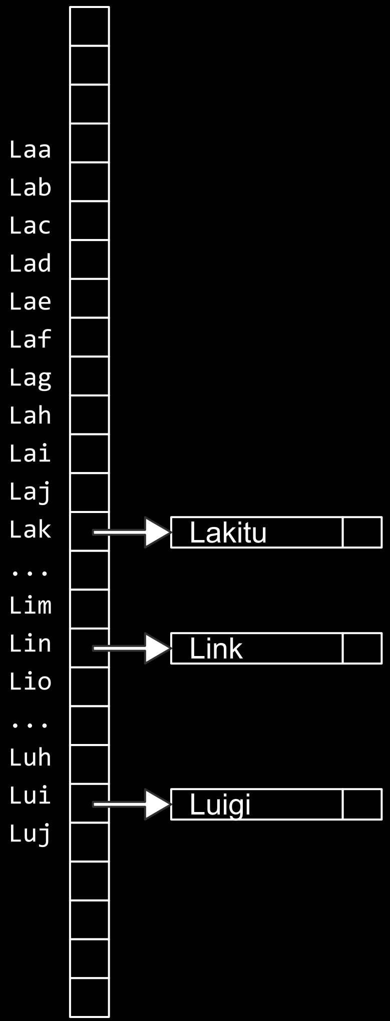
```
node *table[26];
```


input
output

hash function

$$\begin{tikzcd}\text{Mario} \arrow[r] & \boxed{} \arrow[r] & 12\end{tikzcd}$$

$$\begin{tikzcd}\text{Luigi} \arrow[r] & \boxed{} \arrow[r] & 11\end{tikzcd}$$


Laa
Lab
Lac
Lad
Lae
Laf
Lag
Lah
Lai
Laj
Lak
...
Lim
Lin
Lio
...
Luh
Lui
Luj
Lakitu
Link
Luigi

```c
#include <ctype.h>

int hash(char *word)
{
    return toupper(word[0]) - 'A';
}
```

```
#include <ctype.h>

int hash(const char *word)
{
    return toupper(word[0]) - 'A';
}
```

```
#include <ctype.h>

unsigned int hash(const char *word)
{
    return toupper(word[0]) - 'A';
}
```

$O(n)$

$O(n/k)$

$O(n)$

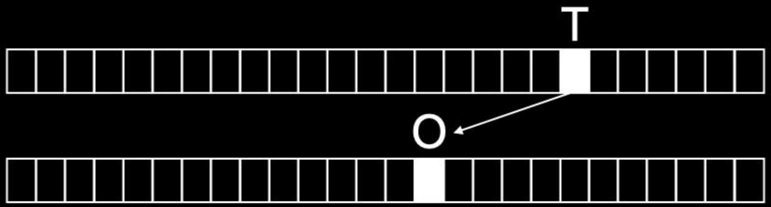
$O(1)$

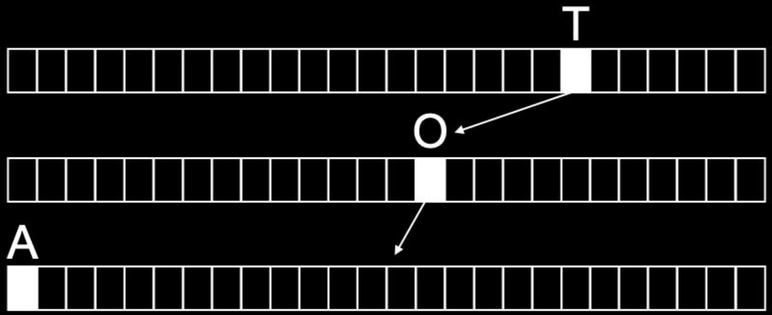
tries

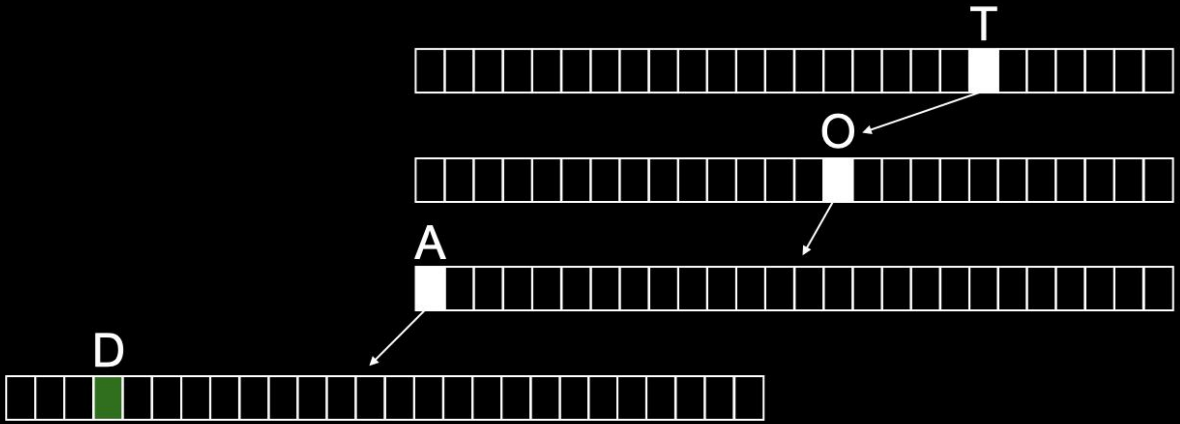
□□□□□□□□□□□□□□□□□□□□□□□□□□

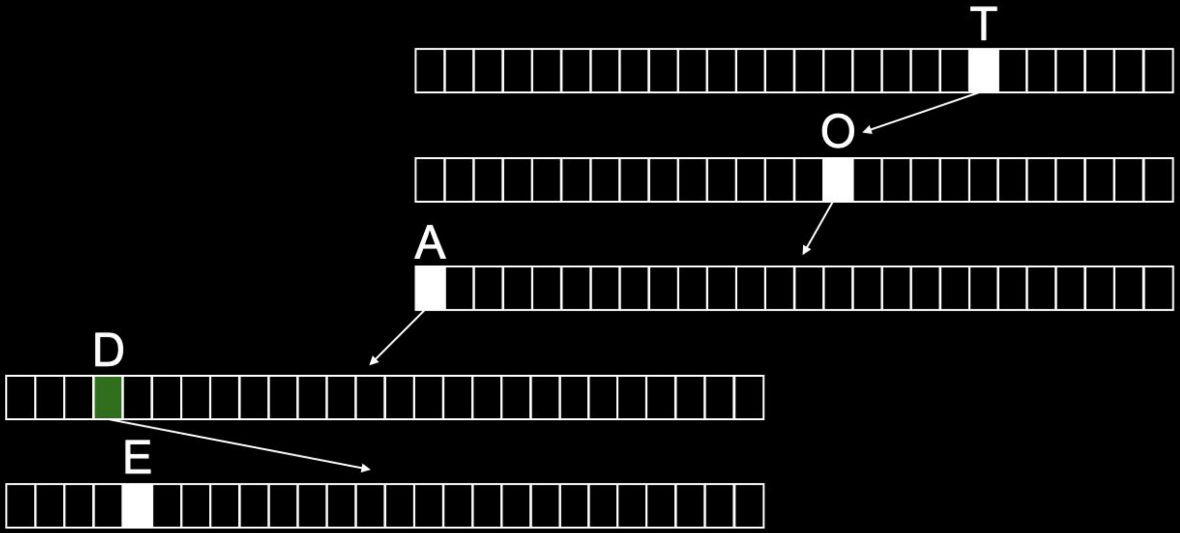
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

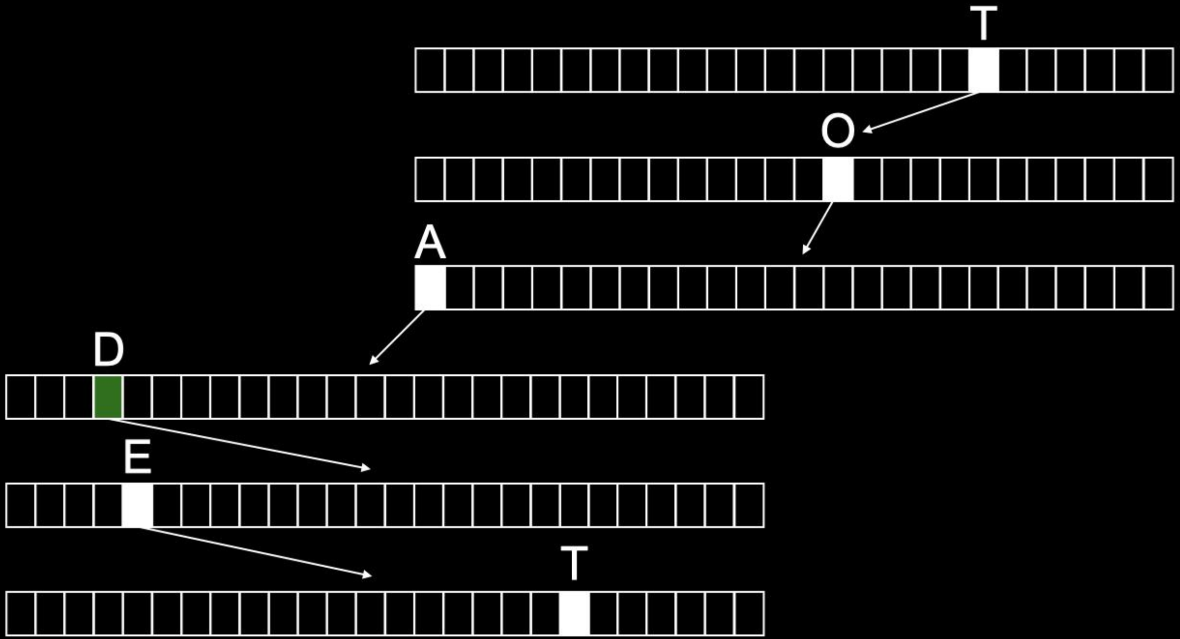
T

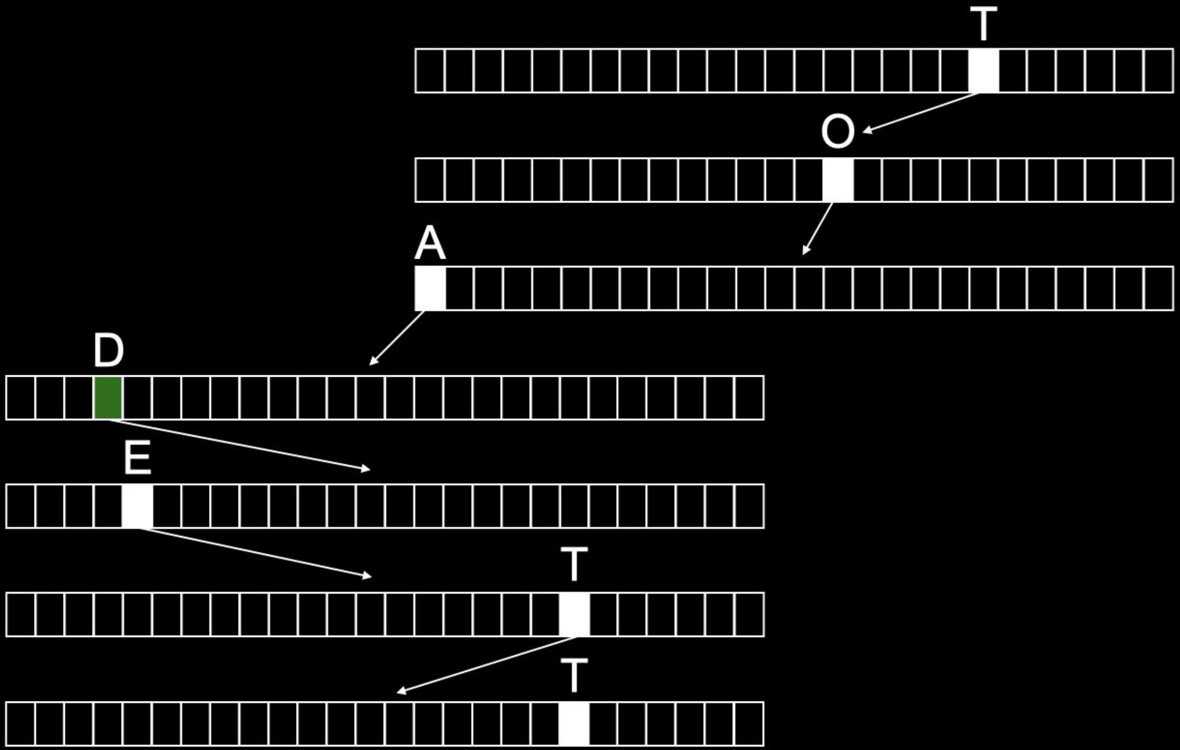

T
O

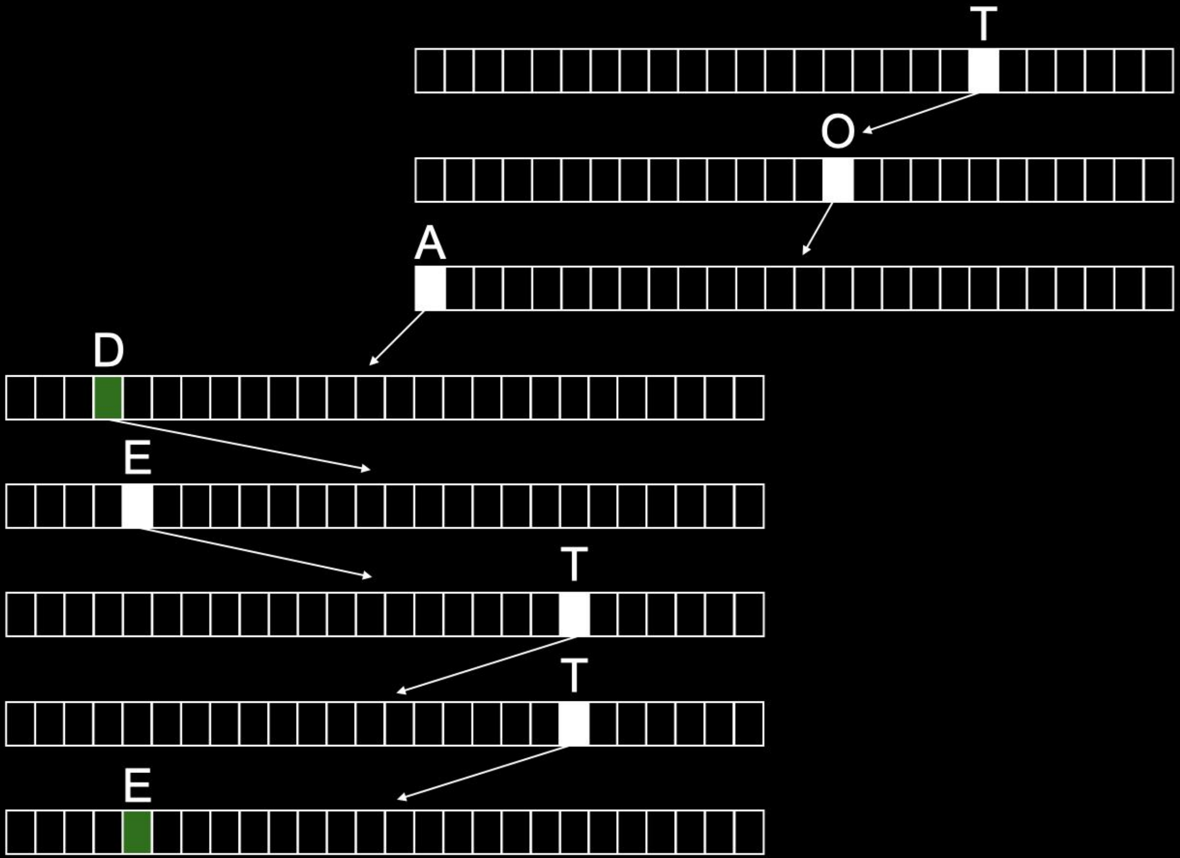

T
O
A

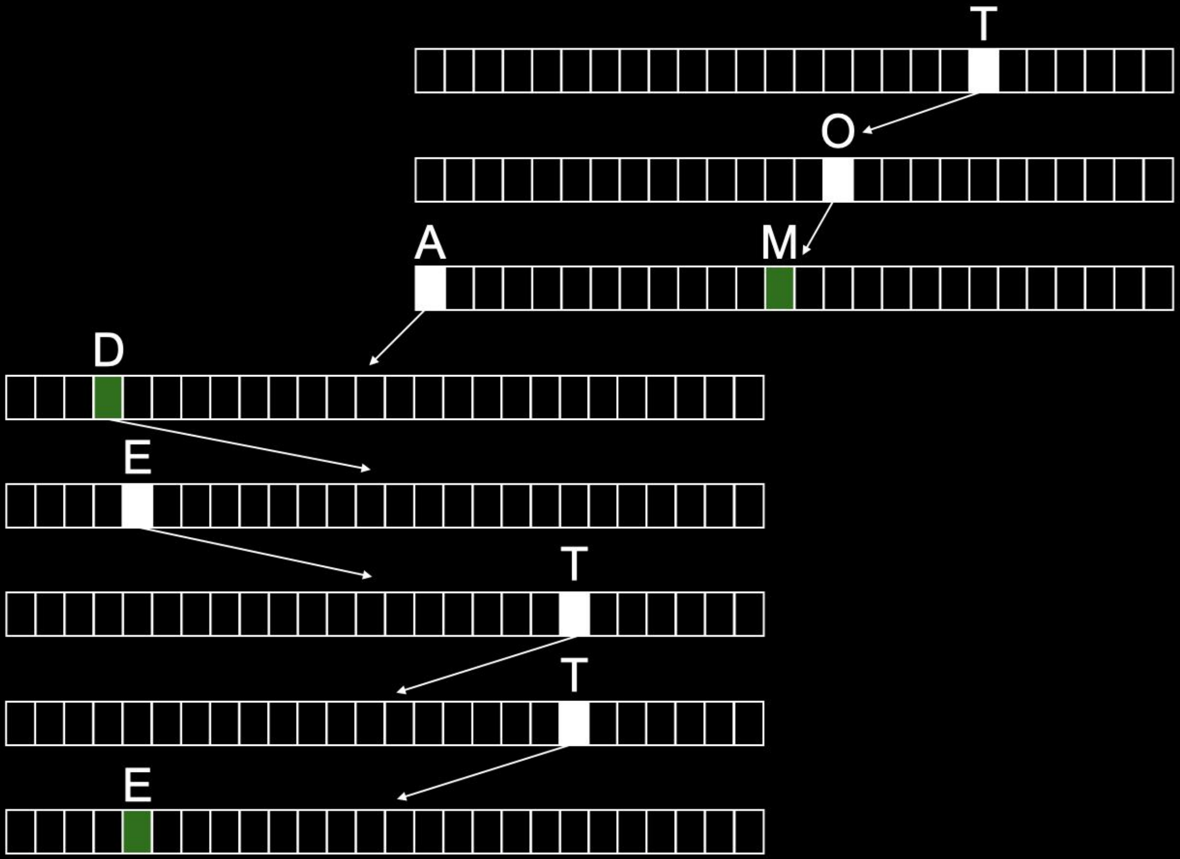

T
O
A
D


T
O
A
D
E


T
O
A
D
E
T


T
O
A
D
E
T
T


T
O
A
D
E
T
T
E


T
O
A
M
D
E
T
T
E

```
typedef struct node
{
    struct node *children[26];
    char *number;
} node;
```

```
node *trie;
```

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

PICK ME UP
A- E
F- J
K- N
O- Z

This is CS50