This is CS50

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;
}
```

```c
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;
}
```

```c
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;
}
```

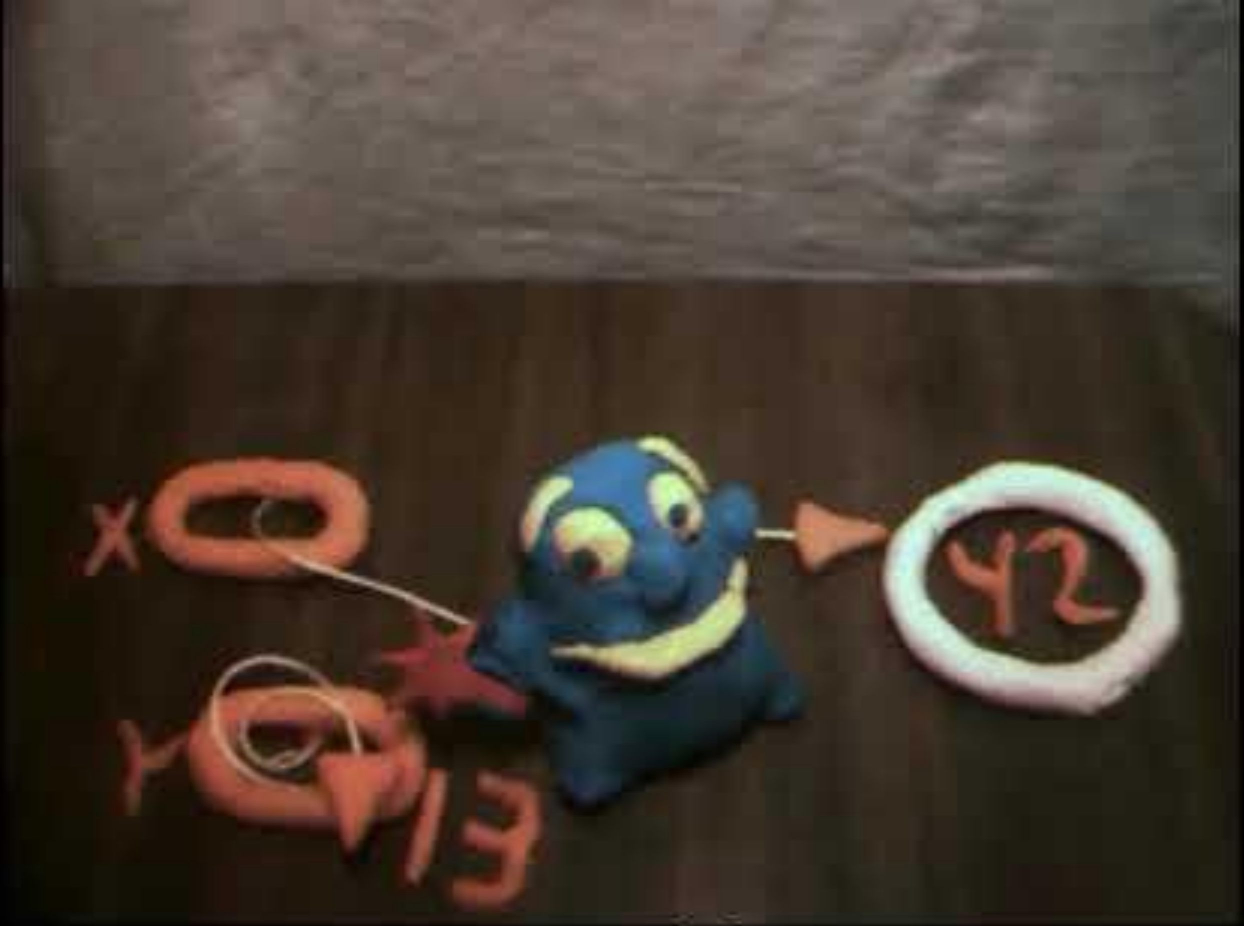
```c
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;




}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;

    y = x;

    *y = 13;
}
```

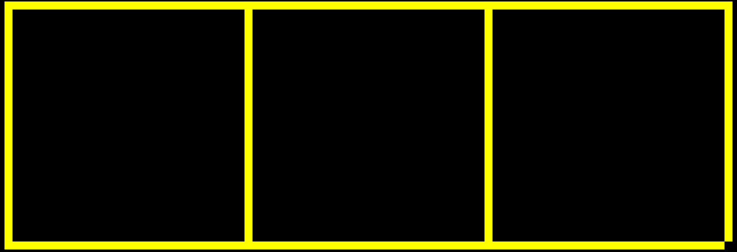
```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;

    y = x;

    *y = 13;
}
```

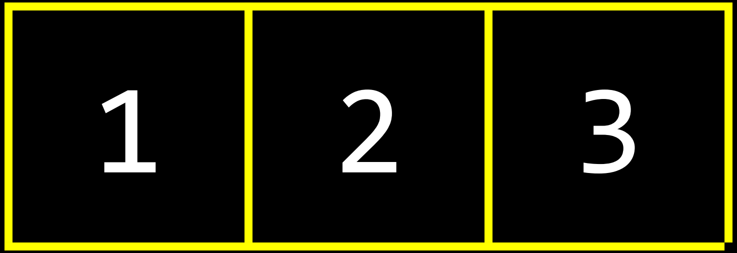
```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;

    y = x;

    *y = 13;
}
```

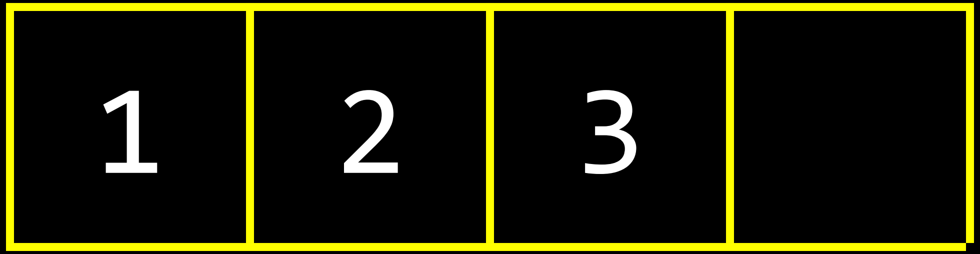


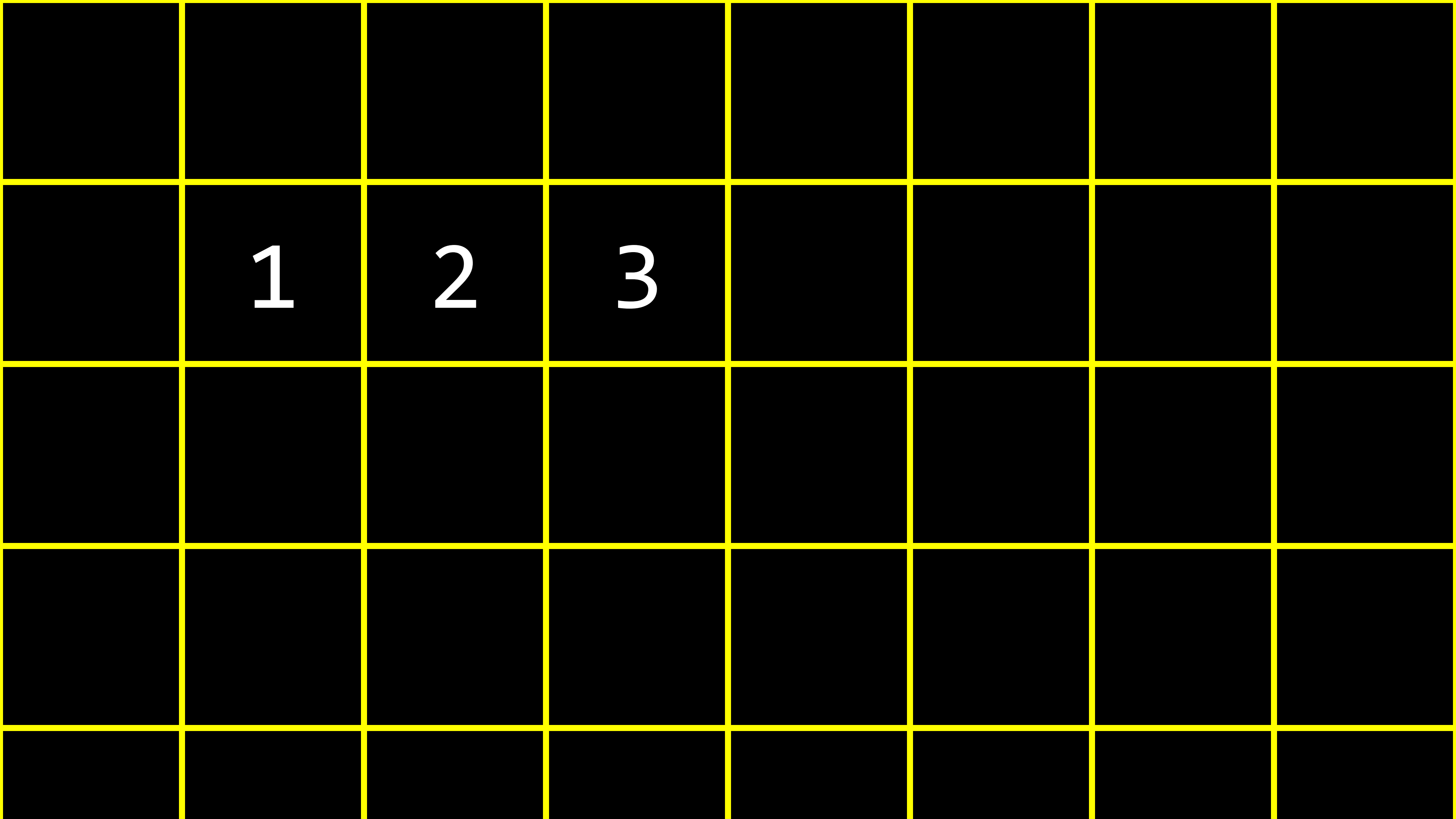
```
*y = 13;
```

arrays

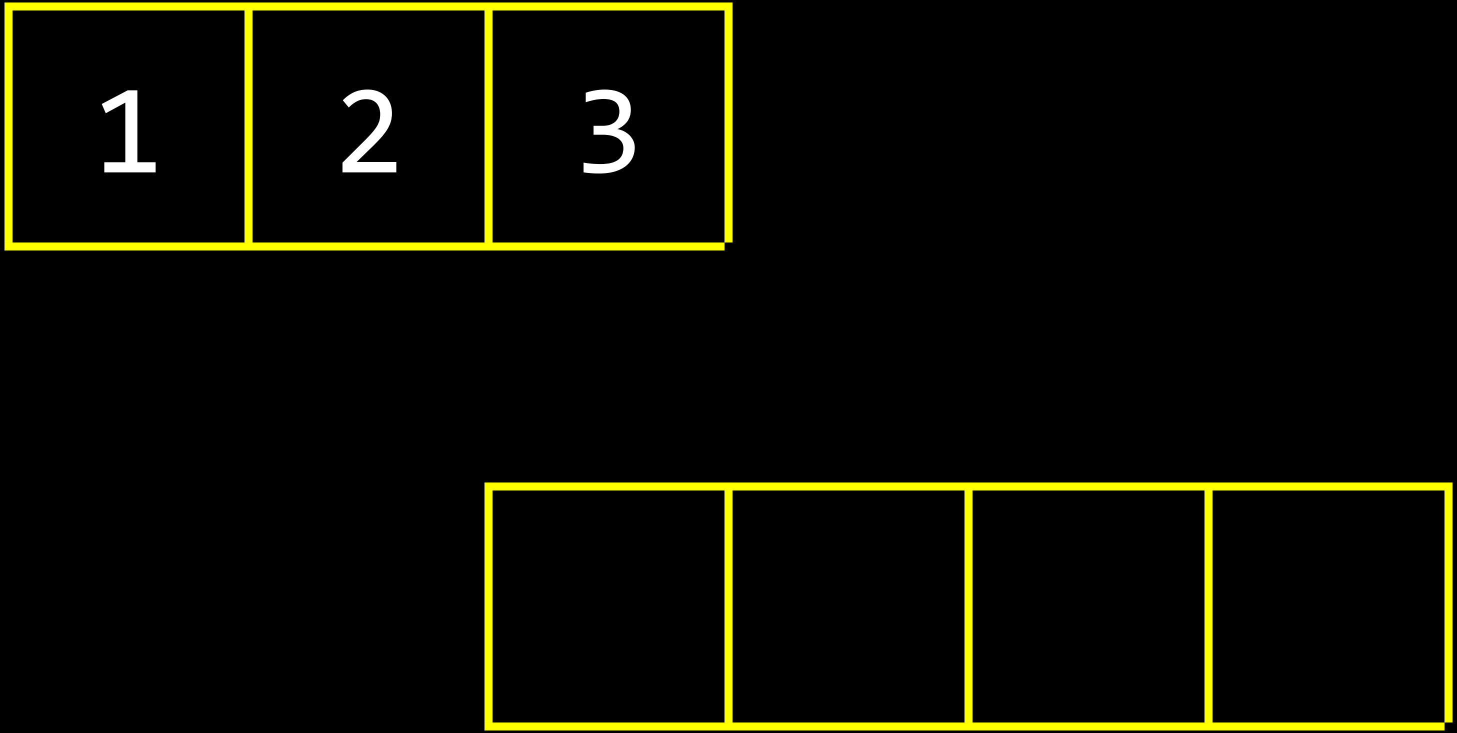


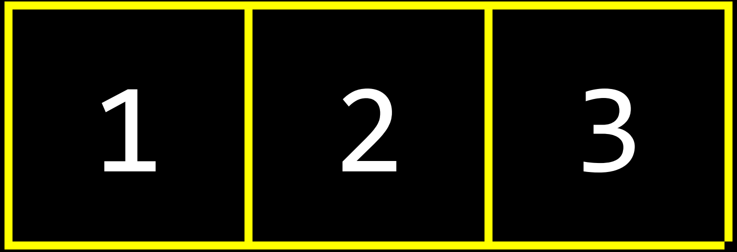
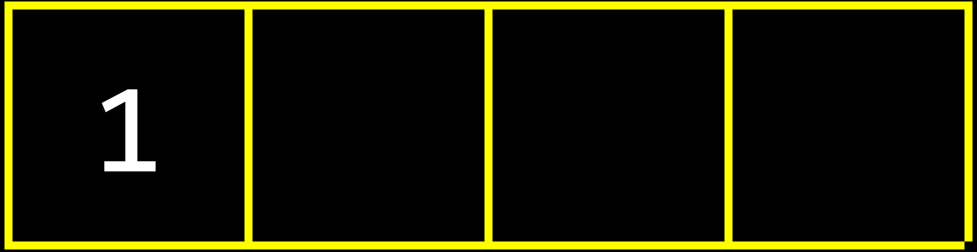

1
2
3

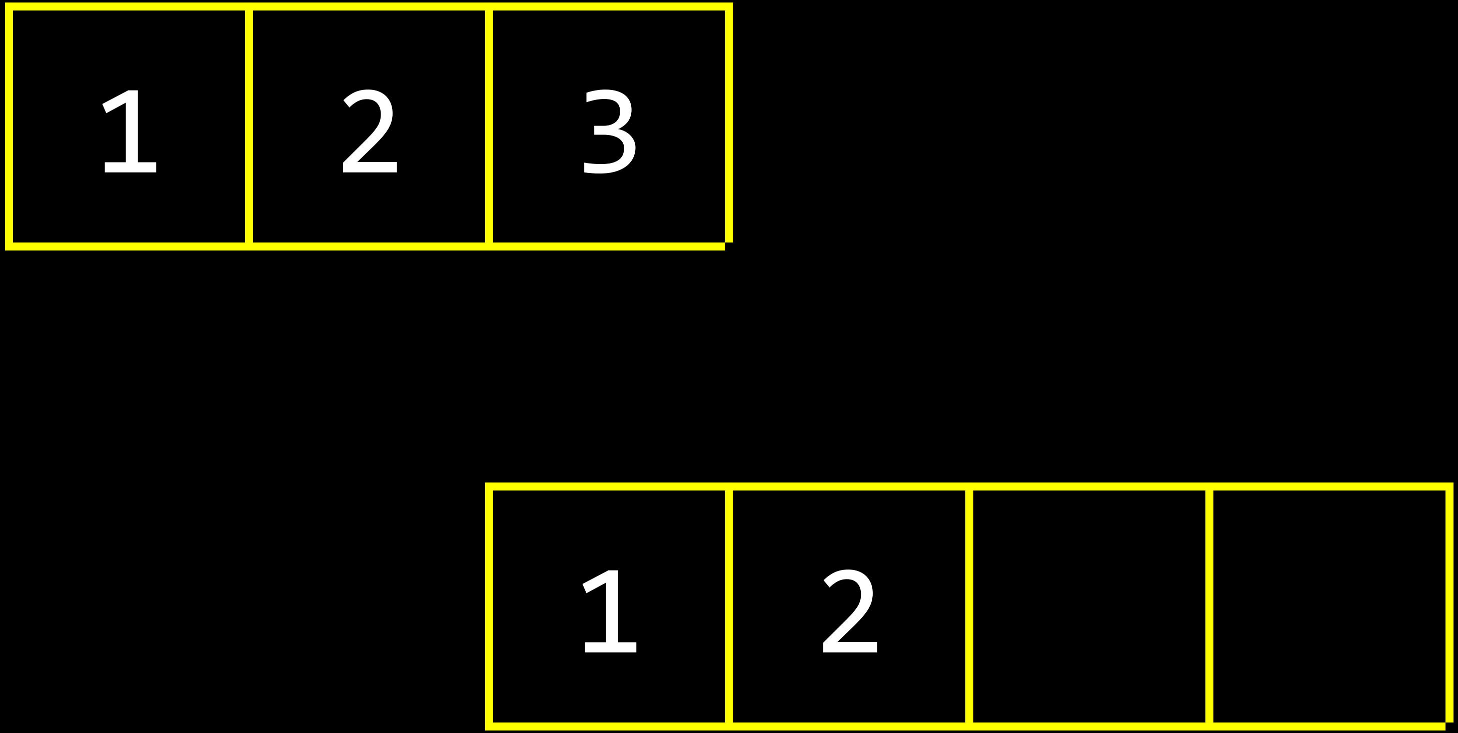

1
2
3

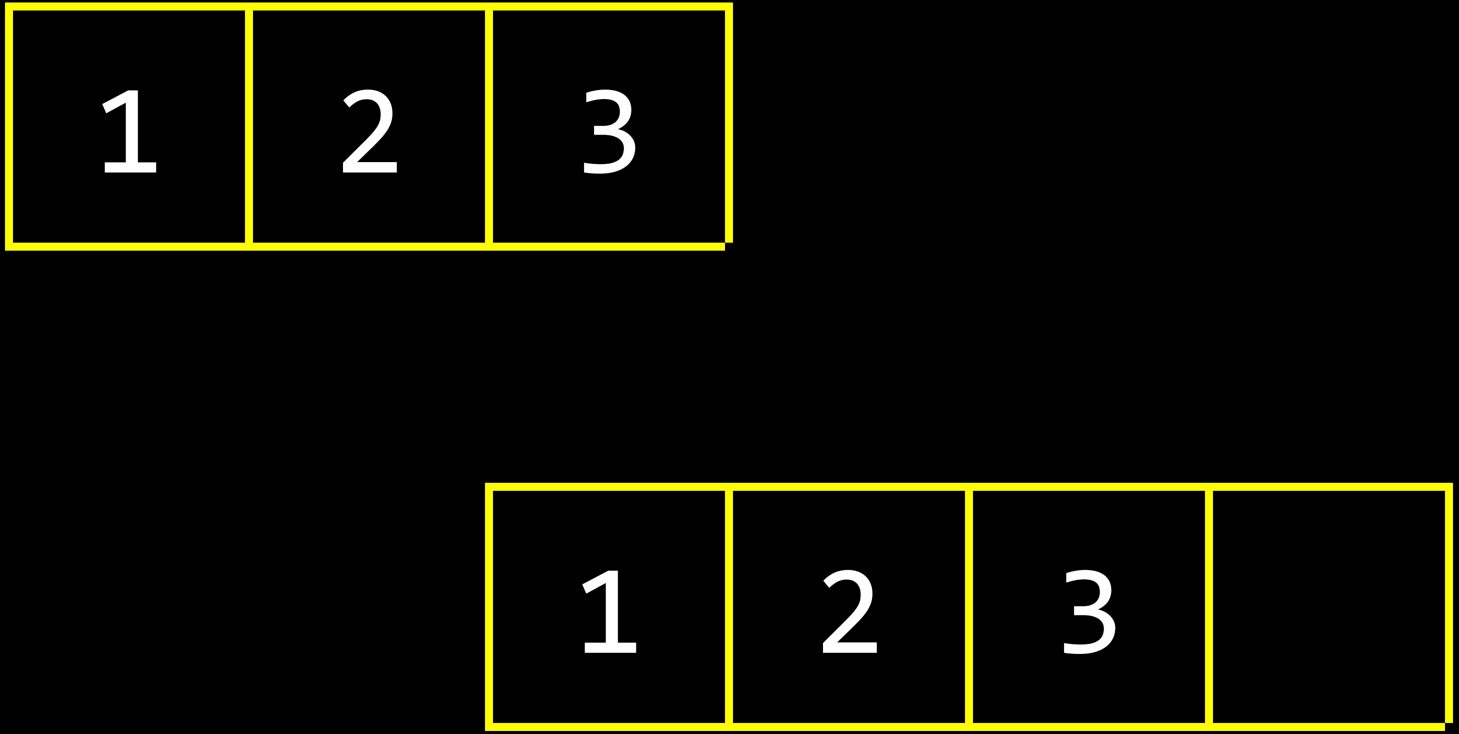

1
2
3

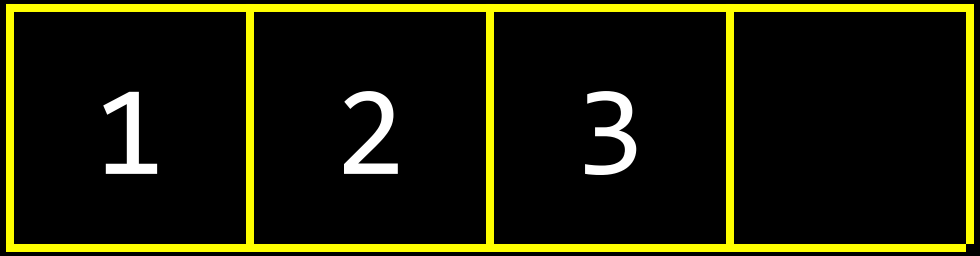
M	M	A	\0	E	M	M	A
\0	1	2	3	E	M	M	A
\0	E	M	M	A	\0	E	M
M	A	\0					

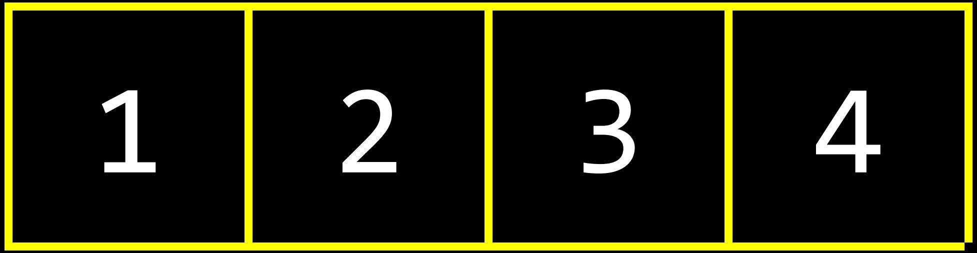

1
2
3


1
2
3

1


1
2
3
1
2


1
2
3
1
2
3


1
2
3


1
2
3
4

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$ search

$O(1)$

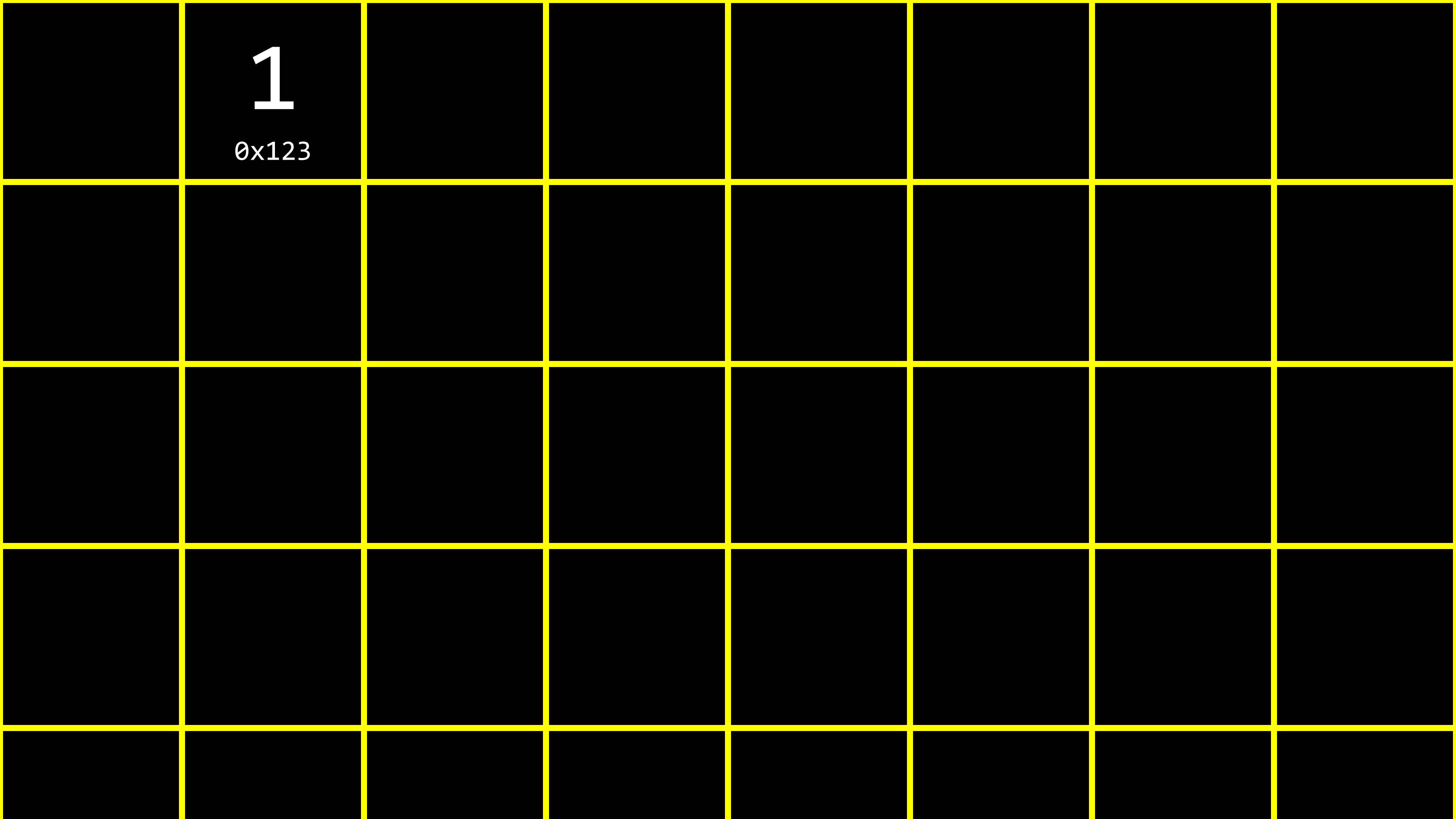
$O(n^2)$	
$O(n \log n)$	
$O(n)$	insert
$O(\log n)$	search
$O(1)$	

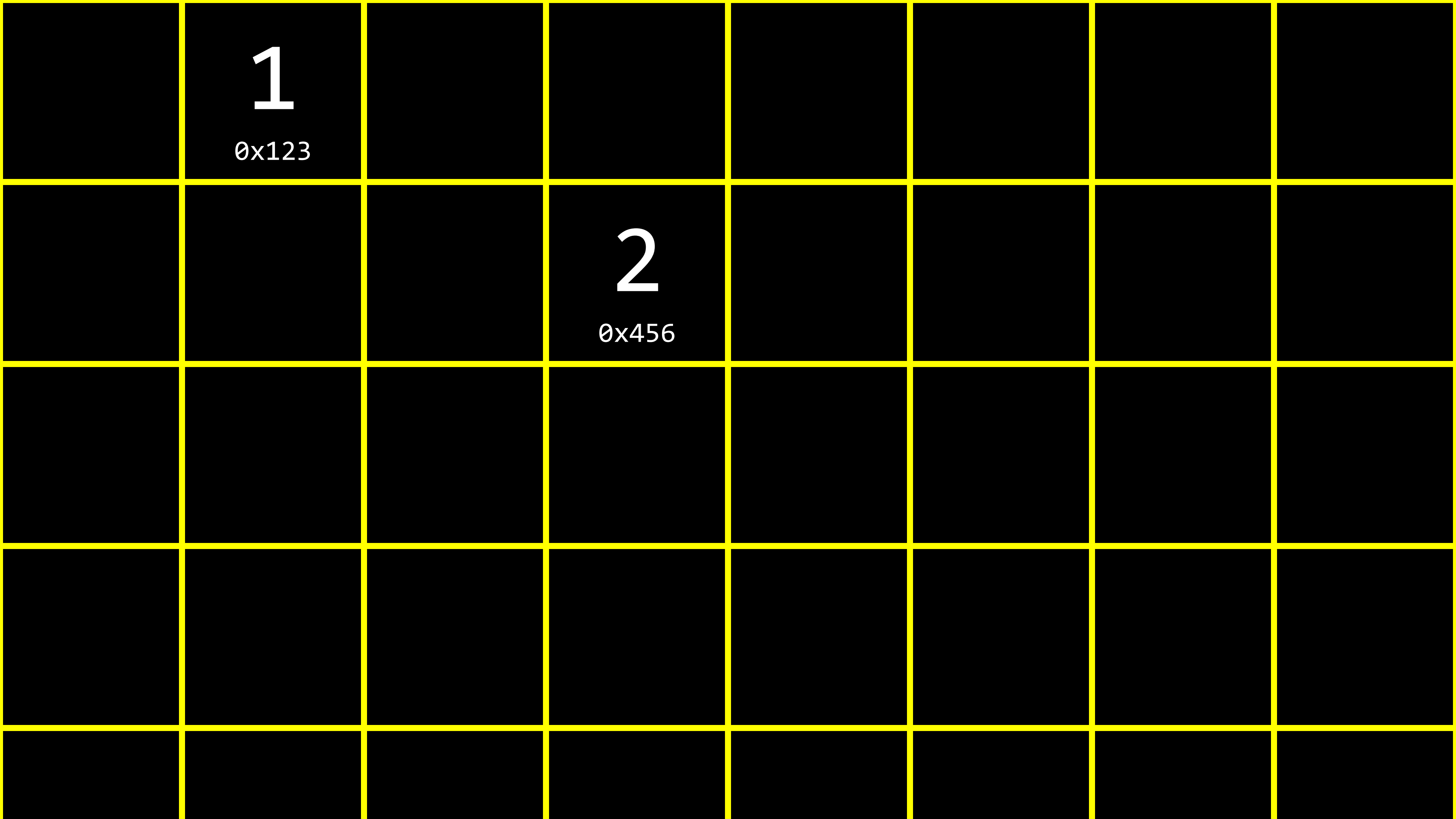
data structures

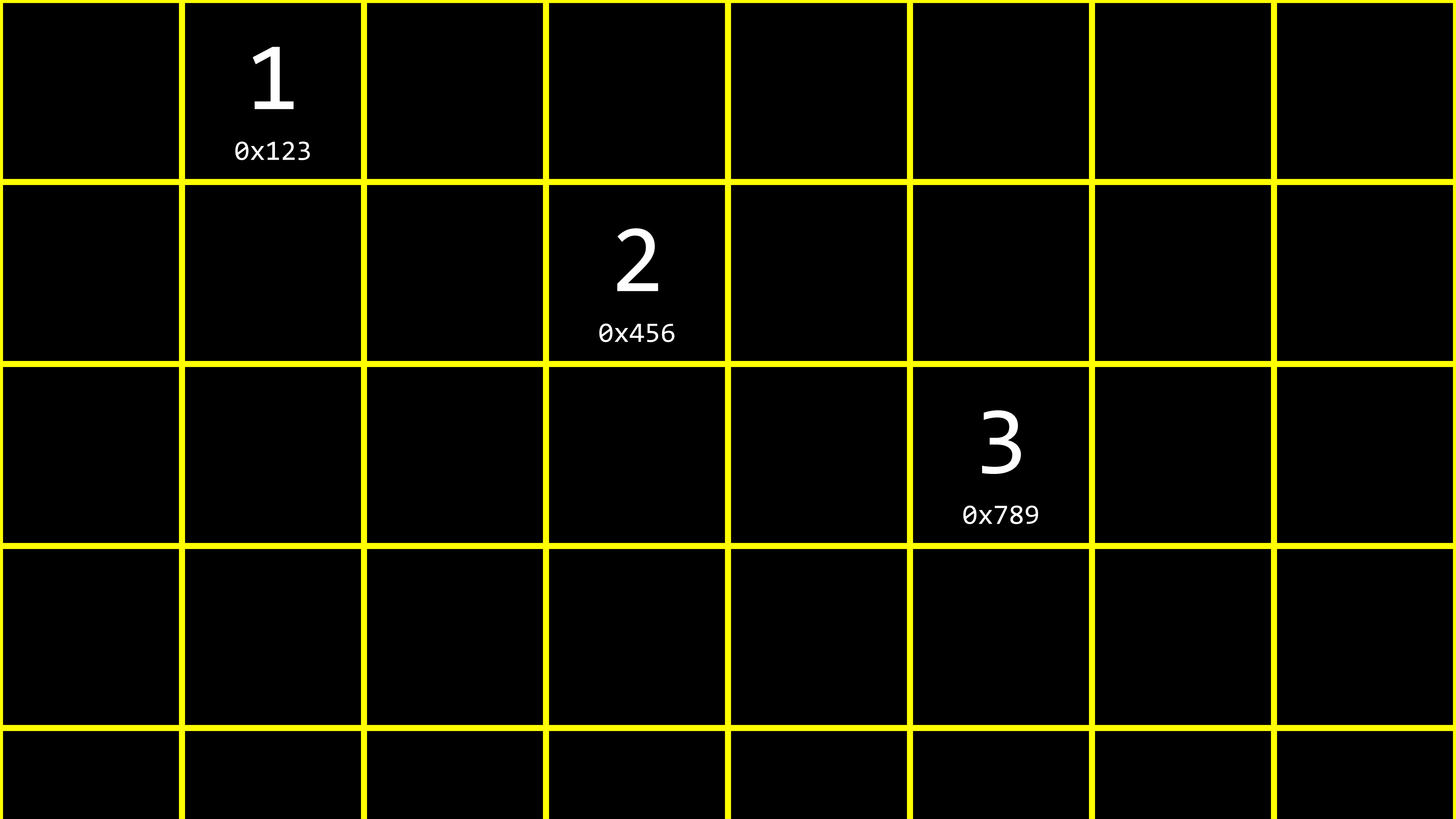
```
struct

.

*
```

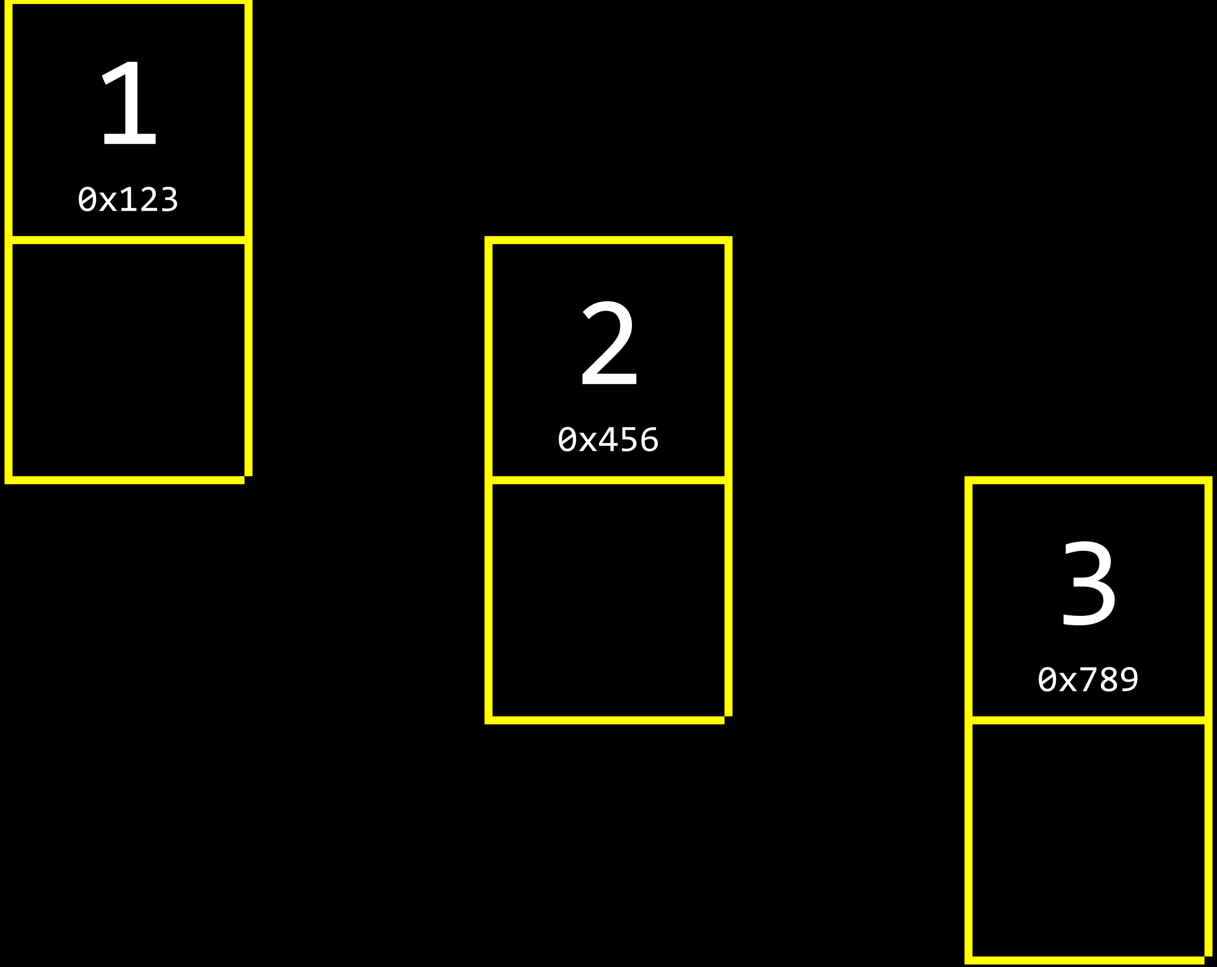
```
struct

->
```

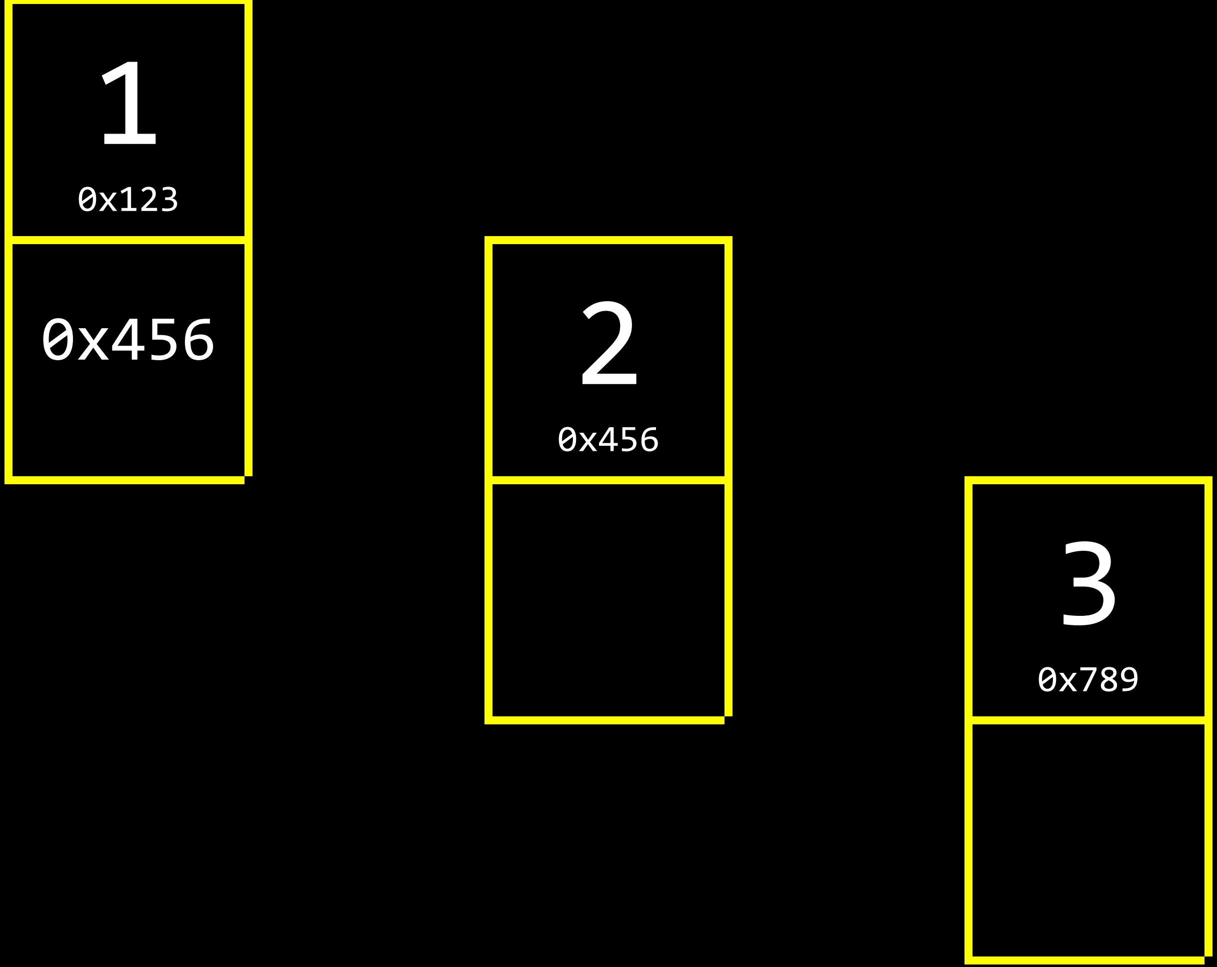
linked lists

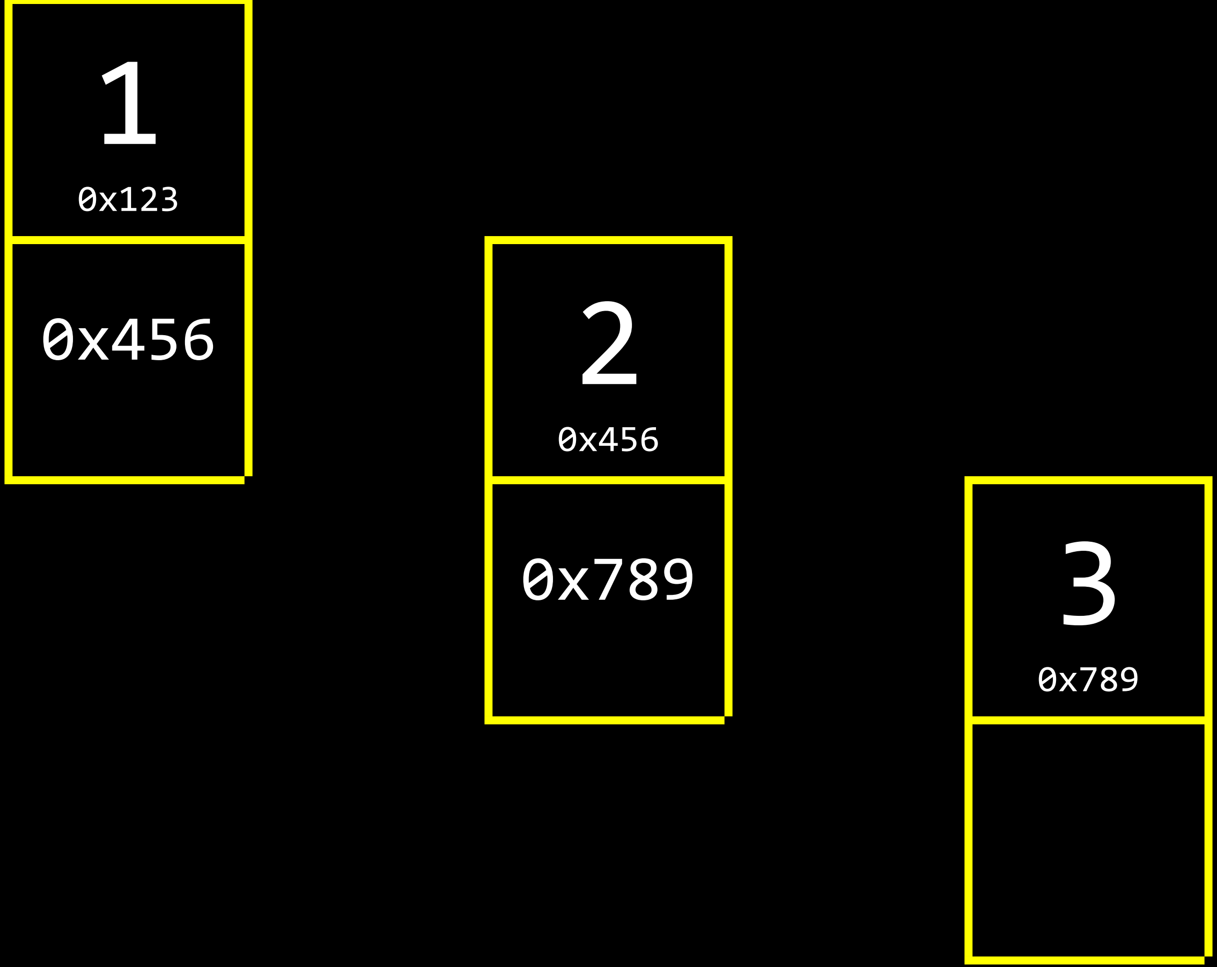

1
0x123

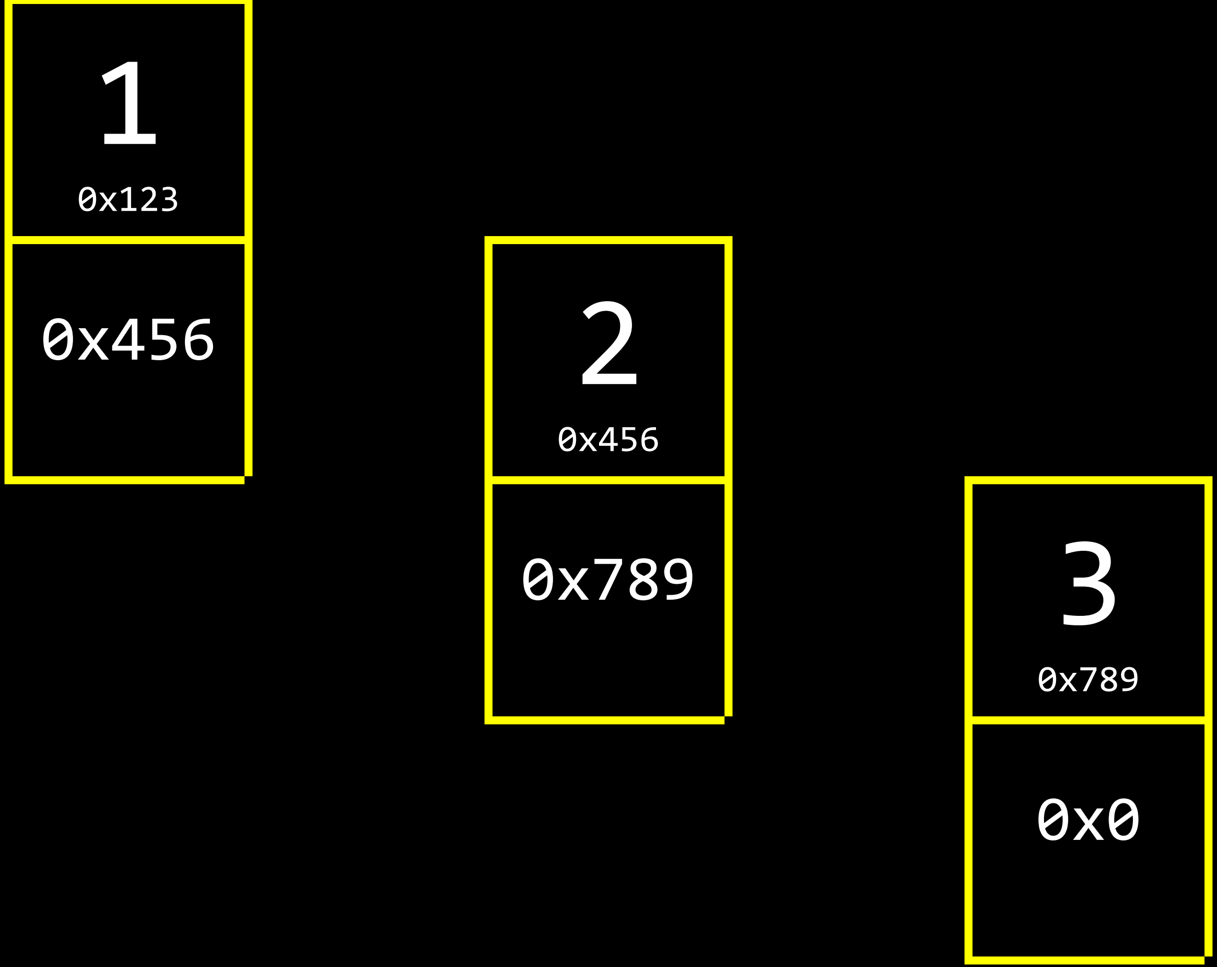

1
0x123
2
0x456

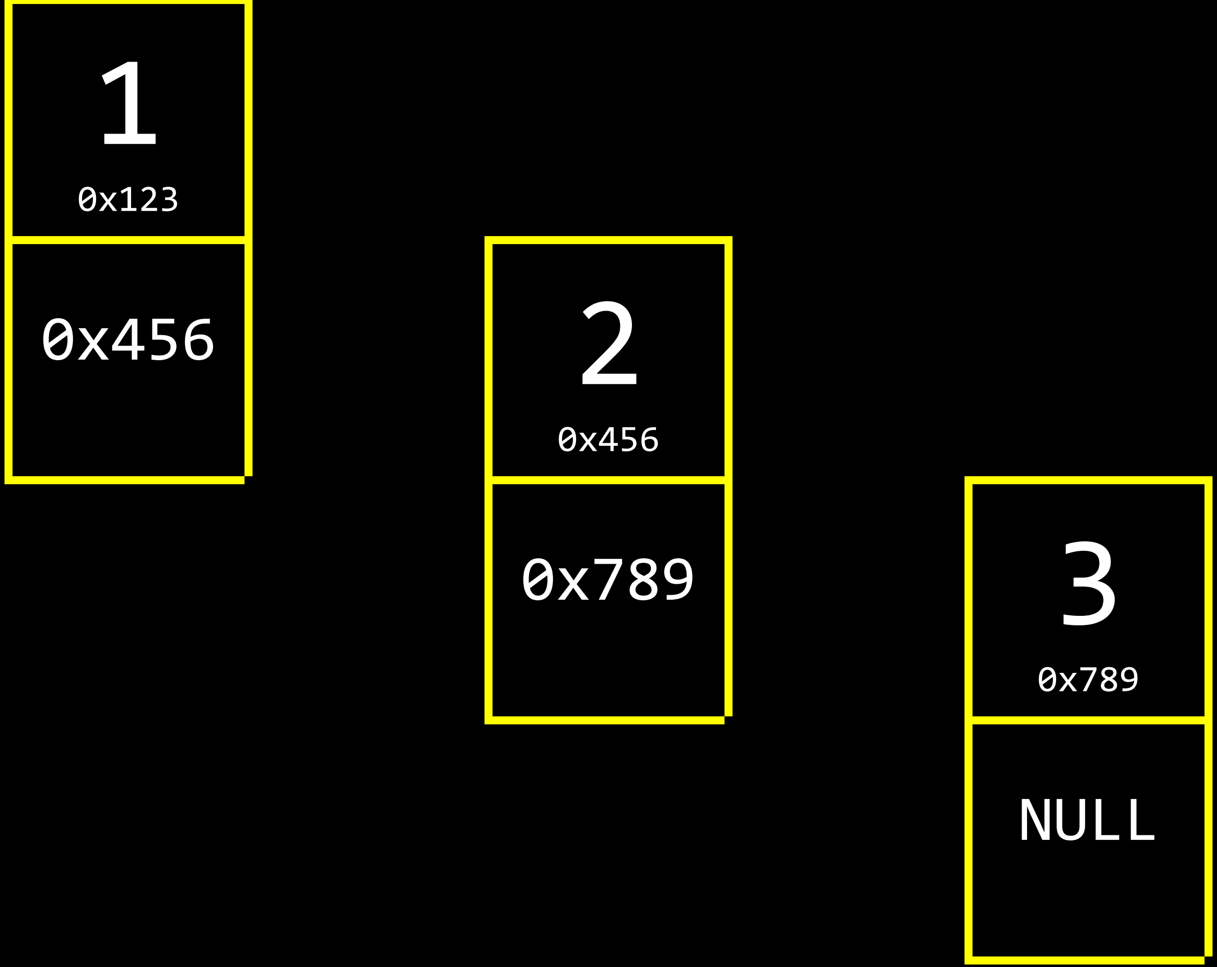

1
0x123
2
0x456
3
0x789

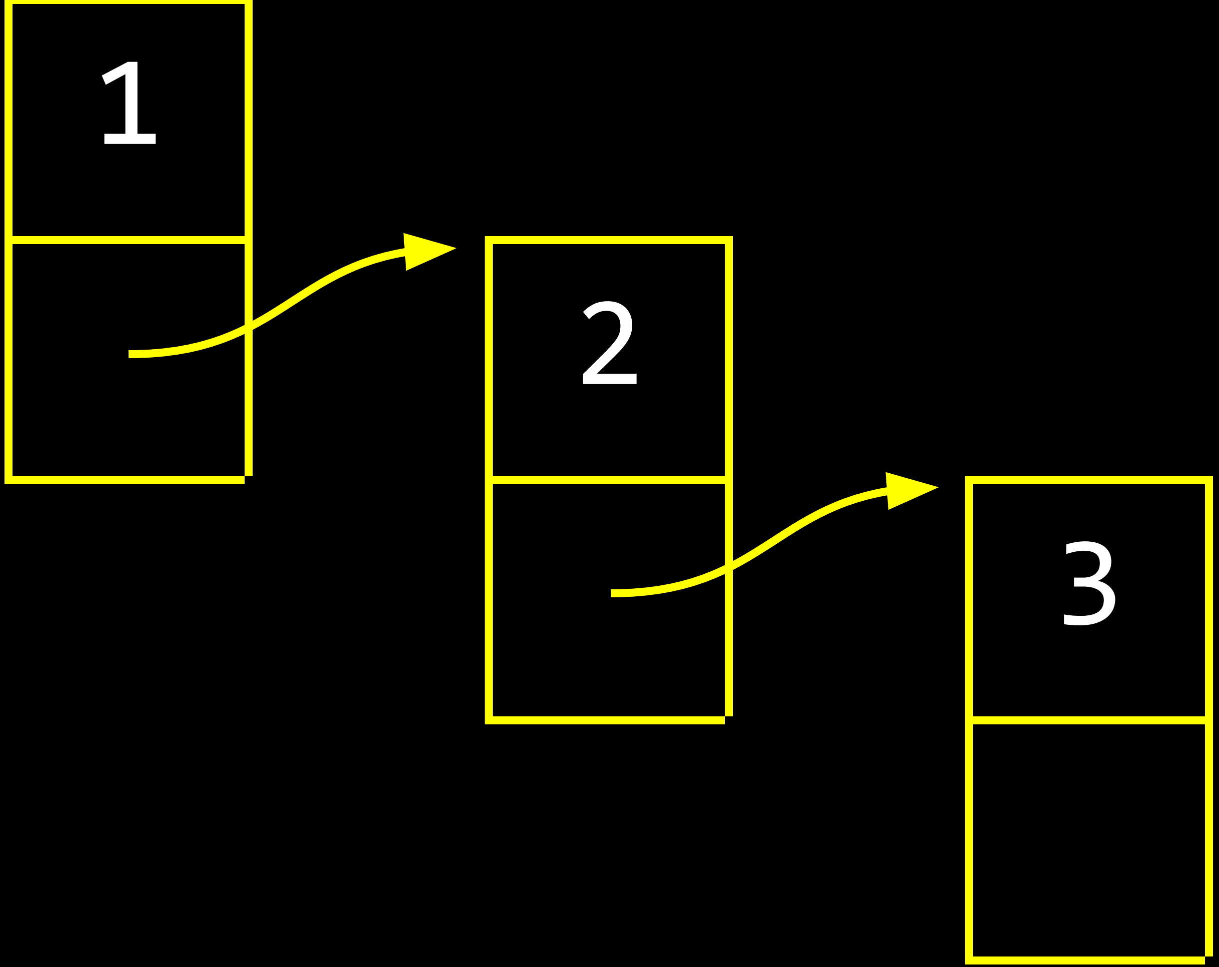

1
0x123
2
0x456
3
0x789


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


1
2
3

```
typedef struct
{
    string name;
    string number;
}
person;
```

```
typedef struct
{



}
person;
```

```
typedef struct
{


}
node;
```

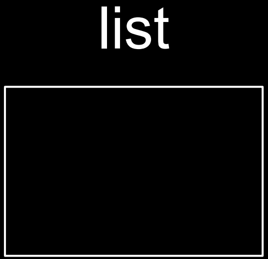
```
typedef struct
{


}
node;
```

```
typedef struct
{
    int number;

}
node;
```

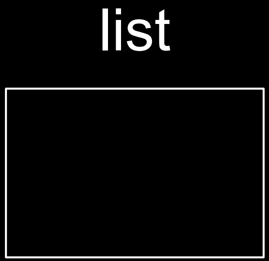
```
typedef struct
{
    int number;
    node *next;
}
node;
```

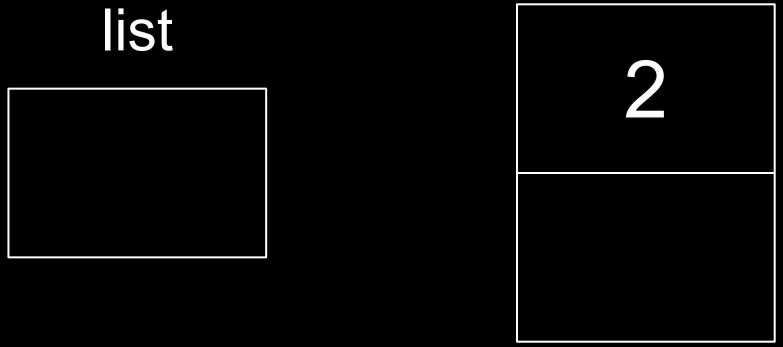
```
typedef struct node
{
    int number;
    node *next;
}
node;
```

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


list

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


list

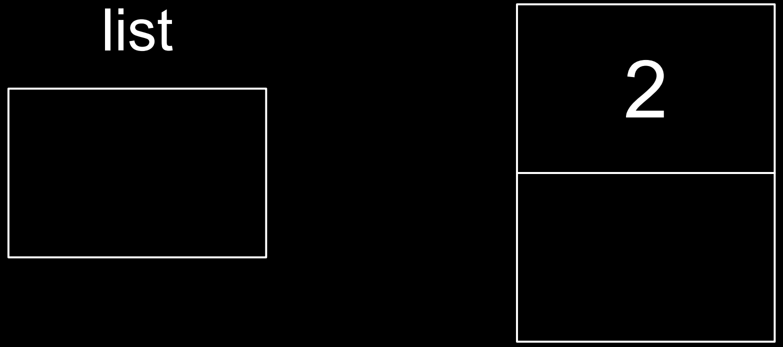

list
2

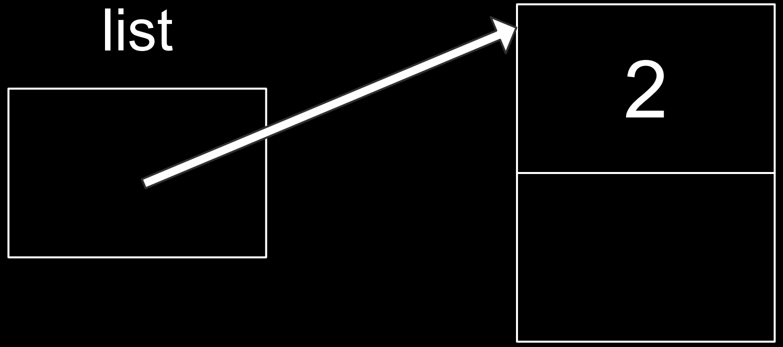
```
node *n = malloc(sizeof(node));
node *n = malloc(sizeof(node));
(*n).number = 2;
```

```
node *n = malloc(sizeof(node));
n->number = 2;
```

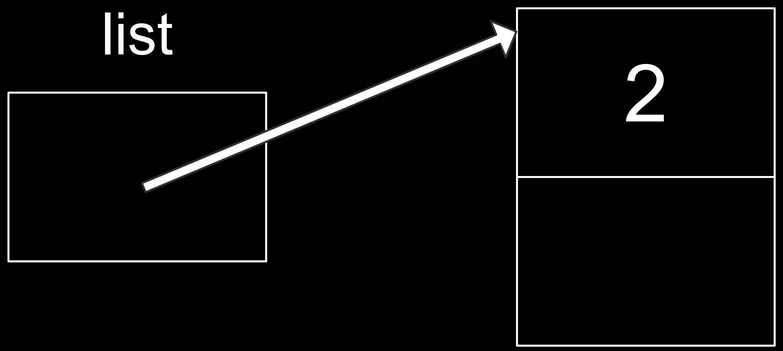
```
node *n = malloc(sizeof(node));
n->number = 2;
n->next = NULL;
```

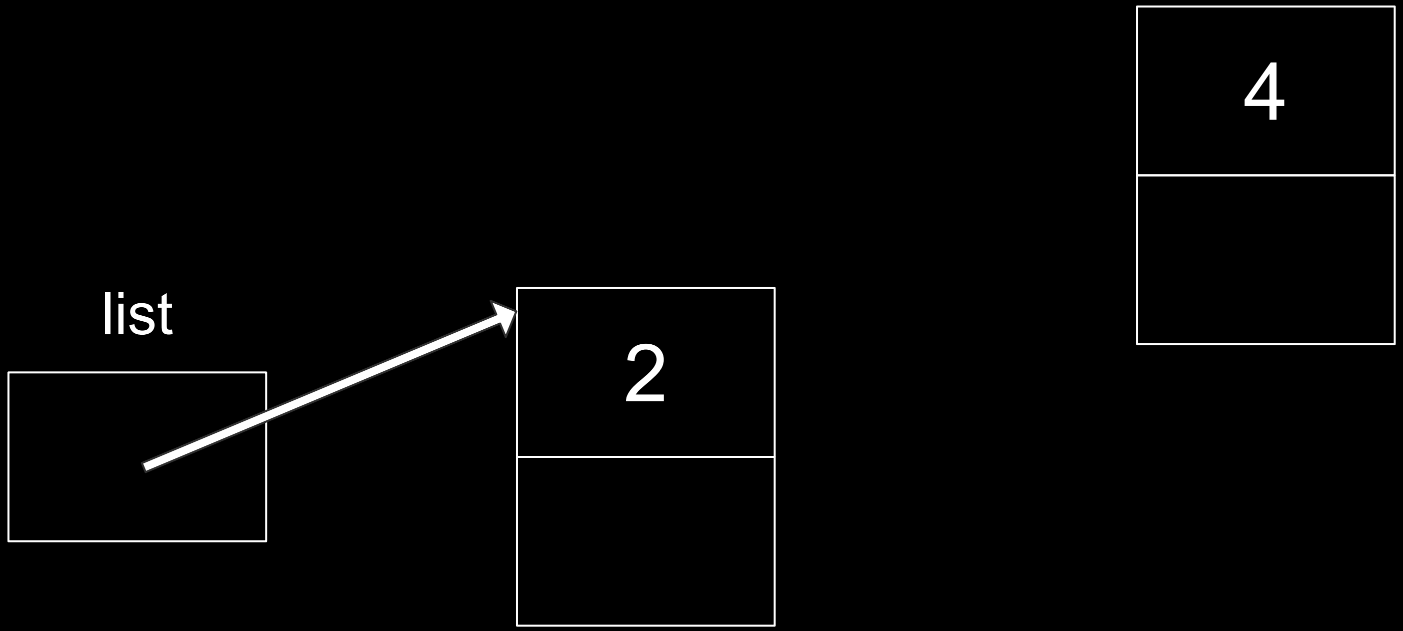
```
node *n = malloc(sizeof(node));
if (n != NULL)
{
    n->number = 2;
    n->next = NULL;
}
```


list
2

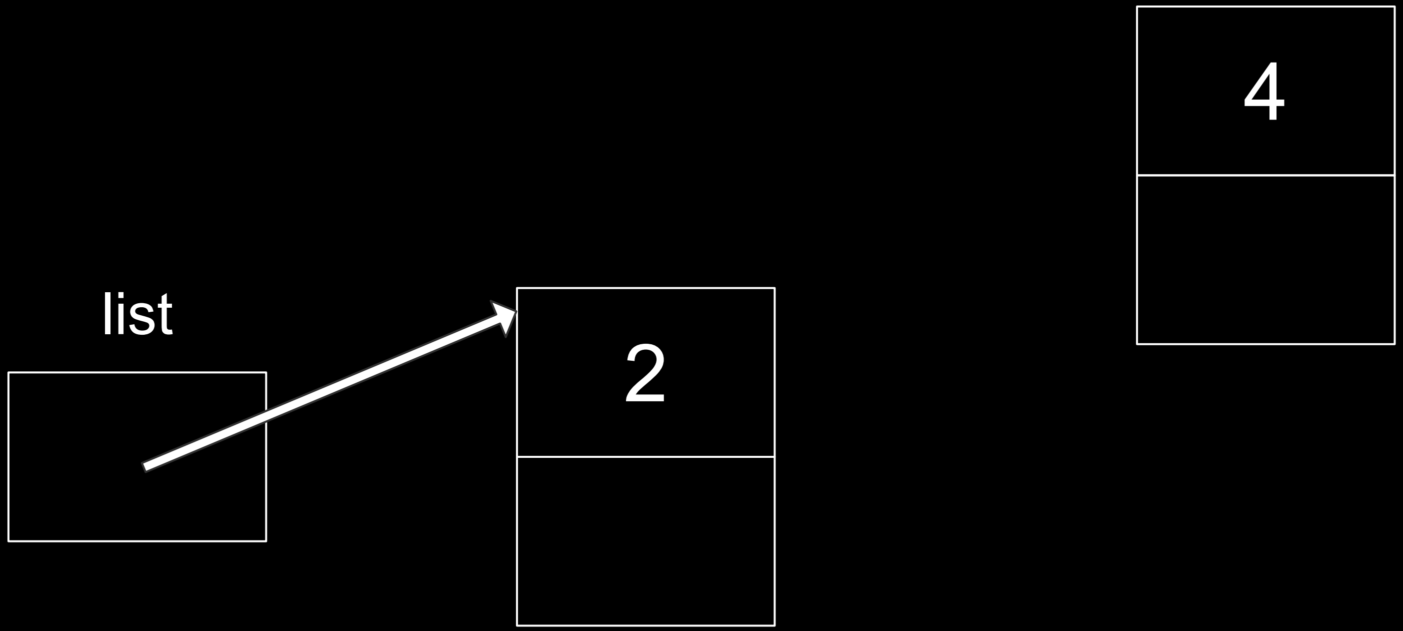

list
2

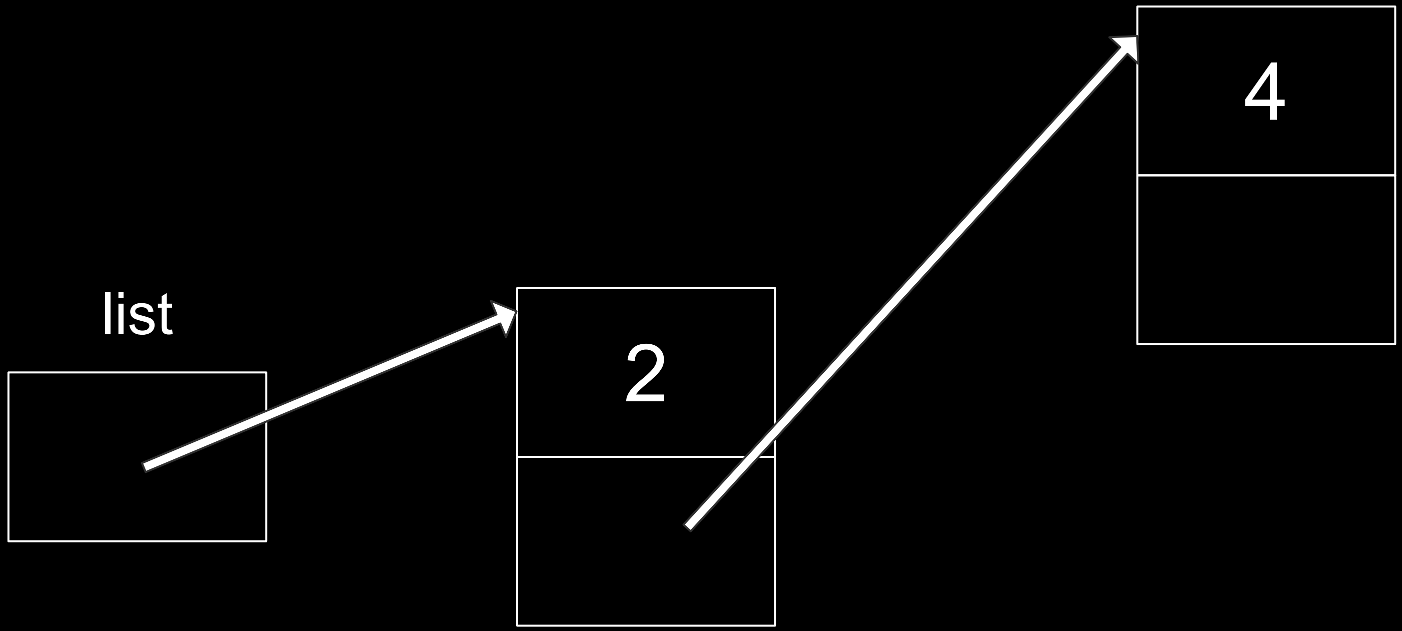
```
list = n;
```


list
2


list
2
4

```
node *n = malloc(sizeof(node));
if (n != NULL)
{
    n->number = 4;
    n->next = NULL;
}
```

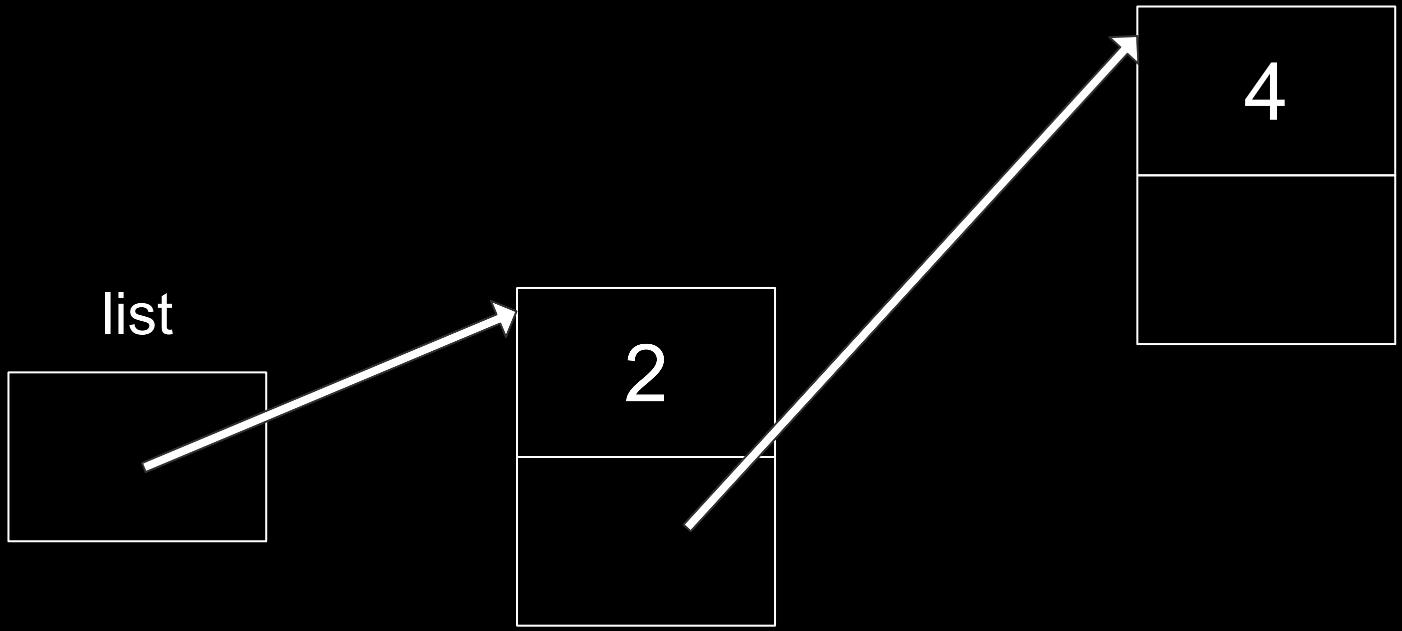

list
2
4

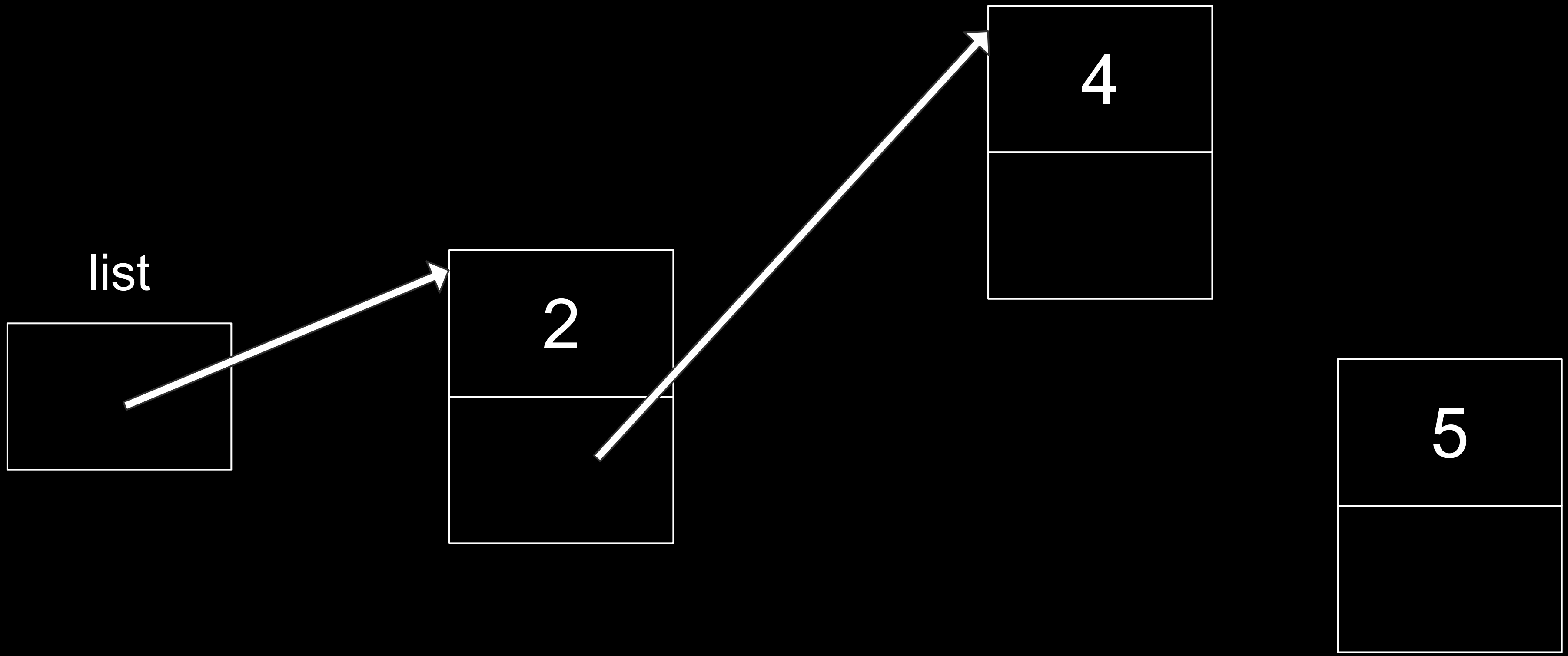

list
2
4

```
node *tmp = list;
node *tmp = list;
while (tmp->next != NULL)
{

}
```

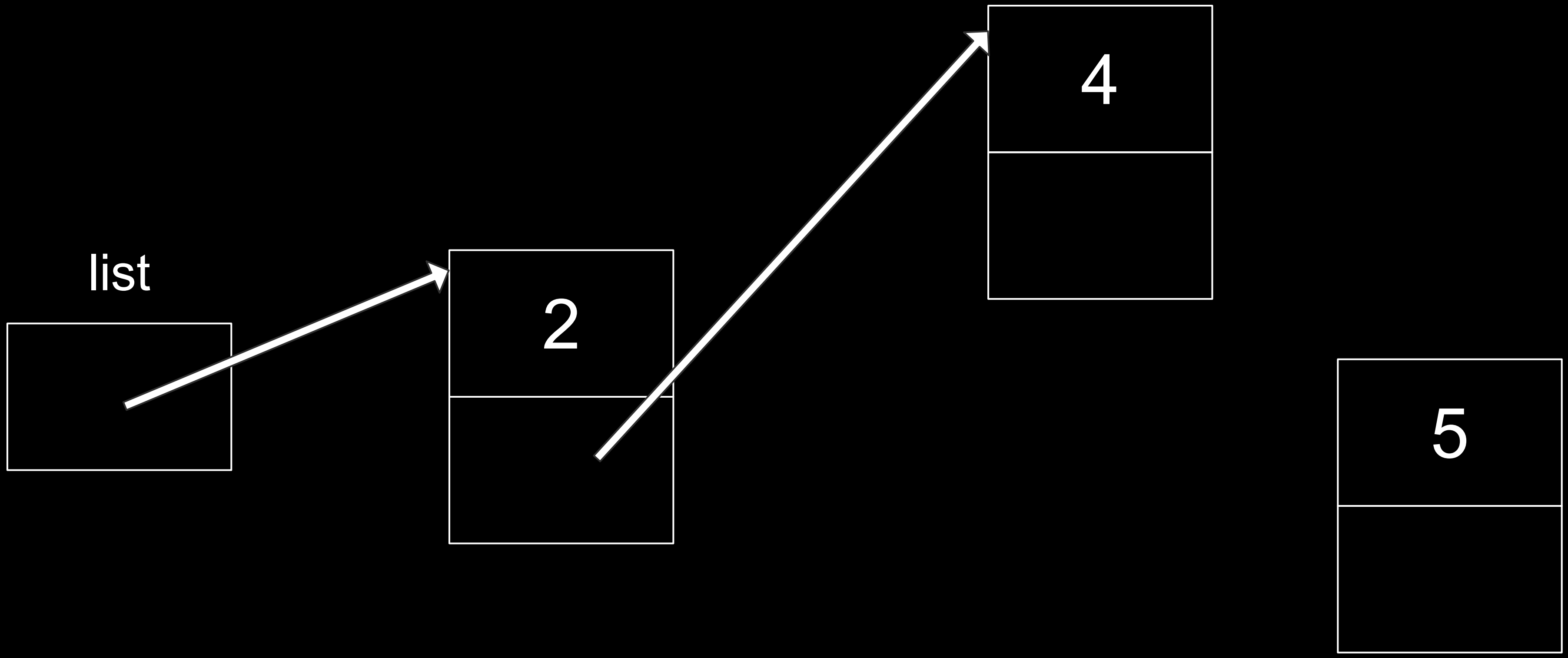
```
node *tmp = list;
while (tmp->next != NULL)
{
    tmp = tmp->next;
}
```

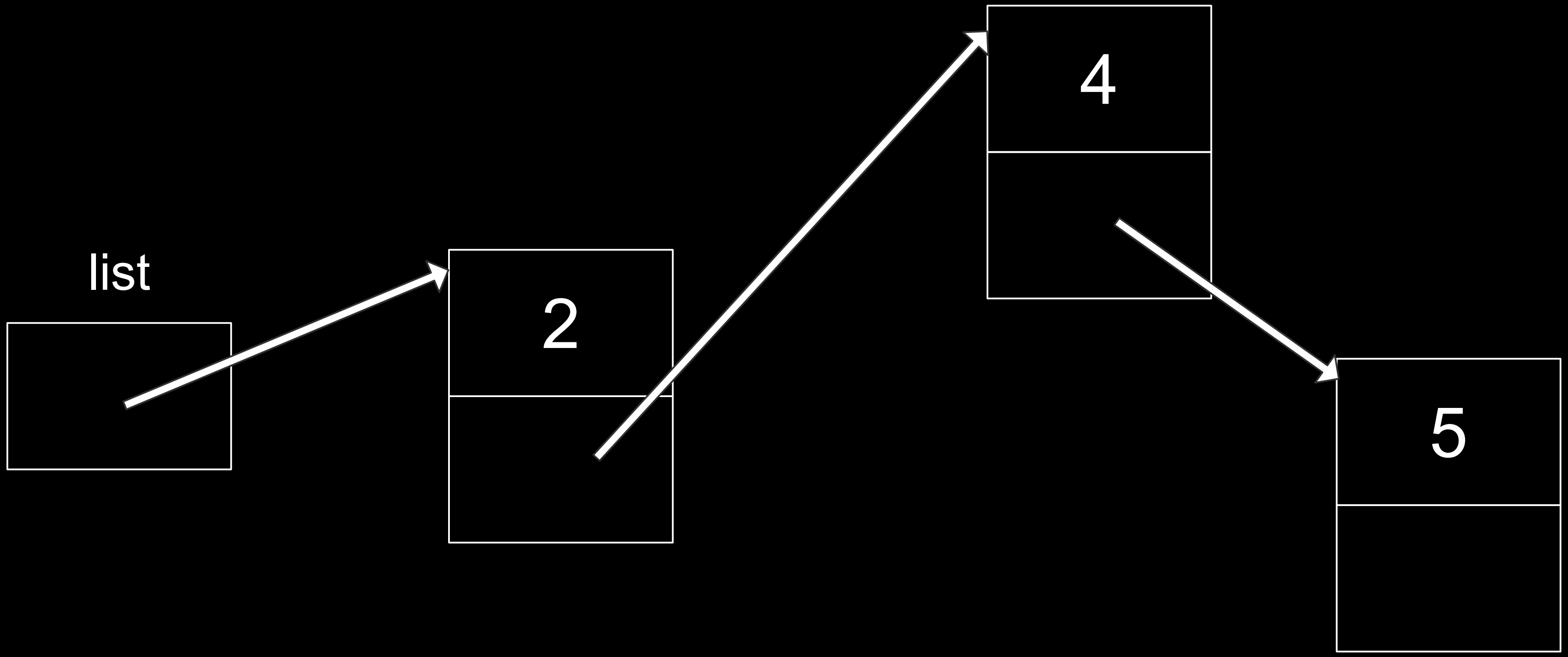
```
node *tmp = list;
while (tmp->next != NULL)
{
    tmp = tmp->next;
}
tmp->next = n;
```


list
2
4

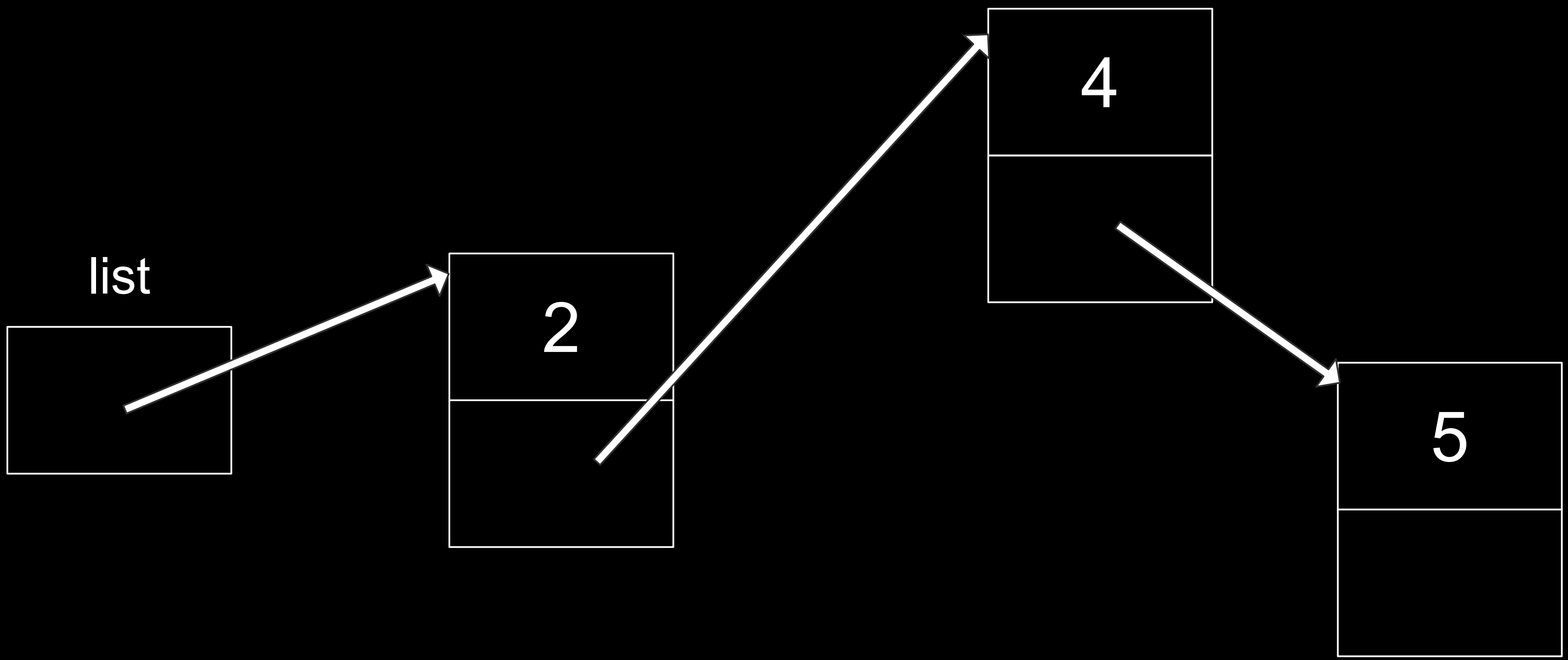

list
2
4
5

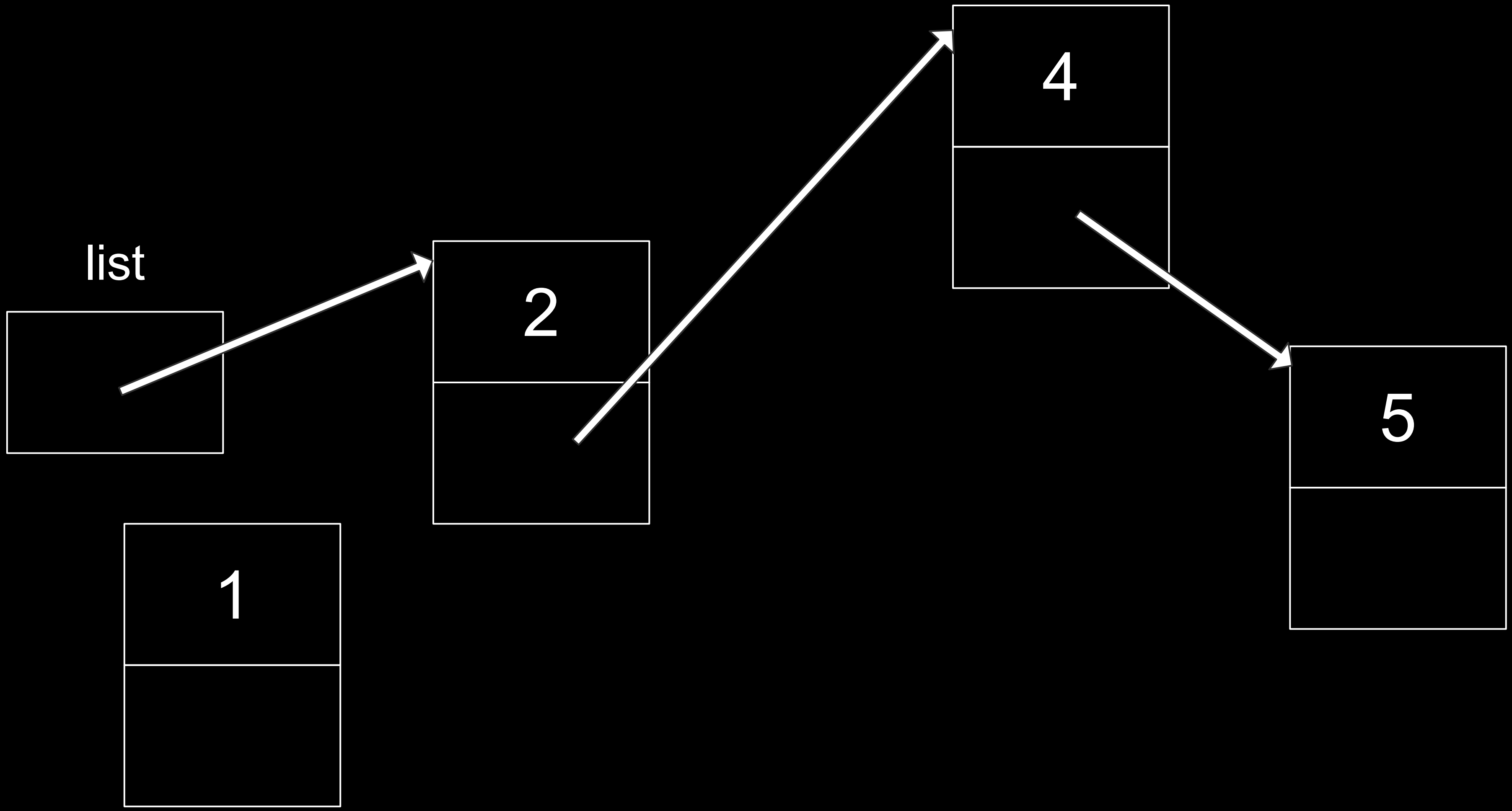
```
node *n = malloc(sizeof(node));
if (n != NULL)
{
    n->number = 5;
    n->next = NULL;
}
```


list
2
4
5

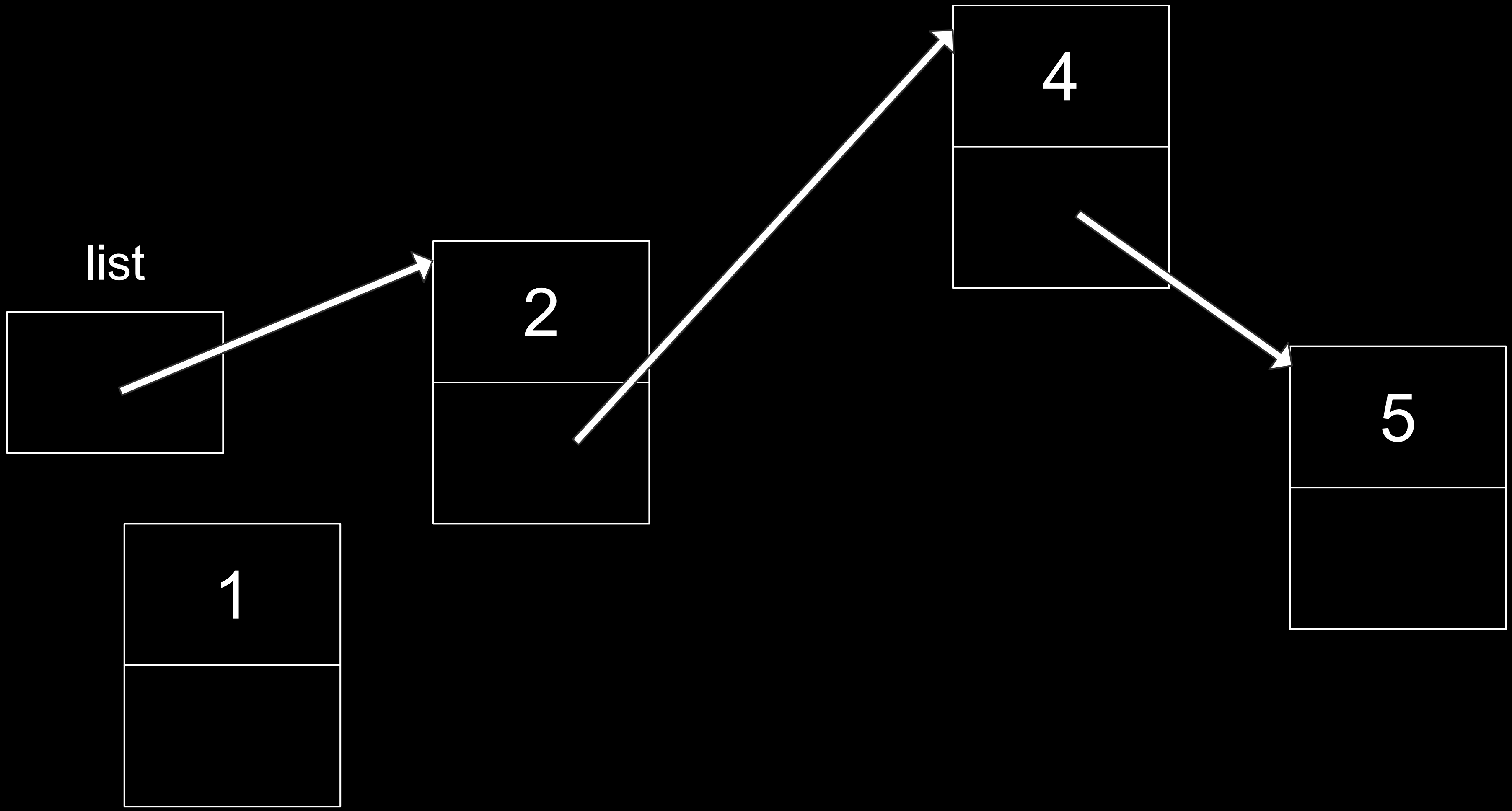

list
2
4
5

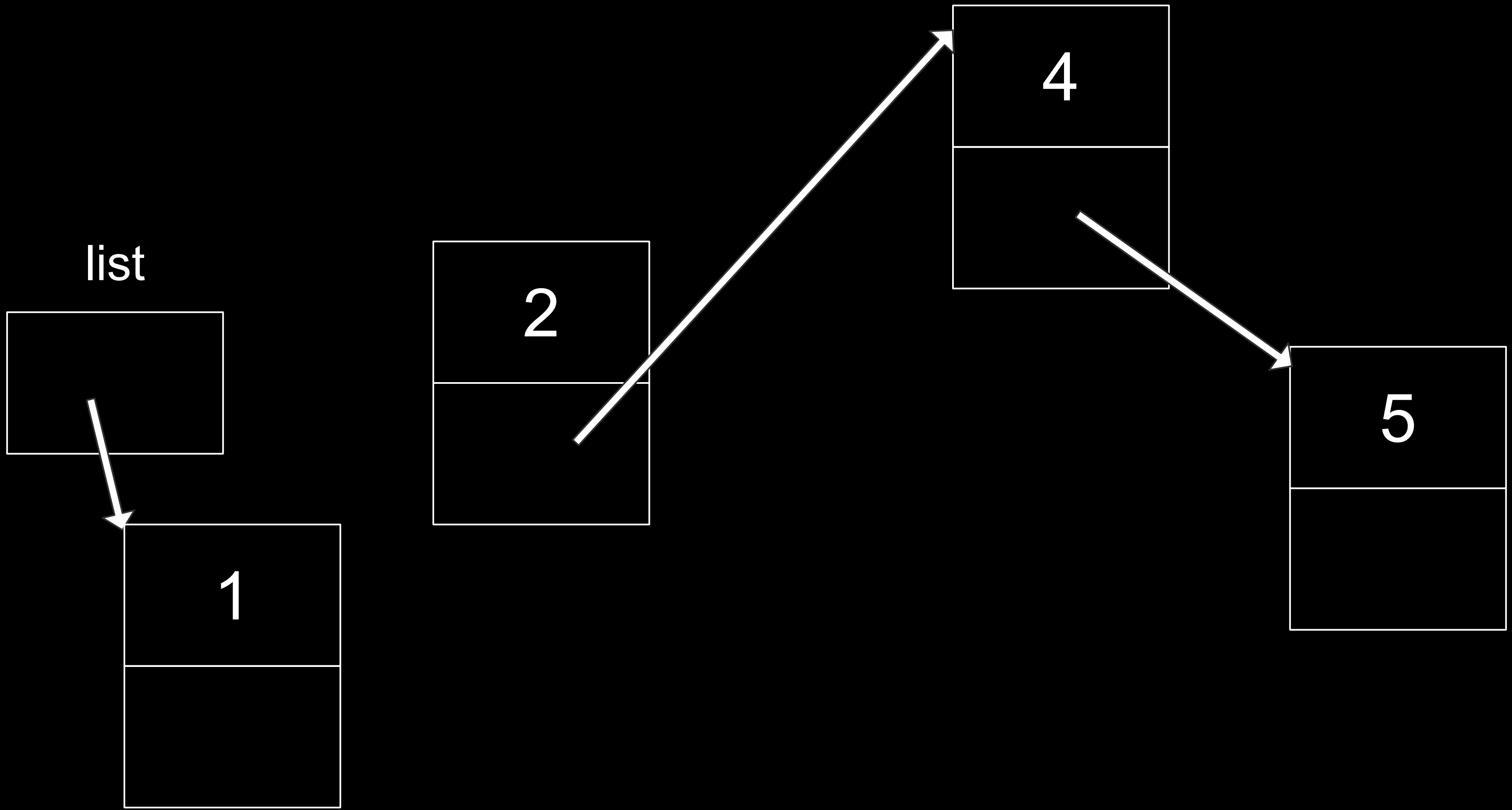
```
node *tmp = list;
while (tmp->next != NULL)
{
    tmp = tmp->next;
}
tmp->next = n;
```

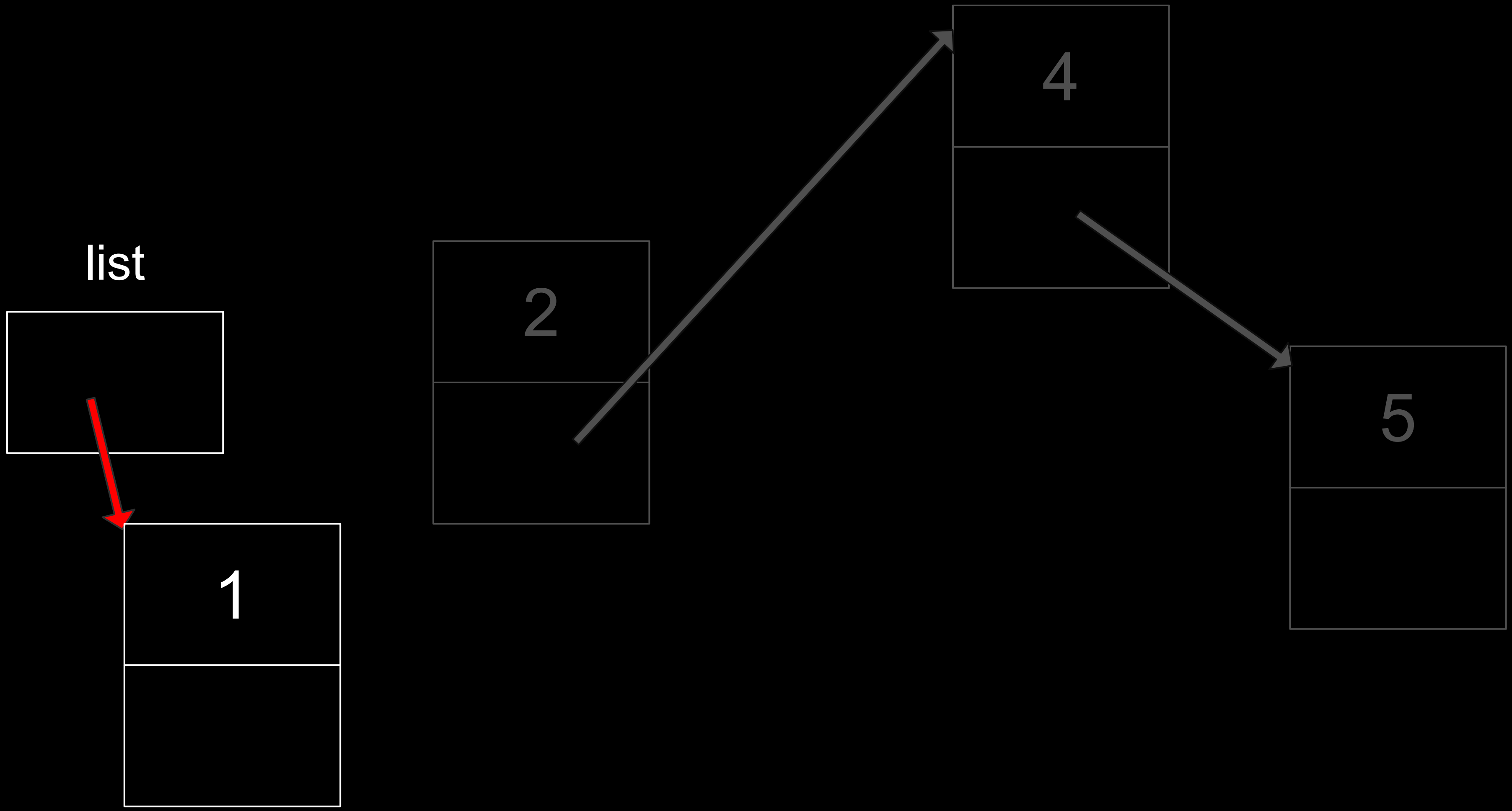

list
2
4
5

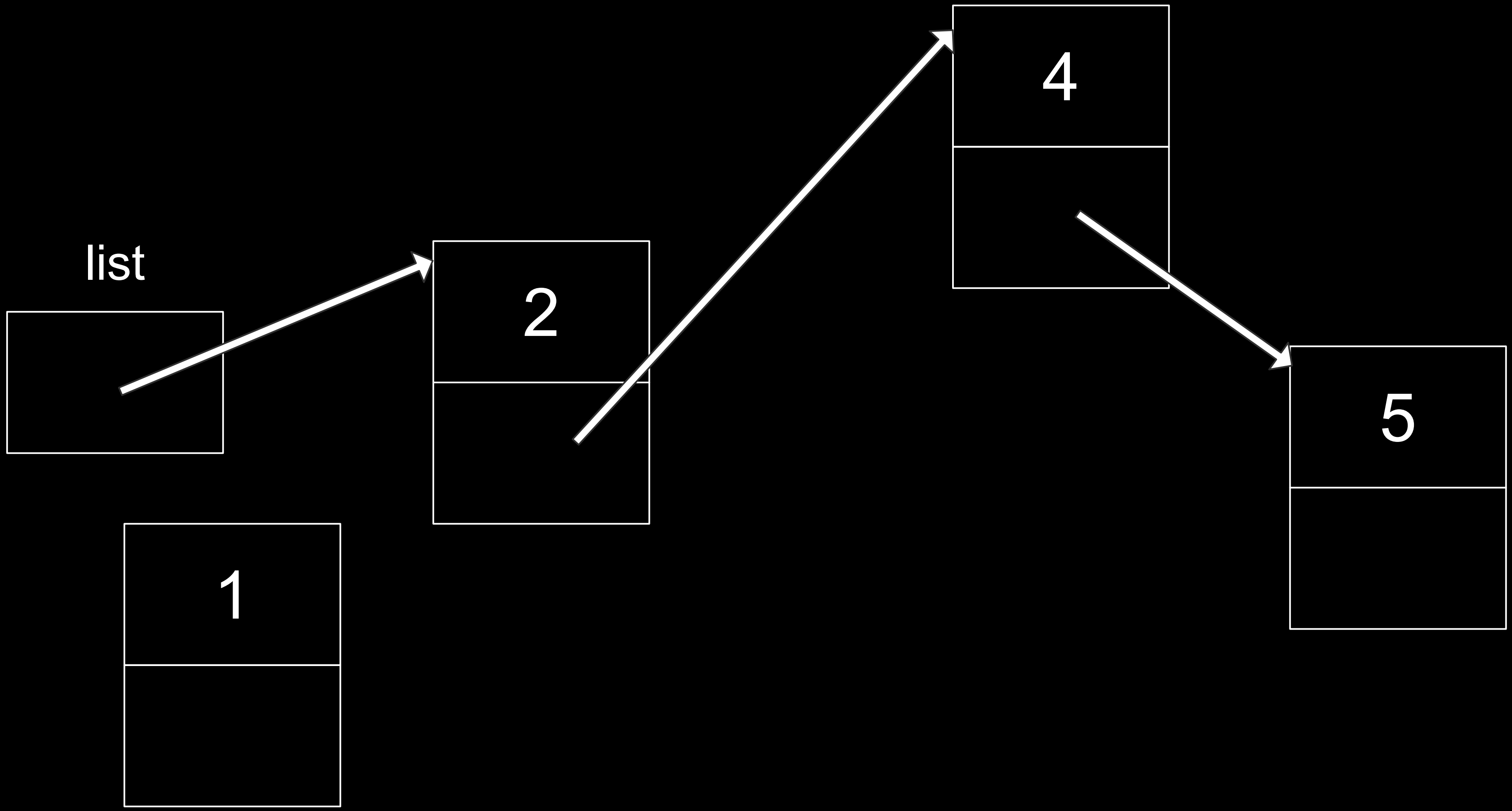

list
2
4
5
1

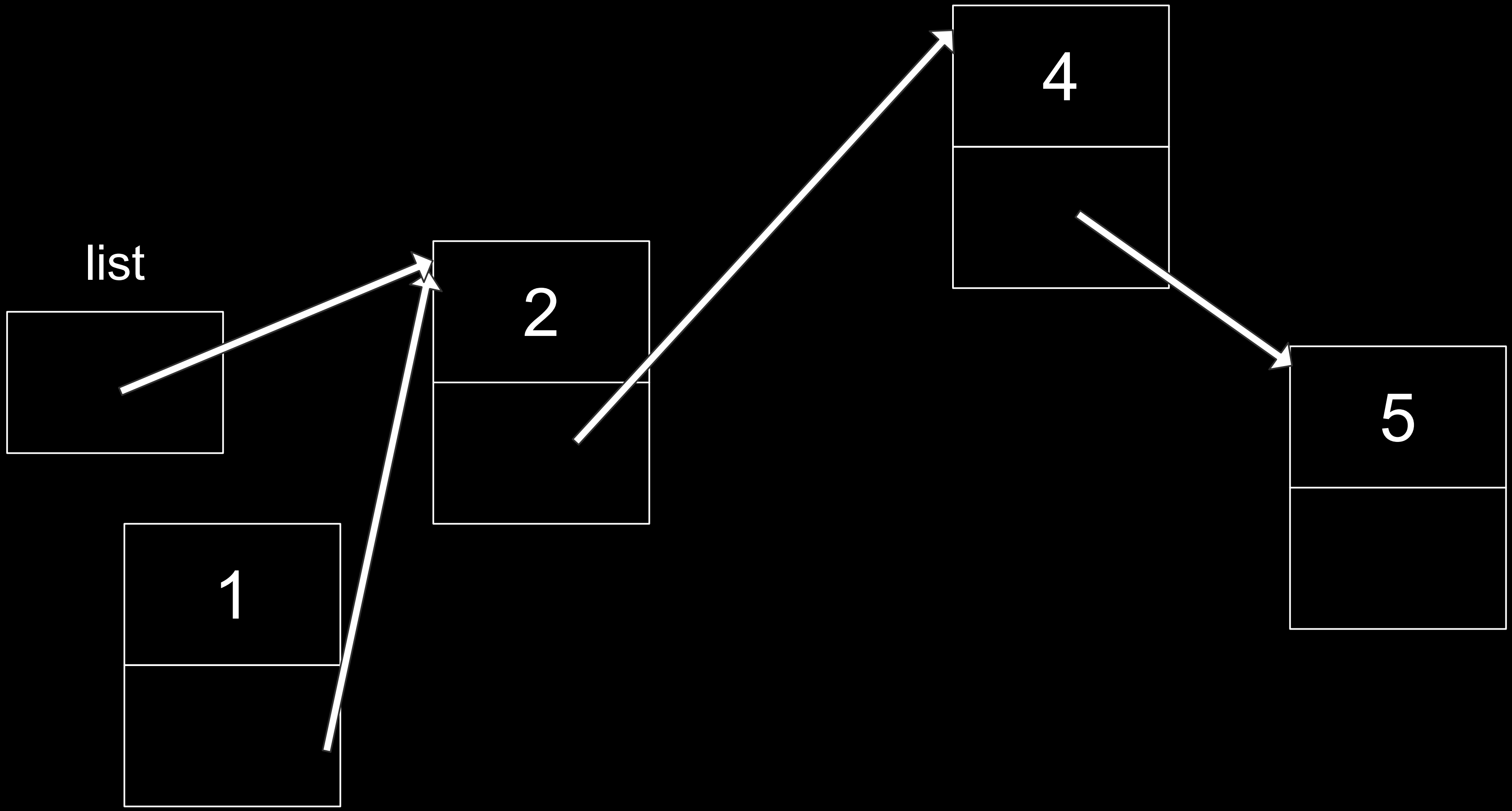
```
node *n = malloc(sizeof(node));
if (n != NULL)
{
    n->number = 1;
    n->next = NULL;
}
```

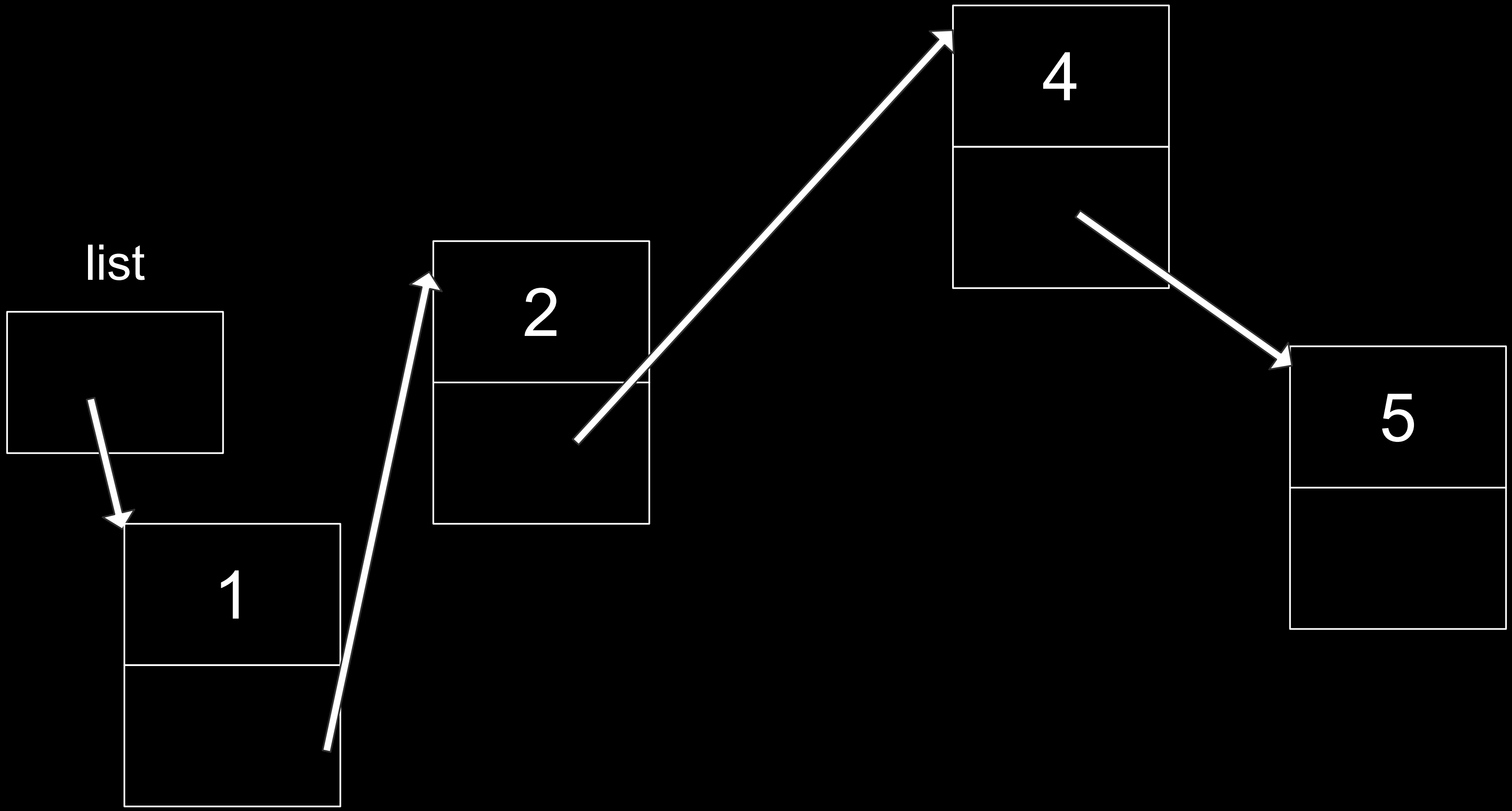

list
2
4
5
1


list
1
2
4
5

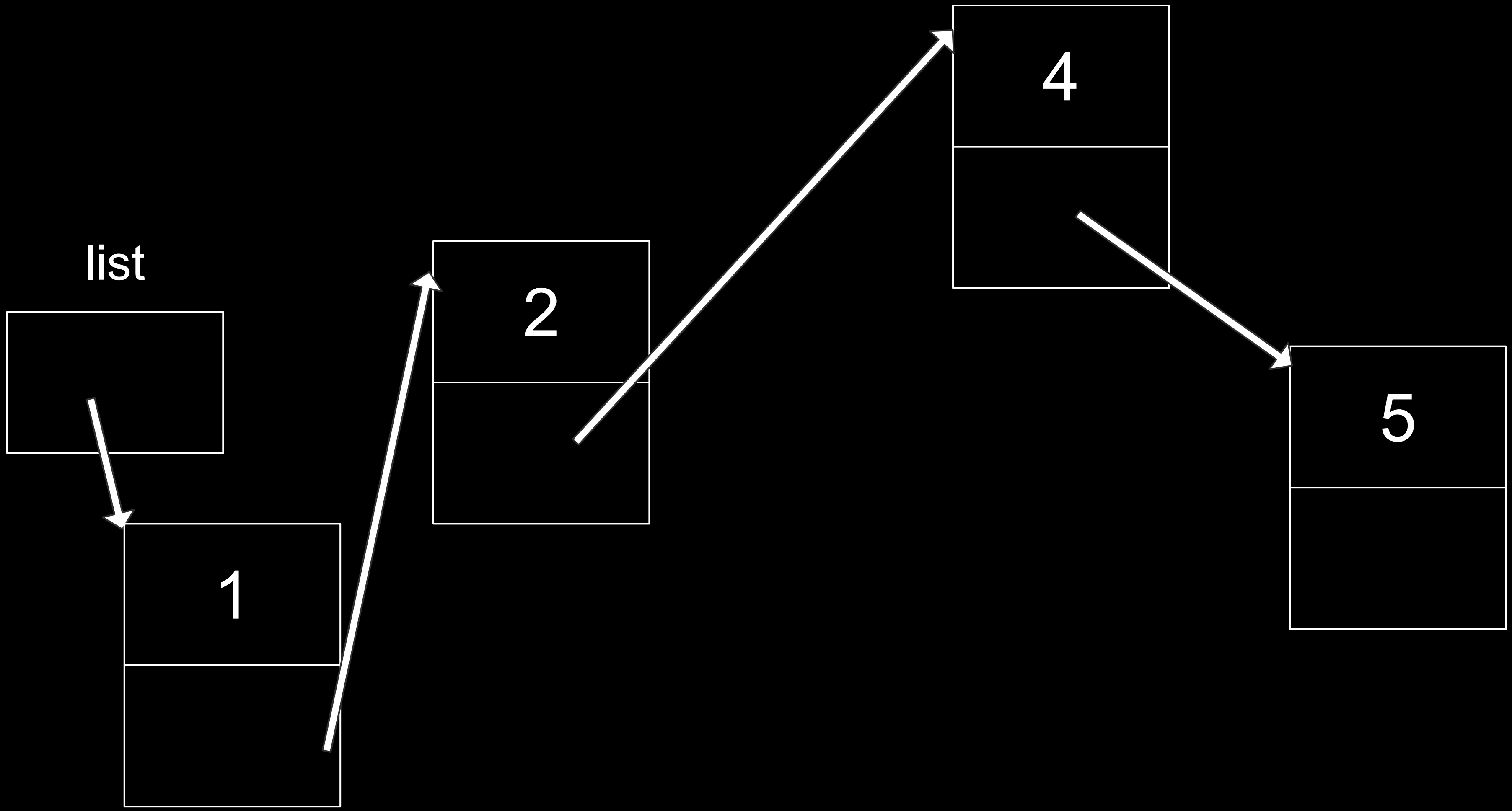

list
1
2
4
5

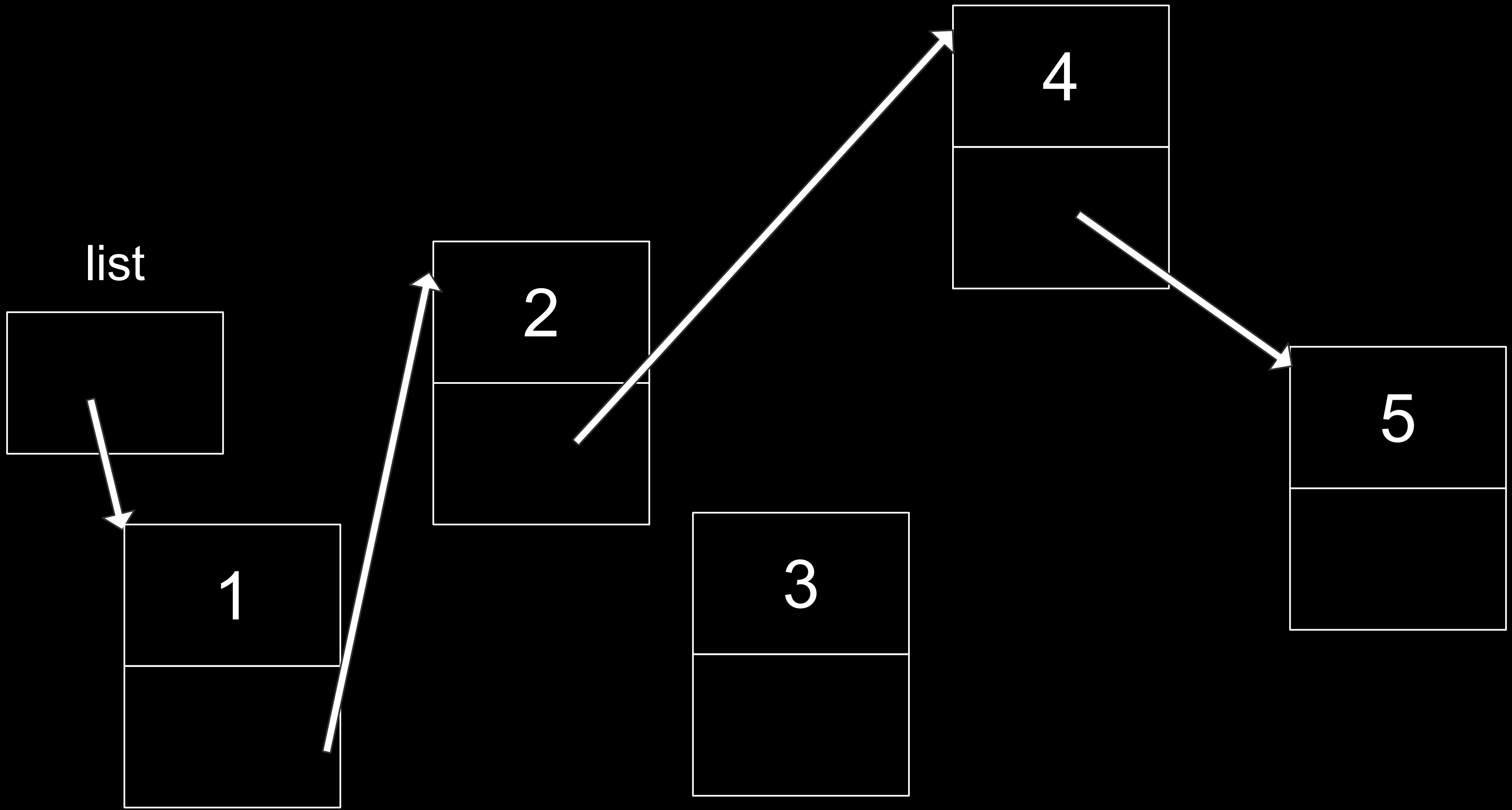

list
2
4
5
1

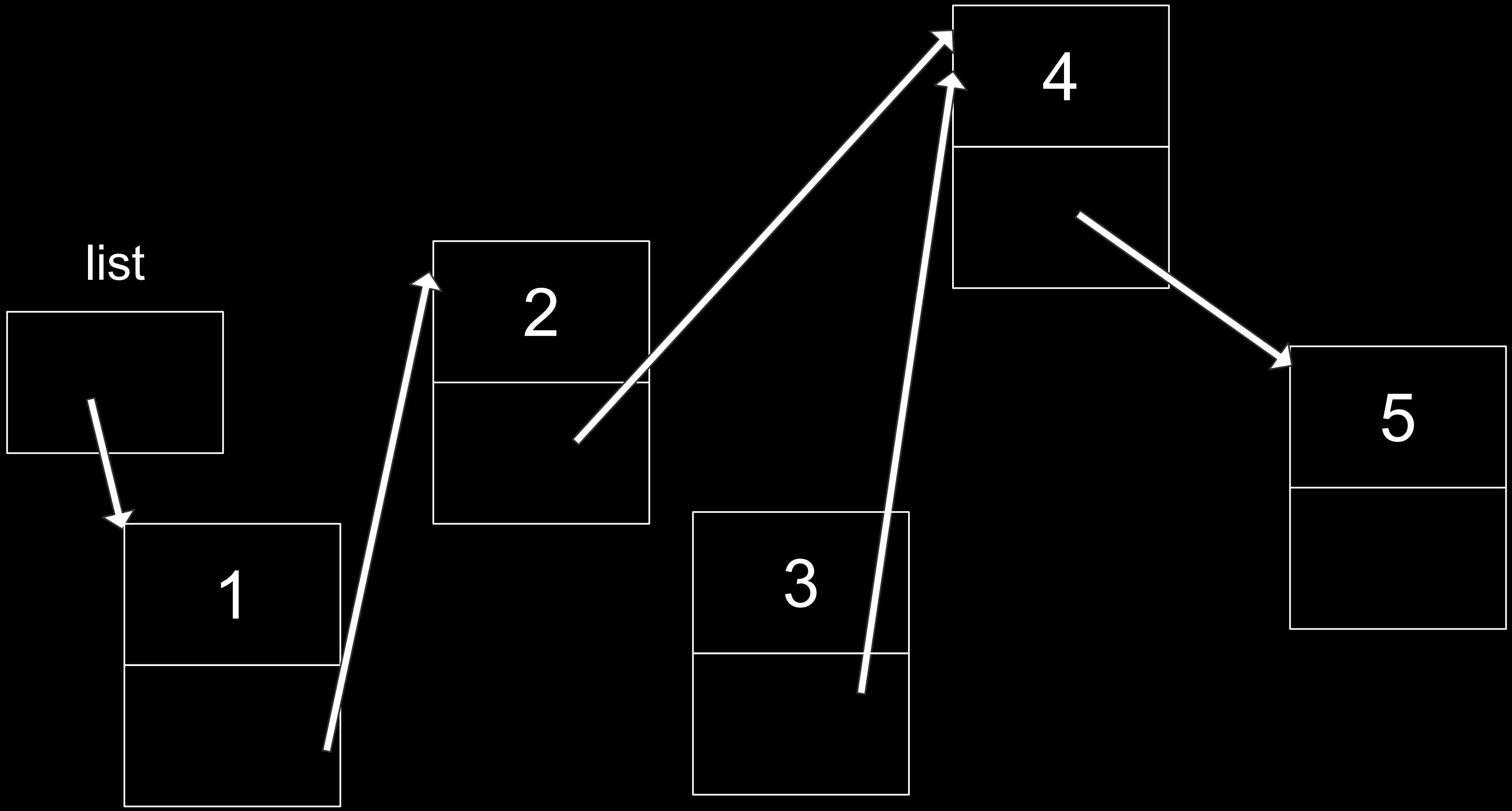

list
1
2
4
5

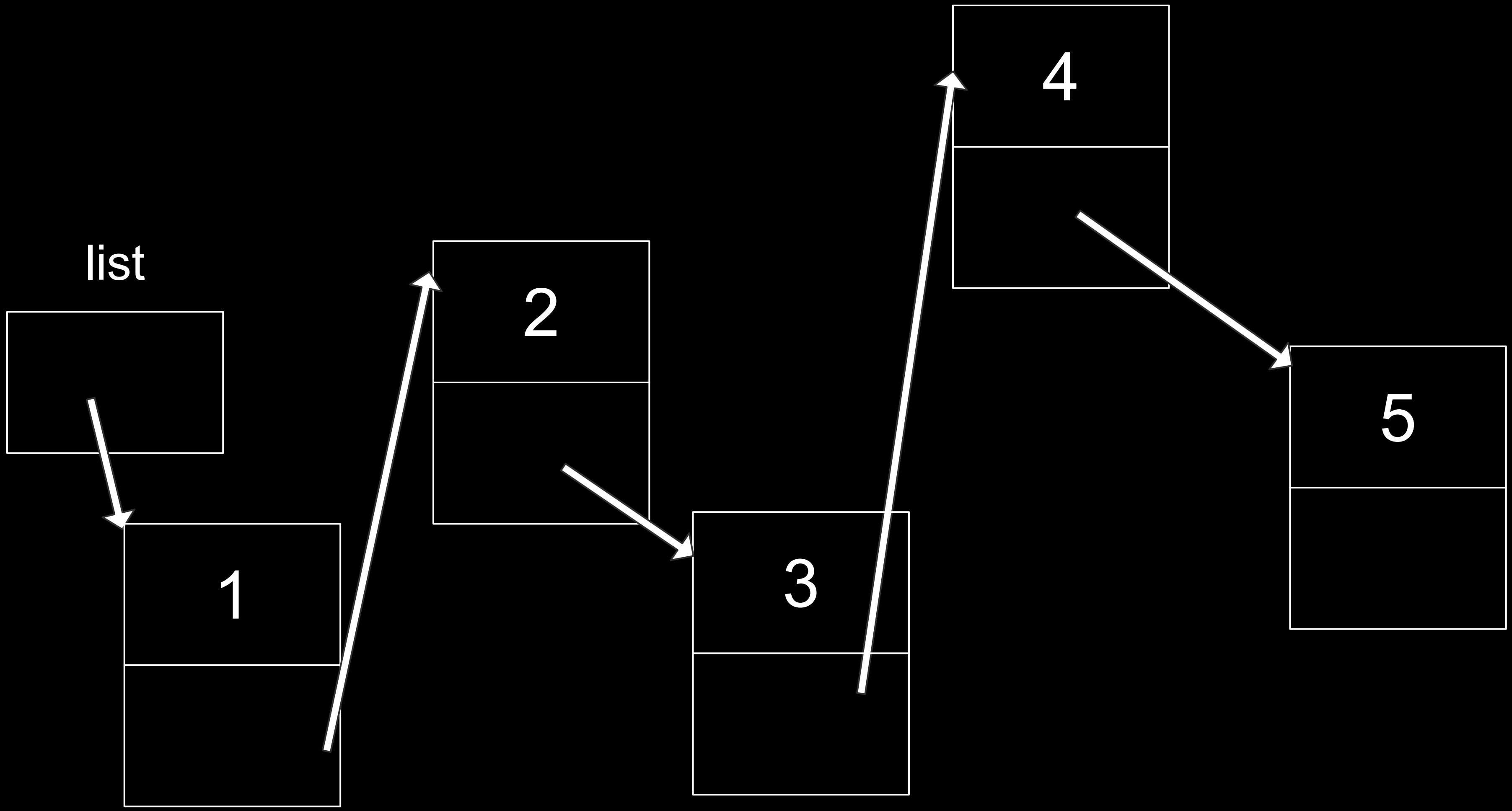

list
1
2
4
5

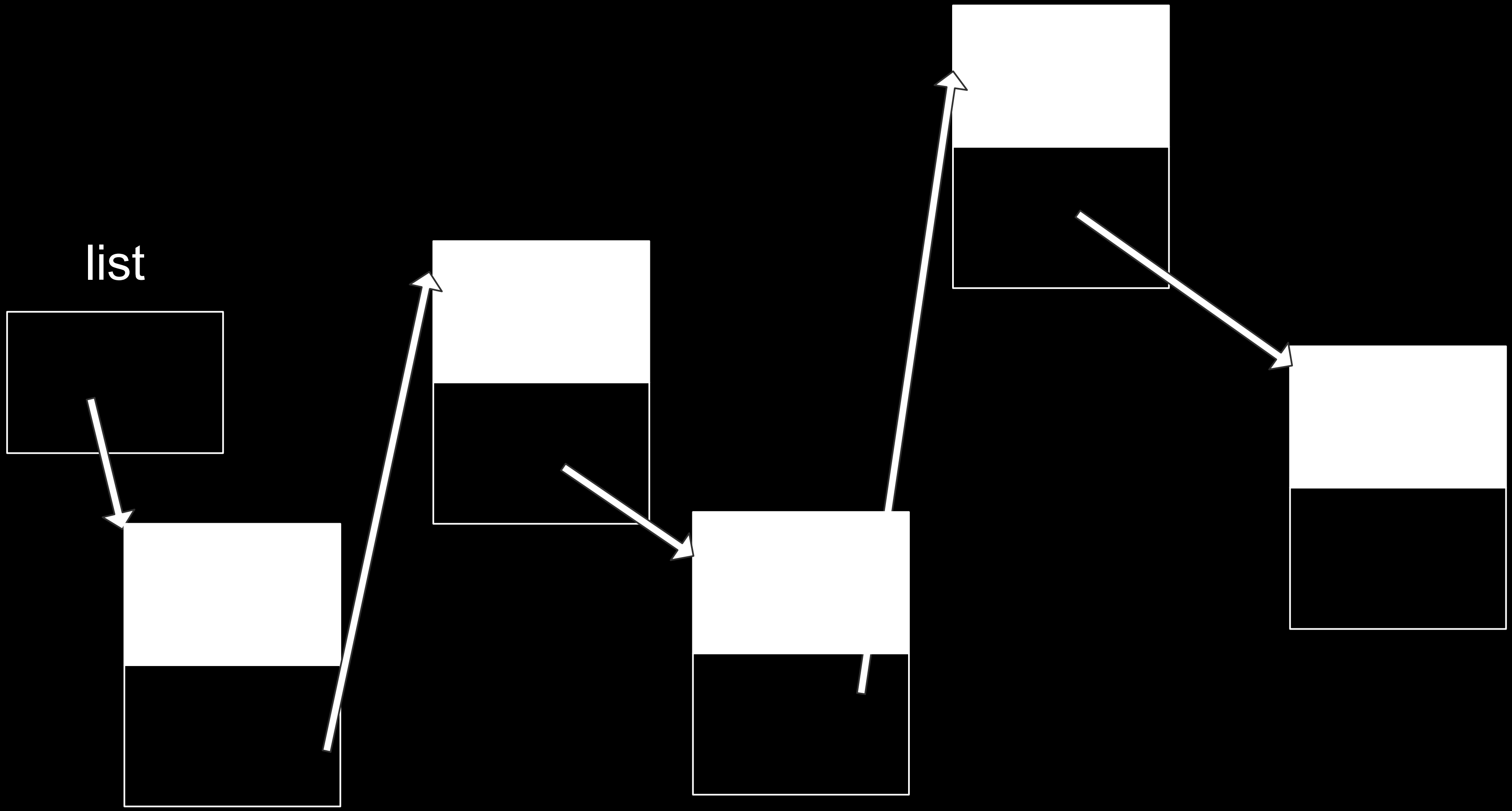
```
n->next = list;
list = n;
```

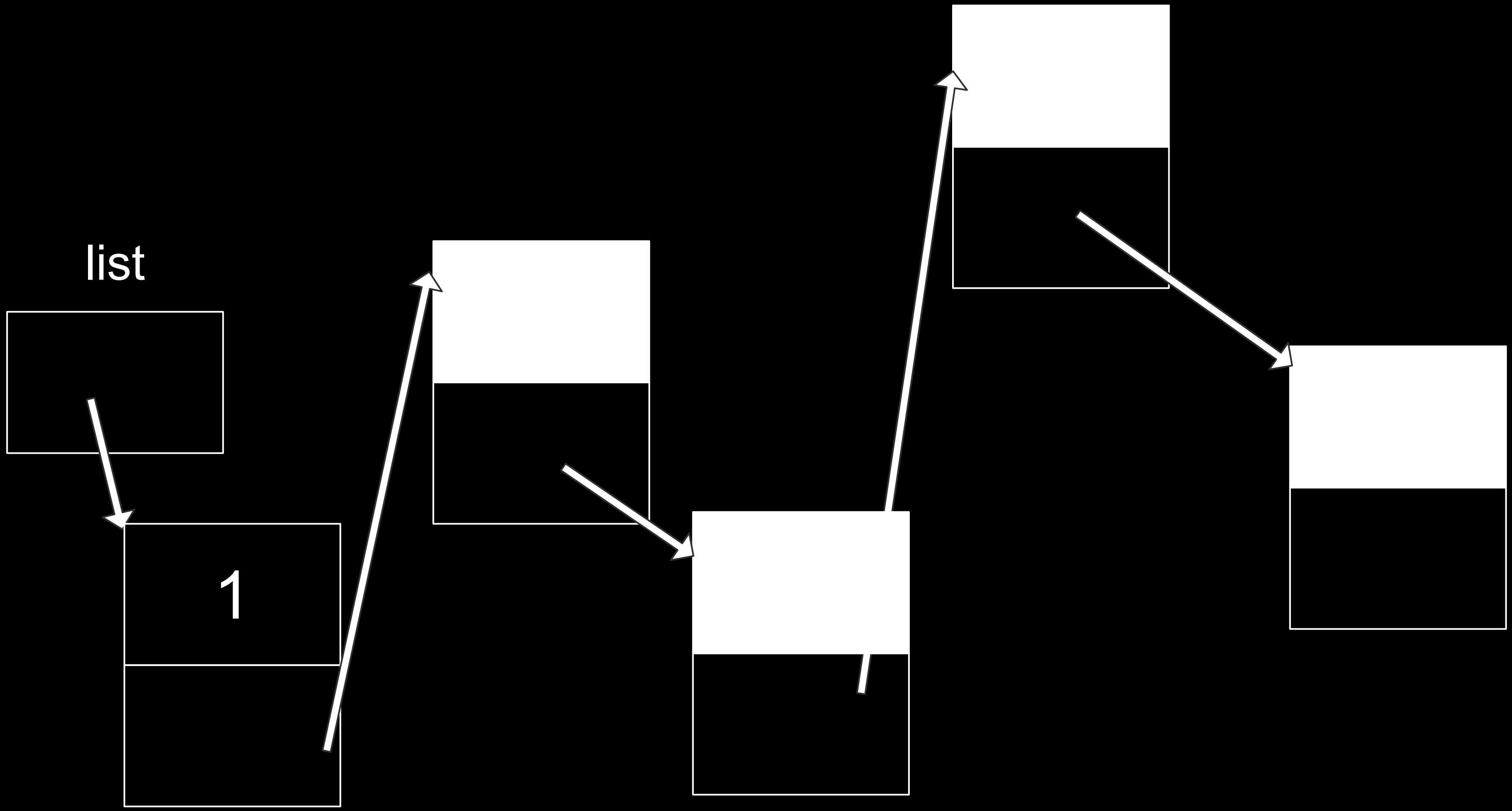

list
1
2
4
5

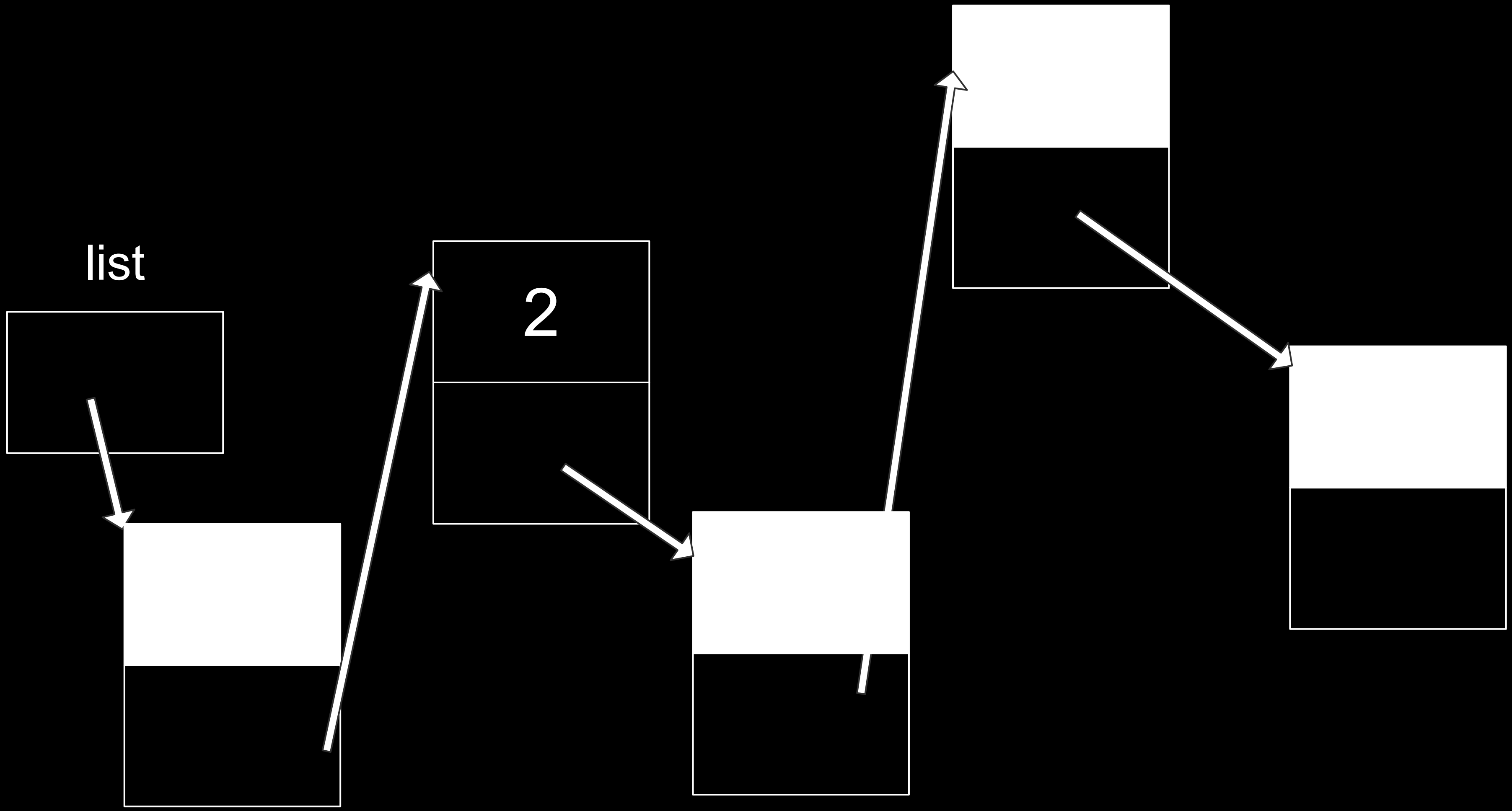

list
1
2
3
4
5

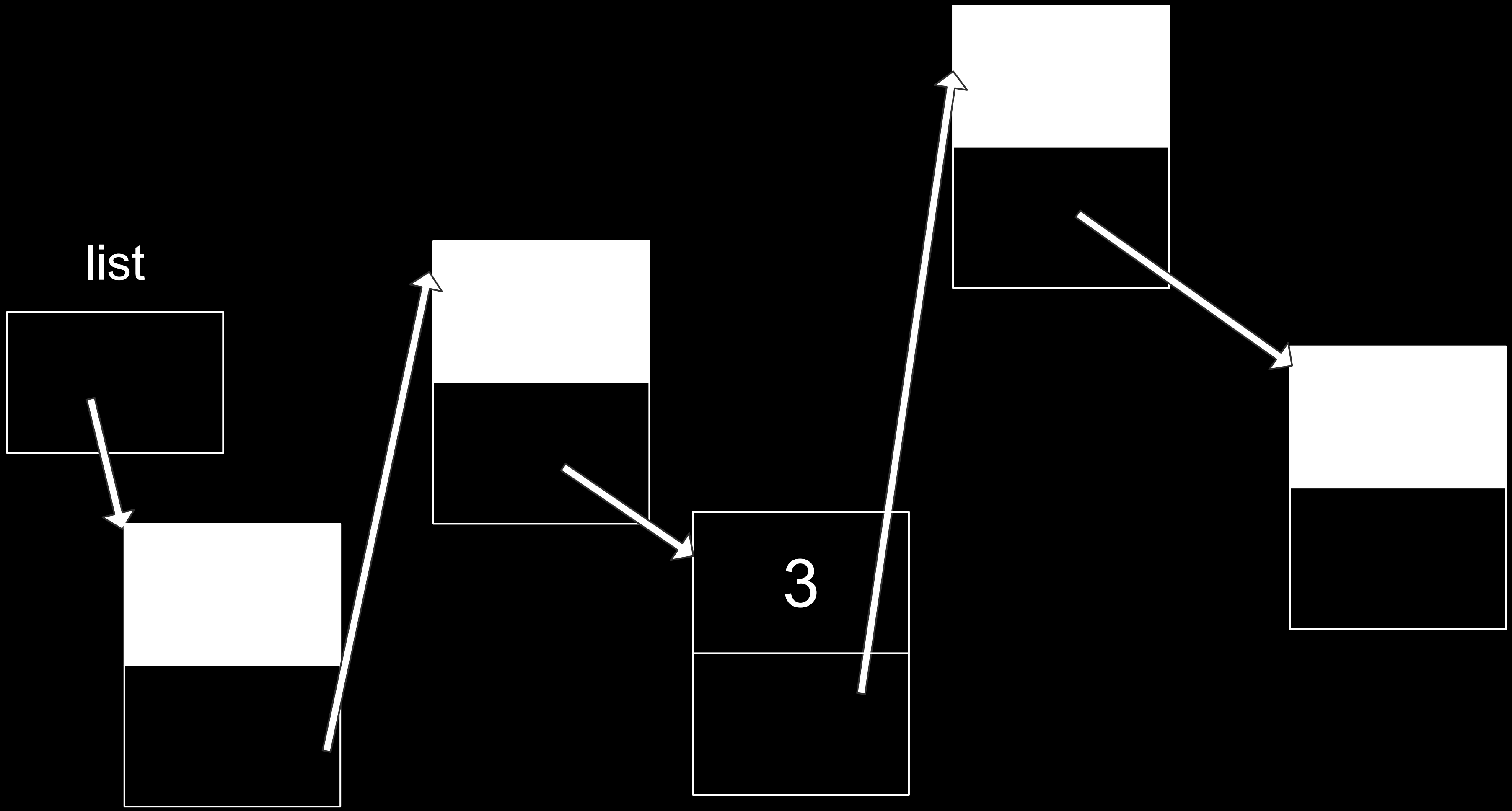

list
1
2
3
4
5

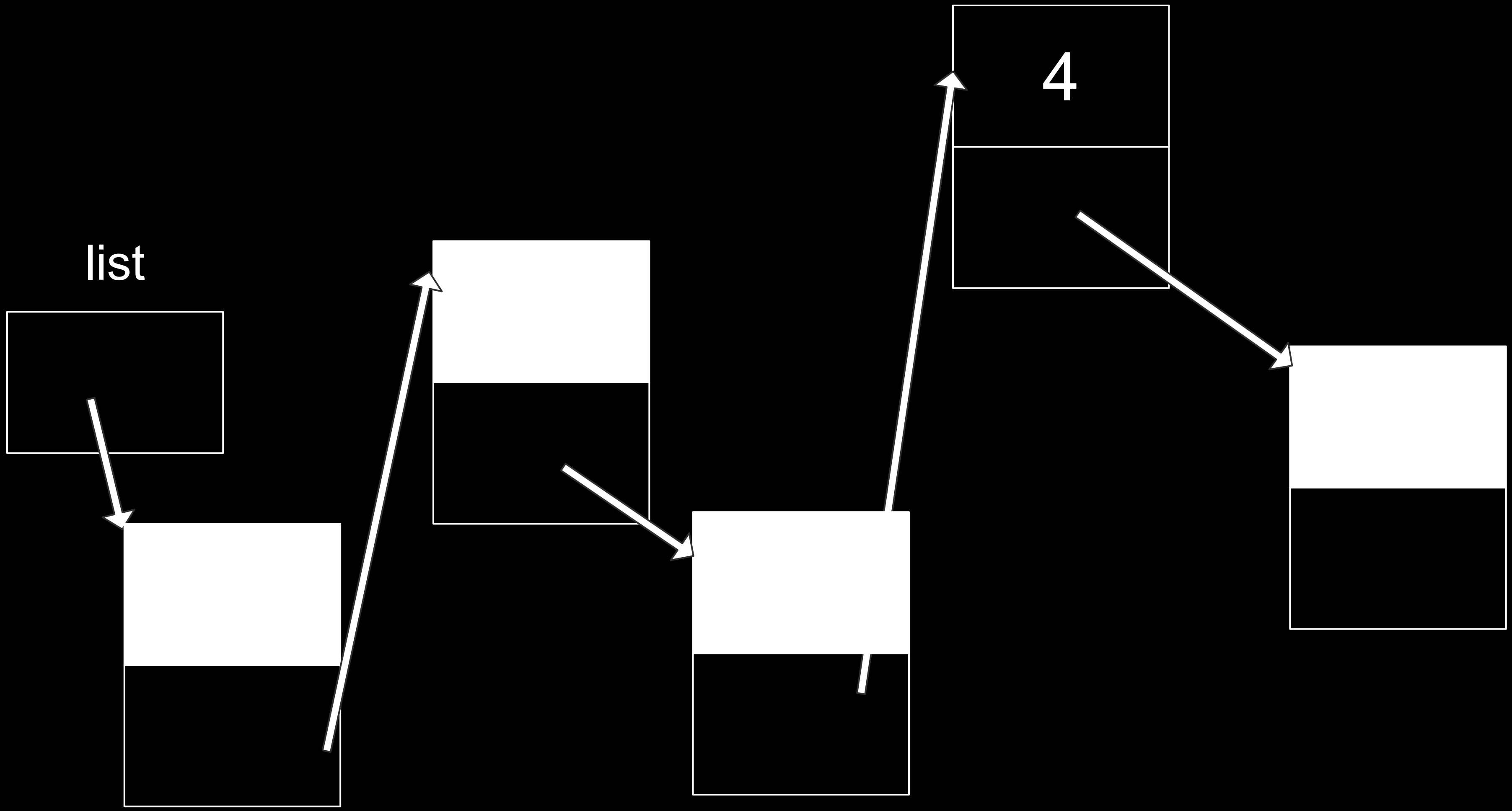

list
1
2
3
4
5

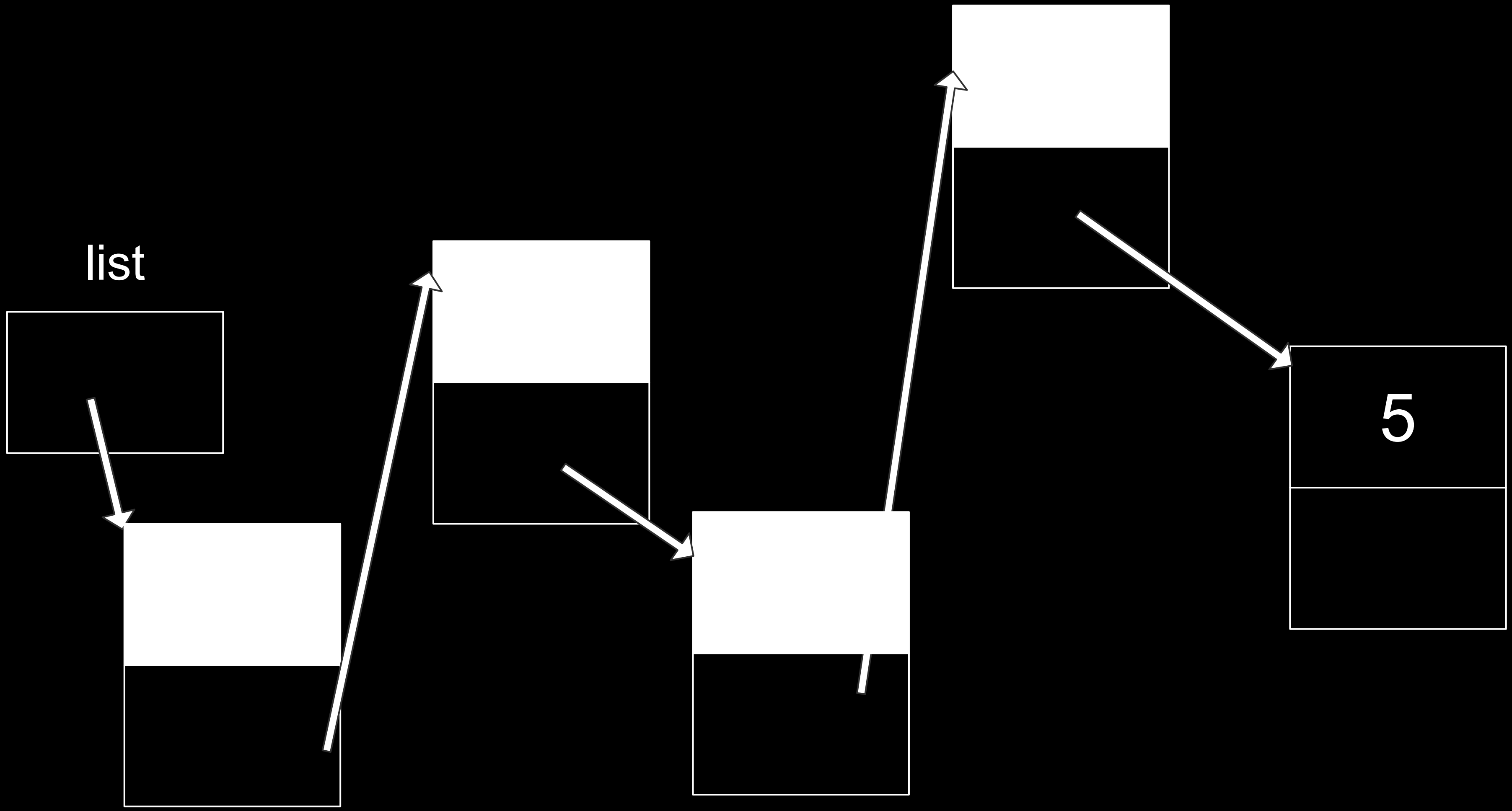

list

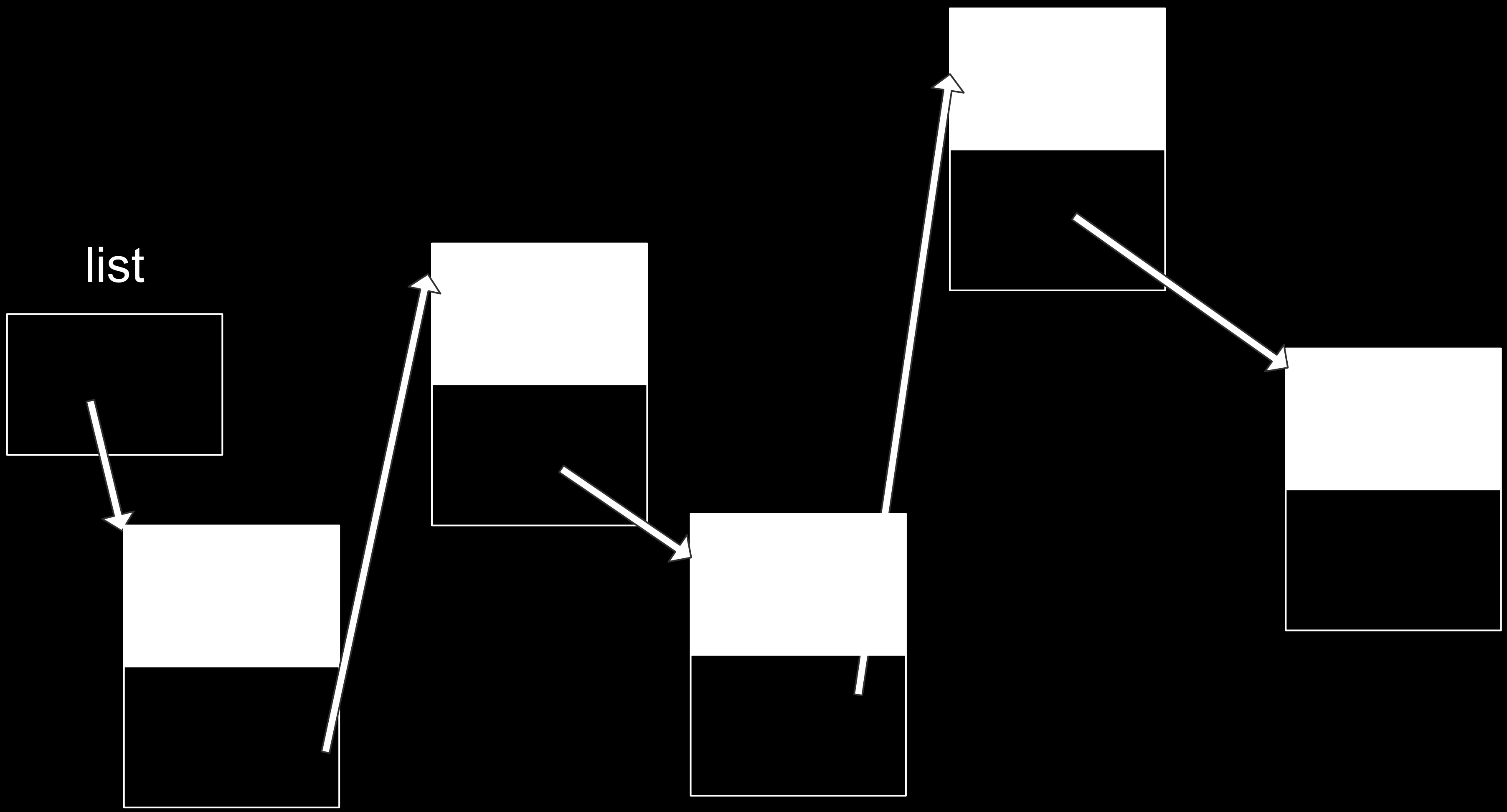

list
1


list
2


list
3


list
4


list
5


list

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

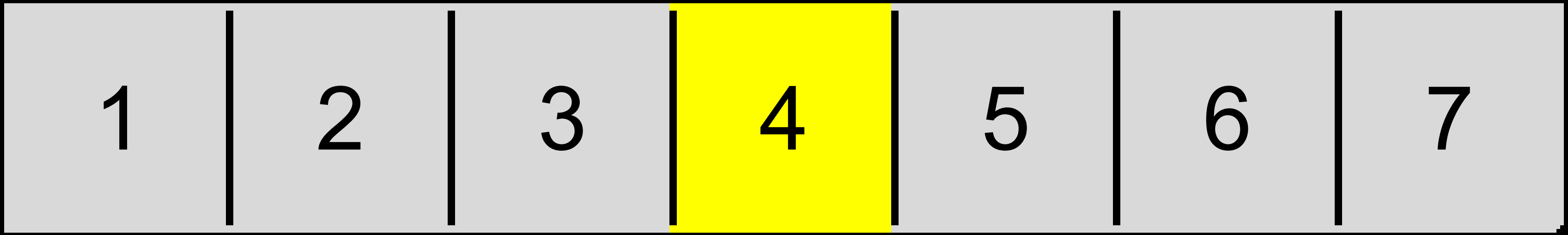
$O(n^2)$	
$O(n \log n)$	
$O(n)$	search
$O(\log n)$	
$O(1)$	

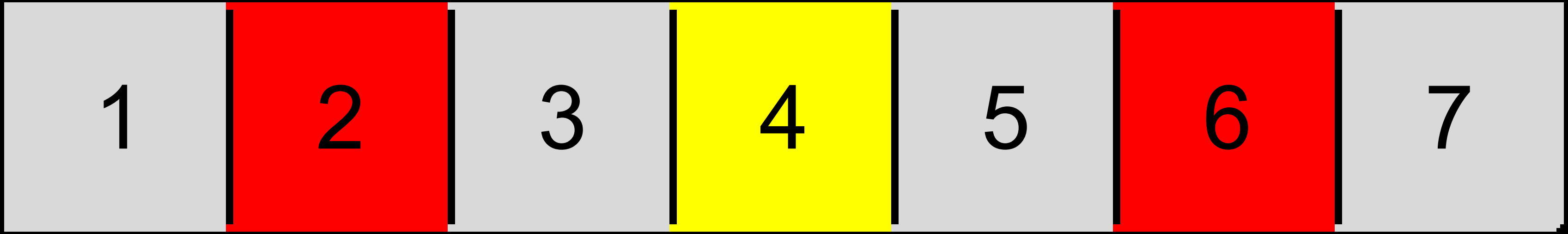
$O(n^2)$	
$O(n \log n)$	
$O(n)$	search, insert
$O(\log n)$	
$O(1)$	

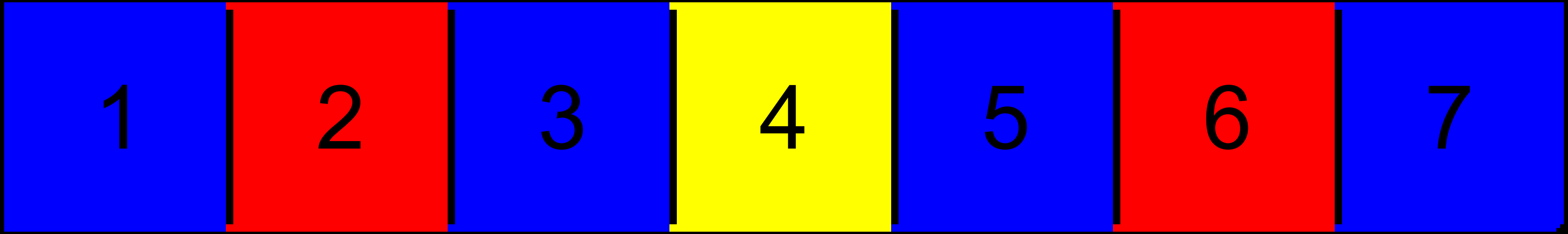
trees

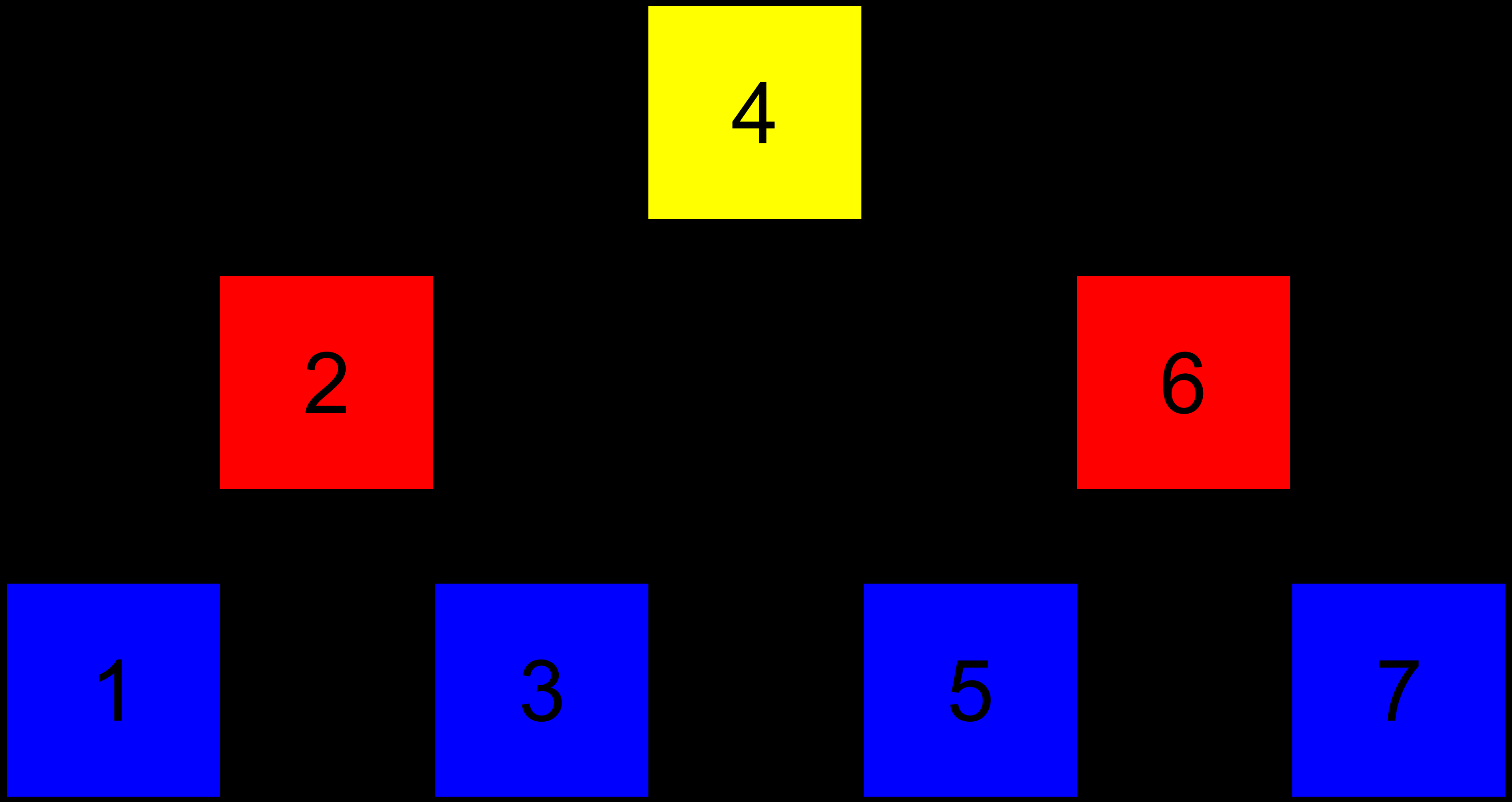
binary search trees

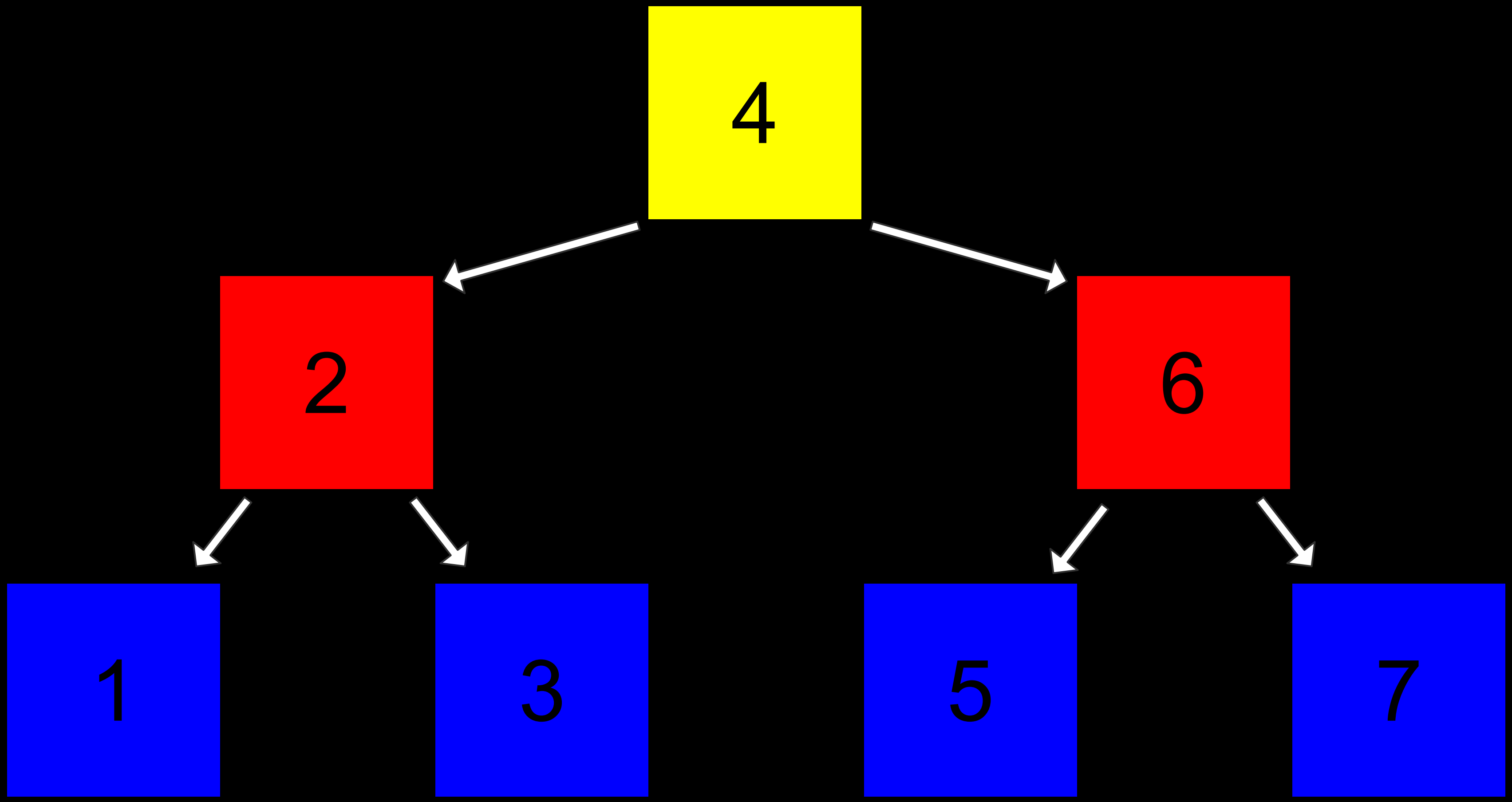
1	2	3	4	5	6	7


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

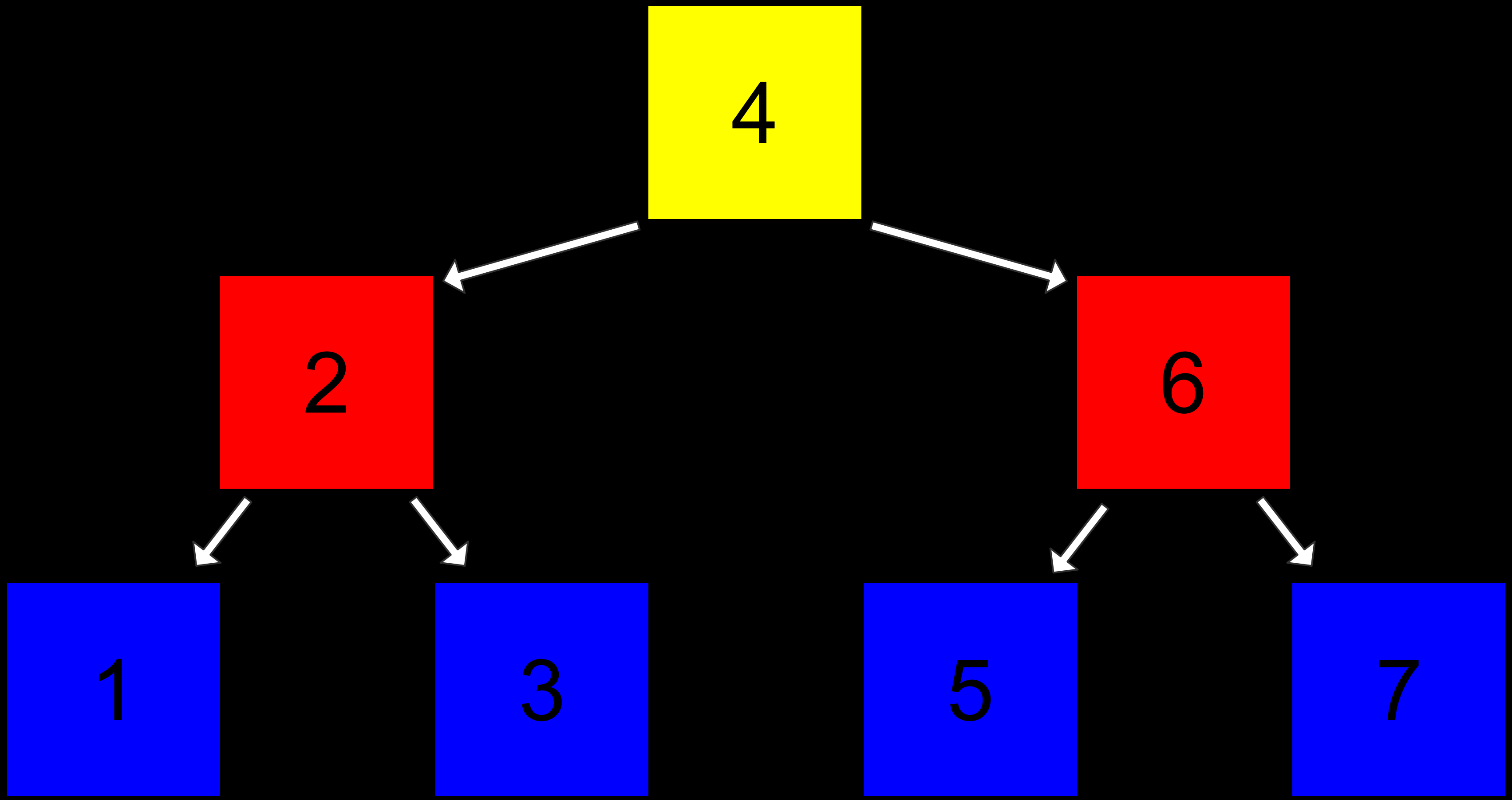

4
2
6
1
3
5
7

```
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)
{





















}
```

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









}
```

```
bool search(node *tree)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (50 < tree->number)
    {
        return search(tree->left);
    }








}
```

```
bool search(node *tree)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (50 < tree->number)
    {
        return search(tree->left);
    }
    else if (50 > tree->number)
    {
        return search(tree->right);
    }




}
```

```
bool search(node *tree)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (50 < tree->number)
    {
        return search(tree->left);
    }
    else if (50 > tree->number)
    {
        return search(tree->right);
    }
    else if (50 == tree->number)
    {
        return true;
    }
}
```

```
bool search(node *tree)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (50 < tree->number)
    {
        return search(tree->left);
    }
    else if (50 > tree->number)
    {
        return search(tree->right);
    }
    else
    {
        return true;
    }
}
```

$O(n^2)$

$O(n \log n)$

$O(n)$

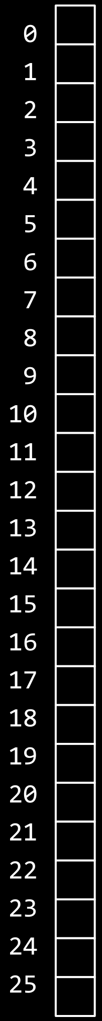
$O(\log n)$

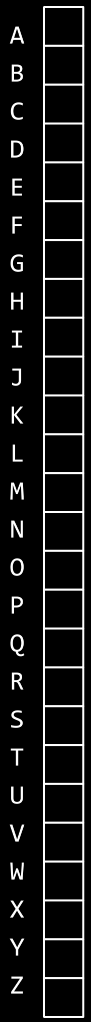
$O(1)$

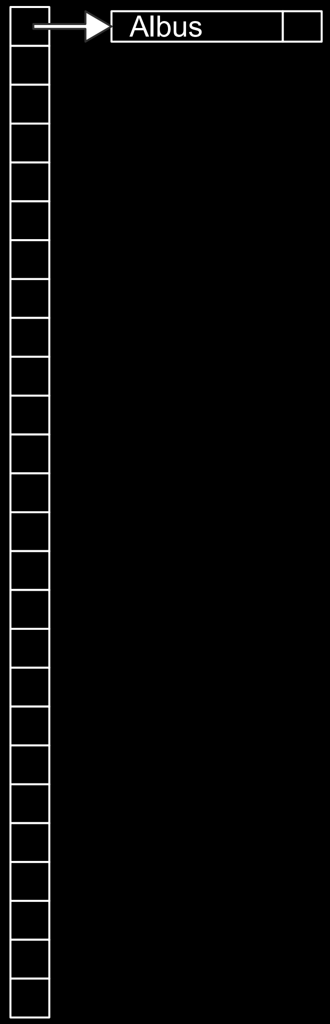
$O(n^2)$	
$O(n \log n)$	
$O(n)$	
$O(\log n)$	search
$O(1)$	

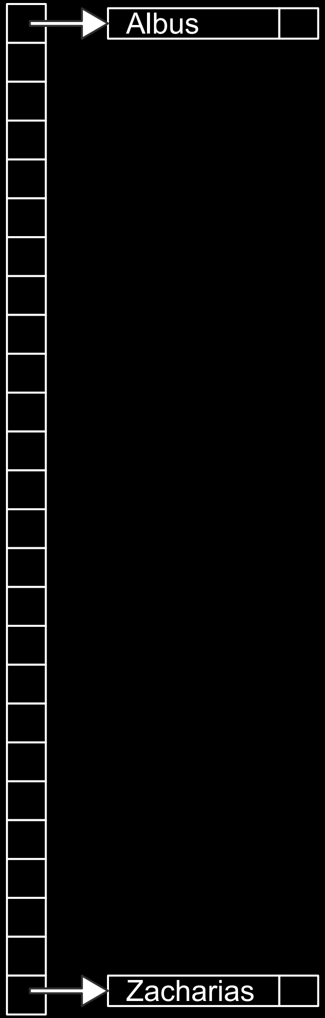
$O(n^2)$	
$O(n \log n)$	
$O(n)$	
$O(\log n)$	search, insert
$O(1)$	

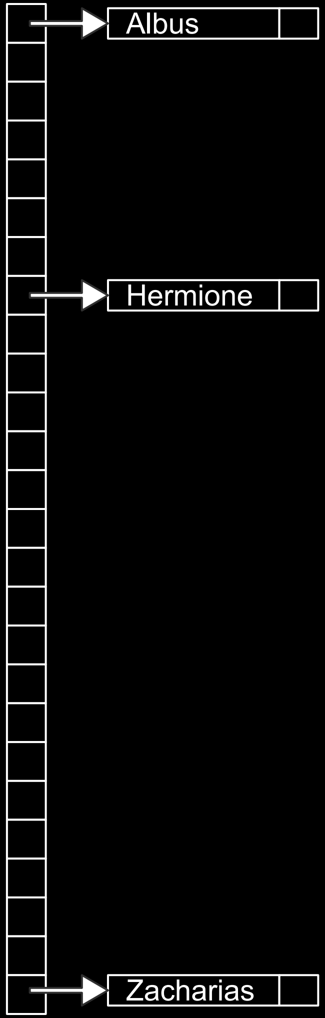
hash tables

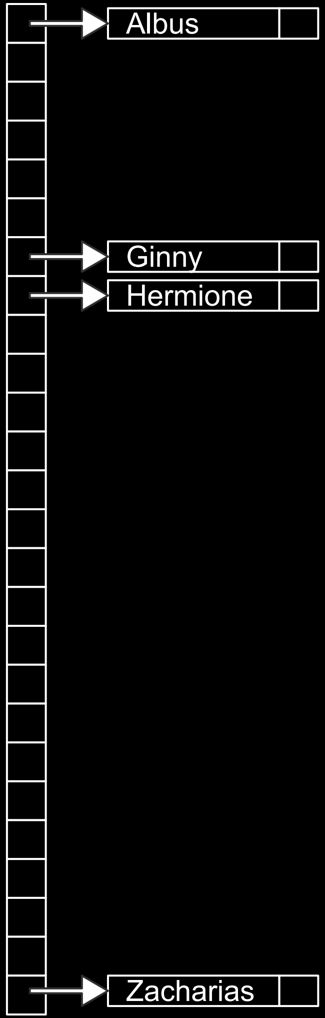

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

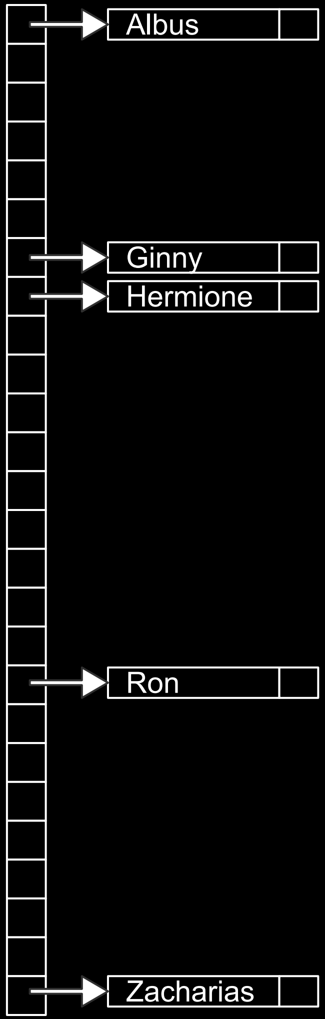

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

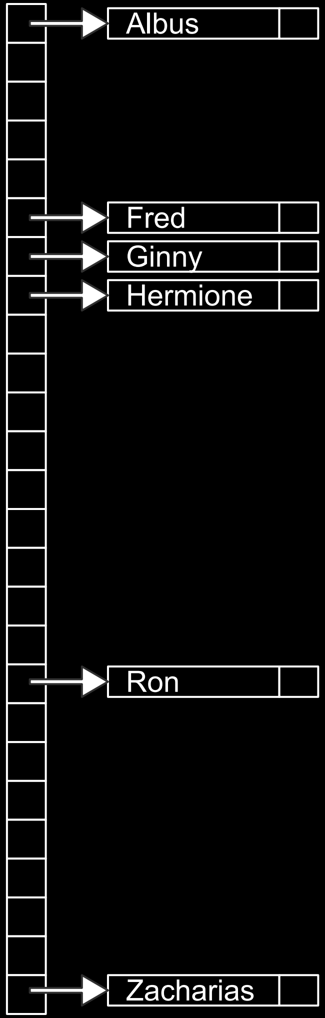

Albus

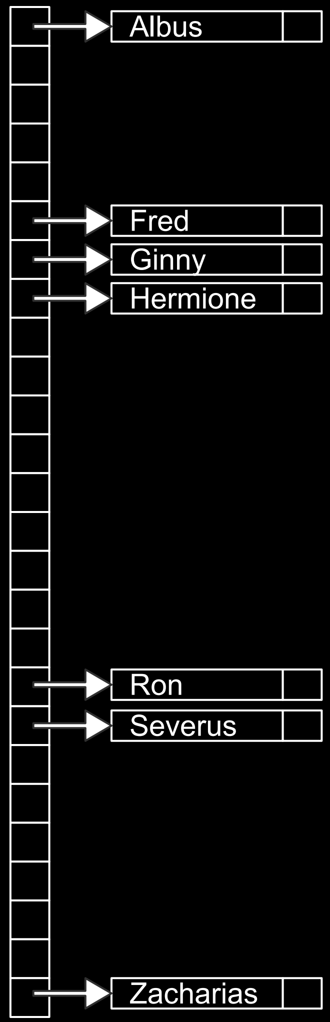

Albus
Zacharias

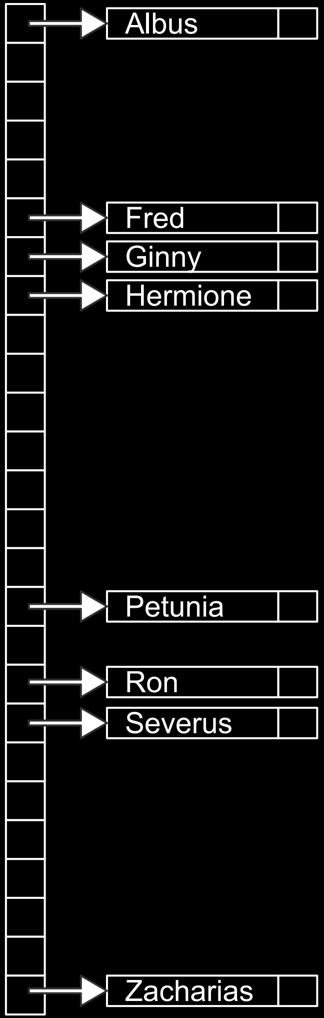

Albus
Hermione
Zacharias


Albus
Ginny
Hermione
Zacharias

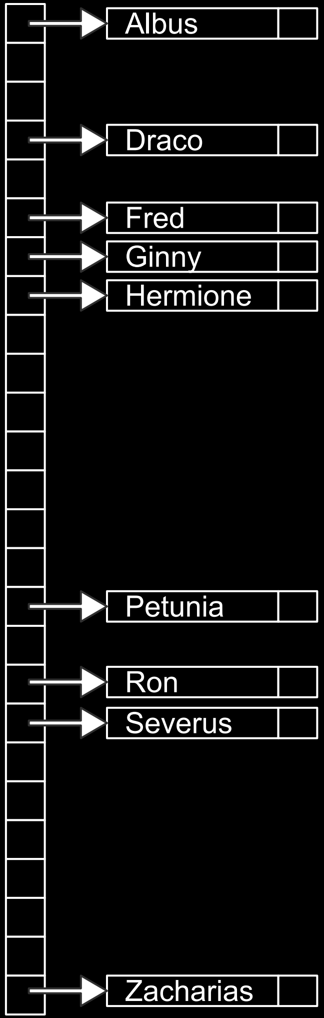

Albus
Ginny
Hermione
Ron
Zacharias

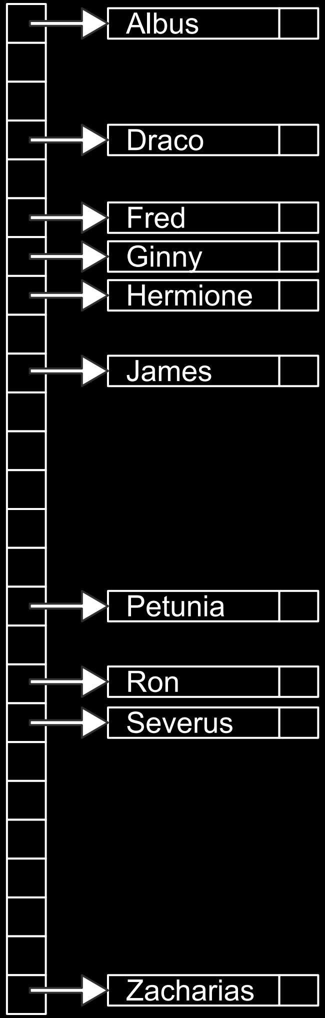

Albus
Fred
Ginny
Hermione
Ron
Zacharias

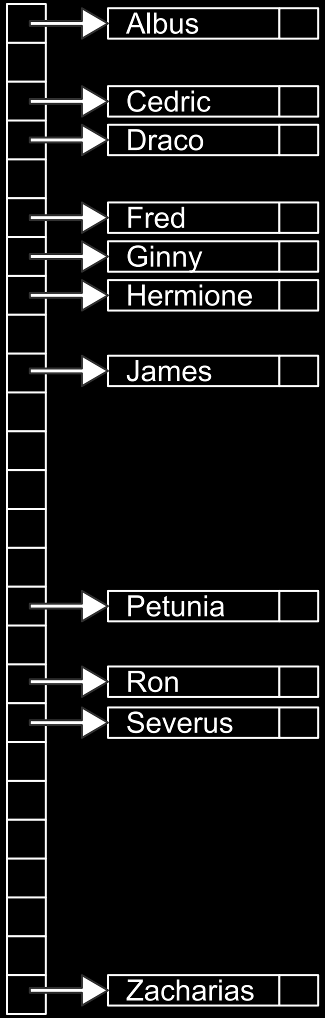

Albus
Fred
Ginny
Hermione
Ron
Severus
Zacharias

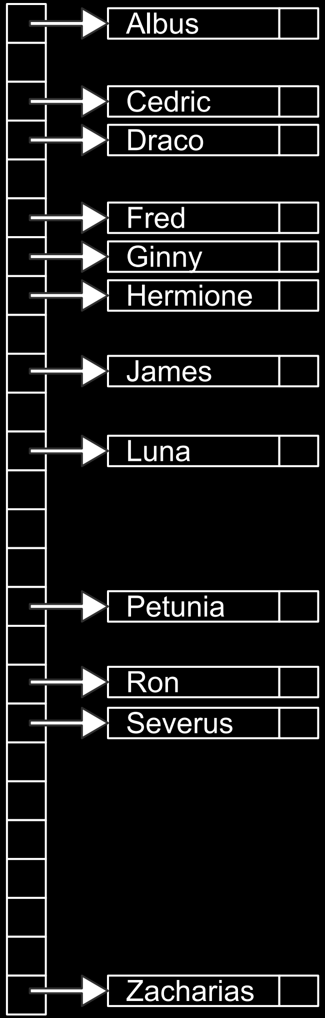


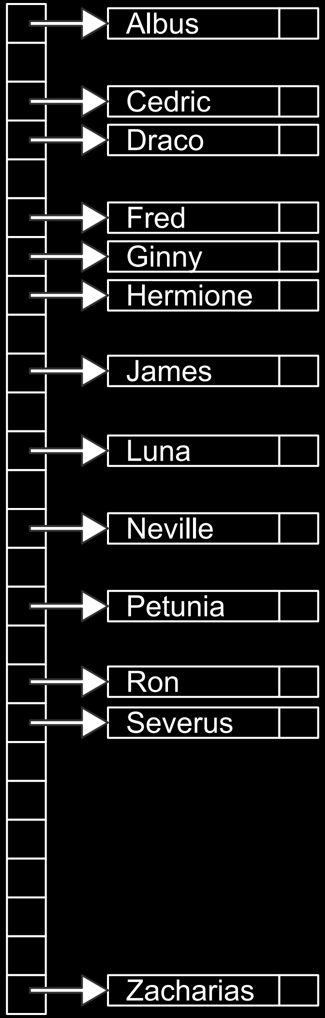
Albus
Fred
Ginny
Hermione
Petunia
Ron
Severus
Zacharias

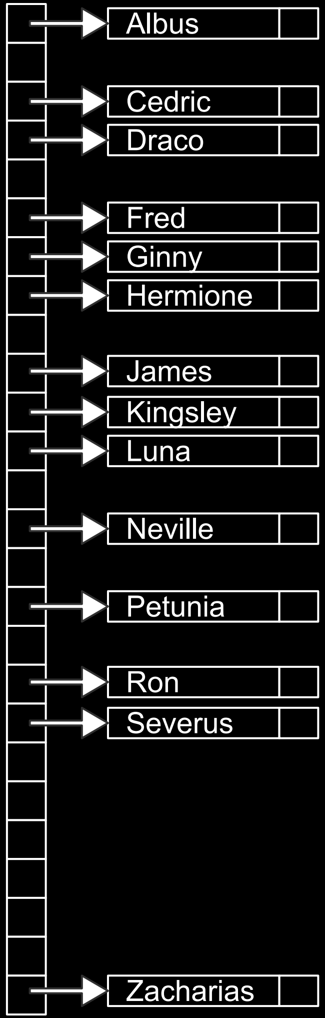

Albus
Draco
Fred
Ginny
Hermione
Petunia
Ron
Severus
Zacharias

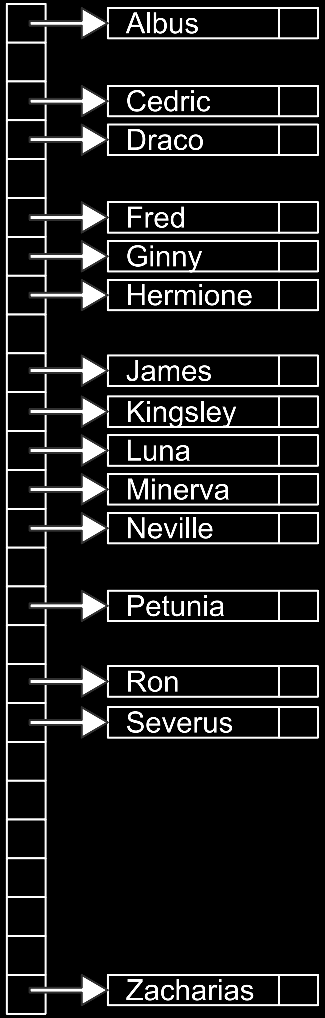

Albus
Draco
Fred
Ginny
Hermione
James
Petunia
Ron
Severus
Zacharias

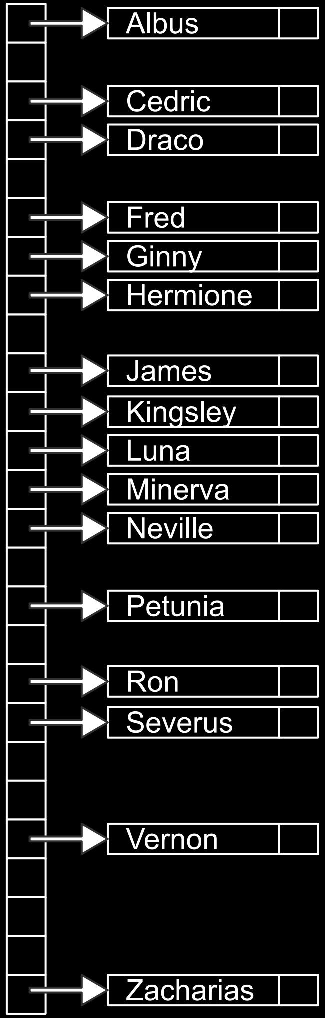

Albus
Cedric
Draco
Fred
Ginny
Hermione
James
Petunia
Ron
Severus
Zacharias

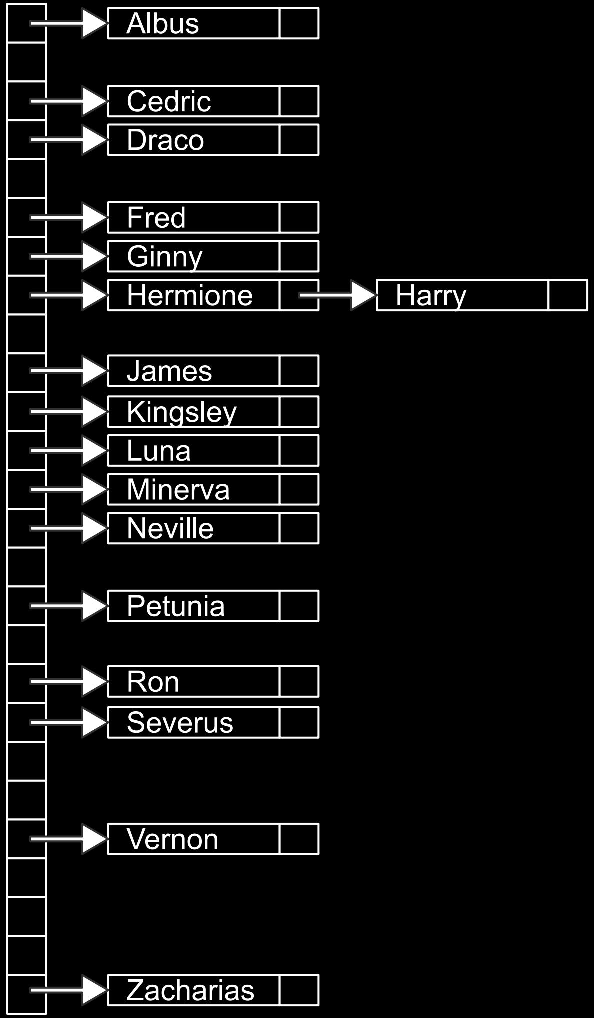

Albus
Cedric
Draco
Fred
Ginny
Hermione
James
Luna
Petunia
Ron
Severus
Zacharias

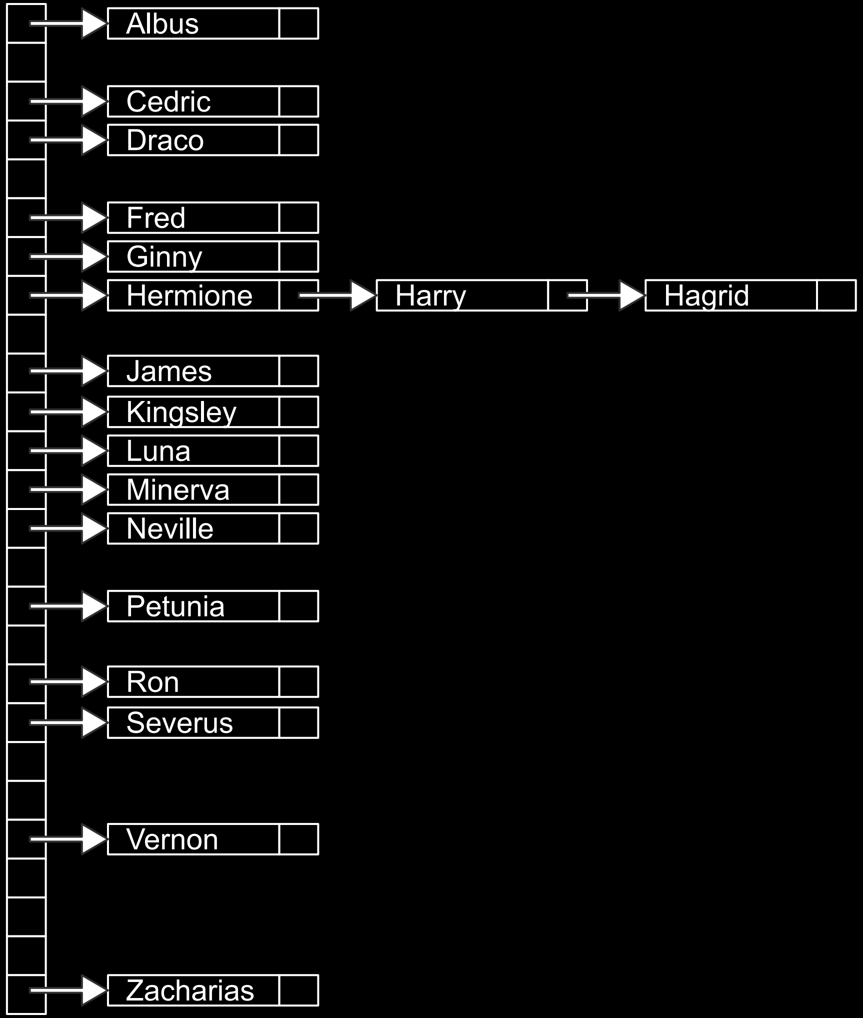

Albus
Cedric
Draco
Fred
Ginny
Hermione
James
Luna
Neville
Petunia
Ron
Severus
Zacharias

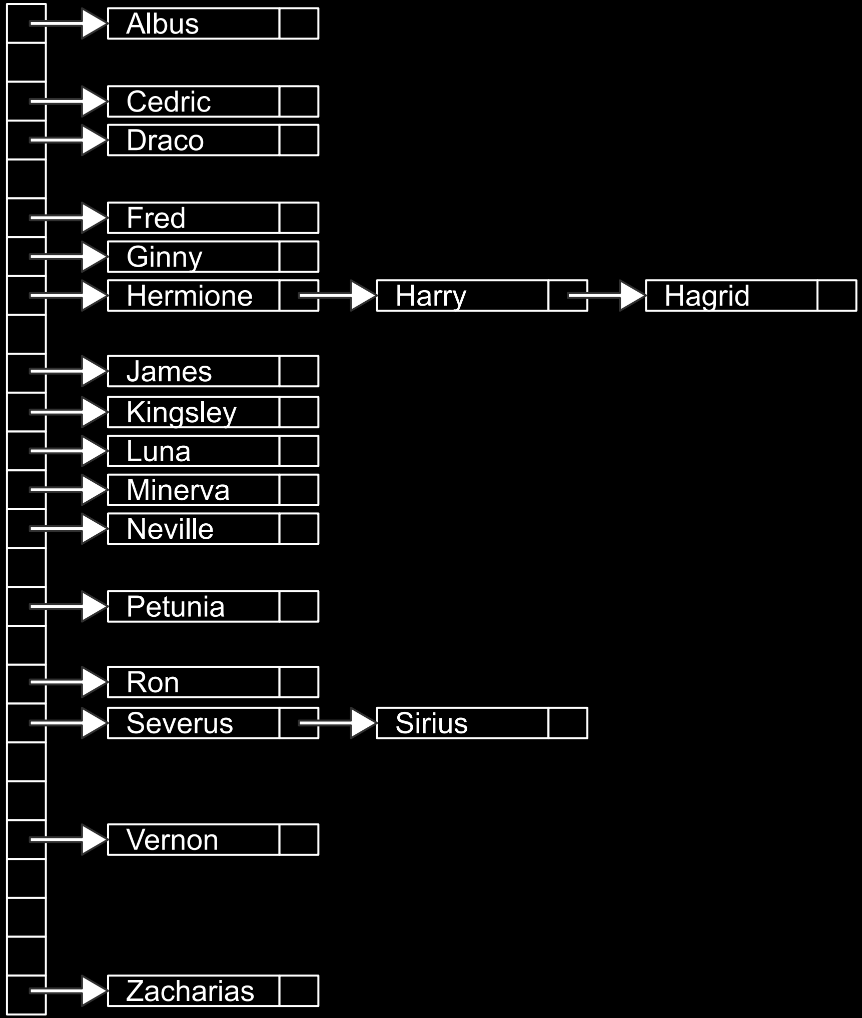

Albus
Cedric
Draco
Fred
Ginny
Hermione
James
Kingsley
Luna
Neville
Petunia
Ron
Severus
Zacharias

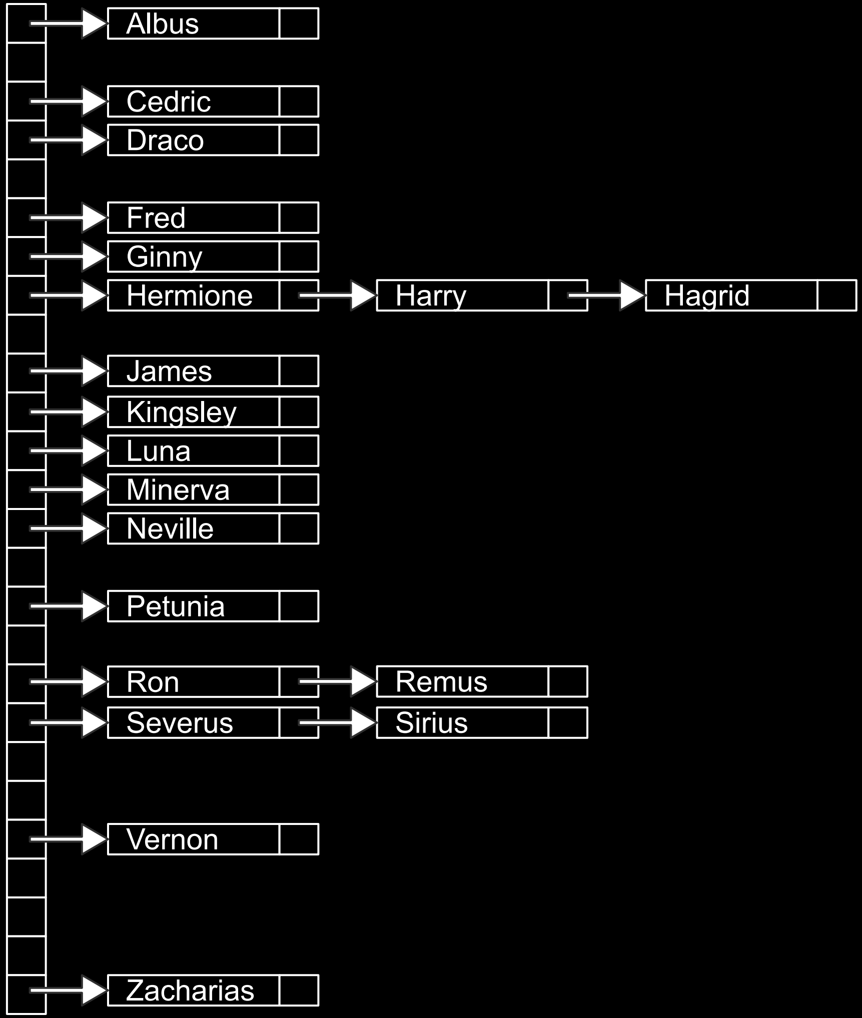

Albus
Cedric
Draco
Fred
Ginny
Hermione
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Zacharias

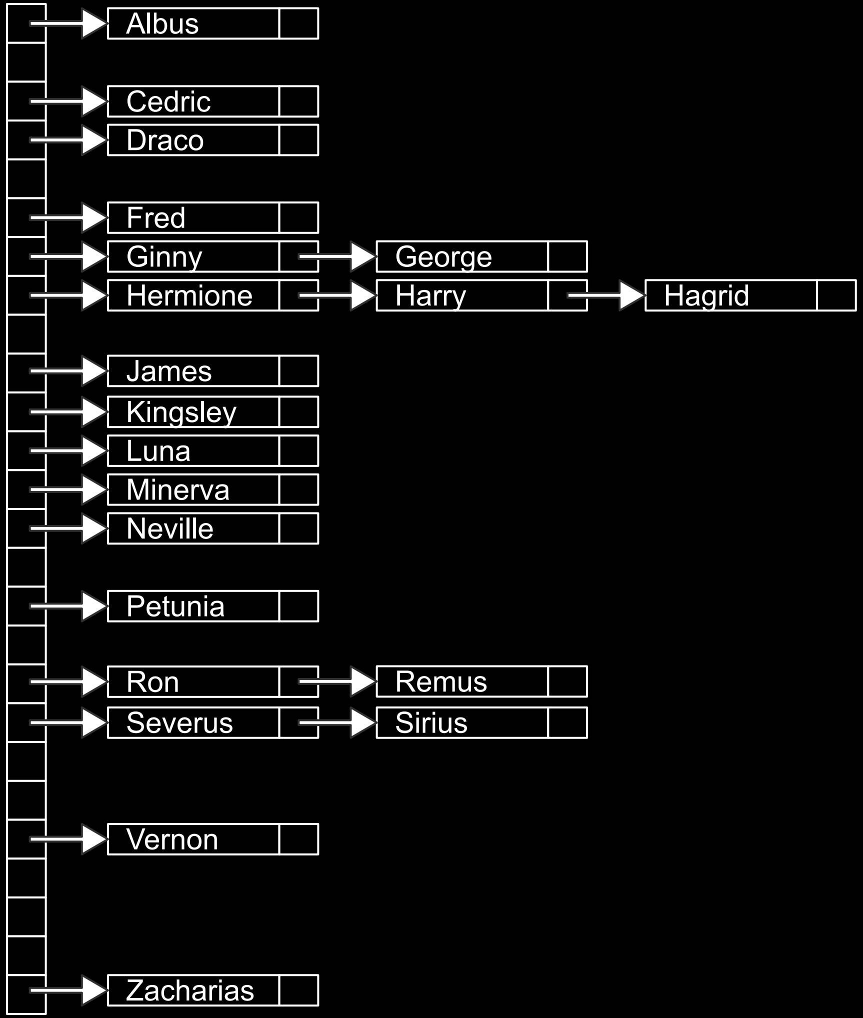

Albus
Cedric
Draco
Fred
Ginny
Hermione
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Vernon
Zacharias

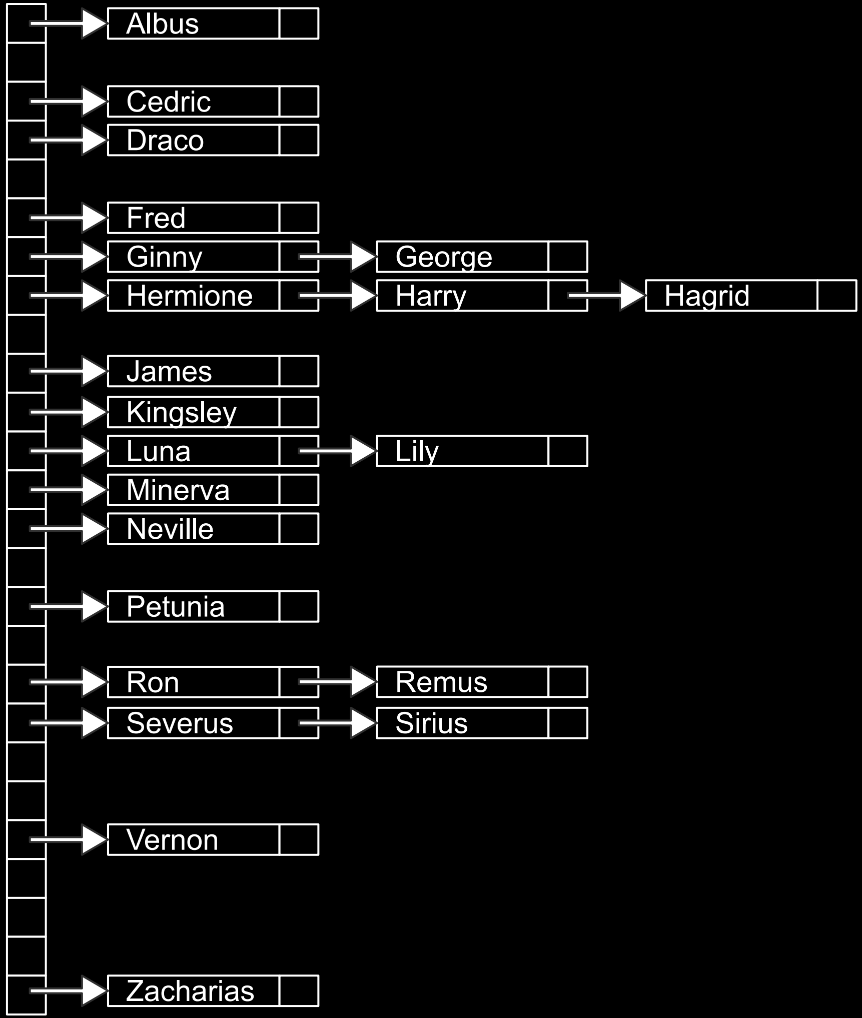

Albus
Cedric
Draco
Fred
Ginny
Hermione
Harry
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Vernon
Zacharias

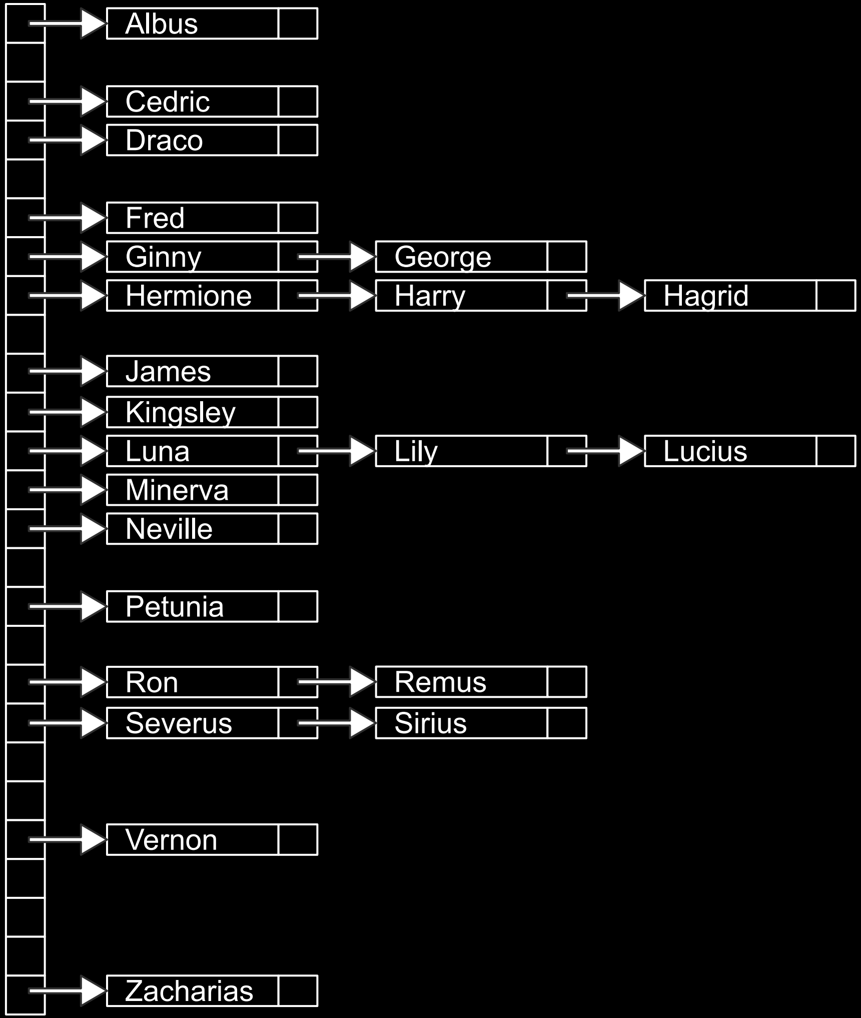

Albus
Cedric
Draco
Fred
Ginny
Hermione
Harry
Hagrid
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Vernon
Zacharias

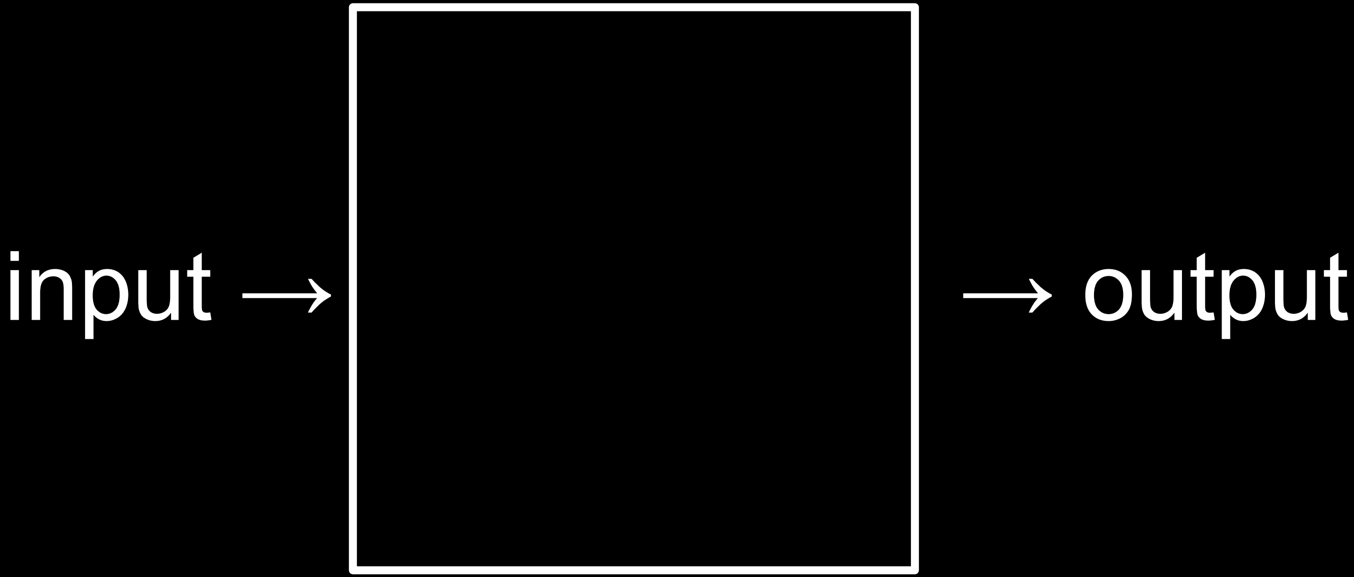

Albus
Cedric
Draco
Fred
Ginny
Hermione
Harry
Hagrid
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Sirius
Vernon
Zacharias


Albus
Cedric
Draco
Fred
Ginny
Hermione
Harry
Hagrid
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

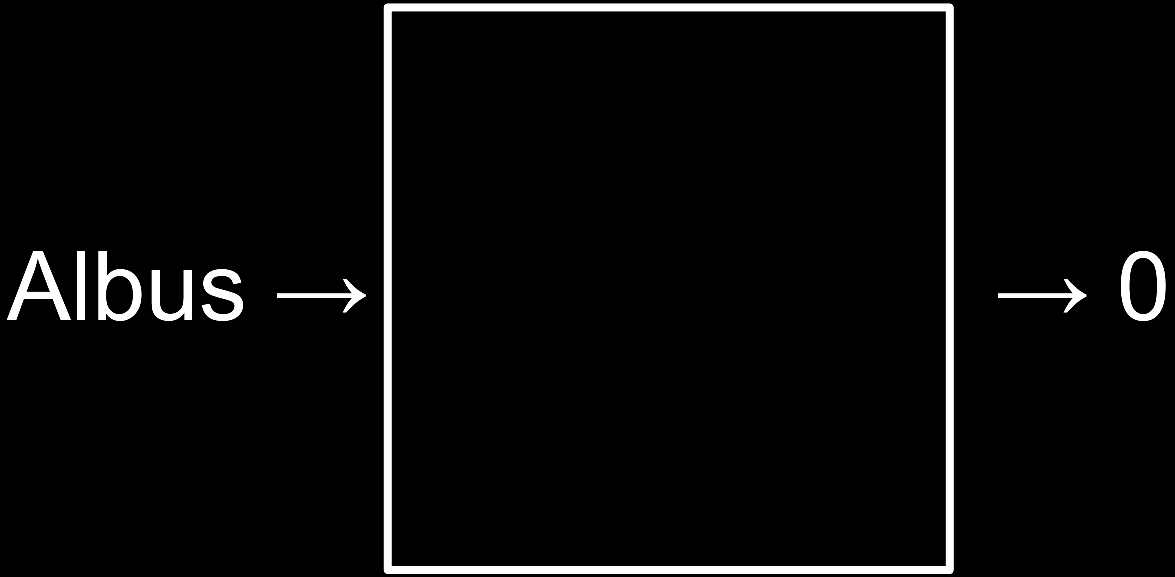

Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias


Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Lily
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

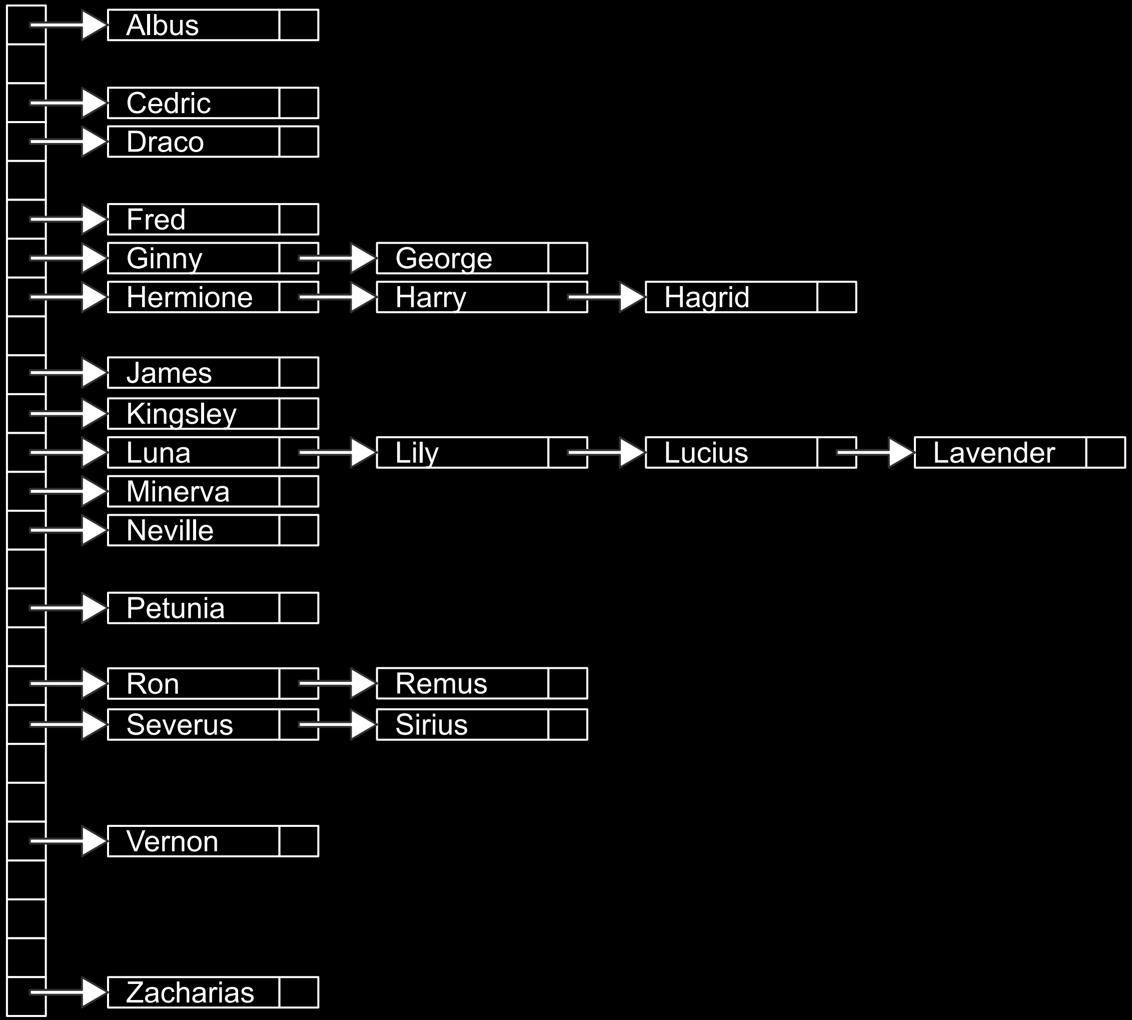

Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Lily
Lucius
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

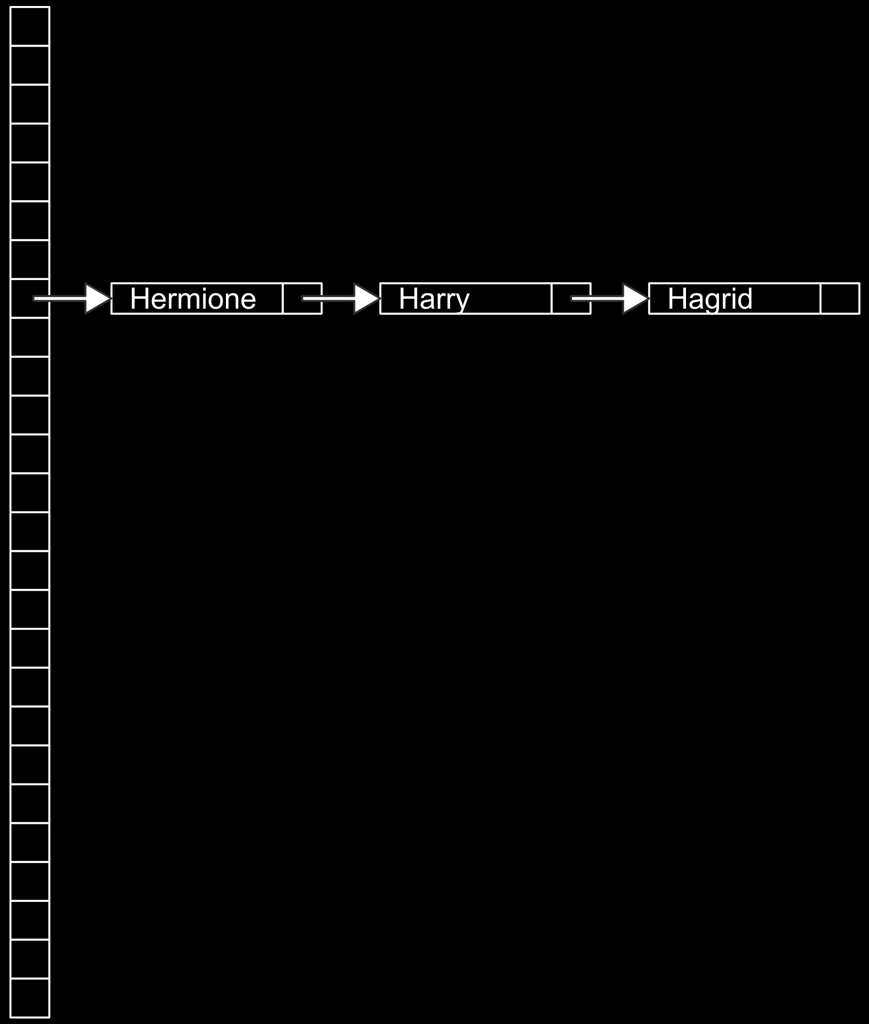

input
output

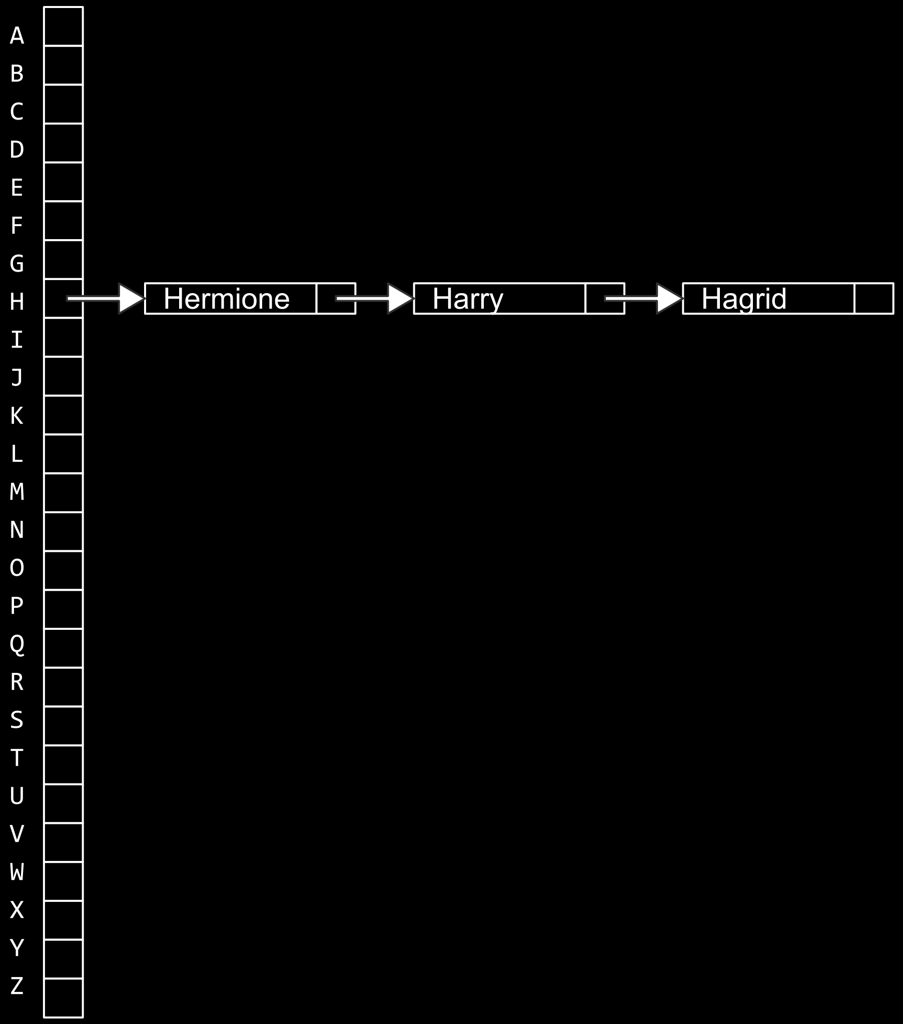
hash function

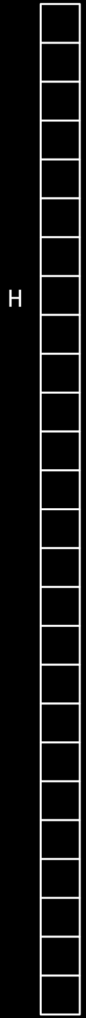

Albus
0

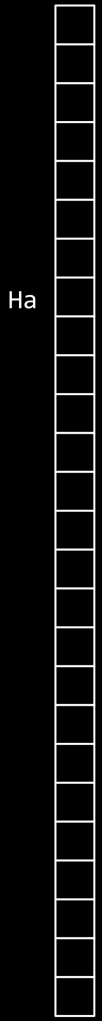
$$\text{Zacharias} \rightarrow \boxed{} \rightarrow 25$$

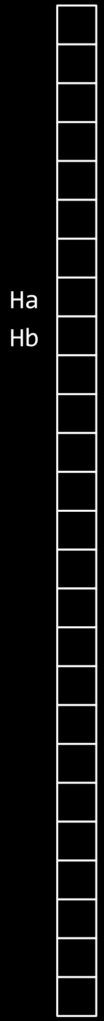

Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Lily
Lucius
Lavender
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

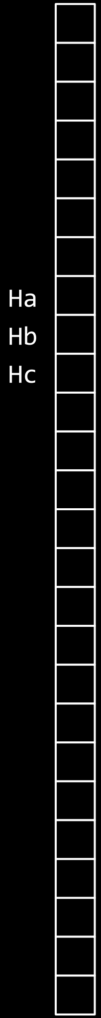

Hermione
Harry
Hagrid

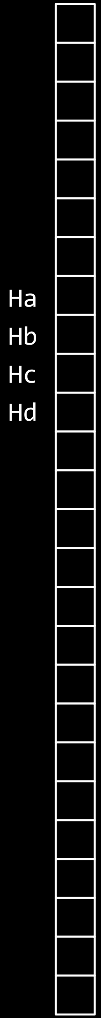

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
Hermione
Harry
Hagrid

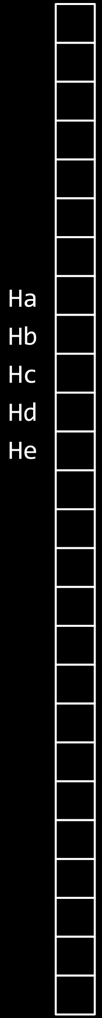

H

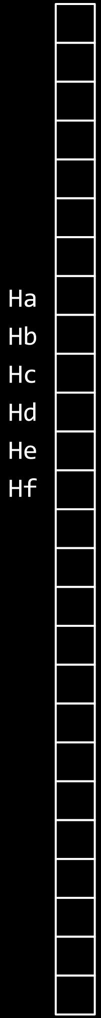

Ha

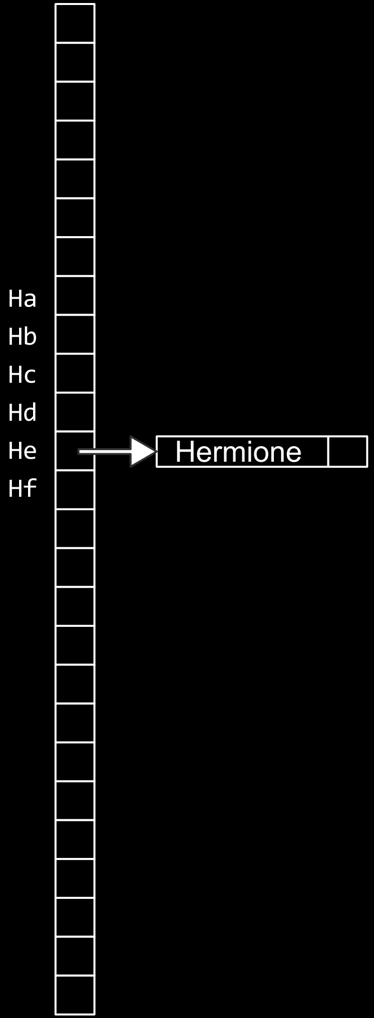

Ha
Hb

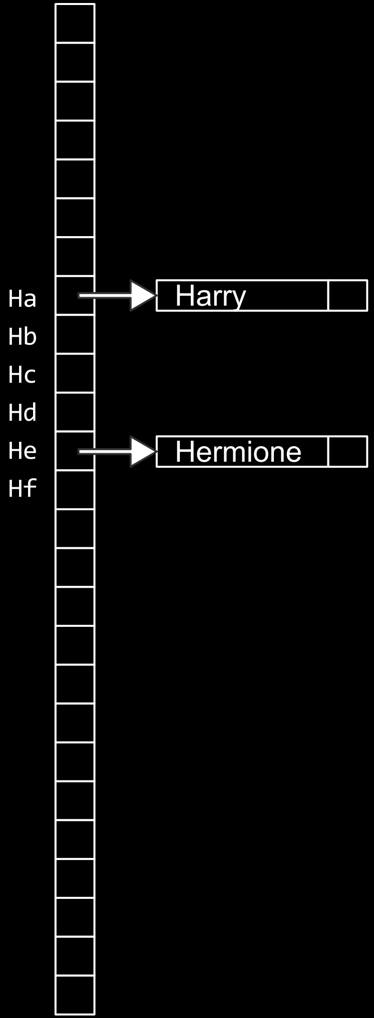

Ha
Hb
Hc

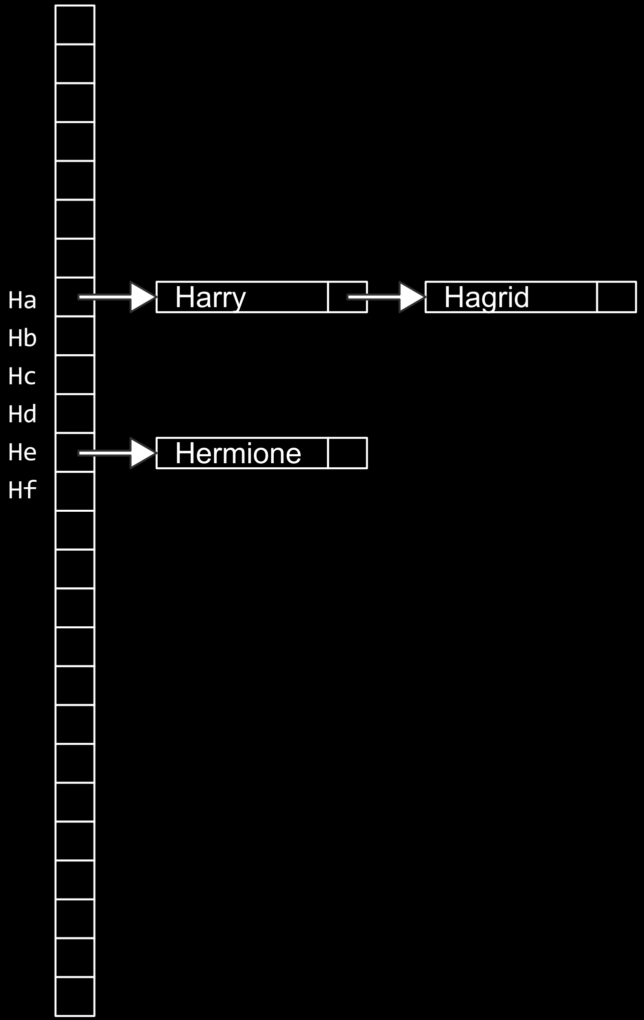

Ha
Hb
Hc
Hd

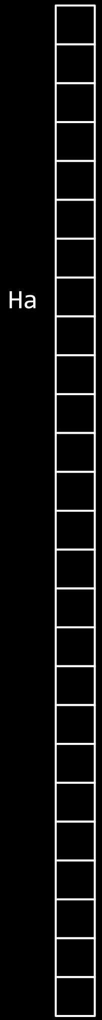

Ha
Hb
Hc
Hd
He

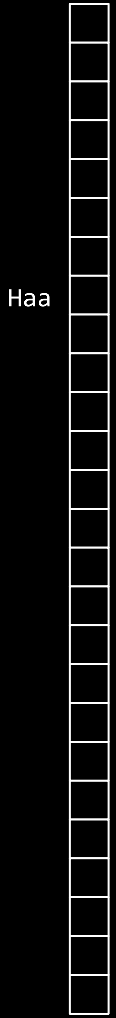

Ha
Hb
Hc
Hd
He
Hf

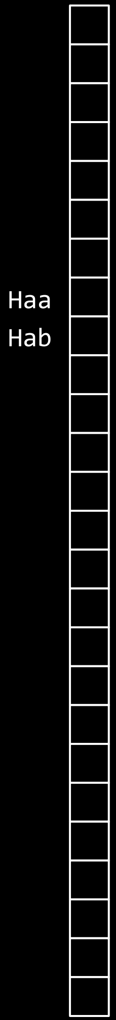

Ha
Hb
Hc
Hd
He
Hf
Hermione

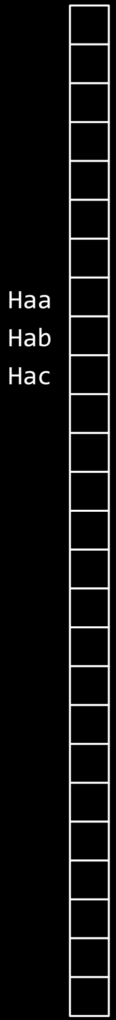

Ha
Hb
Hc
Hd
He
Hf
Harry
Hermione

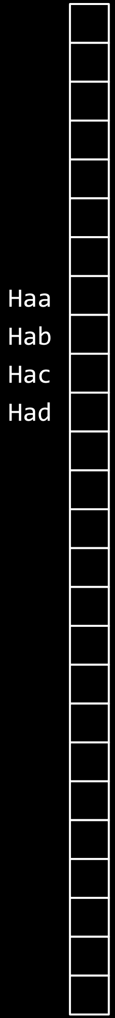

Ha
Hb
Hc
Hd
He
Hf
Harry
Hagrid
Hermione

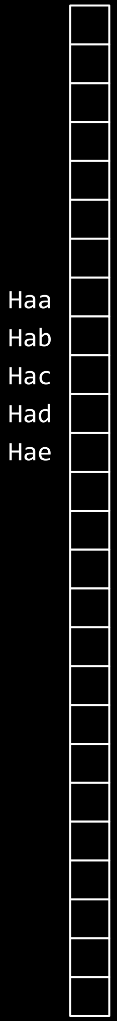

Ha

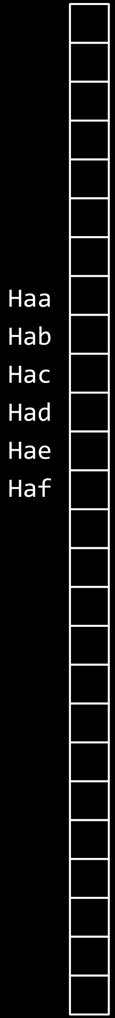

Haa

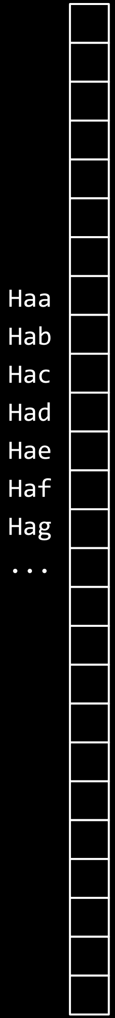

Haa
Hab

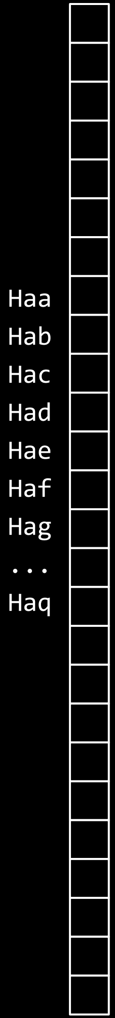

Haa
Hab
Hac

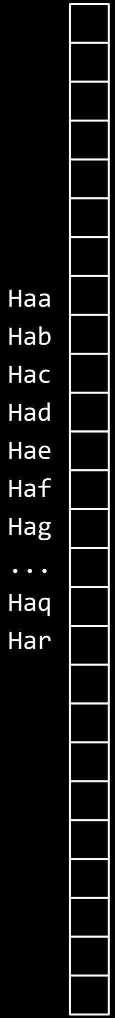

Haa
Hab
Hac
Had

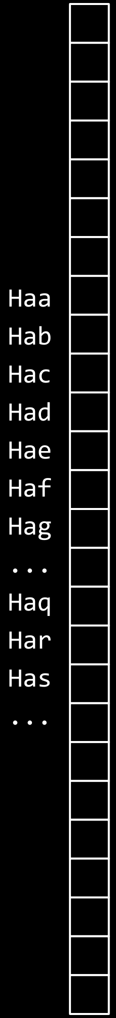

Haa
Hab
Hac
Had
Hae

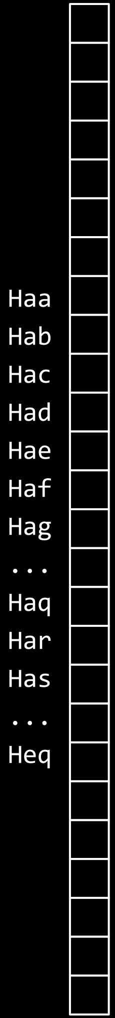

Haa
Hab
Hac
Had
Hae
Haf

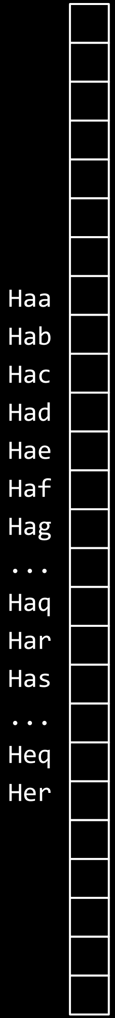

Haa
Hab
Hac
Had
Hae
Haf
Hag
...

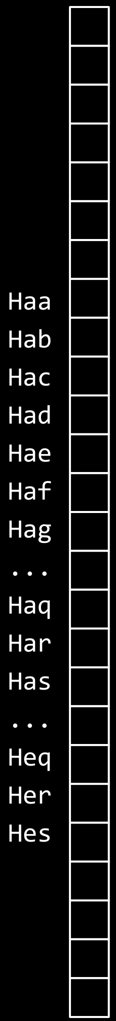

Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq

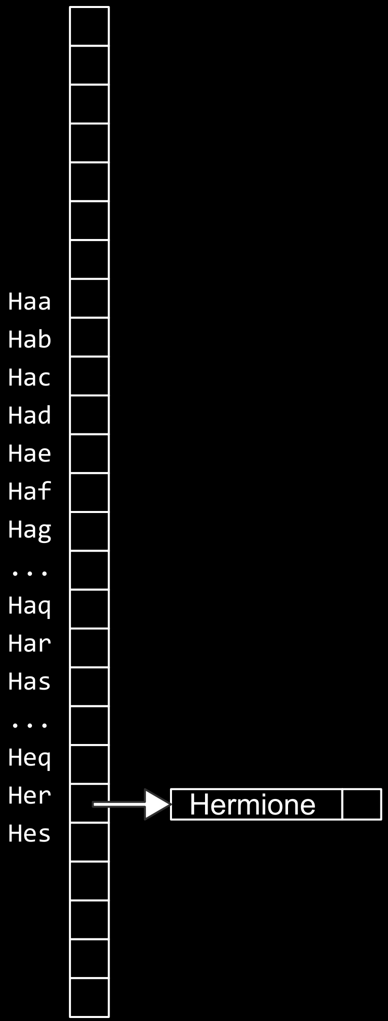

Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har

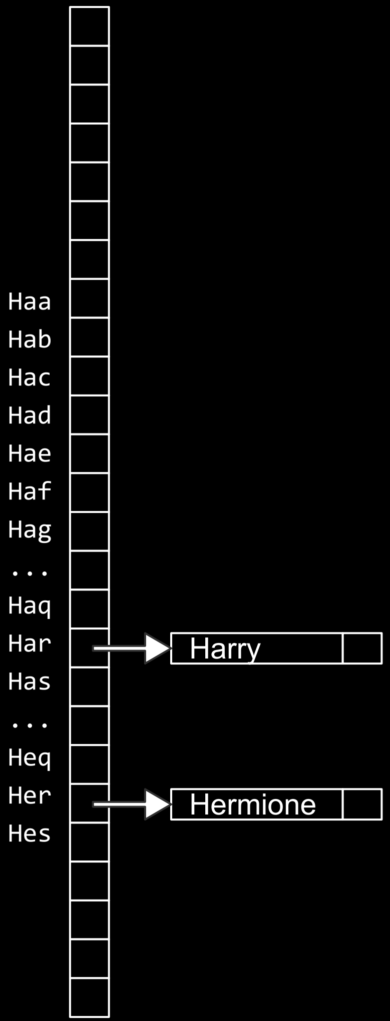

Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...

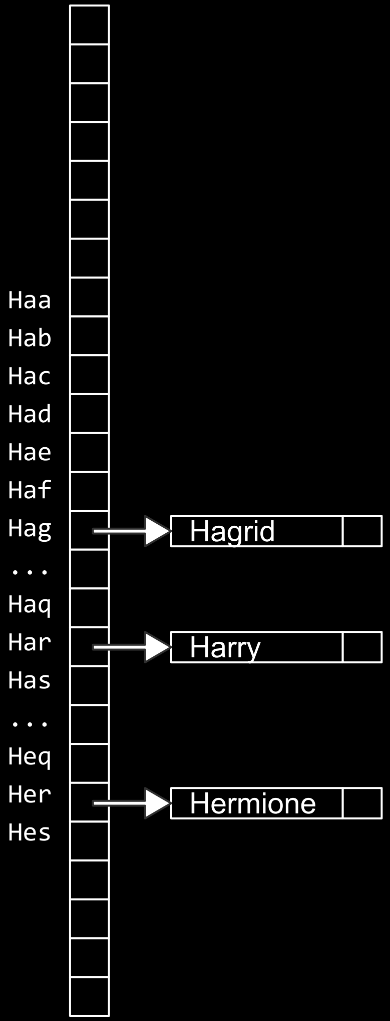

Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes
Hermione


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes
Harry
Hermione


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes
Hagrid
Harry
Hermione

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

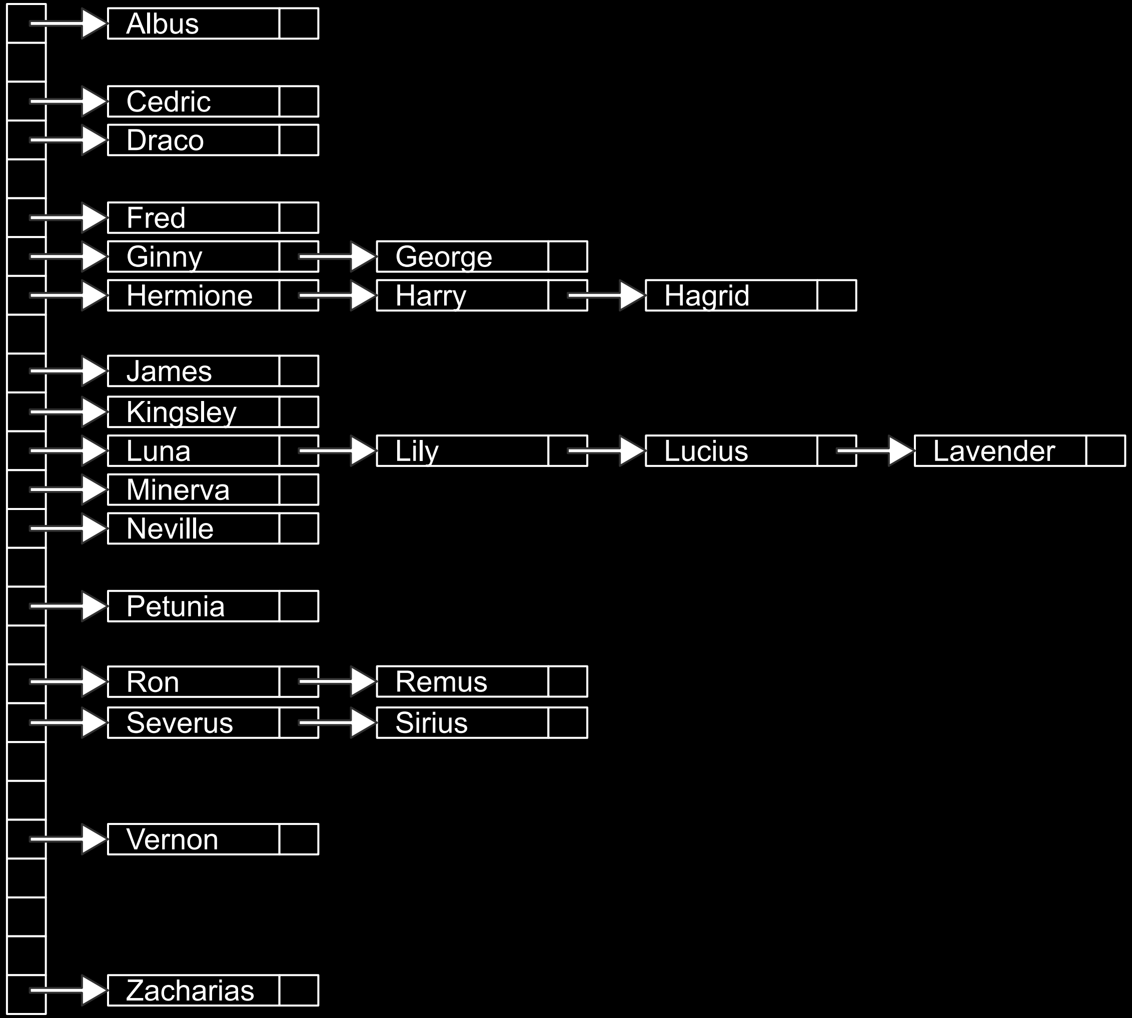
$O(1)$

$O(n^2)$	
$O(n \log n)$	
$O(n)$	
$O(\log n)$	
$O(1)$	search

$O(n^2)$	
$O(n \log n)$	
$O(n)$	search
$O(\log n)$	
$O(1)$	

$O(n^2)$	
$O(n \log n)$	
$O(n)$	search, insert
$O(\log n)$	
$O(1)$	

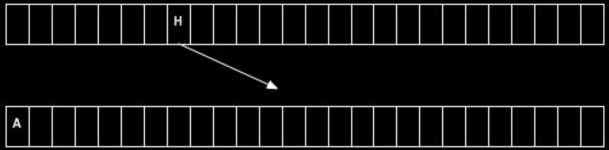
$O(n^2)$	
$O(n \log n)$	
$O(n)$	search
$O(\log n)$	
$O(1)$	insert

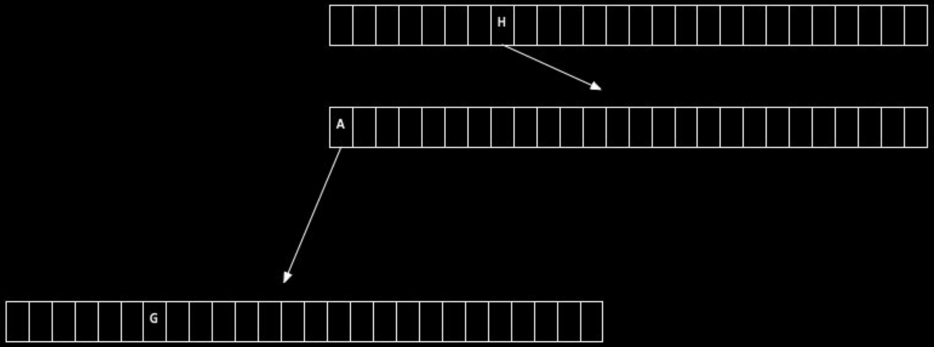

Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Lily
Lucius
Lavender
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

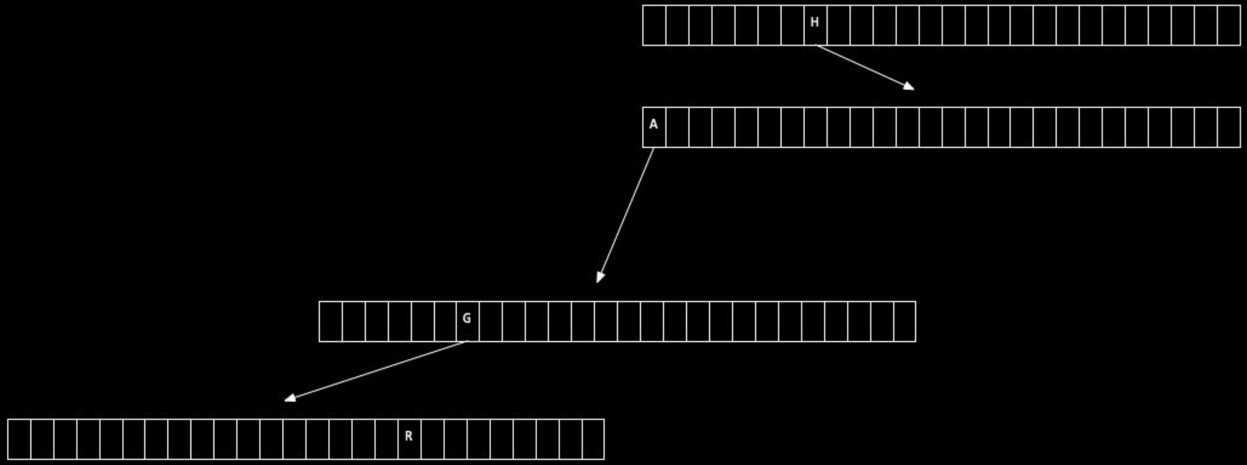
tries

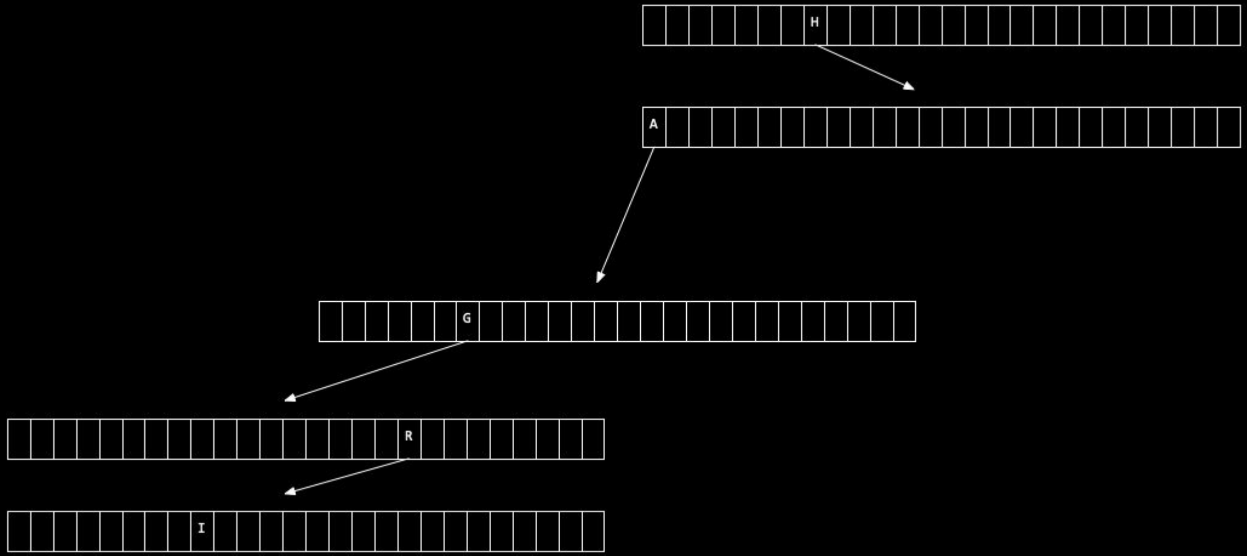
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

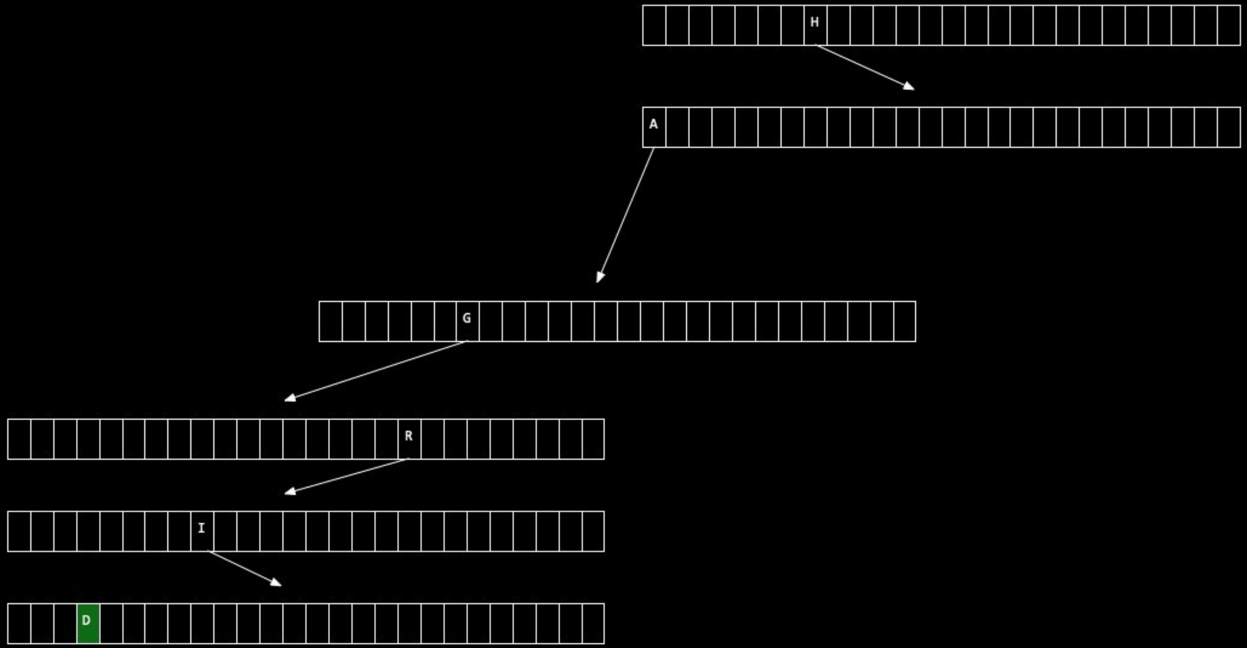
							H																		

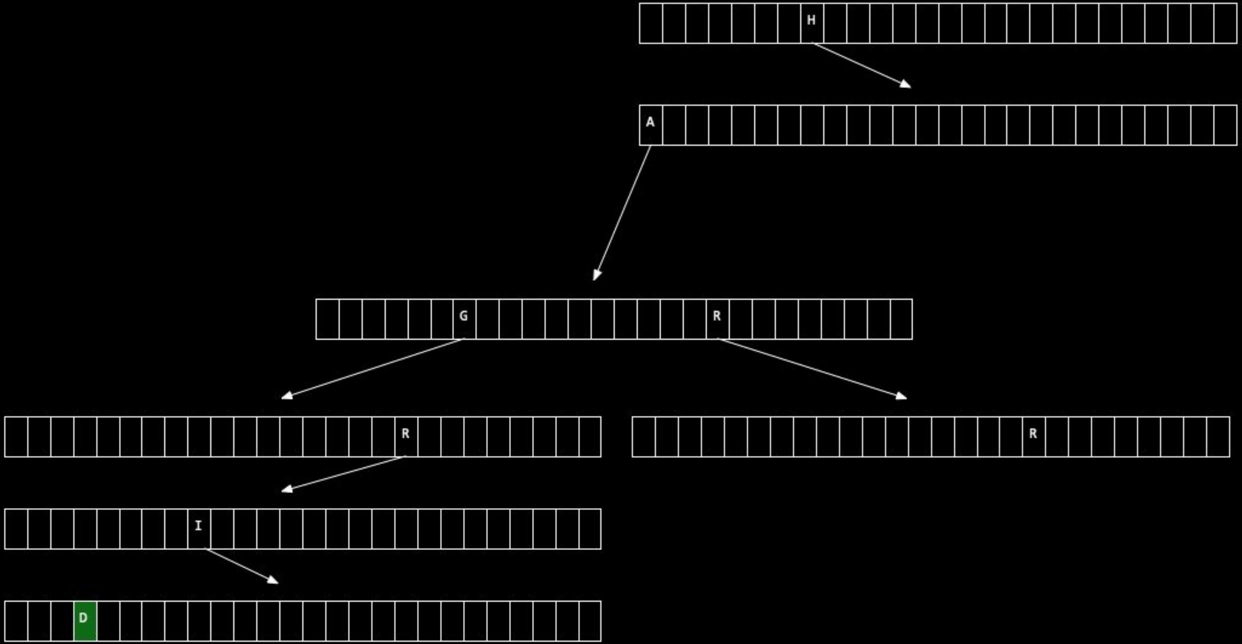

H
A

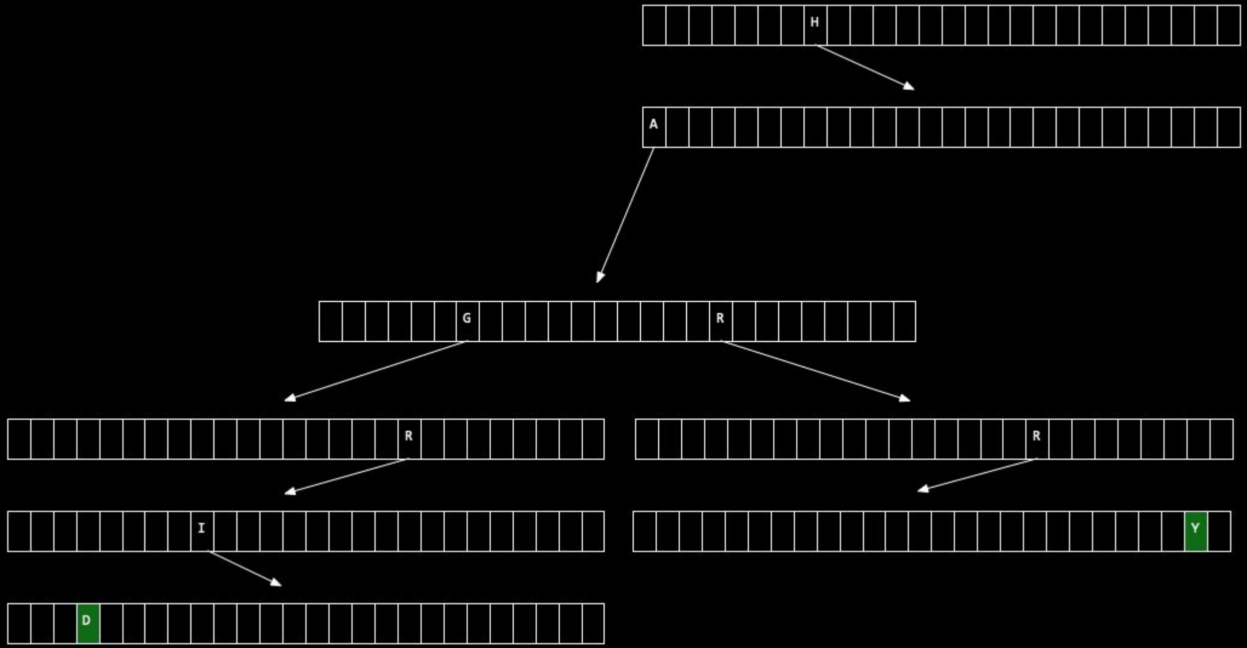

H
A
G

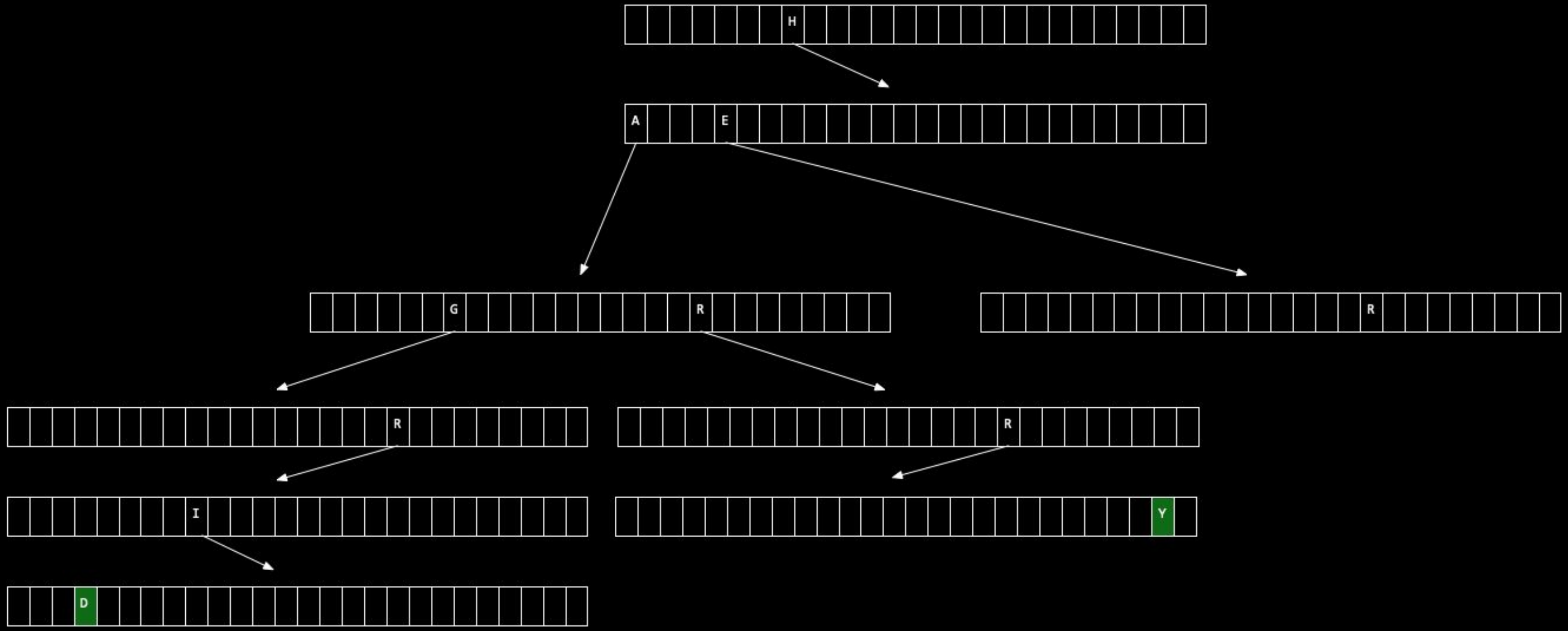

H
A
G
R

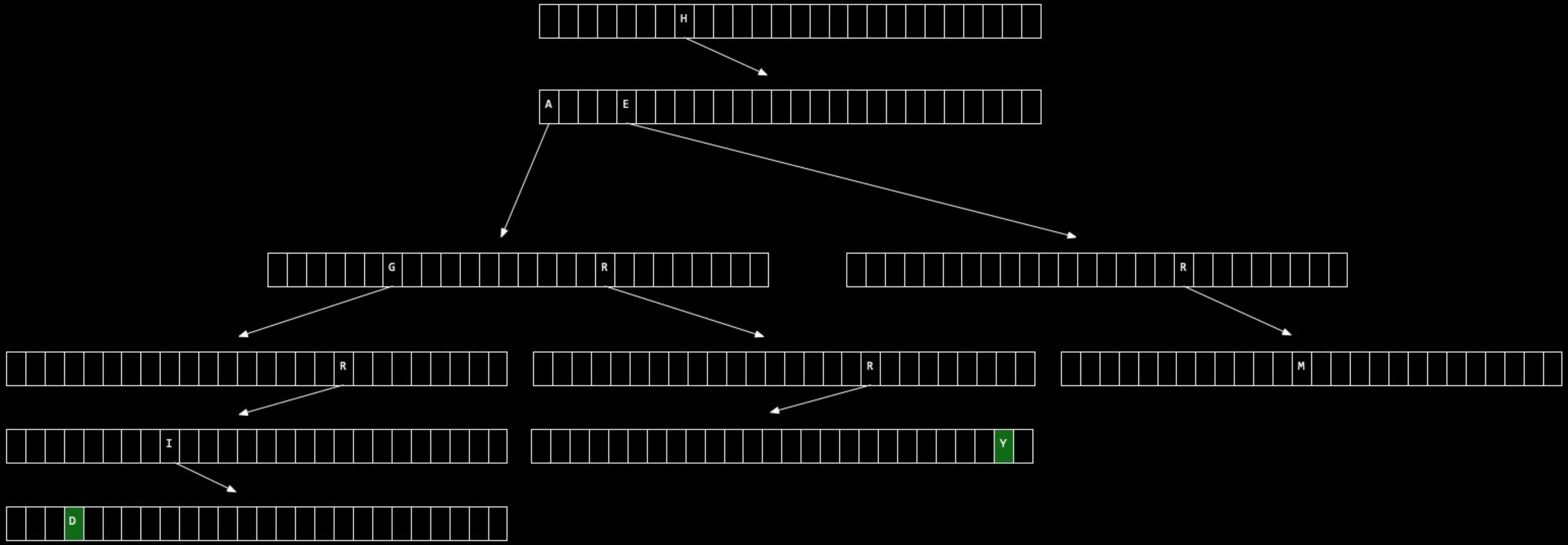

H
A
G
R
I

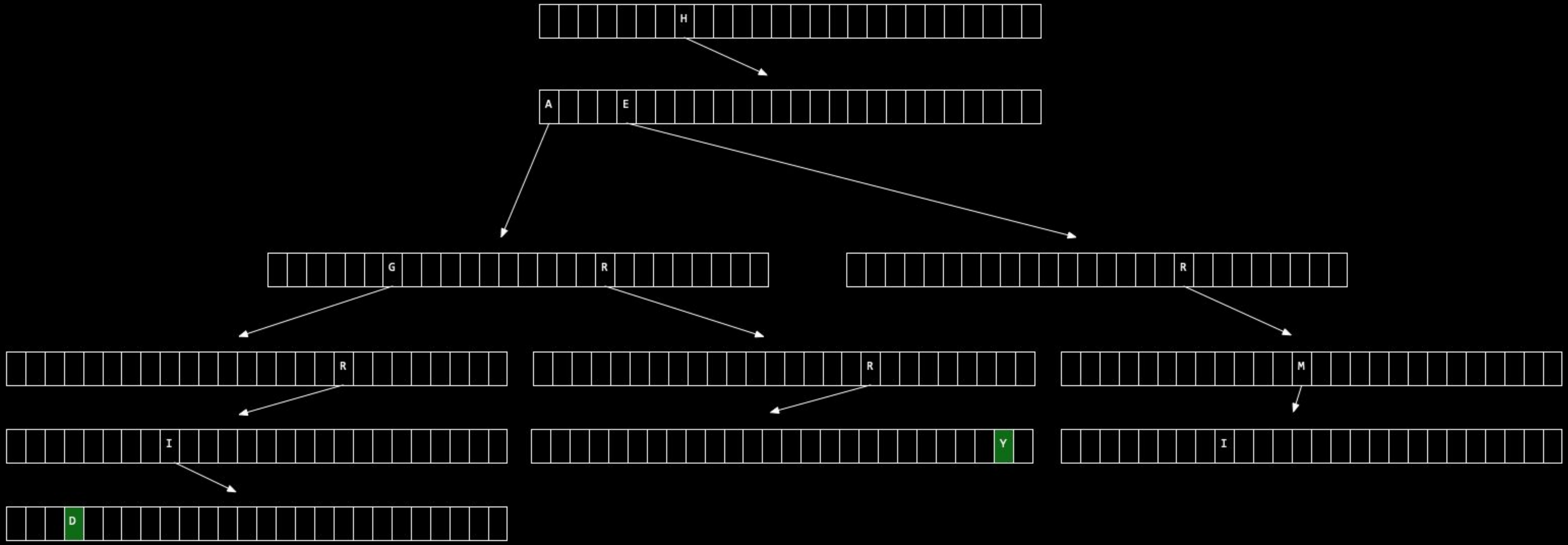

H
A
G
R
I
D

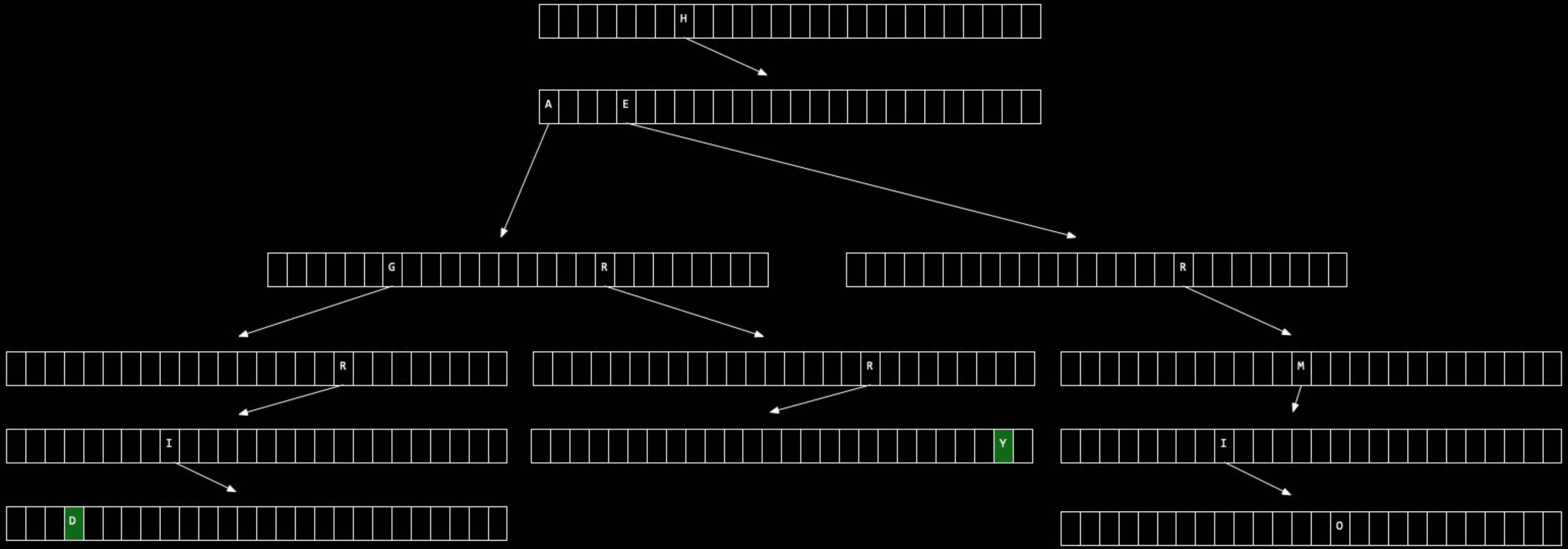

H
A
G
R
R
R
I
D

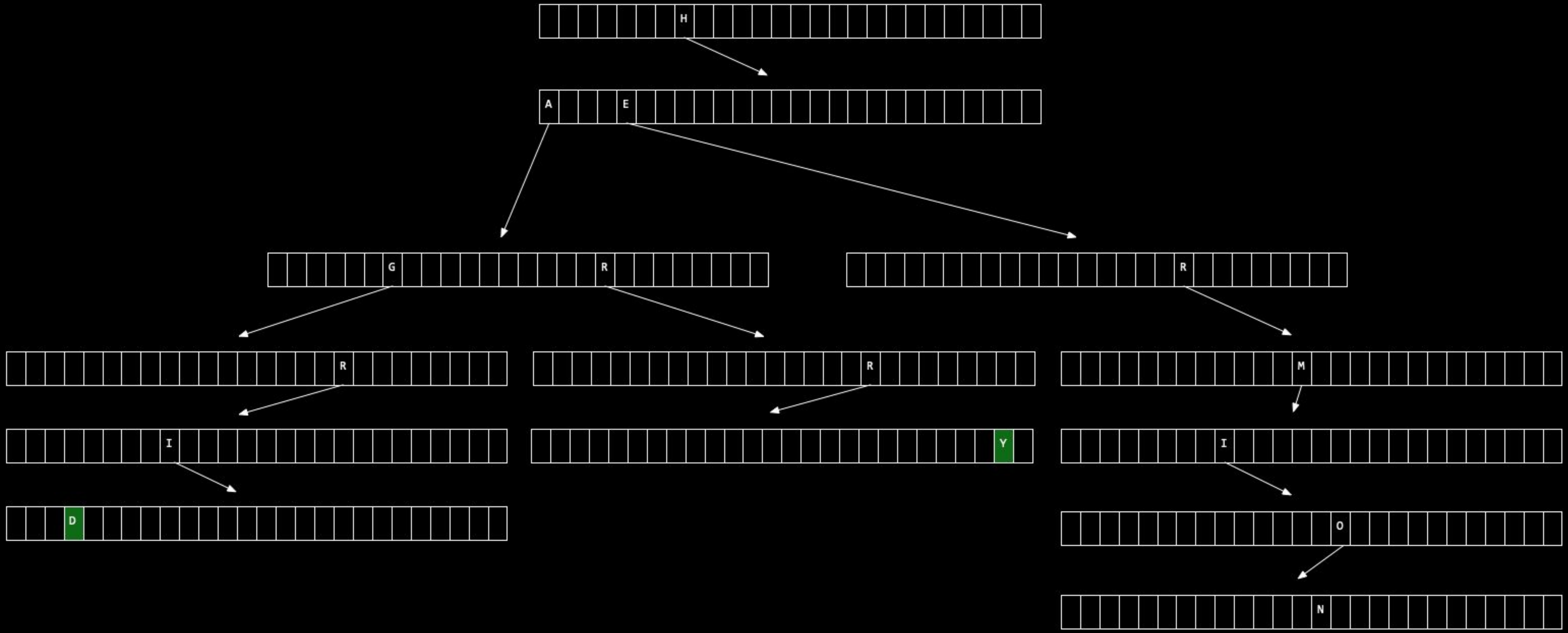

H
A
G
R
R
R
I
Y
D

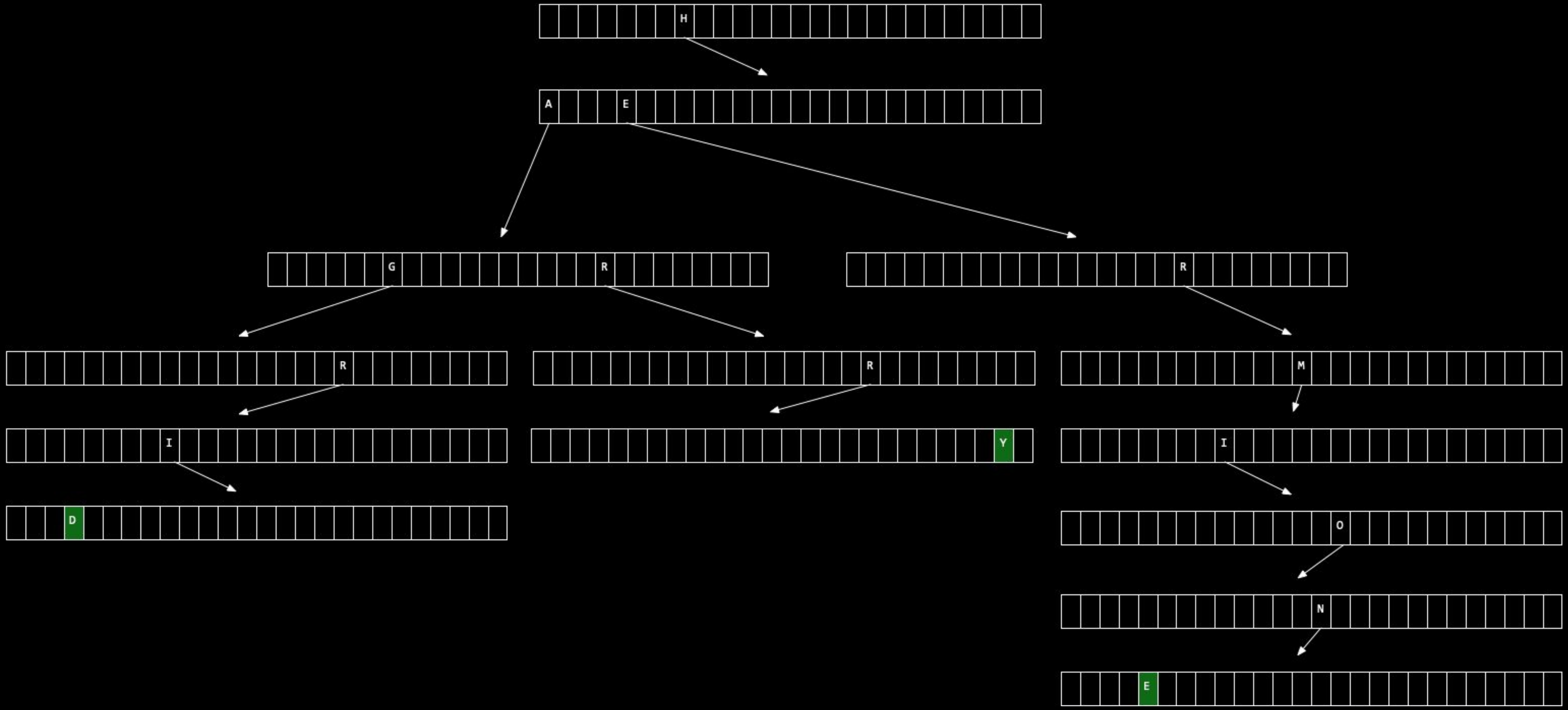

H
A
E
G
R
R
R
R
I
Y
D

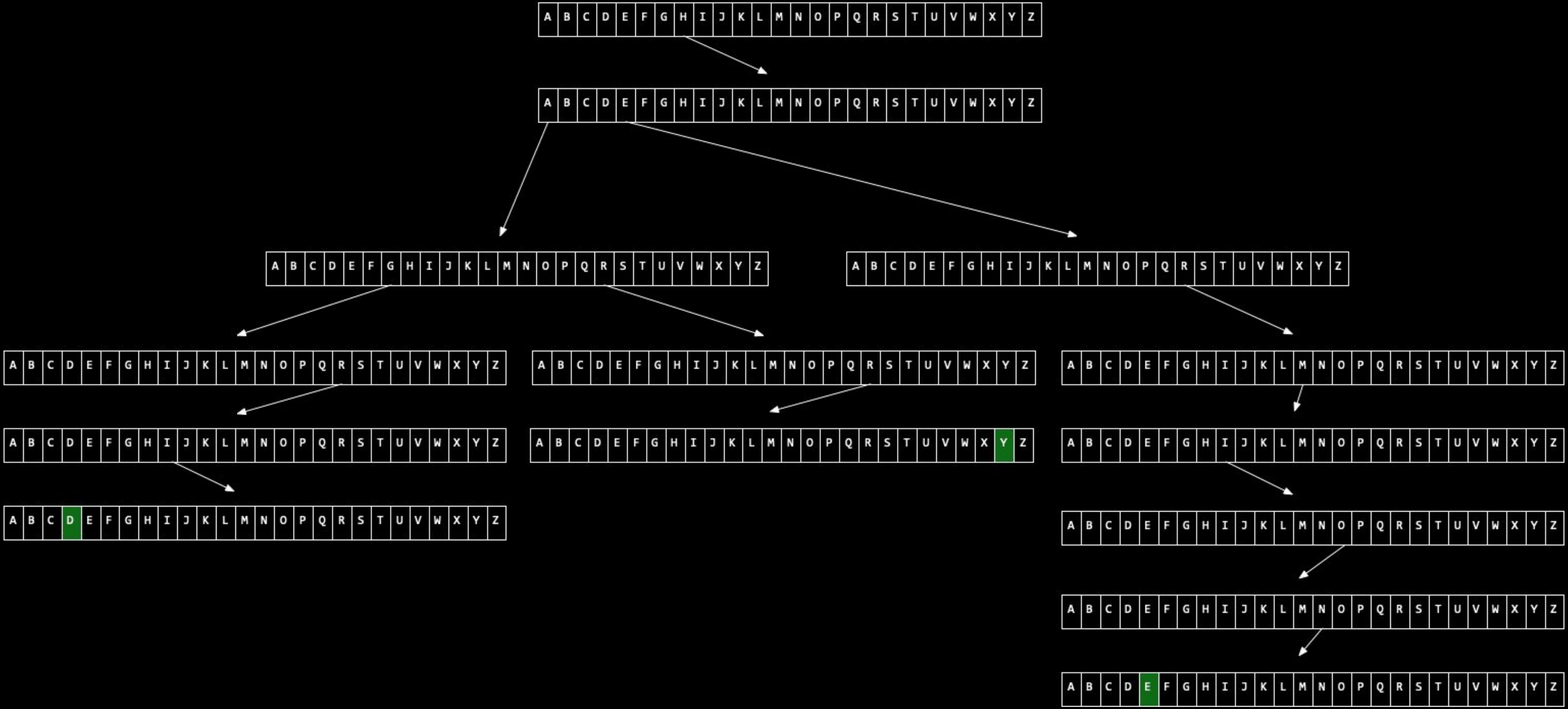

H
A
E
G
R
R
R
R
M
I
Y
D


H
A
E
G
R
R
R
R
M
I
Y
I
D


H
A
E
G
R
R
R
R
M
I
Y
I
D
O


H
A
E
G
R
R
R
R
M
I
Y
I
D
O
N


H
A
E
G
R
R
R
R
M
I
Y
I
D
O
N
E


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
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
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
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
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
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
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
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
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
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
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
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
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
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

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(k)$ search

$O(k)$ search, insert

$O(1)$ search, insert

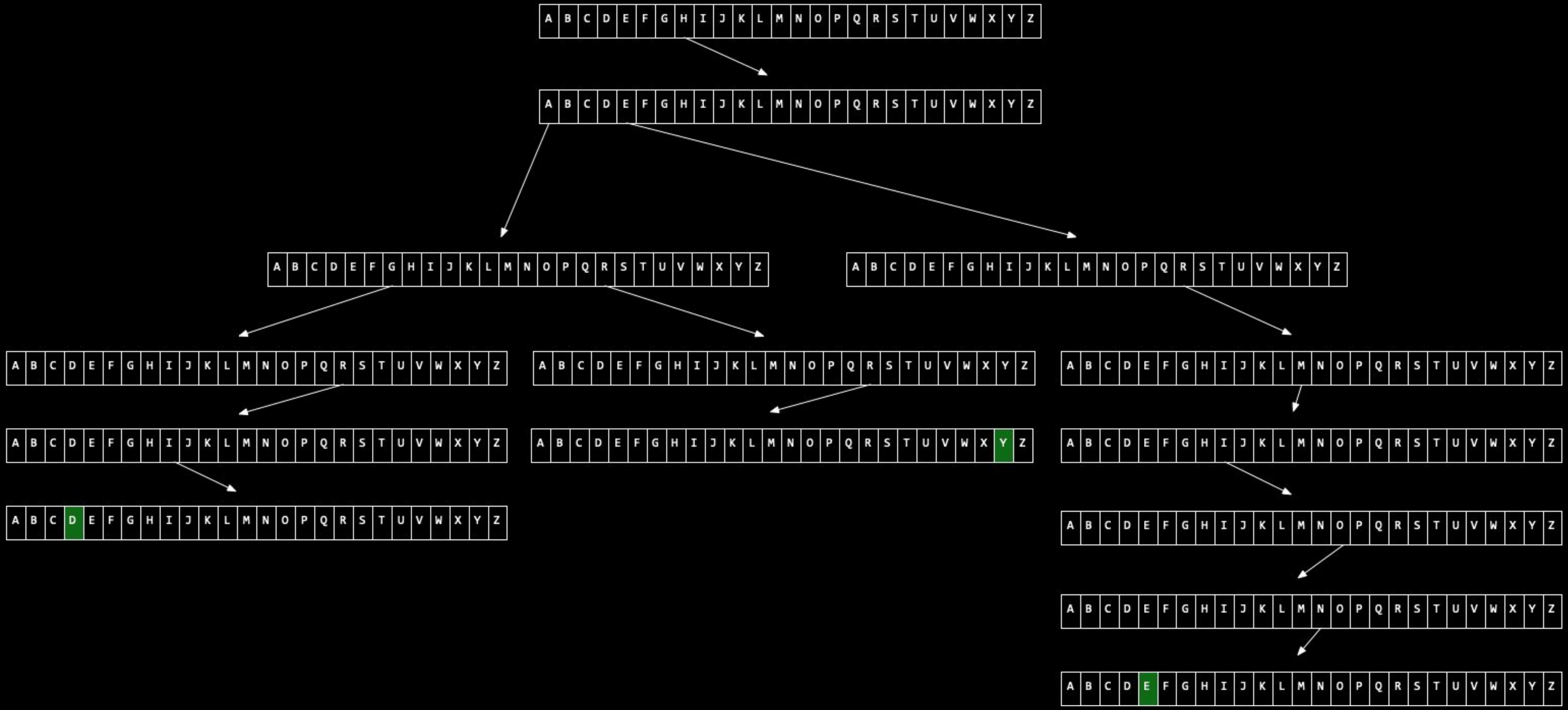
$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$	search, insert


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
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
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
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
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
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
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
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
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
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
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
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
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
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
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

abstract data structures

queues

enqueue

dequeue

FIFO

stacks

push

pop

LIFO

dictionaries

ICK ME UP
CHOKIN
B
C
D
E
F
G
H
I
K
L
M
N
O
P
Q
R
T
U-
V
W
X
Y
Z

This is CS50