

Permutation from Swapping (1/4)

- **Recursive**

- 1 + permutations of {2, 3, 4}
- 2 + permutations of {1, 3, 4}
- 3 + permutations of {2, 1, 4}
- 4 + permutations of {2, 3, 1}

1	2	3	4
1	2	4	3
1	3	2	4
1	3	4	2
1	4	3	2
1	4	2	3
2	1	3	4
2	1	4	3
...
3	2	1	4
3	2	4	1
...
4	2	3	1
4	2	1	3
...
4	1	2	3

Permutation from Swapping (1/4)

- **Recursive**

1 + permutations of {2, 3, 4}

2 + permutations of {1, 3, 4}

3 + permutations of {2, 1, 4}

4 + permutations of {2, 3, 1}

```
for (i=0; i<n; i++) a[i] = i+1;  
permutation(a, 0, n-1);
```

1	2	3	4
1	2	4	3
1	3	2	4
1	3	4	2
1	4	3	2
1	4	2	3
2	1	3	4
2	1	4	3
...
3	2	1	4
3	2	4	1
...
4	2	3	1
4	2	1	3
...
4	1	2	3

Permutation from Swapping (1/4)

- **Recursive**

- 1 + permutations of {2, 3, 4}
- 2 + permutations of {1, 3, 4}
- 3 + permutations of {2, 1, 4}
- 4 + permutations of {2, 3, 1}

```
for (i=0; i<n; i++) a[i] = i+1;  
permutation(a, 0, n-1);
```

```
void permutation(int perm[], int start, int end) {  
    if (start == end) printPerm(perm, end+1);  
    for (int i=start; i<=end; i++) {  
        swap(&perm[start], &perm[i]);  
        permutation(perm, start+1, end);  
        swap(&perm[start], &perm[i]);  
    }  
}
```

1	2	3	4
1	2	4	3
1	3	2	4
1	3	4	2
1	4	3	2
1	4	2	3
2	1	3	4
2	1	4	3
...
3	2	1	4
3	2	4	1
...
4	2	3	1
4	2	1	3
...
4	1	2	3

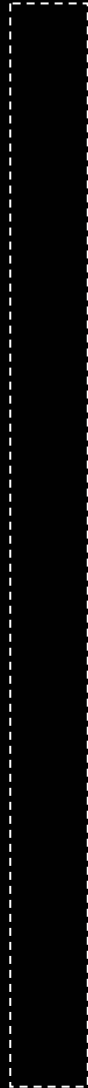
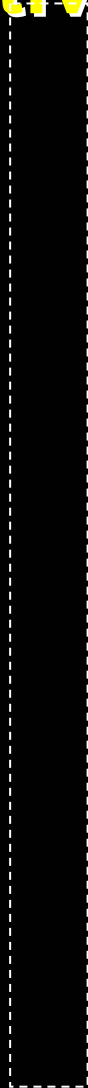
Permutation from Swapping (2/4)

- **Iterative**

1	2	3	4	}
1	2	4	3	
1	3	2	4	
1	...			
1	4	2	3	}
2	1	3	4	
2	...			
2	4	1	3	}
3	2	1	4	
3	2	4	1	
3	1	2	4	
3	1	4	2	
3	4	1	2	
3	4	2	1	}
4	2	3	1	
...				

Permutation from Swapping (2/4)

- **Iterative**



1	2	3	4
1	2	4	3
1	3	2	4
	...		
1	4	2	3

2	1	3	4
	...		
2	4	1	3

3	2	1	4
3	2	4	1
3	1	2	4
3	1	4	2
3	4	1	2
3	4	2	1

4	2	3	1

...

Permutation from Swapping (2/4)

- **Iterative**

1 2 3 4

1	2	3	4
1	2	4	3
1	3	2	4
...			
1	4	2	3

2	1	3	4
...			
2	4	1	3

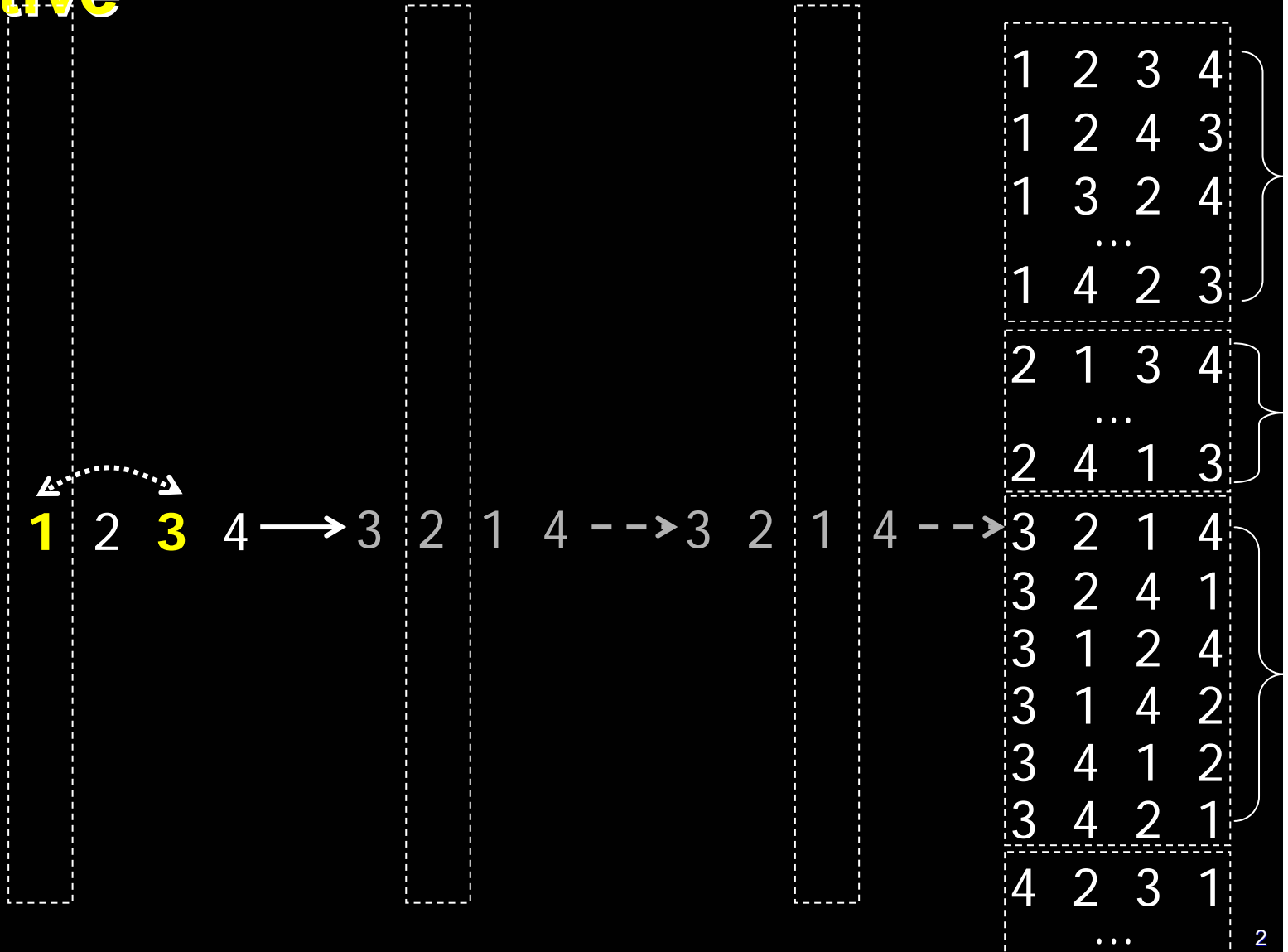
3	2	1	4
3	2	4	1
3	1	2	4
3	1	4	2
3	4	1	2
3	4	2	1

4	2	3	1

...

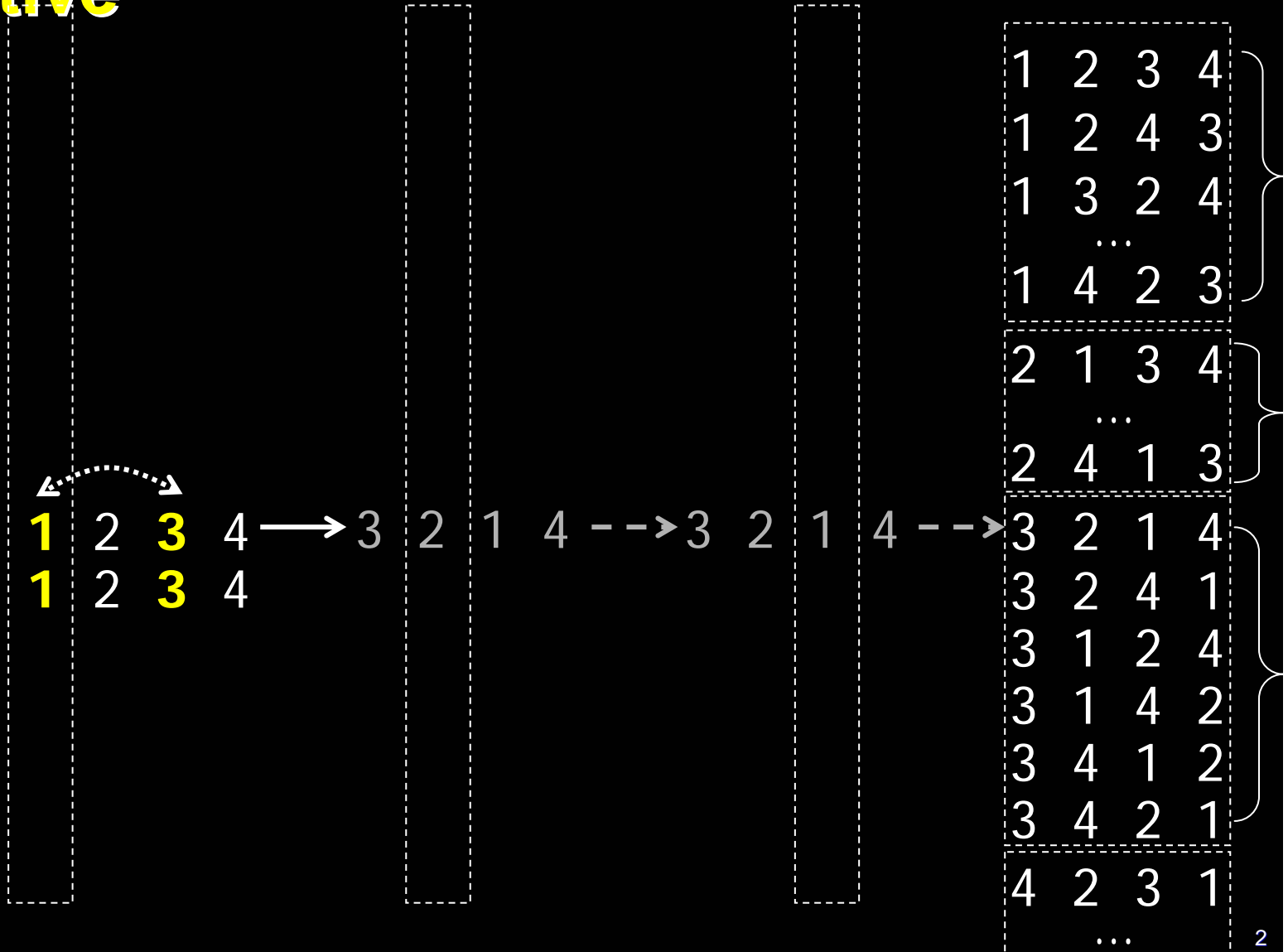
Permutation from Swapping (2/4)

- **Iterative**



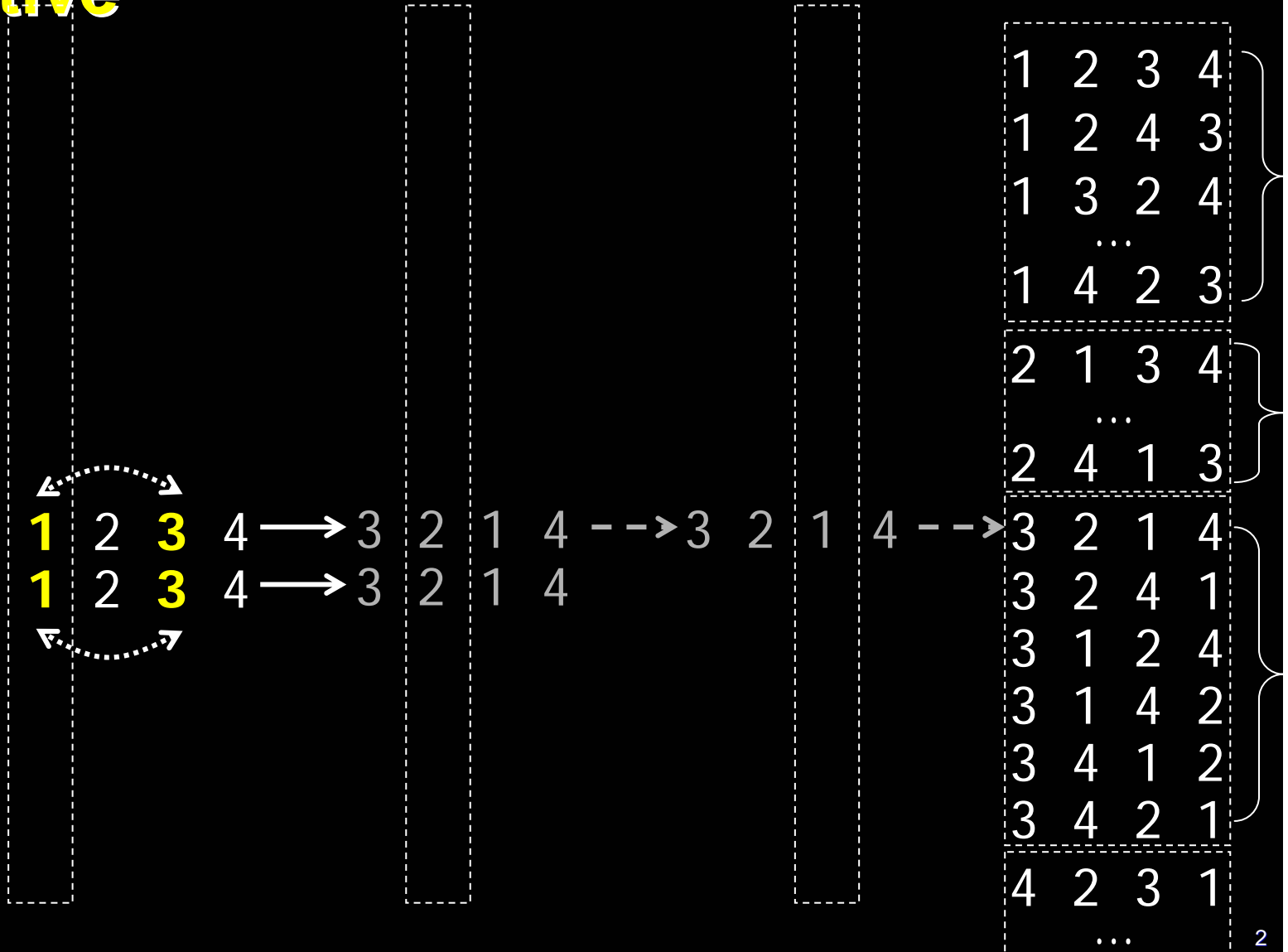
Permutation from Swapping (2/4)

- **Iterative**



Permutation from Swapping (2/4)

- **Iterative**



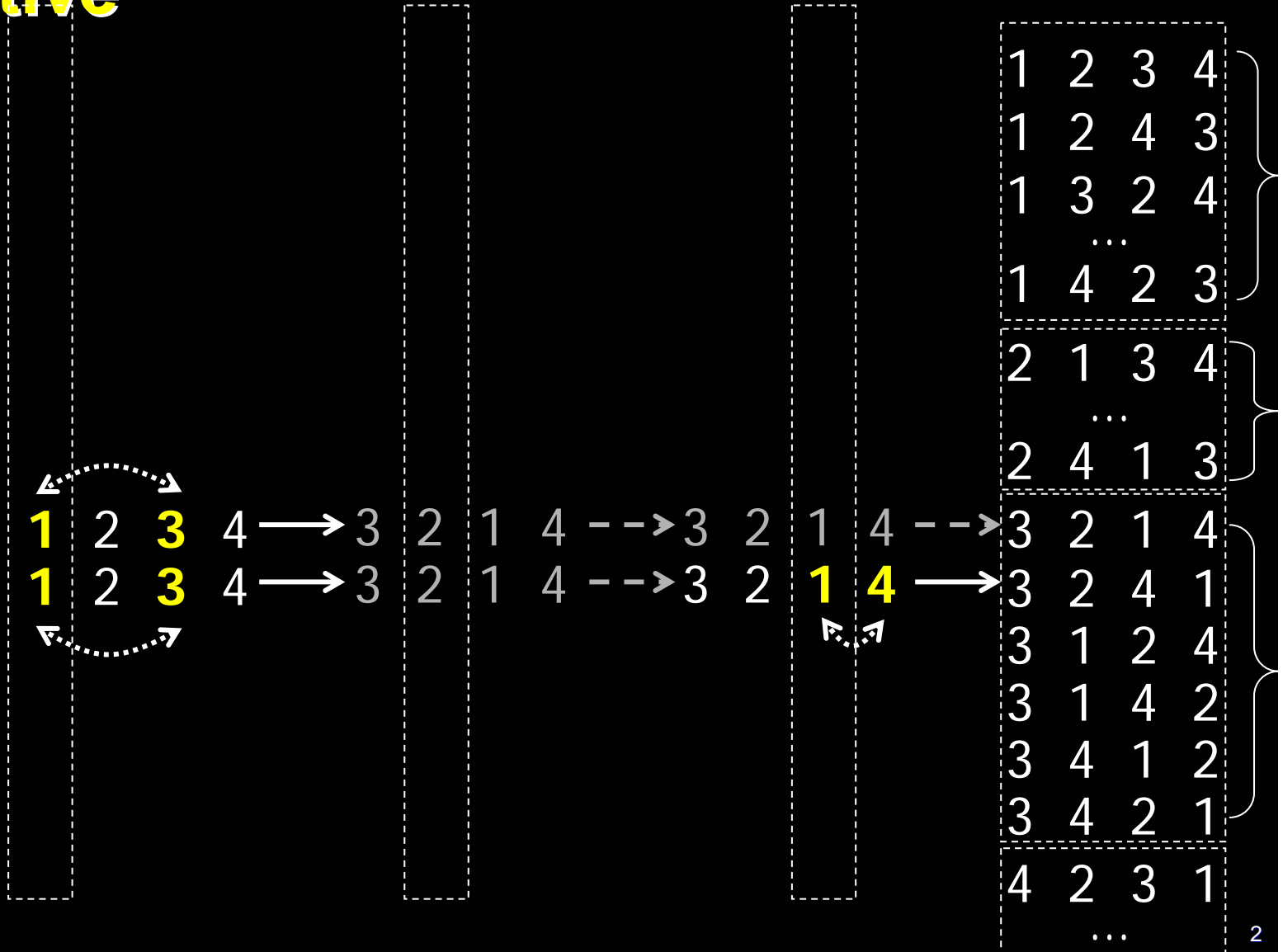
Permutation from Swapping (2/4)

- Iterative



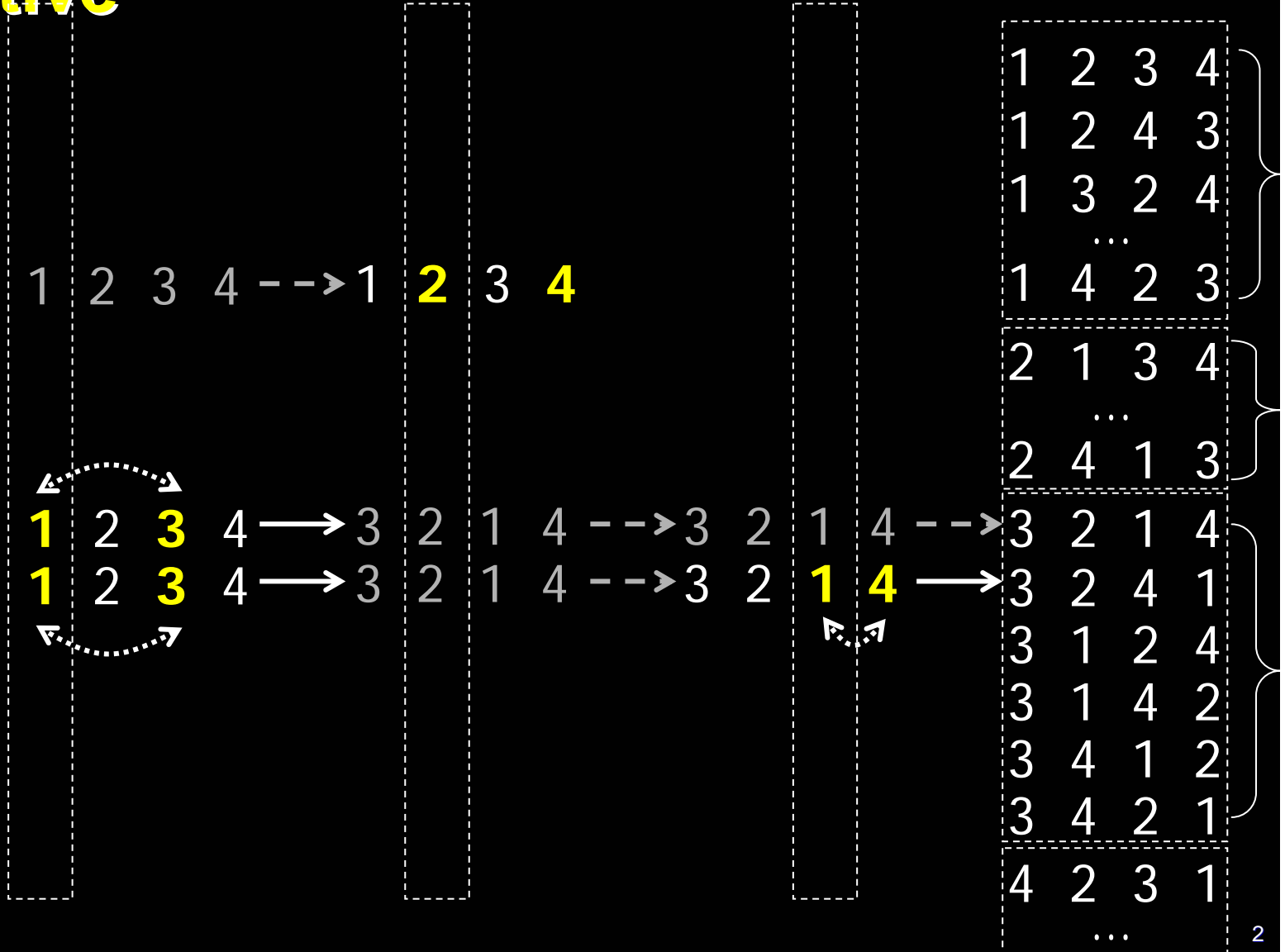
Permutation from Swapping (2/4)

- Iterative



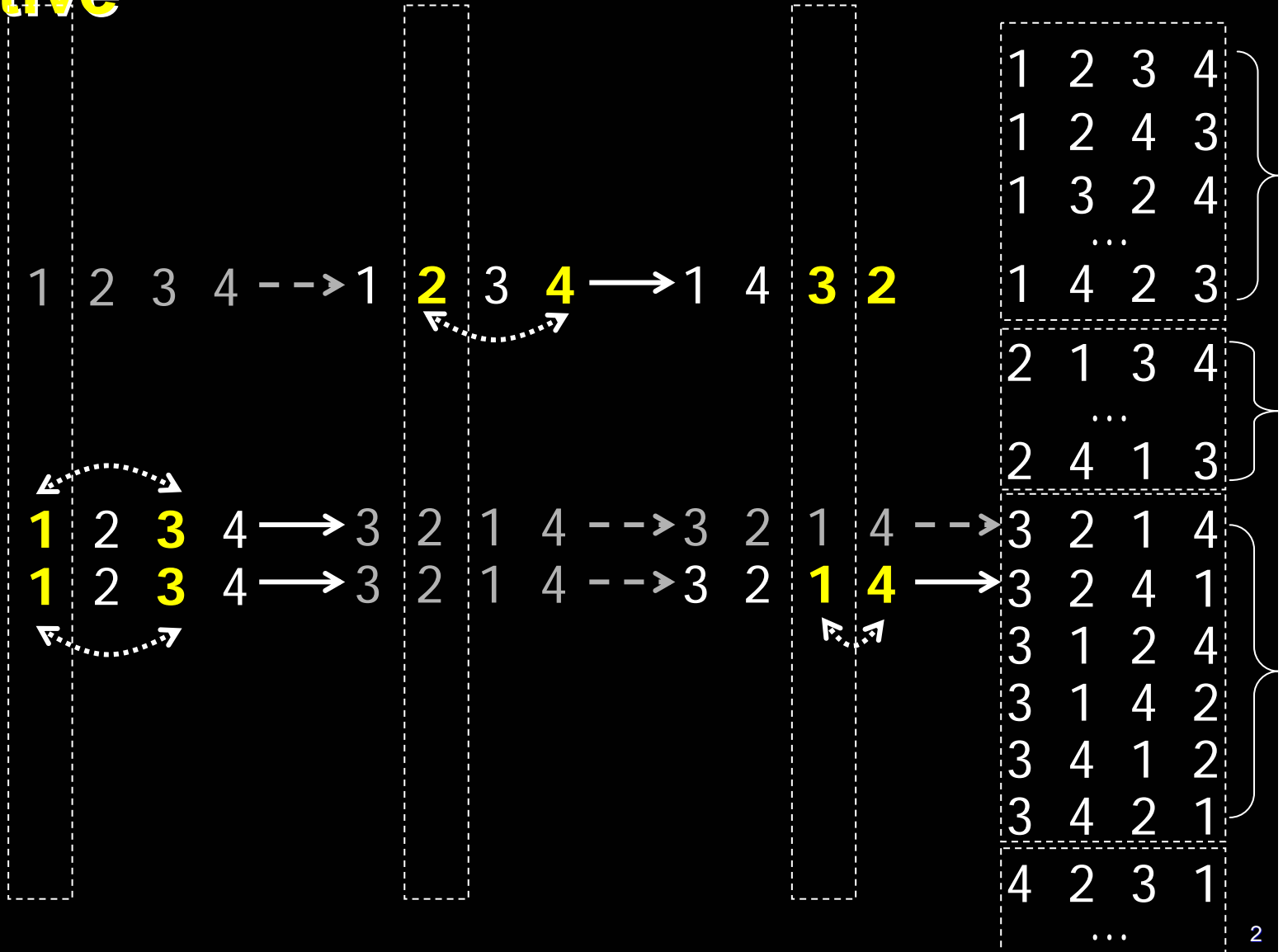
Permutation from Swapping (2/4)

- Iterative



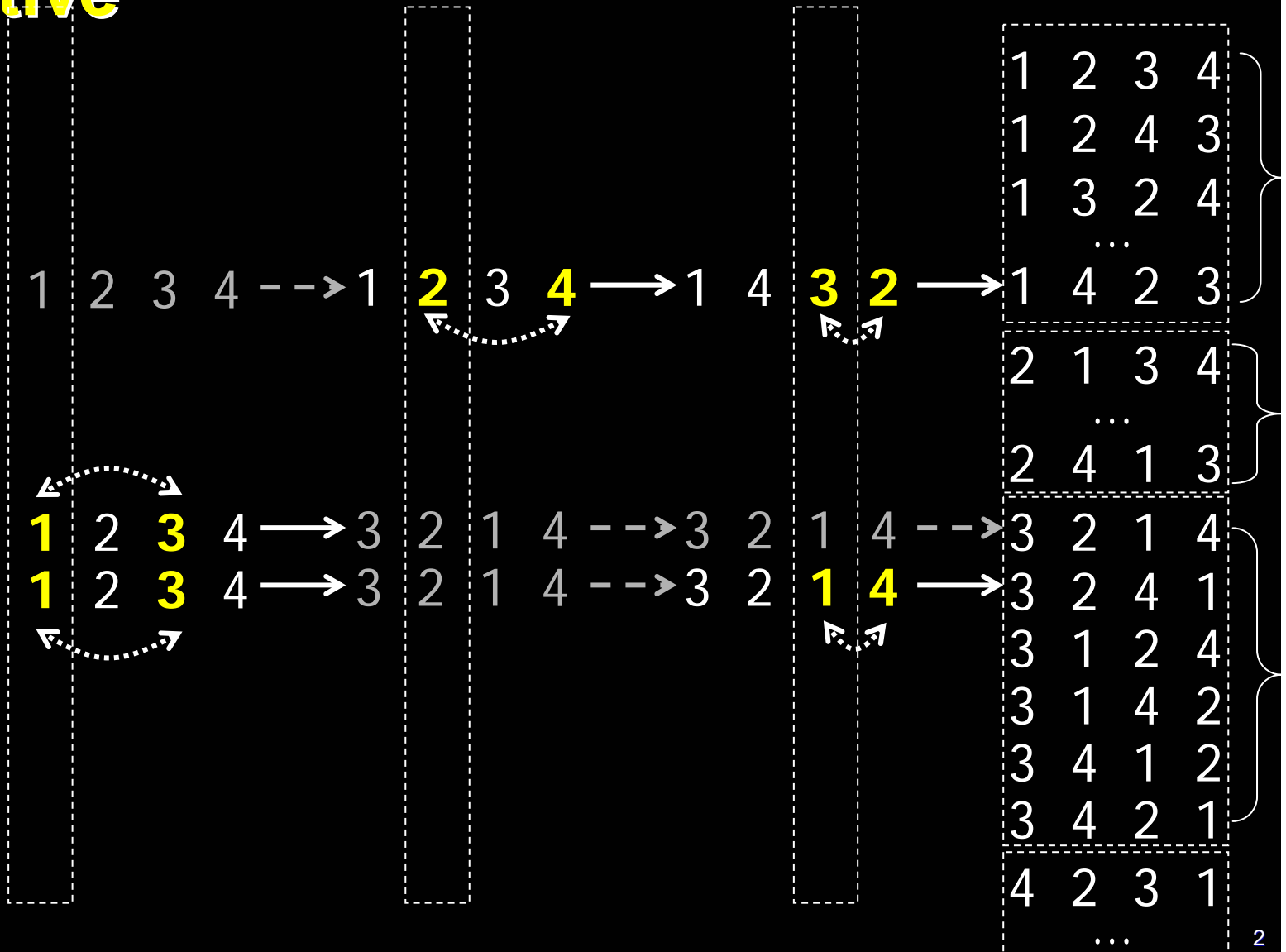
Permutation from Swapping (2/4)

- Iterative



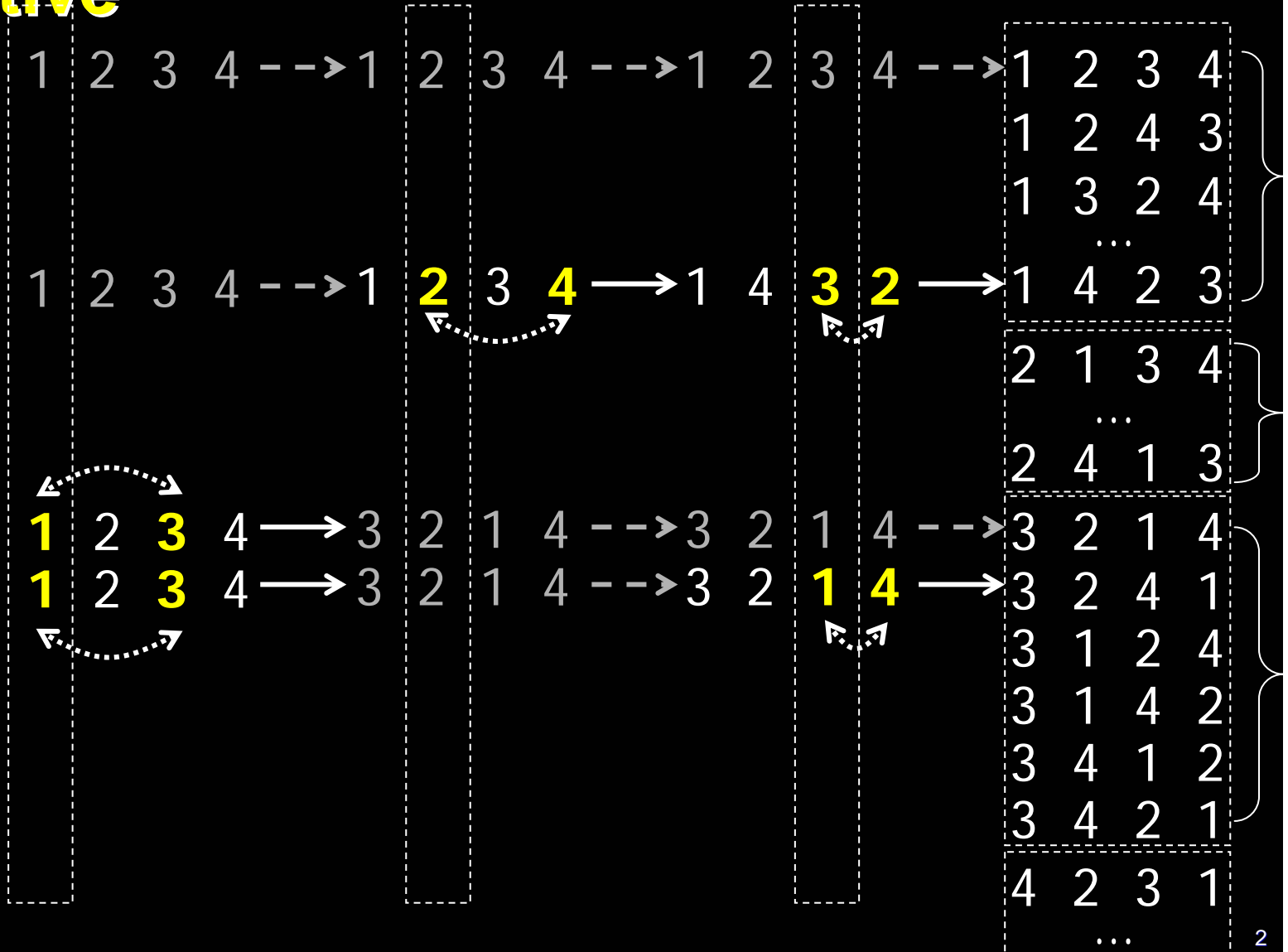
Permutation from Swapping (2/4)

- Iterative



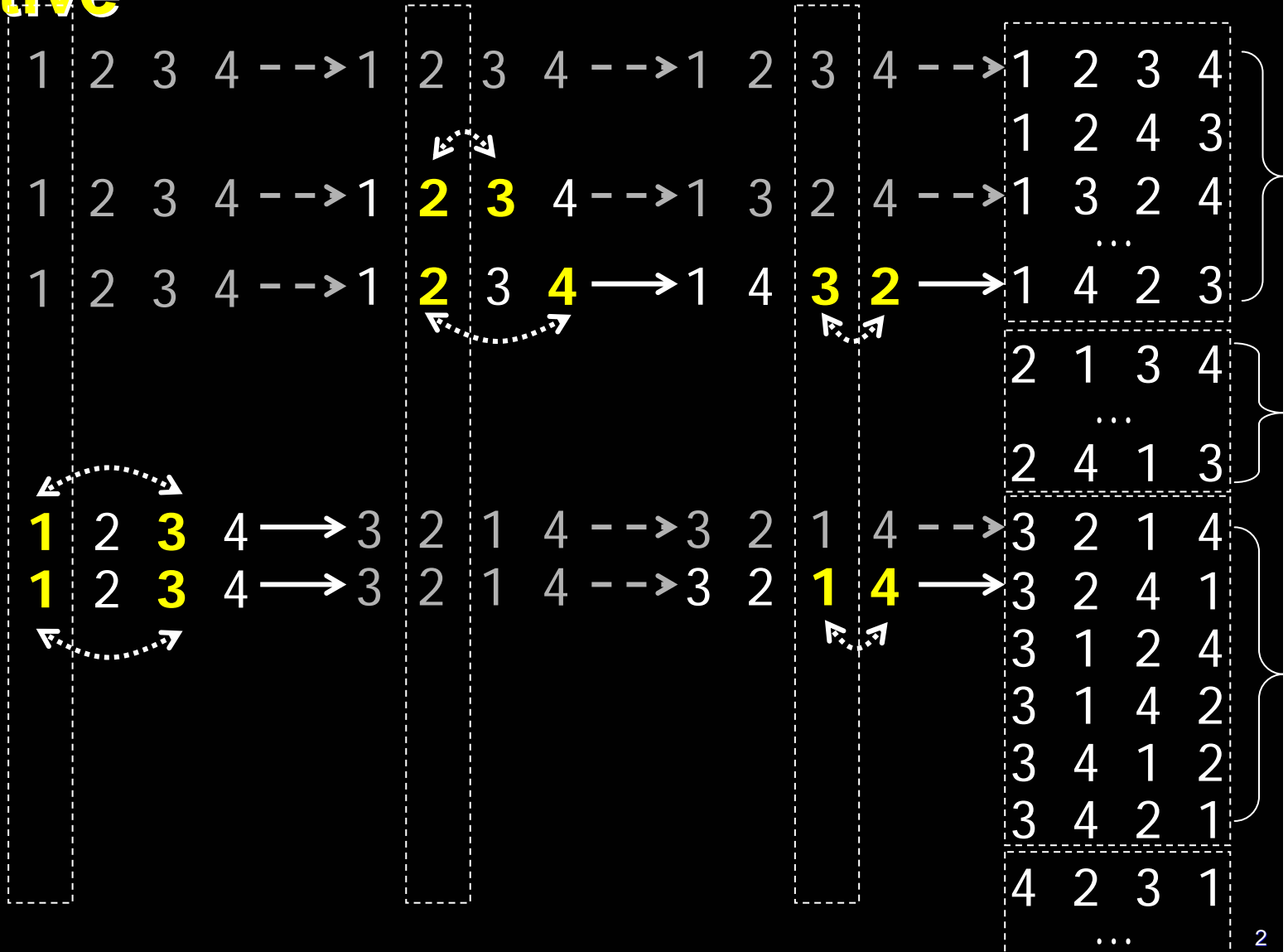
Permutation from Swapping (2/4)

- Iterative



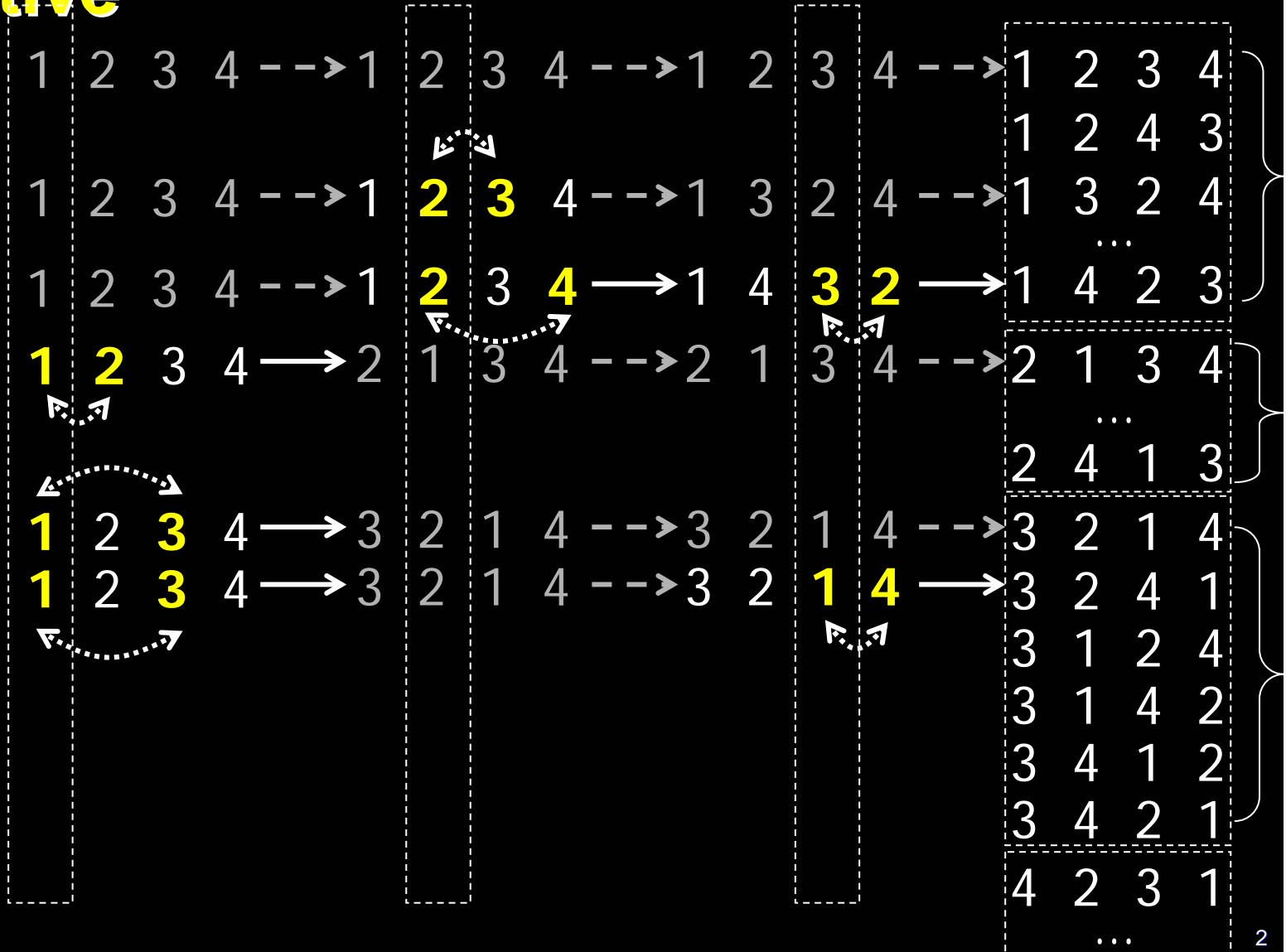
Permutation from Swapping (2/4)

- Iterative



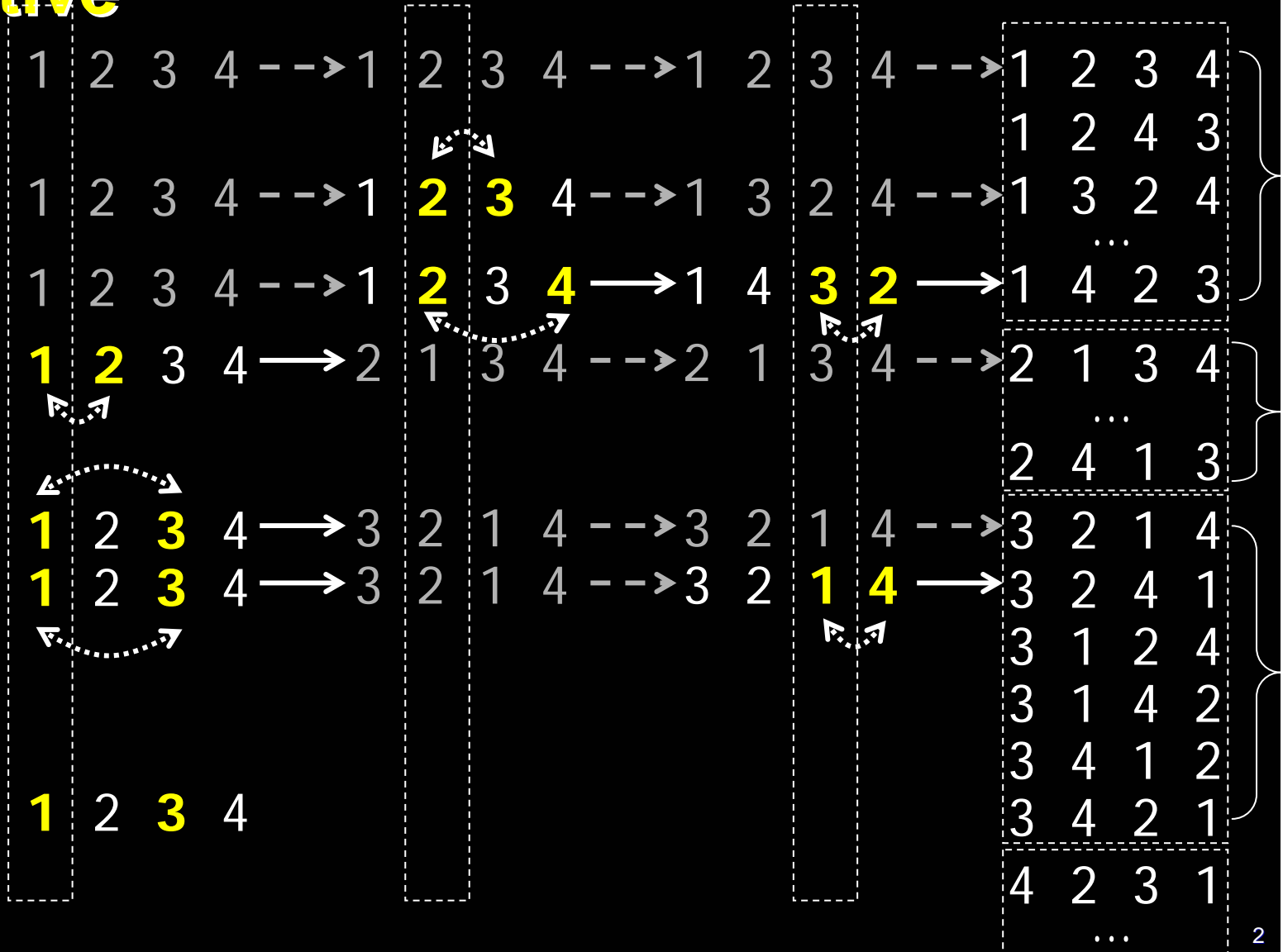
Permutation from Swapping (2/4)

- Iterative



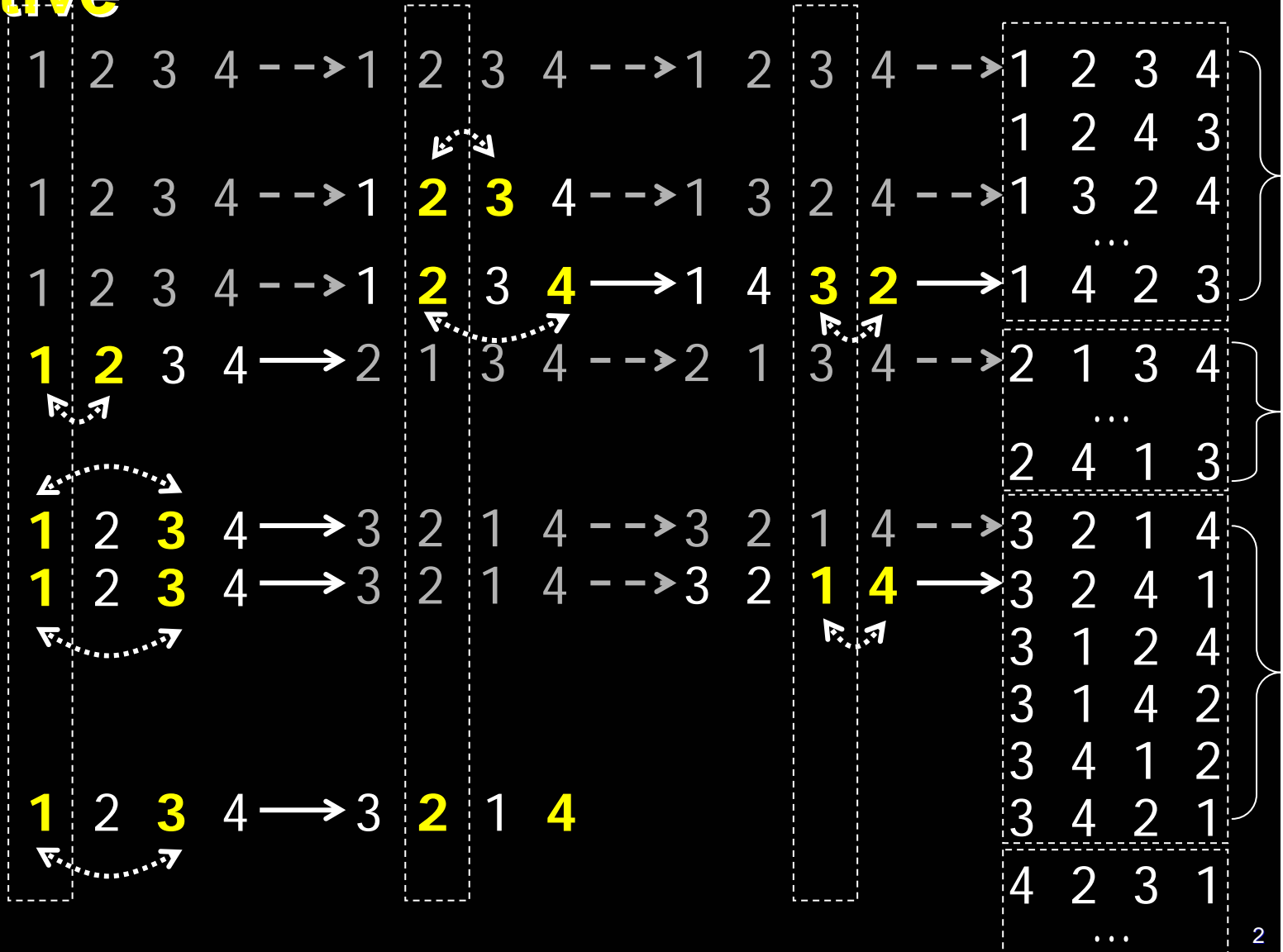
Permutation from Swapping (2/4)

- Iterative



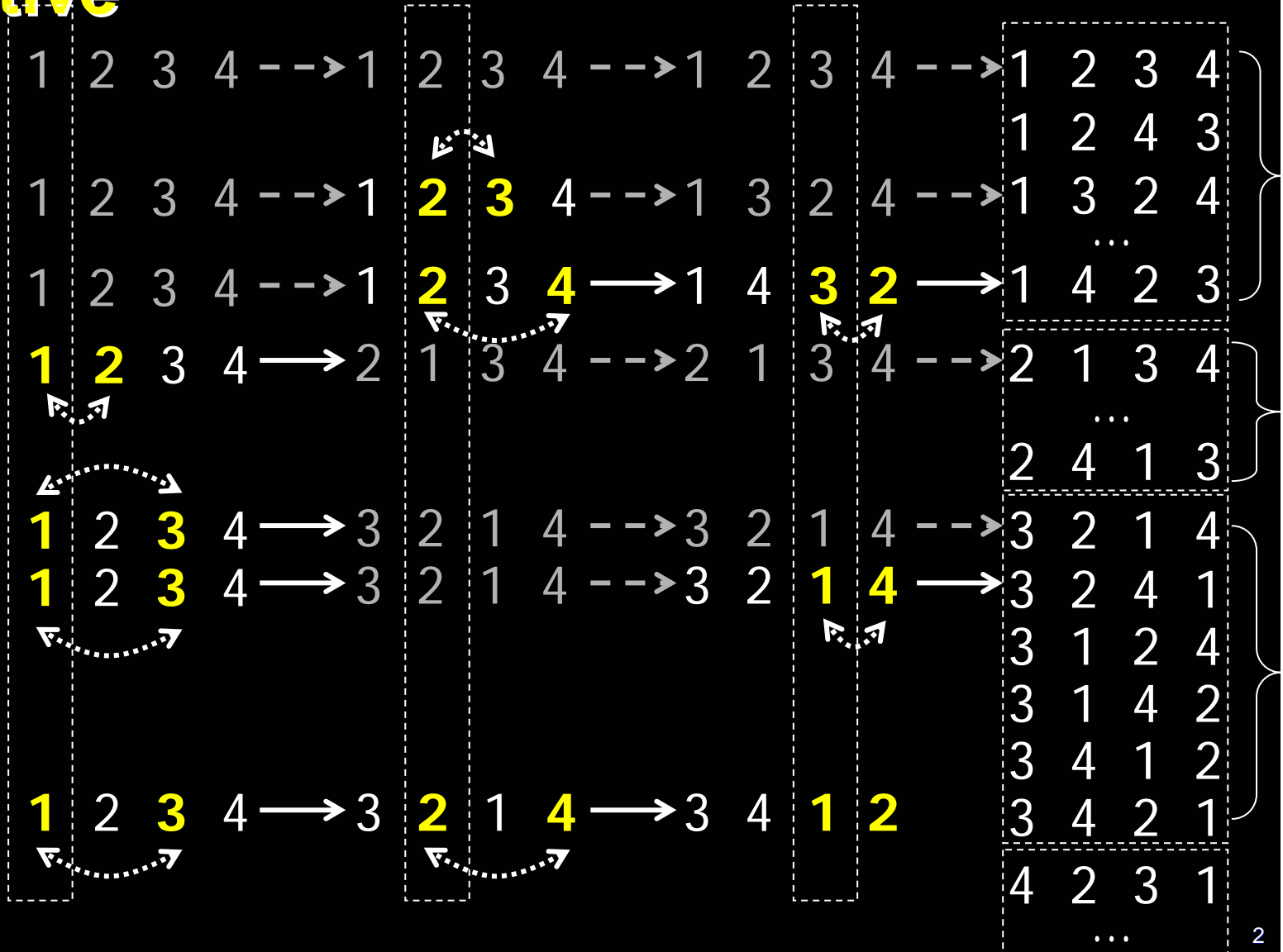
Permutation from Swapping (2/4)

- Iterative



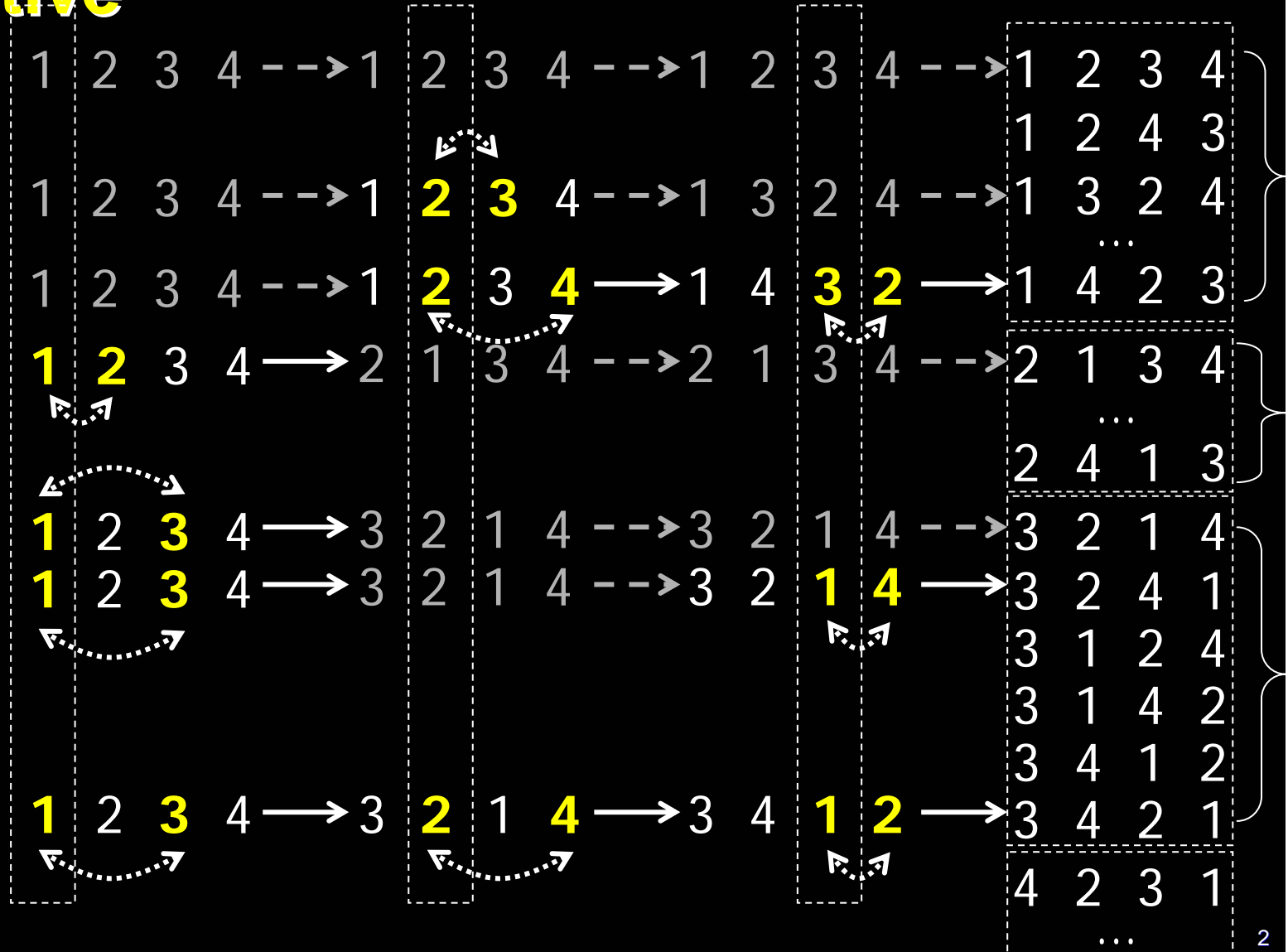
Permutation from Swapping (2/4)

- Iterative



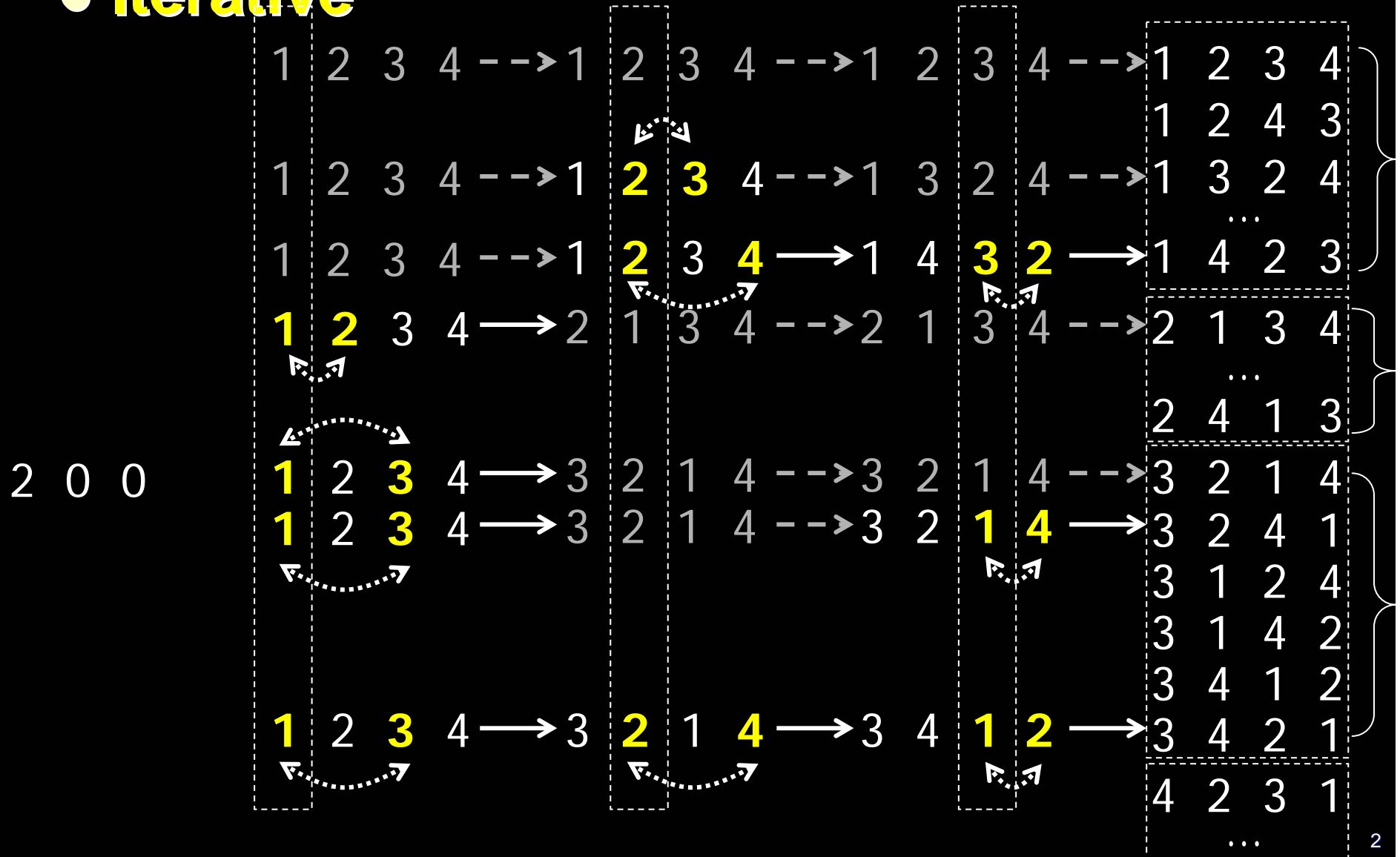
Permutation from Swapping (2/4)

- Iterative



Permutation from Swapping (2/4)

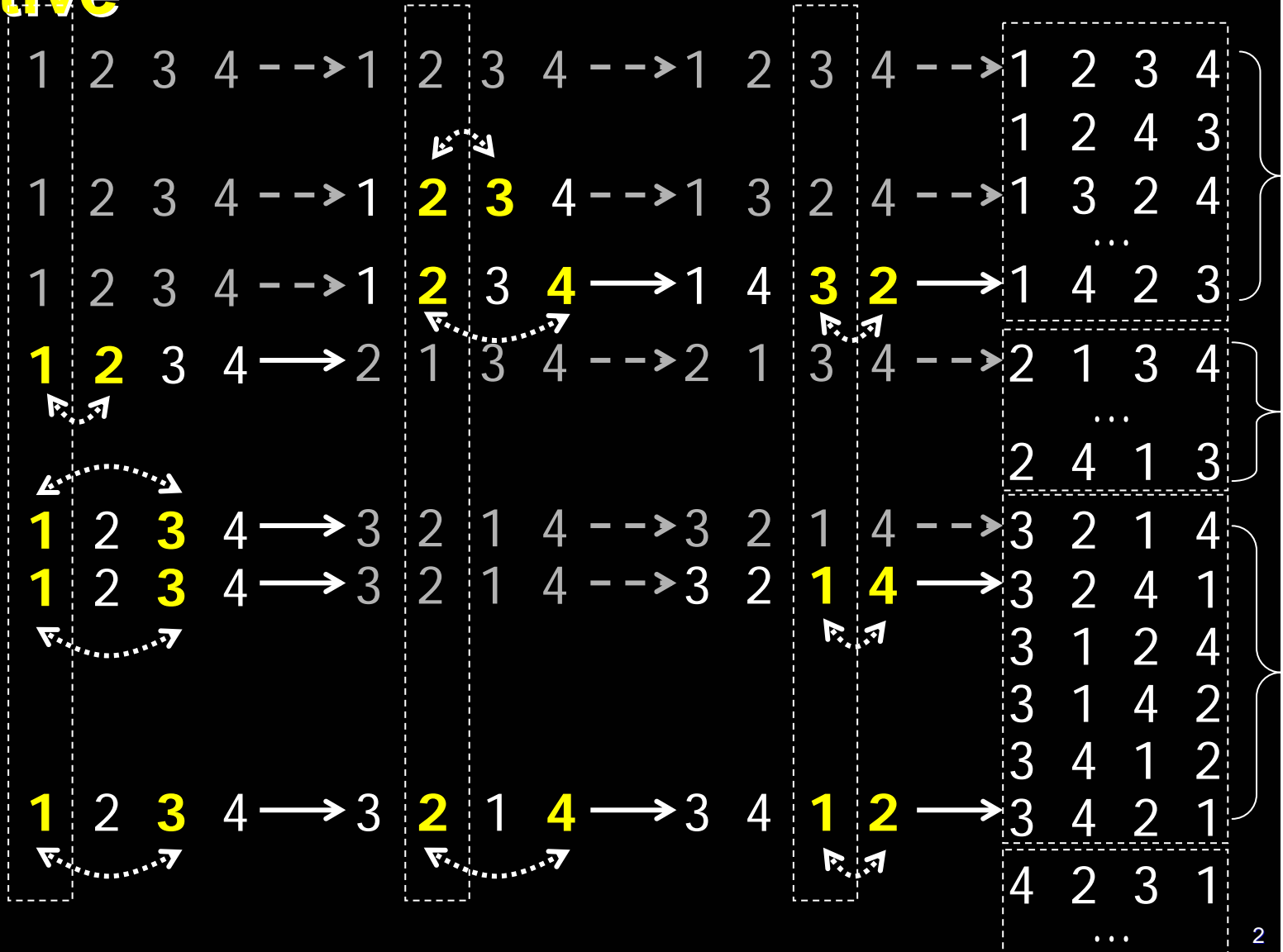
- Iterative



Permutation from Swapping (2/4)

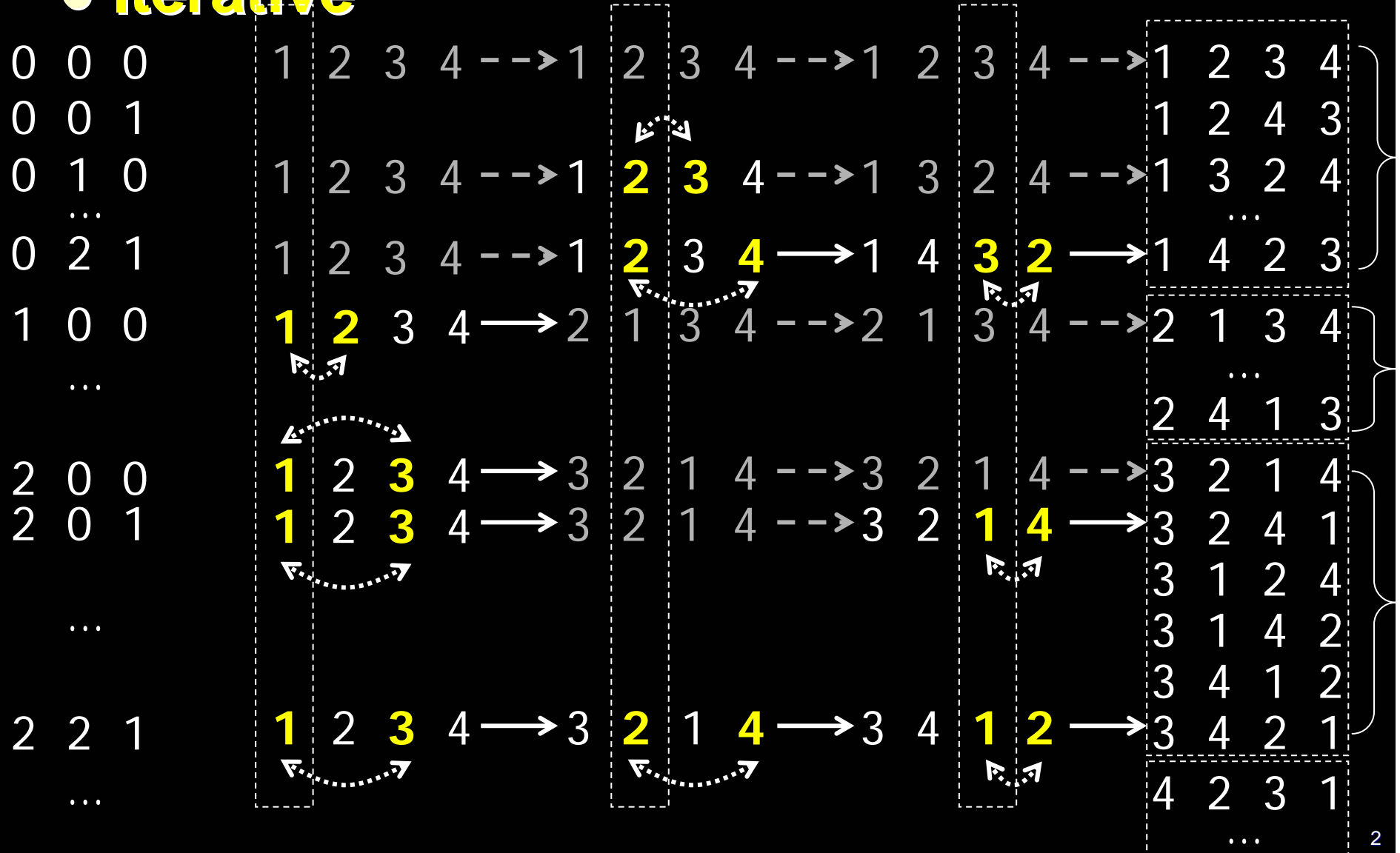
- Iterative

2 0 0
2 0 1



Permutation from Swapping (2/4)

- Iterative**



Permutation from Swapping (3/4)

- counting up

Permutation from Swapping (3/4)

- counting up

0	0	0			
1	1	1			
2	2				
3					

Permutation from Swapping (3/4)

- counting up

0	0	0	
1	1	1	
2	2		
3			

--	--

0	0	0
0	0	1
0	1	0
0	1	1
0	2	0
0	2	1

Permutation from Swapping (3/4)

- counting up

0	0	0	
1	1	1	
2	2		
3			

--	--

0	0	0
0	0	1
0	1	0
0	1	1
0	2	0
0	2	1
1	0	0
1	0	1
1	1	0
1	1	1
1	2	0
1	2	1

Permutation from Swapping (3/4)

- counting up

0	0	0	
1	1	1	
2	2		
3			

--	--

0	0	0	2	0	0
0	0	1	2	0	1
0	1	0	2	1	0
0	1	1	2	1	1
0	2	0	2	2	0
0	2	1	2	2	1
1	0	0			
1	0	1			
1	1	0			
1	1	1			
1	2	0			
1	2	1			

Permutation from Swapping (3/4)

- counting up

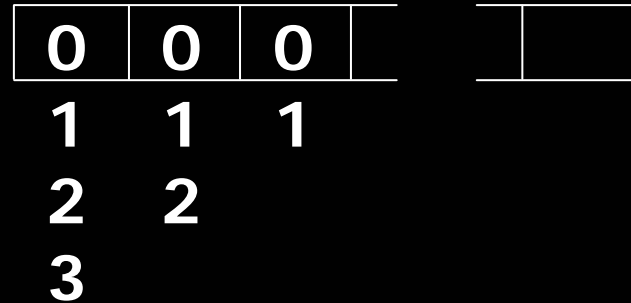
0	0	0	
1	1	1	
2	2		
3			

--	--

0	0	0	2	0	0
0	0	1	2	0	1
0	1	0	2	1	0
0	1	1	2	1	1
0	2	0	2	2	0
0	2	1	2	2	1
1	0	0	3	0	0
1	0	1	3	0	1
1	1	0	3	1	0
1	1	1	3	1	1
1	2	0	3	2	0
1	2	1	3	2	1

Permutation from Swapping (3/4)

- counting up

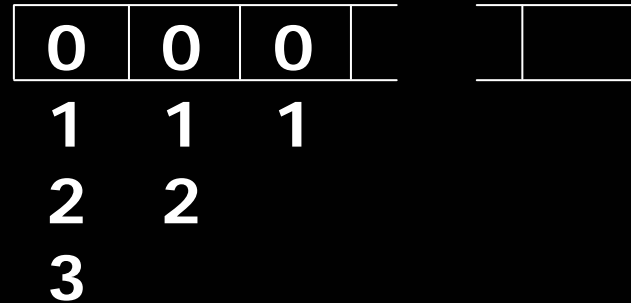


0	0	0	2	0	0
0	0	1	2	0	1
0	1	0	2	1	0
0	1	1	2	1	1
0	2	0	2	2	0
0	2	1	2	2	1
1	0	0	3	0	0
1	0	1	3	0	1
1	1	0	3	1	0
1	1	1	3	1	1
1	2	0	3	2	0
1	2	1	3	2	1

```
int next(int counter[], int size) {
    int i;
    for (i=size-1; i>=0; i--) {
        counter[i]++;
        if (counter[i]<=size-i)
            return 1;
        counter[i] = 0;
    }
    return 0;
}
```

Permutation from Swapping (3/4)

- counting up



0	0	0	2	0	0
0	0	1	2	0	1
0	1	0	2	1	0
0	1	1	2	1	1
0	2	0	2	2	0
0	2	1	2	2	1
1	0	0	3	0	0
1	0	1	3	0	1
1	1	0	3	1	0
1	1	1	3	1	1
1	2	0	3	2	0
1	2	1	3	2	1

```
int next(int counter[], int size) {
    int i;
    for (i=size-1; i>=0; i--) {
        counter[i]++;
        if (counter[i]<=size-i)
            return 1;
        counter[i] = 0;
    }
    return 0;
}
```

Initially, 0 0 -1