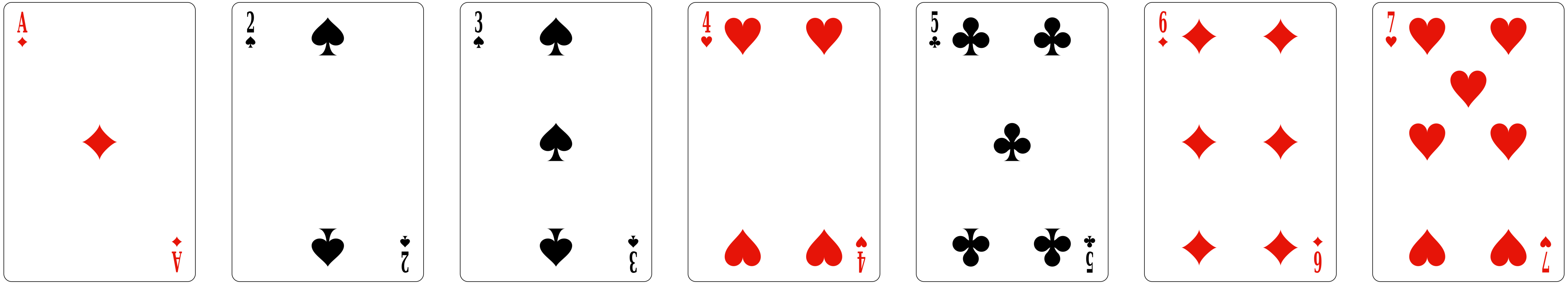
Random Permutation

Shusen Wang

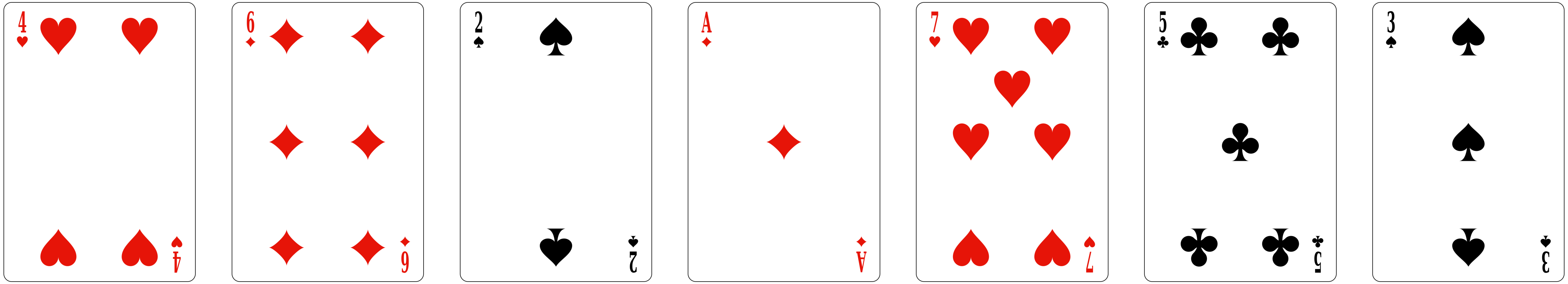
http://wangshusen.github.io/

Random Permutation



Random Permutation

Now, the cards have random order.



What is uniform random permutation?

Number of Permutations

- The permutations of `{A, B, C}`:
  - `A, B, C.`
  - `A, C, B.`
  - `B, A, C.`
  - `B, C, A.`
  - `C, A, B.`
  - `C, B, A.`

Number of Permutations

- The permutations of **`{A, B, C}`**:
  - **`A, B, C.`**
  - **`A, C, B.`**
  - **`B, A, C.`**
  - **`B, C, A.`**
  - **`C, A, B.`**
  - **`C, B, A.`**
- If a set contains n items, then there are $\boxed{n!}$ permutations.
- The factorial of n is

$$\underline{n! = n\times(n-1)\times(n-2)\times\cdots\times3\times2\times1}.$$

Number of Permutations

- The permutations of **`{A, B, C}`**:
  - **`A, B, C.`**
  - **`A, C, B.`**
  - **`B, A, C.`**
  - **`B, C, A.`**
  - **`C, A, B.`**
  - **`C, B, A.`**

There are $3! = 3\times2\times1 = 6$ possible arrangements.

- If a set contains n items, then there are $n!$ permutations.
- The factorial of n is

$$n! = n\times(n-1)\times(n-2)\times\cdots\times3\times2\times1.$$

Uniform Random Permutations

- The permutations of **`{A, B, C}`**:
  - **`A, B, C.`**
  - **`A, C, B.`**
  - **`B, A, C.`**
  - **`B, C, A.`**
  - **`C, A, B.`**
  - **`C, B, A.`**

Uniformly sample one out of the $3! = 6$ sequences.

Uniformly selecting one out of the $n!$ possible sequences.

Uniform Random Permutations

- The permutations of **`{A, B, C}`**:
  - **`A, B, C.`**
  - **`A, C, B.`**
  - **`B, A, C.`**
  - **`B, C, A.`**
  - **`C, A, B.`**
  - **`C, B, A.`**

An element appears in any of the n the positions with probability $\frac{1}{n}$.

Uniform Random Permutations

- The permutations of **`{A, B, C}`**:
  - `A, B, C.`
  - `A, C, B.`
  - `B, A, C.`
  - `B, C, A.`
  - `C, A, B.`
  - `C, B, A.`

An element appears in any of the n the positions with probability $\frac{1}{n}$.

Uniform Random Permutations

- The permutations of **`{A, B, C}`**:
  - **`A, B, C.`**
  - **`A, C, B.`**
  - **`B, A, C.`**
  - **`B, C, A.`**
  - **`C, A, B.`**
  - **`C, B, A.`**

A position is filled with any of the n items with probability $\frac{1}{n}$.

Uniform Random Permutations

- The permutations of **`{A, B, C}`**:
  - `A, B, C.`
  - `A, C, B.`
  - `B, A, C.`
  - `B, C, A.`
  - `C, A, B.`
  - `C, B, A.`

A position is filled with any of the n items with probability $\frac{1}{n}$.

Fisher-Yates Shuffle: Original Version

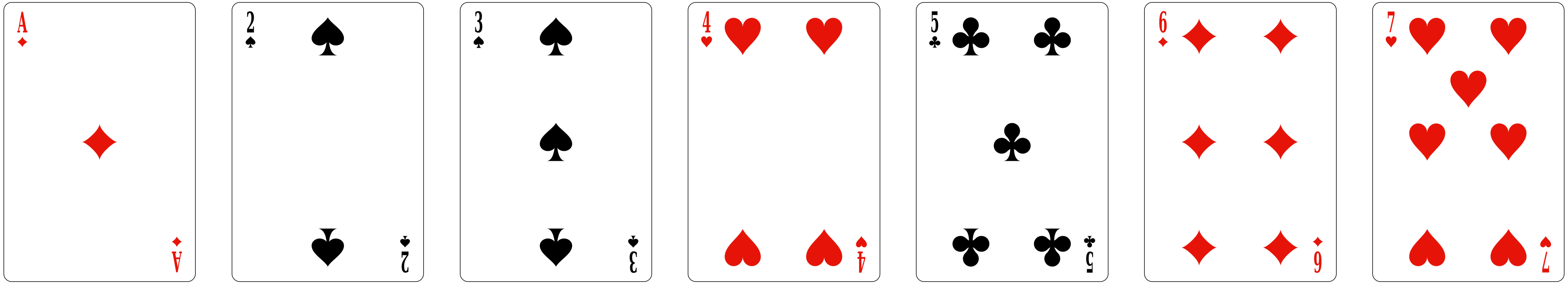
Reference

- Fisher & Yates. Statistical tables for biological, agricultural and medical research, 1938.

Random Integer Generator

- Assume we have a random integer generator.
- **Input:** integer n.
- **Output:** an element sampled from $\{0, 1, 2, \dots, n-1\}$ uniformly at random.

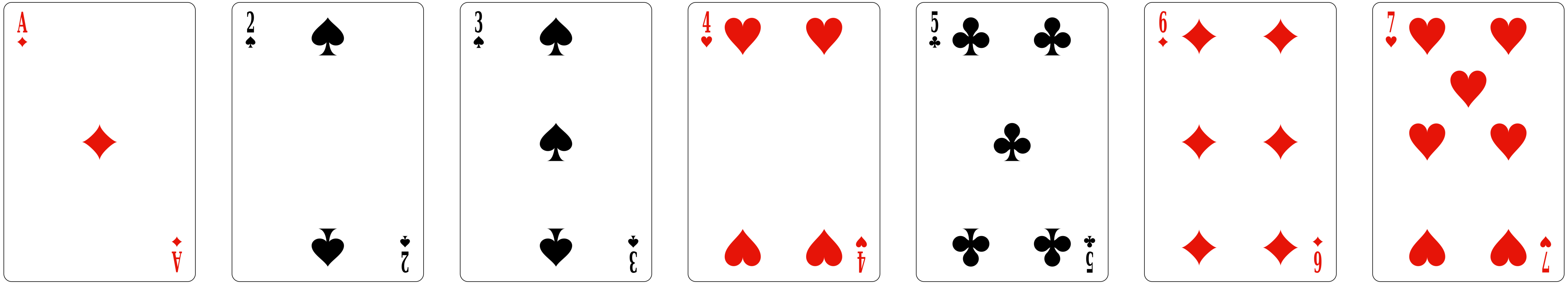
Fisher-Yates Shuffle: Original Version



0th 1st 2nd 3rd 4th 5th 6th

Iteration 0

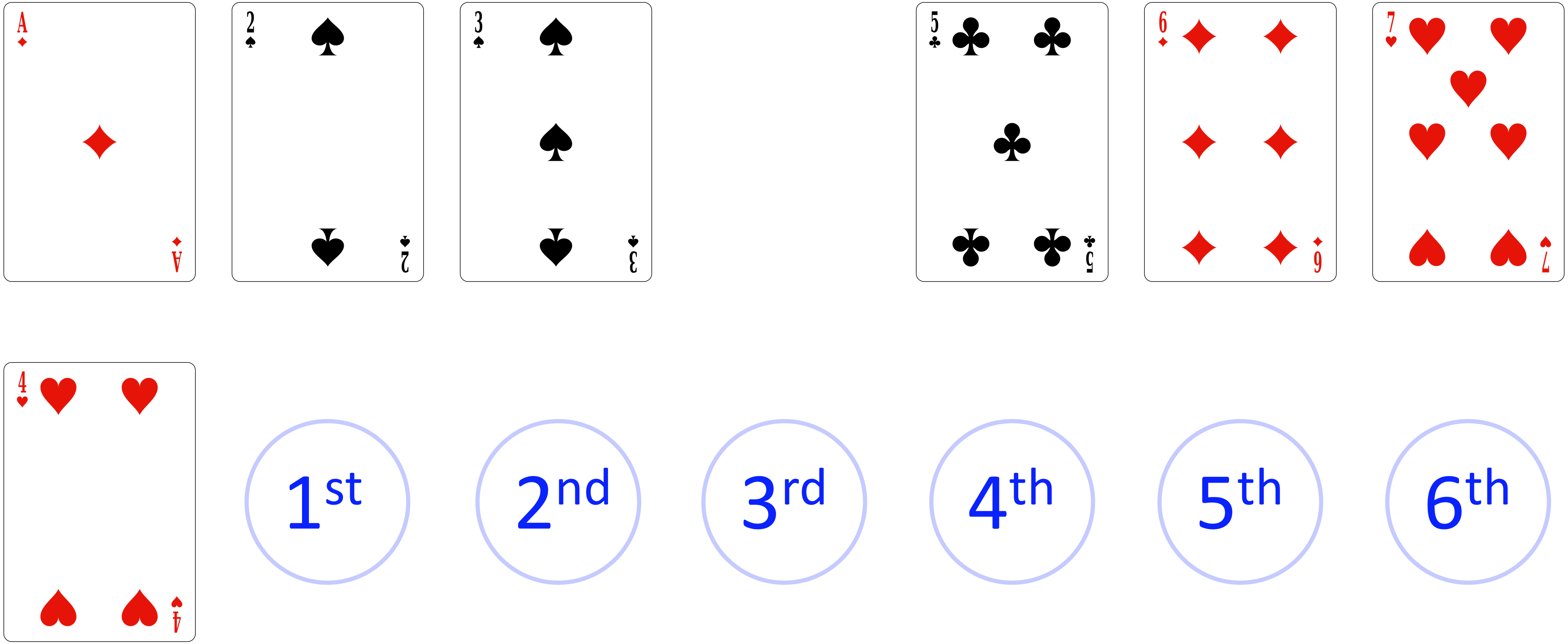
Sample an element from the set $\{1, 2, 3, 4, 5, 6, 7\}$ uniformly at random.



0th 1st 2nd 3rd 4th 5th 6th

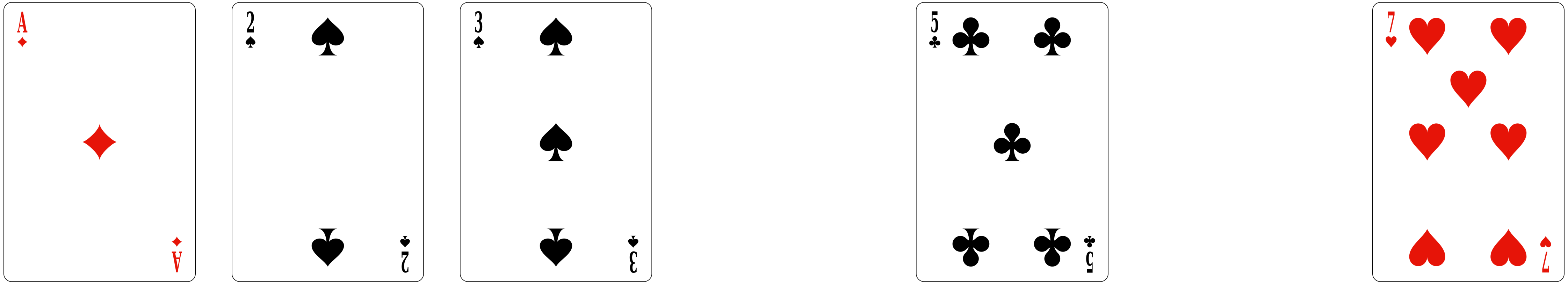
Iteration 1

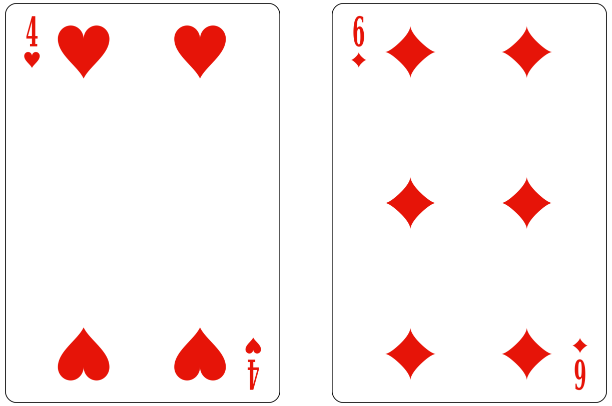
Sample an element from the set $\{1, 2, 3, 5, 6, 7\}$ uniformly at random.



Iteration 2

Sample an element from the set $\{1, 2, 3, 5, 7\}$ uniformly at random.





2nd

3rd

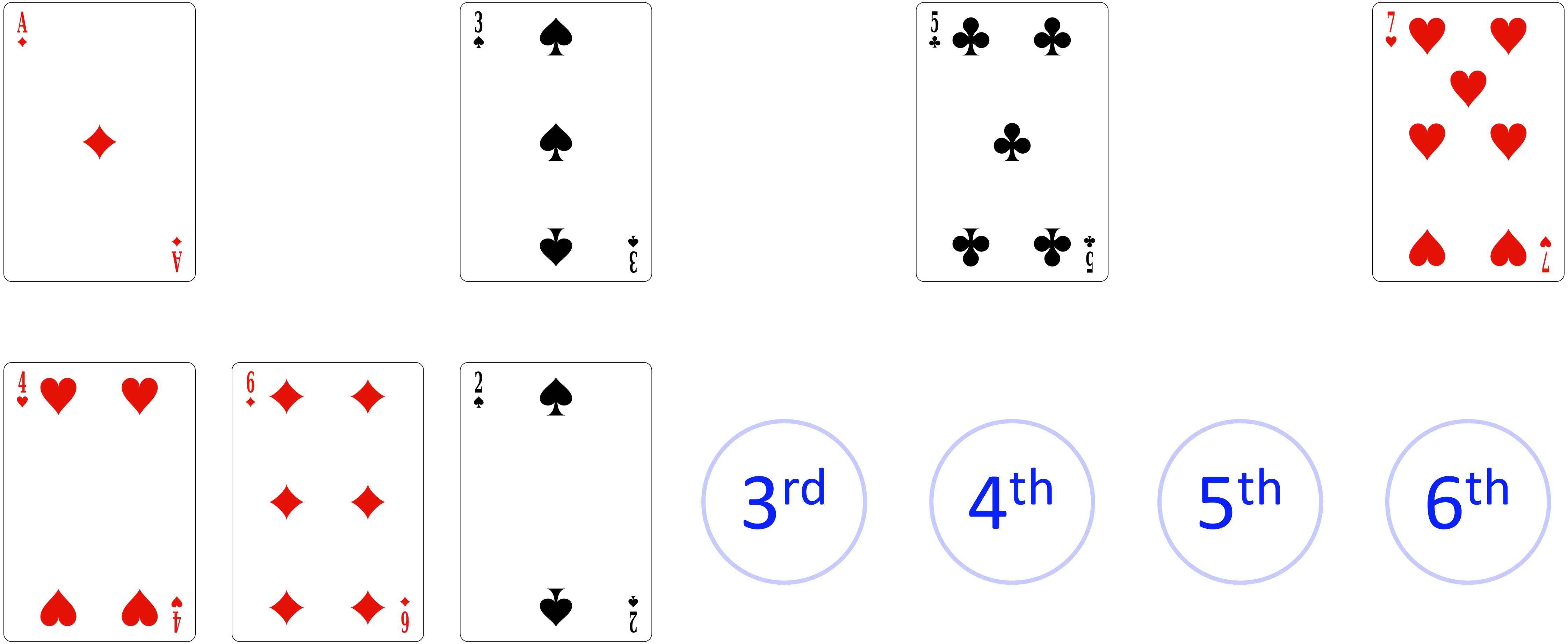
4th

5th

6th

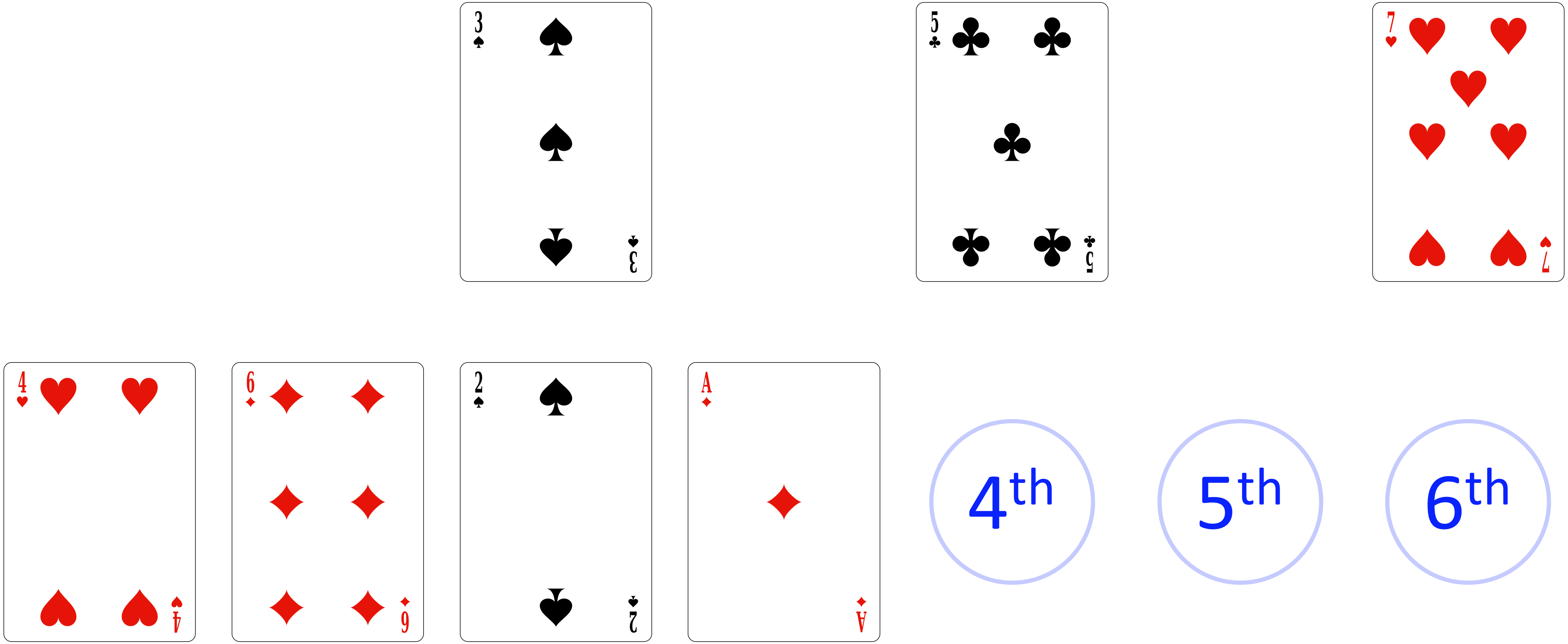
Iteration 3

Sample an element from the set $\{1, 3, 5, 7\}$ uniformly at random.



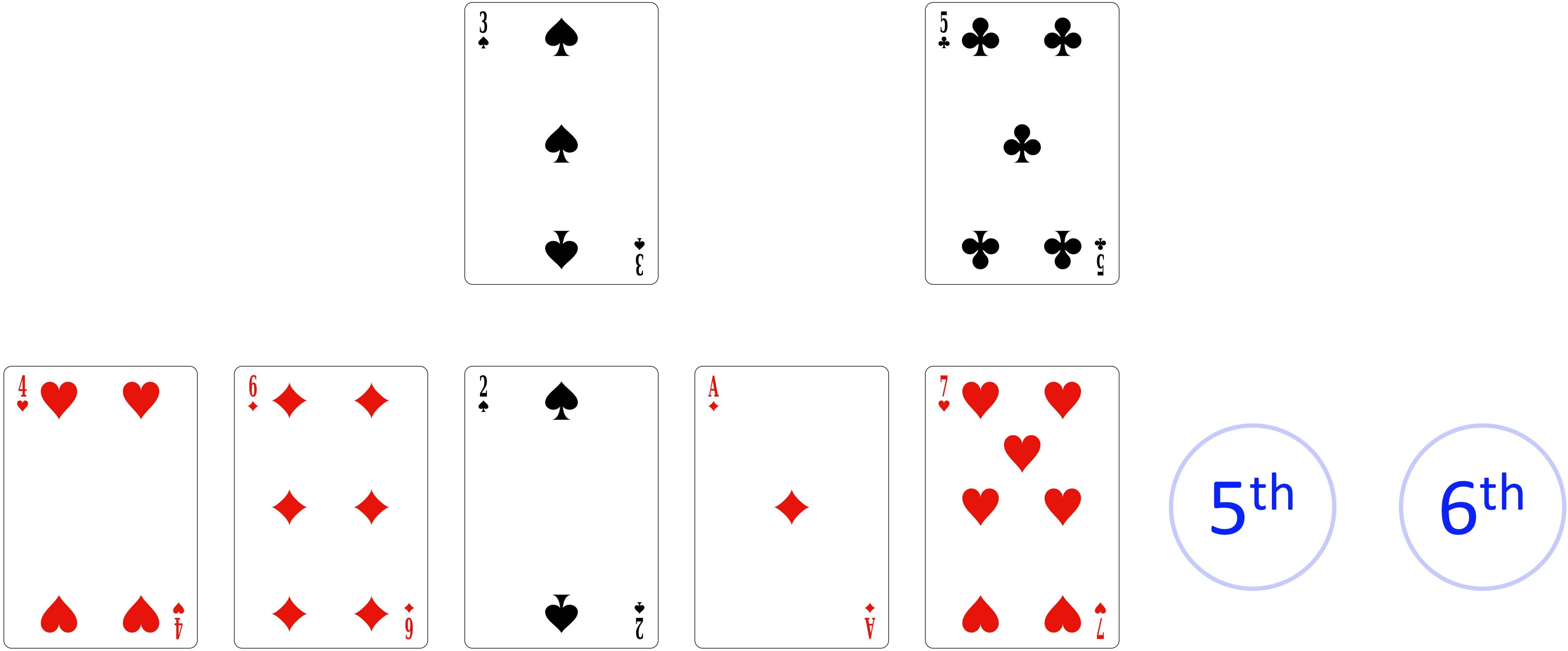
Iteration 4

Sample an element from the set $\{3, 5, 7\}$ uniformly at random.



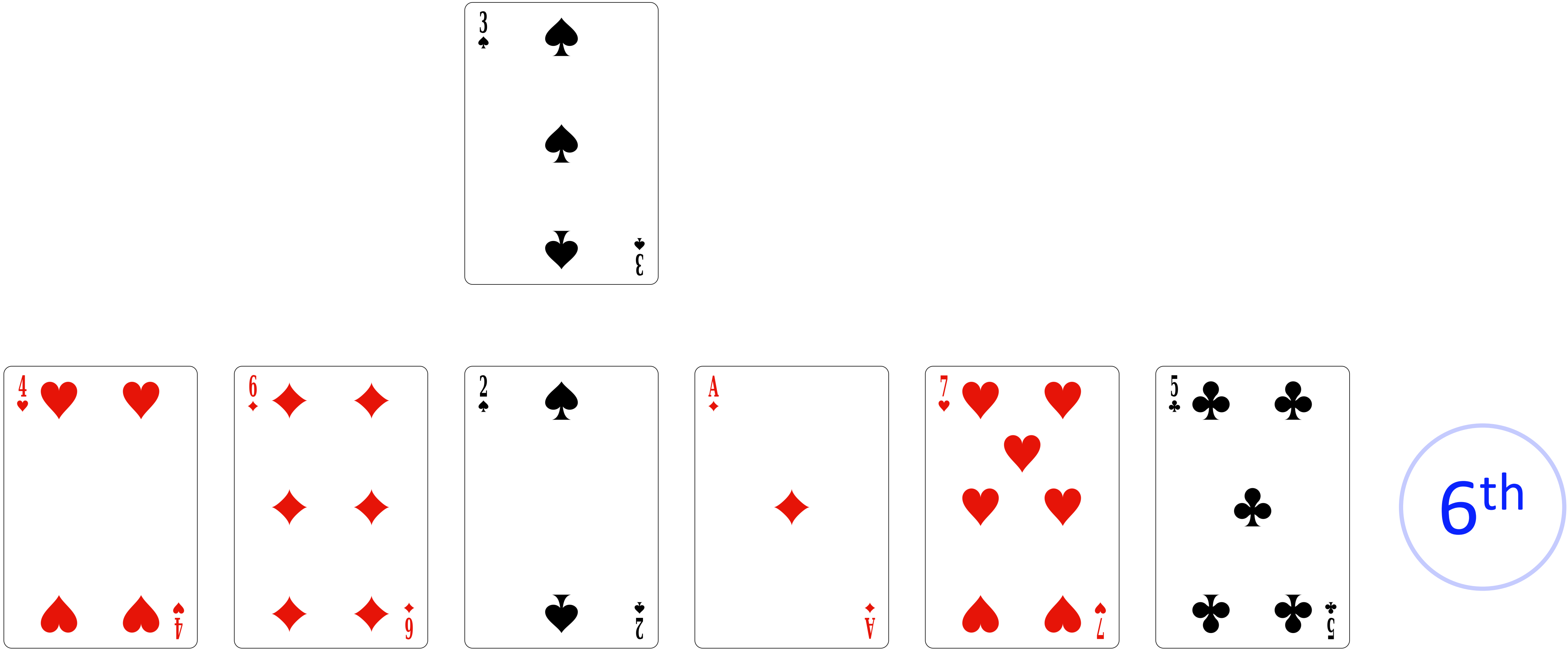
Iteration 5

Sample an element from the set $\{3, 5\}$ uniformly at random.



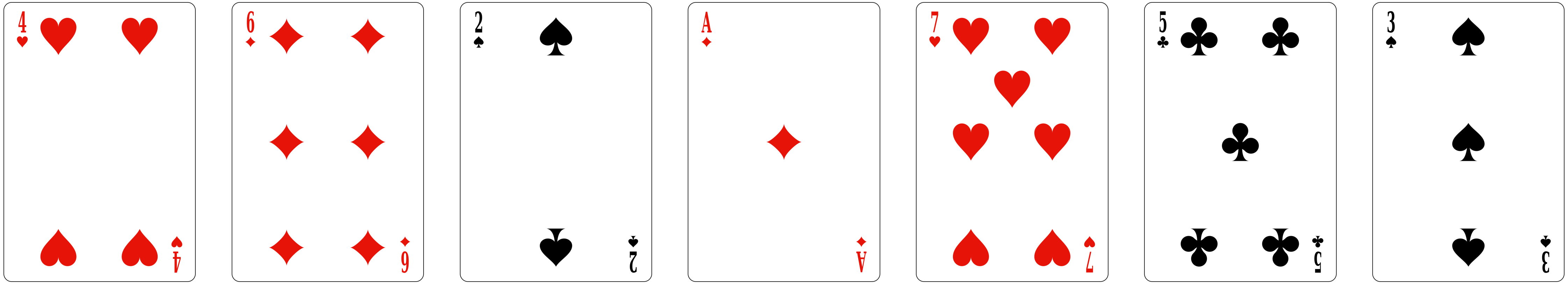
Iteration 6

Put the remaining card at the end.



End of Procedure

The obtained sequence is a uniform random permutation.



Implementation

Initial State:

A	B	C	D	E	F	G

- Assume we have a random integer generator:

```
int k = uniform(int n);
```

Implementation

Initial State:

A	B	C	D	E	F	G

- Assume we have a random integer generator:

```
int k = uniform(int n);
```

- **k** is sampled from $\{0, 1, 2, \cdots, n-1\}$ uniformly at random.

Implementation

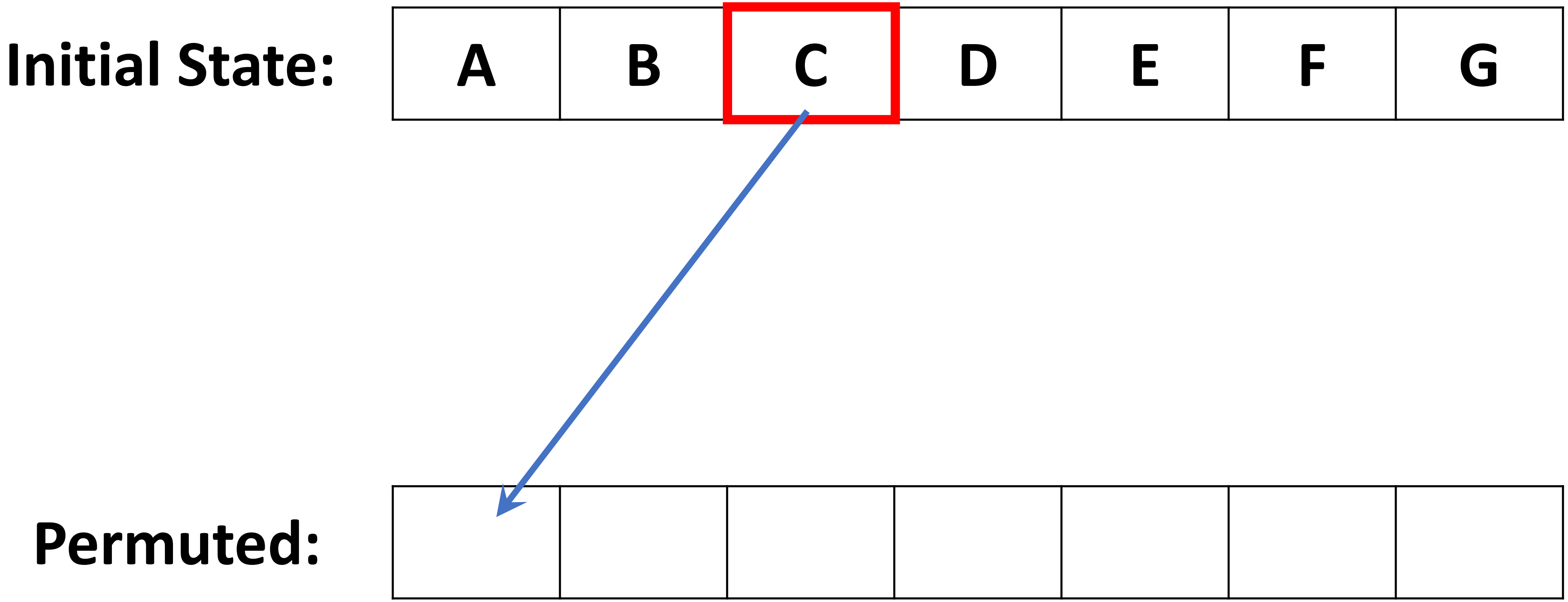
Initial State:

A	B	C	D	E	F	G

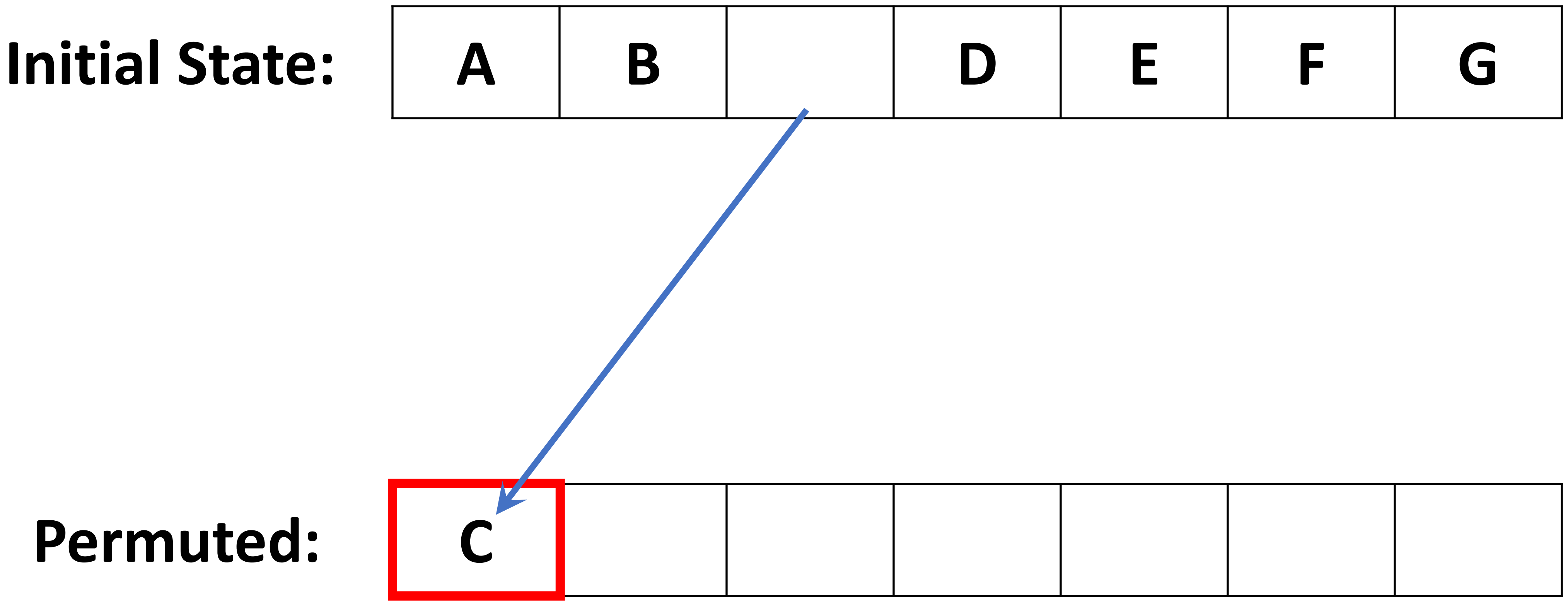
Randomly sample an element

Permuted:

Implementation



Implementation



Implementation

Initial State:

A	**B**		**D**	**E**	**F**	**G**

Fill the hole by moving the rest elements leftward.

Permuted:

C						

Implementation

Initial State:	A	B		D	E	F	G

Permuted:	C						

Time Complexity of Original Version

- In each iteration:
  - $\boxed{O(1)}$ time for random sampling.
  - $\boxed{O(n)}$ time (on average) for filling a hole.
- Overall time complexity:

$$\underline{n + (n-1) + \cdots + 3 + 2 + 1} = \underline{O(n^2)}.$$

Fisher-Yates Shuffle: Modern Version

Reference

- Durstenfeld. Algorithm 235: Random permutation. *Communications of the ACM*, 1964.

Initial State

Elements:

A	B	C	D	E	F	G
0	1	2	3	4	5	6

Iteration 0

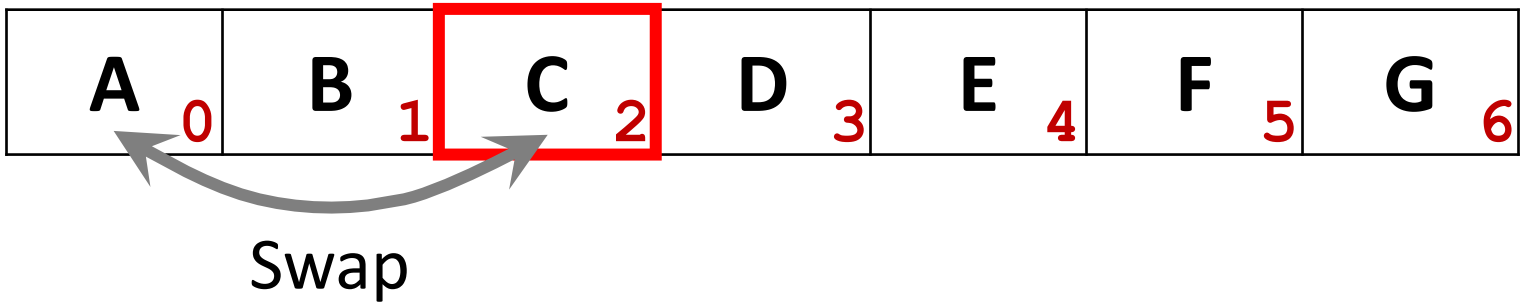
Elements:

A 0	B 1	C 2	D 3	E 4	F 5	G 6

Randomly sample an element

Iteration 0

Elements:



Iteration 0

Elements:	**C**	**B**	**A**	**D**	**E**	**F**	**G**
	0	1	2	3	4	5	6

This is a randomly sampled element.

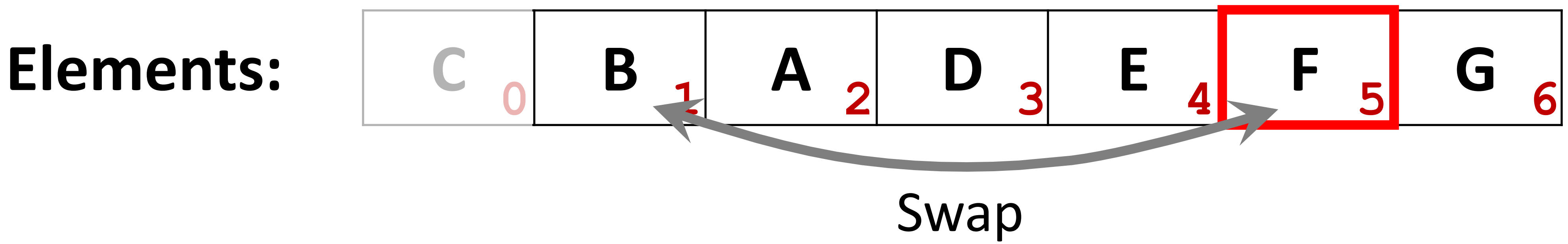
Iteration 1

Elements:

C 0	B 1	A 2	D 3	E 4	F 5	G 6

Randomly sample an element

Iteration 1



Iteration 1

Elements:	C 0	F 1	A 2	D 3	E 4	B 5	G 6

This is a randomly sampled element.

Iteration 2

Elements:	C	F	A	D	E	B	G
	0	1	2	3	4	5	6

Randomly sample an element

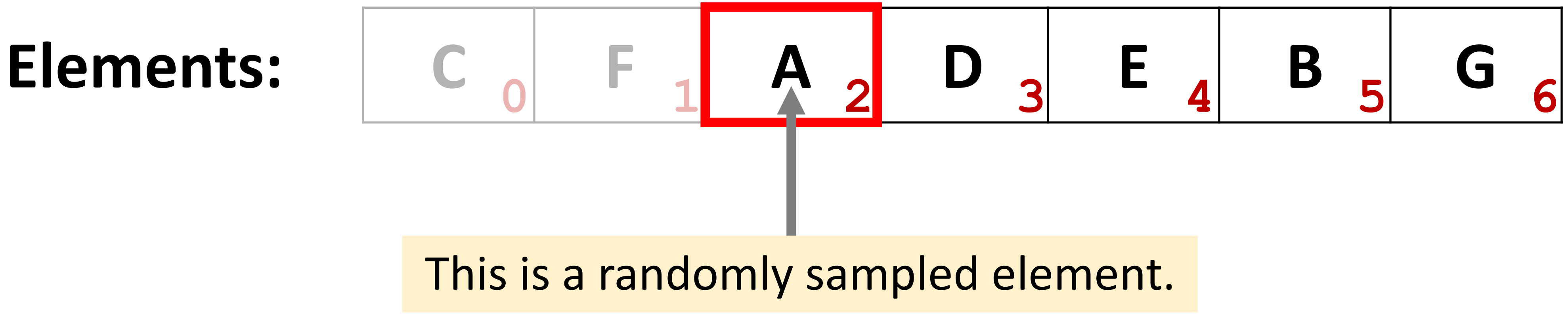
Iteration 2

Elements:

C 0	F 1	A 2	D 3	E 4	B 5	G 6

Swap it with itself

Iteration 2


Elements:
C 0
F 1
A 2
D 3
E 4
B 5
G 6
This is a randomly sampled element.

Iteration 3

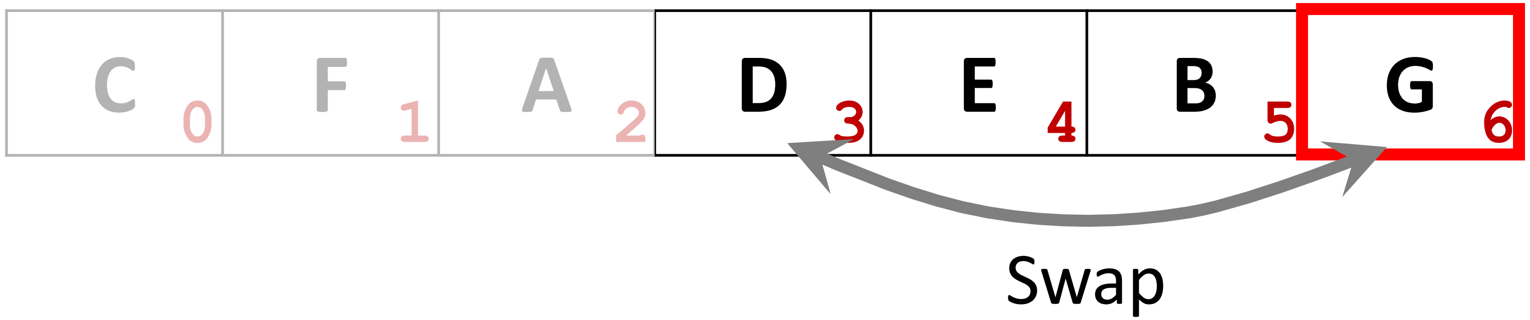
Elements:

C	F	A	D	E	B	G
0	1	2	3	4	5	6

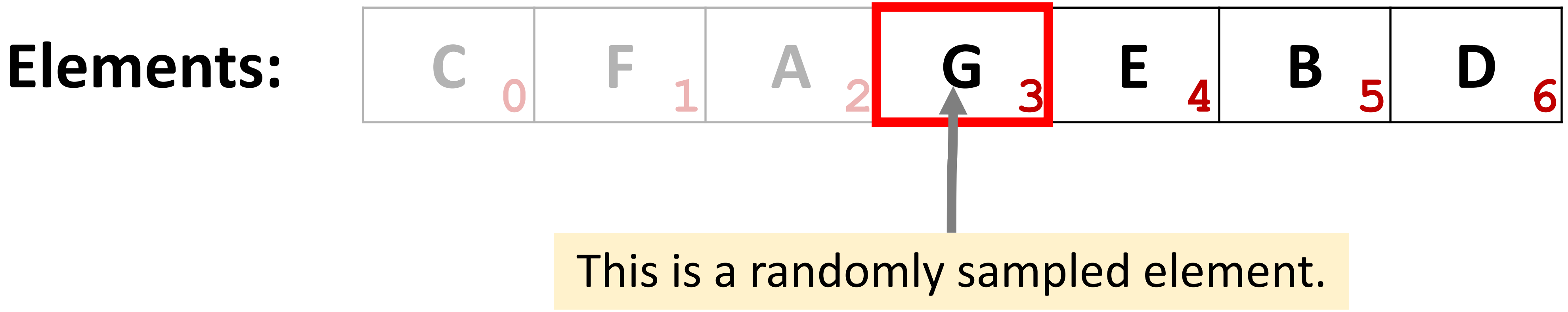
Randomly sample an element

Iteration 3

Elements:

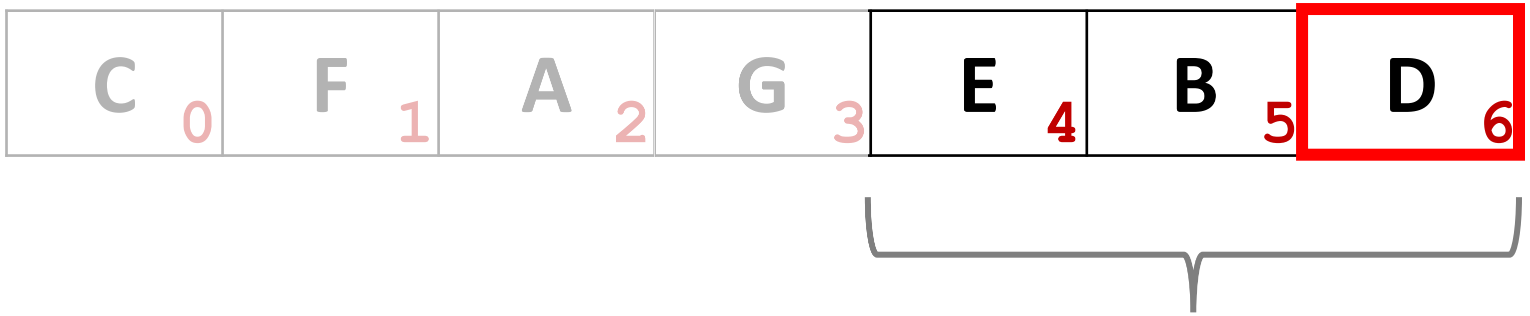


Iteration 3


Elements:
C 0
F 1
A 2
G 3
E 4
B 5
D 6
This is a randomly sampled element.

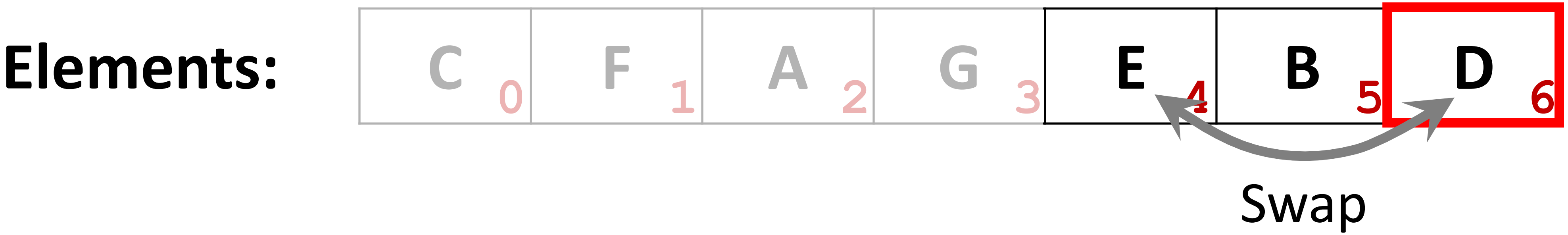
Iteration 4

Elements:



Randomly sample an element

Iteration 4



Iteration 4

Elements:

C 0	F 1	A 2	G 3	D 4	B 5	E 6

This is a randomly sampled element.

Iteration 5

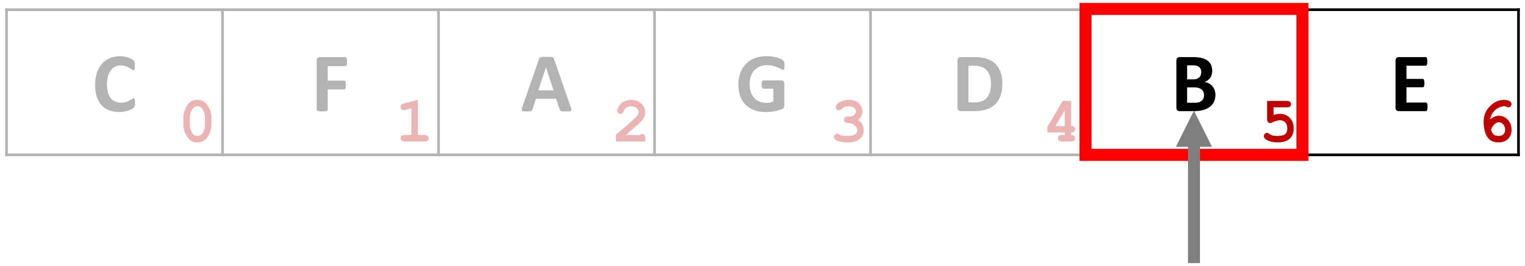
Elements:

C 0	F 1	A 2	G 3	D 4	$\boxed{\text{B } 5}$	E 6

Randomly sample an element

Iteration 5

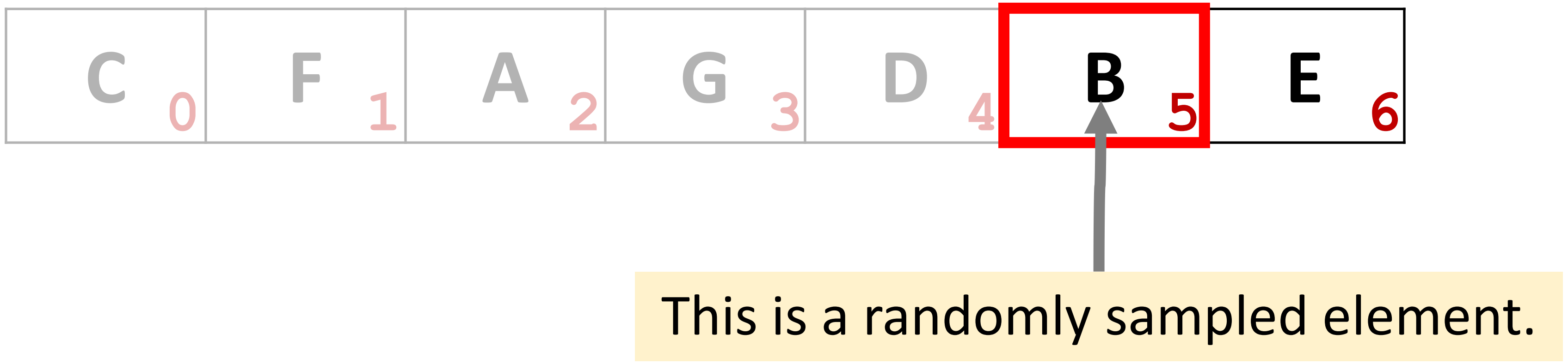
Elements:



Swap it with itself

Iteration 5

Elements:



End of Procedure

Elements:

C 0	F 1	A 2	G 3	D 4	B 5	E 6

Leave the last element alone.

End of Procedure

Elements:	C 0	F 1	A 2	G 3	D 4	B 5	E 6

```c
void permute(int arr[], int n) {
    int i;
    for (i=0; i <= n-2; i++) {
        // k is sampled from {0, 1, ..., n-i-1}
        int k = uniform(n-i);
        // j is in {i, i+1, ..., n-1}
        int j = i + k;
        // put arr[j] at the i-th position
        swap(arr, i, j);
    }
}
```

```
void permute(int arr[], int n) {
  int i;
  for (i=0; i <= n-2; i++) {
        // k is sampled from {0, 1, ..., n-i-1}
        int k = uniform(n-i);
        // j is in {i, i+1, ..., n-1}
        int j = i + k;
        // put arr[j] at the i-th position
        swap(arr, i, j);
  }
}
```

```
void permute(int arr[], int n) {
    int i;
    for (i=0; i <= n-2; i++) {
        // k is sampled from {0, 1, ..., n-i-1}
        int k = uniform(n-i);
        // j is in {i, i+1, ..., n-1}
        int j = i + k;
        // put arr[j] at the i-th position
        swap(arr, i, j);
    }
}
```

Explain the code

Elements:

C	B	A	D	E	F	G

i is the current position

Currently, it is the i-th iteration.

Explain the code

Elements:

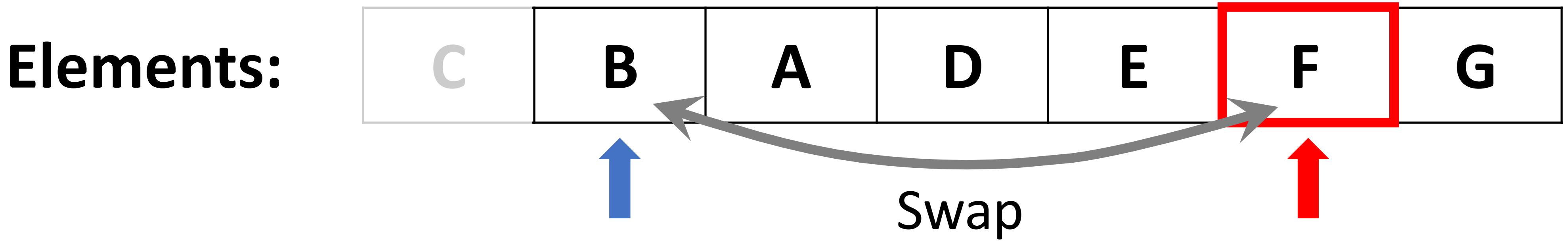
C	B	A	D	E	F	G

i is the current position

j is a random integer

Currently, it is the i-th iteration.

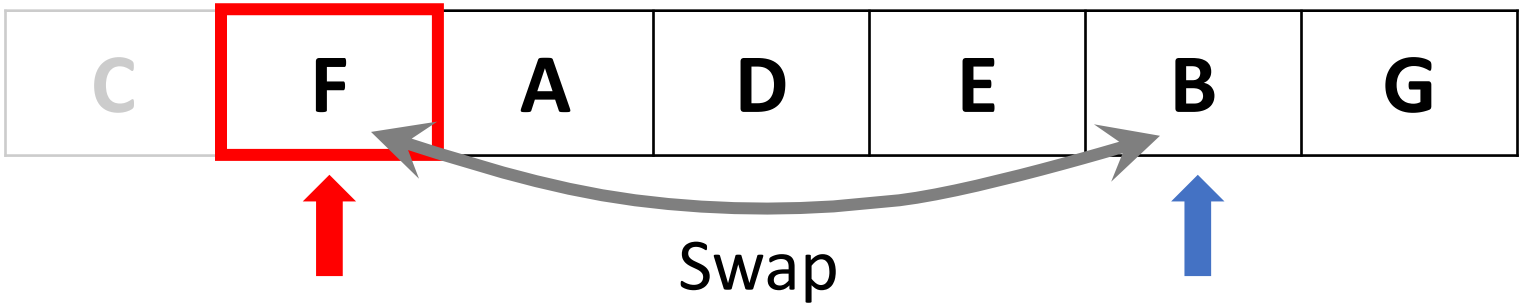
Explain the code



Currently, it is the i-th iteration.

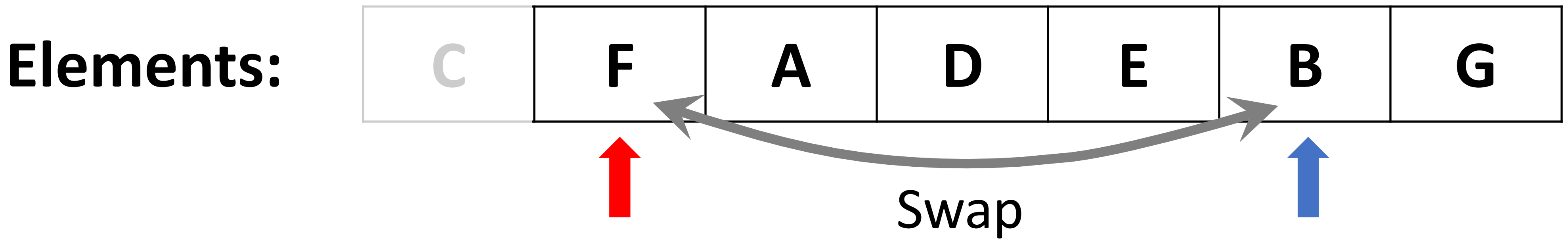
Explain the code

Elements:



After the i-th iteration.

Time Complexity



- The per-iteration time complexity is $O(1)$.
- Totally $n - 1$ iterations.
- Thus, the overall time complexity is $O(n)$.

Summary

Uniform Random Permutation

- What is a uniform random permutation?
- We are given n items.
- **Definition 1:** Select one out of the $n!$ permutations uniformly at random.
- **Definition 2:** The order of an item, after the permutation, can be any of $\{0, 1, \cdots, n-1\}$, with equal probability.
- **Definition 3:** A position, after the permutation, can be filled with any one of the n items, with equal probability.

Fisher-Yates Shuffle

- **Basic idea:** In the i-th iteration, randomly select an item from the remaining items, and put it in the i-th position.
- **Original version:** $O(n^2)$ time complexity, requires an extra array.
- **Modern version:** $O(n)$ time complexity, in place.

Thank You!