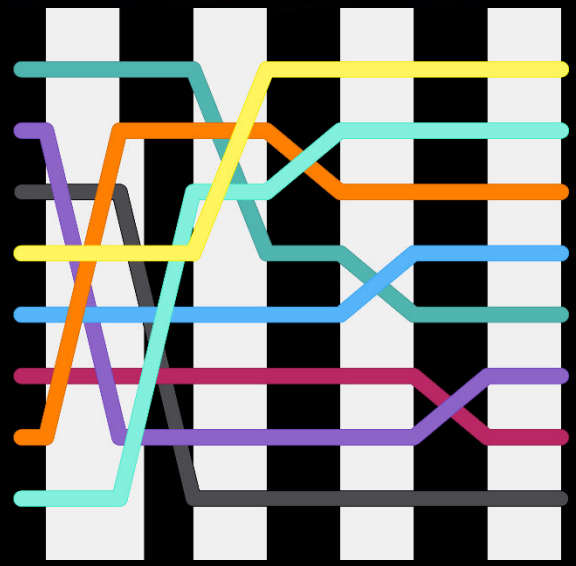
Sorting and Searching Algorithms

Sorting, Searching, Shuffling



SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

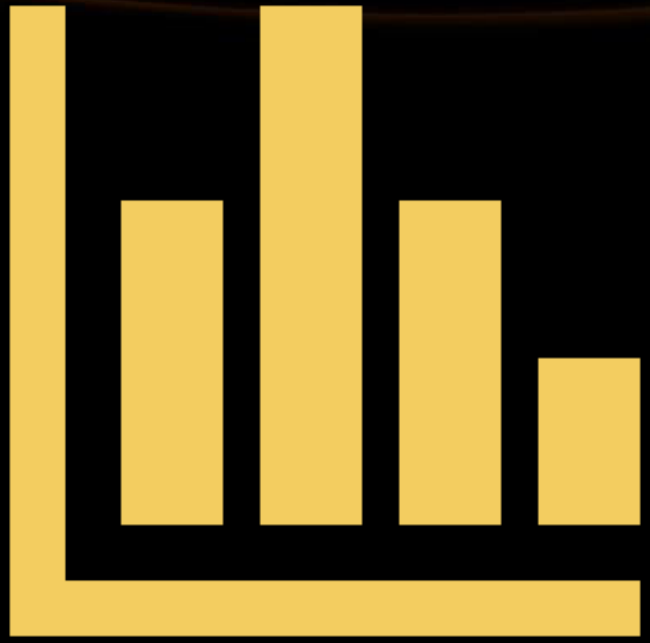
Have a Question?

SOFTWARE UNIVERSITY FOUNDATION

sli.do

#DsAlgo

SOFTWARE UNIVERSITY FOUNDATION

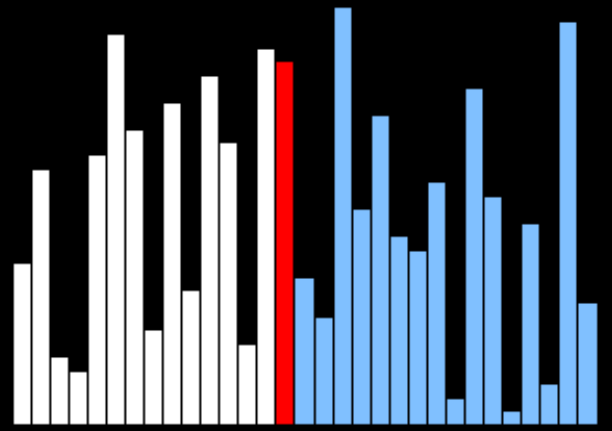


Simple Sorting Algorithms

BubbleSort, InsertionSort, SelectSort

What is a Sorting Algorithm?

- Sorting algorithm
  - An algorithm that rearranges elements in a list
    - In non-decreasing order
  - Elements must be comparable
- More formally
  - The input is a sequence / list of elements
  - The output is an rearrangement / permutation of elements
    - In non-decreasing order



Sorting – Example

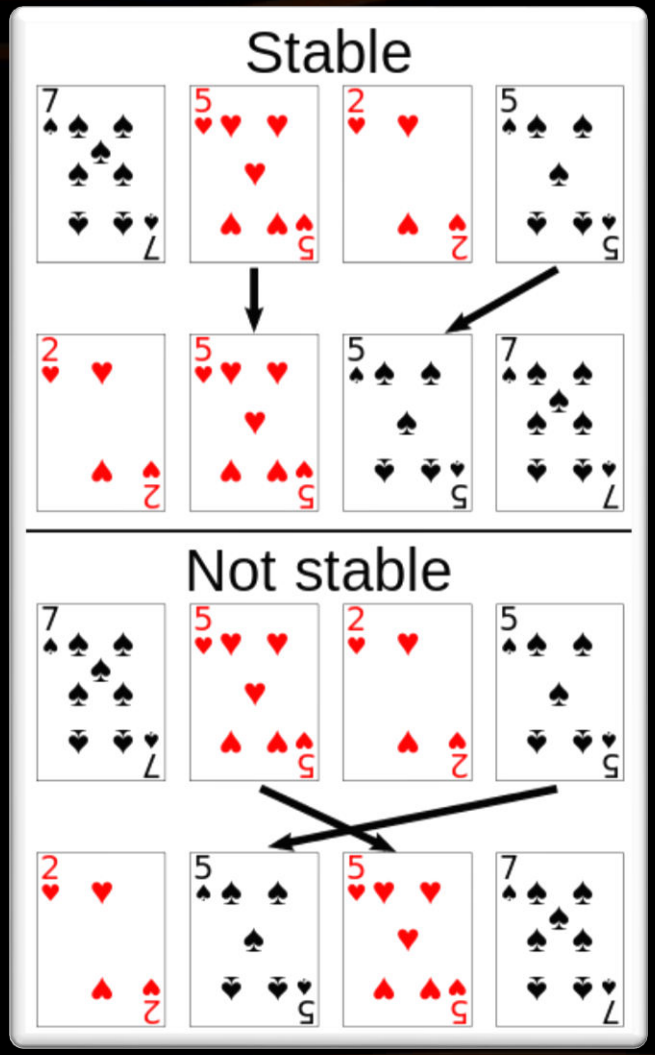
- Efficient sorting algorithms are important for
  - Producing human-readable output
  - Canonicalizing data – making data uniquely arranged
  - In conjunction with other algorithms, like binary searching
- Example of sorting:

Sorting Algorithms: Classification

- Sorting algorithms are often classified by
  - Computational complexity and memory usage
    - Worst, average and best case behavior
  - Recursive / non-recursive
  - Stability – stable / unstable
  - Comparison-based sort / non-comparison based
  - Sorting method: insertion, exchange (bubble sort and quicksort), selection (heapsort), merging, serial / parallel, etc.

Stability of Sorting

- Stable sorting algorithms
  - Maintain the order of equal elements
  - If two items compare as equal, their relative order is preserved
- Unstable sorting algorithms
  - Rearrange the equal elements in unpredictable order
- Often different elements have same key used for equality comparing



Sorting Helper Methods

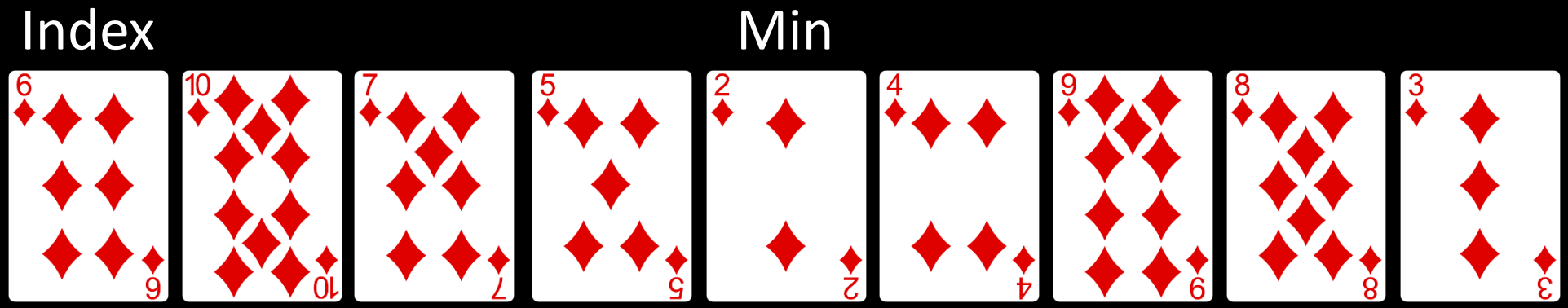

```
static void Swap<T>(T[] collection, int from, int to)
{
  //TODO: Swap the elements
}
```

```
static bool Less(IComparable first, IComparable second)
{
  return first.CompareTo(second) < 0;
}
```

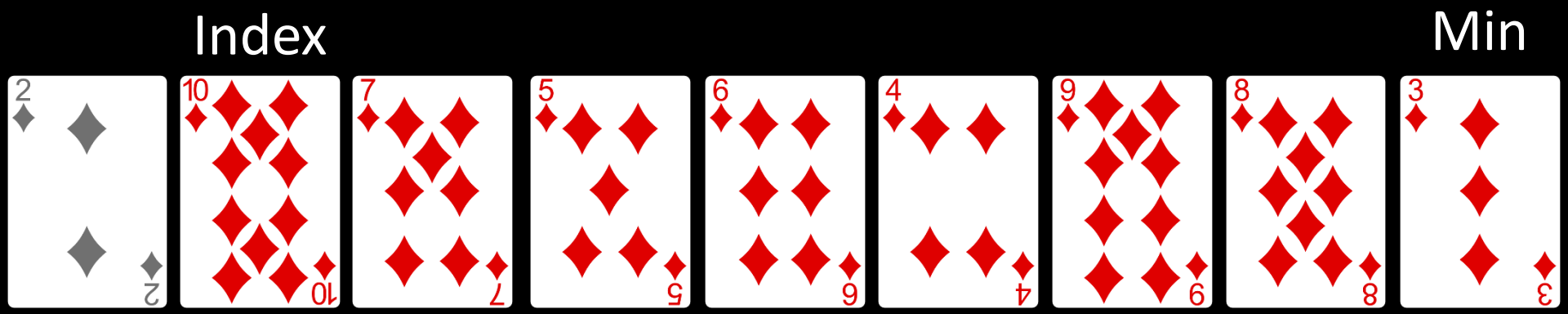
Selection Sort

- Selection sort – simple, but inefficient algorithm (visualize)
  - Swap the first with the min element on the right, then the second, etc.
  - Memory: O(1)
  - Stable: No
  - Method: Selection

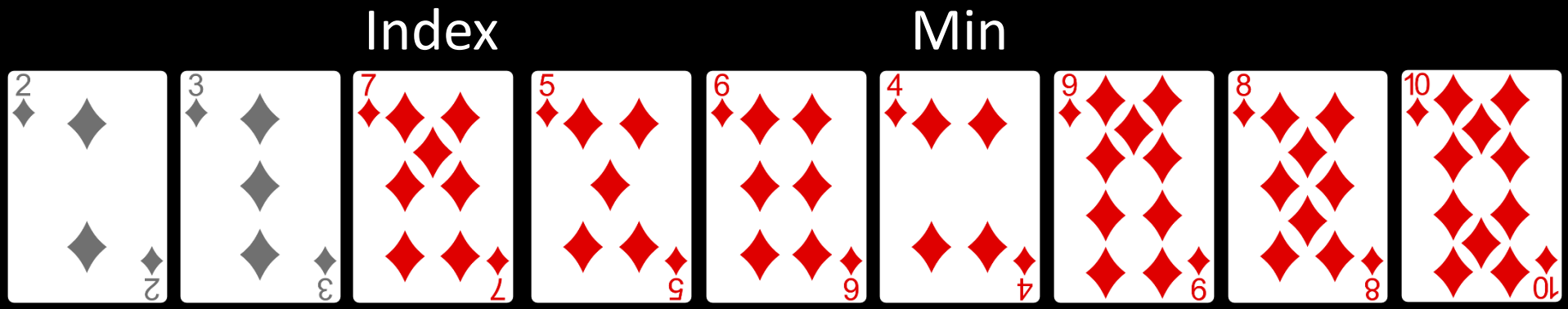
Selection Sort Visualization



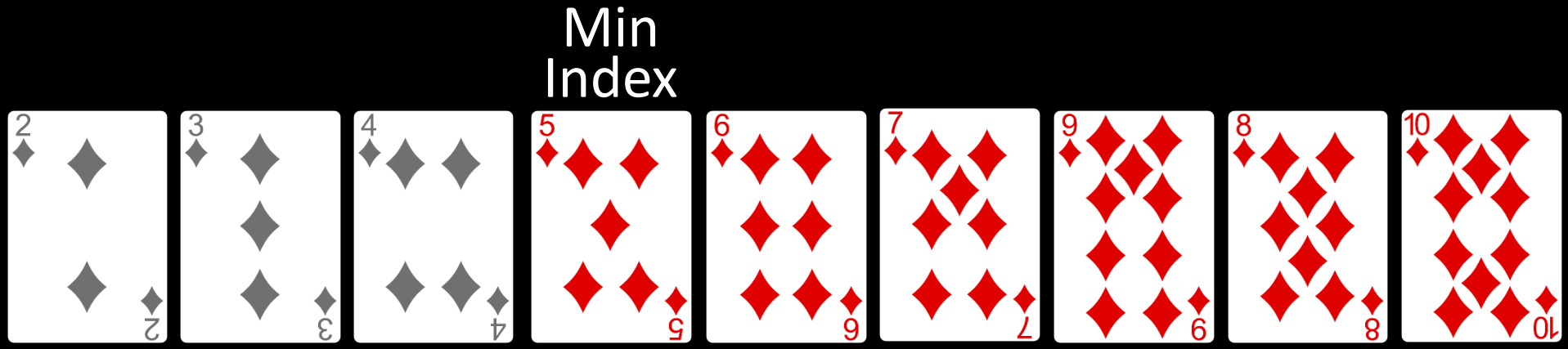
Selection Sort Visualization



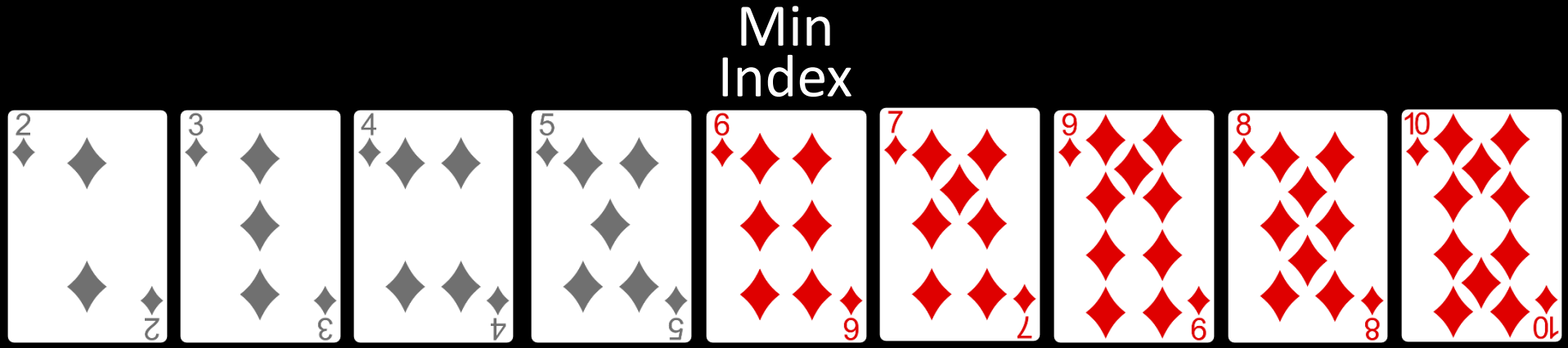
Selection Sort Visualization



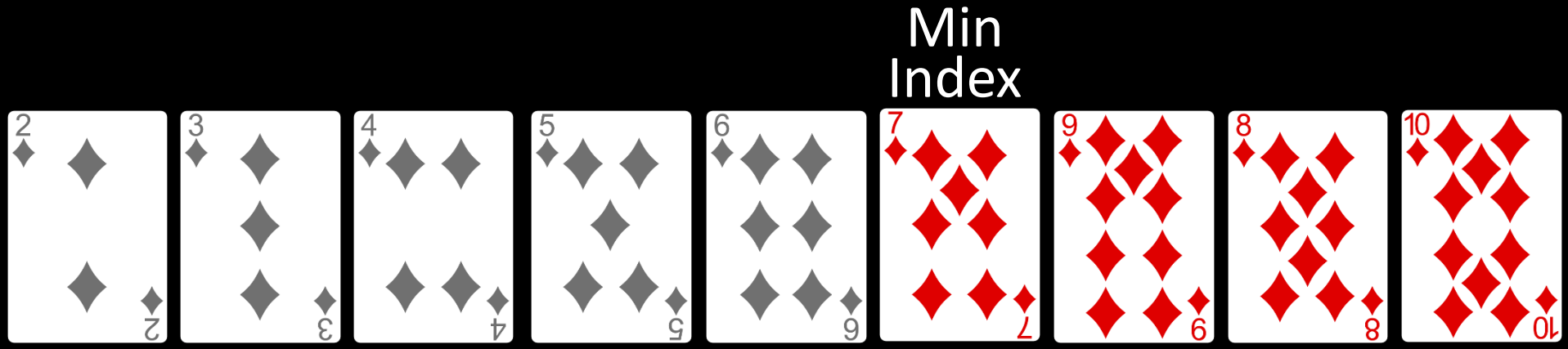
Selection Sort Visualization



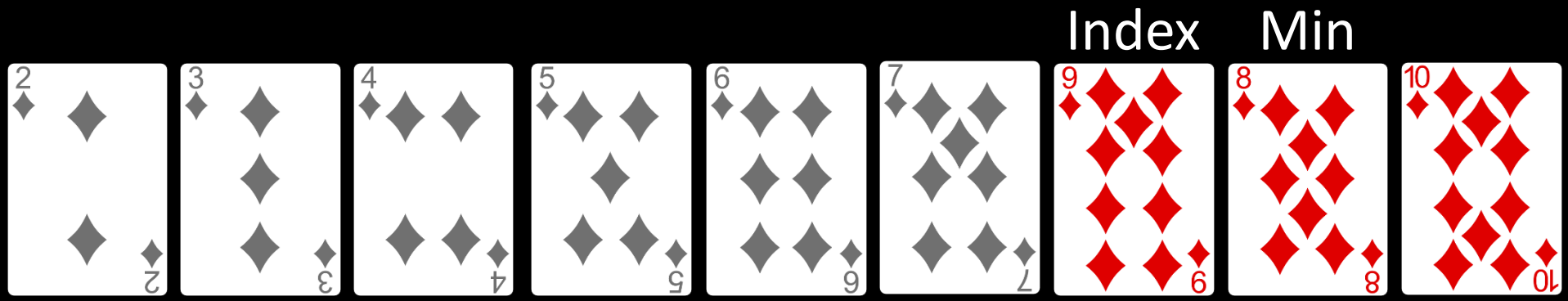
Selection Sort Visualization



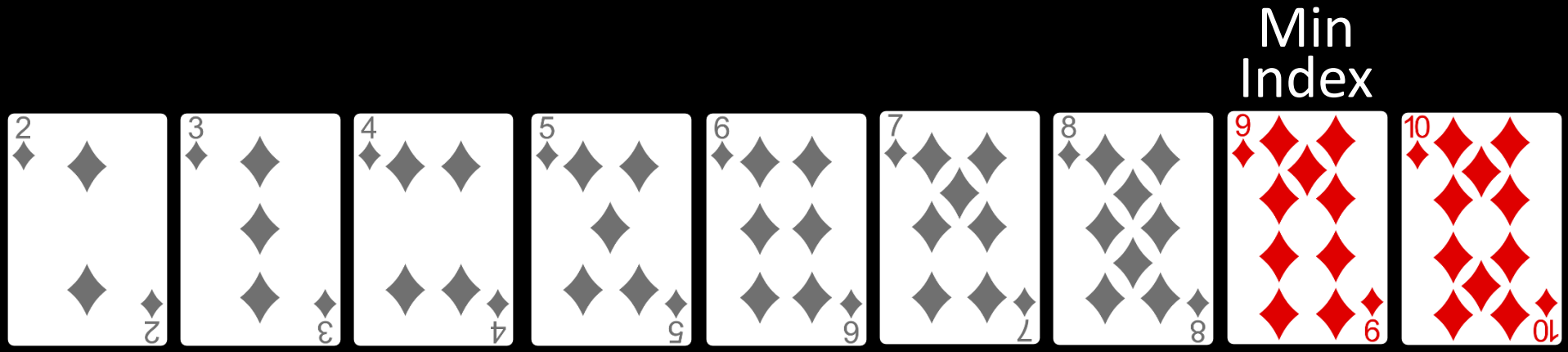
Selection Sort Visualization



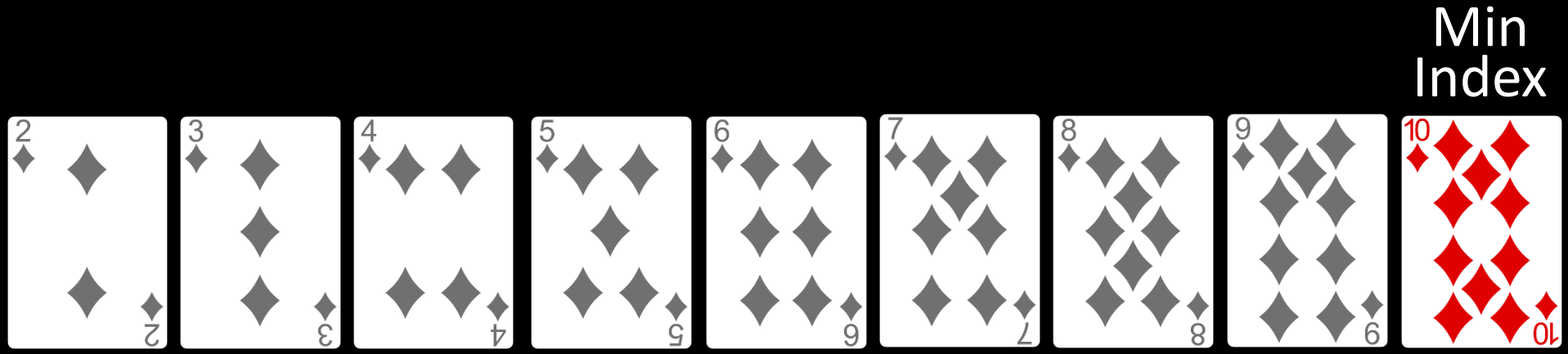
Selection Sort Visualization



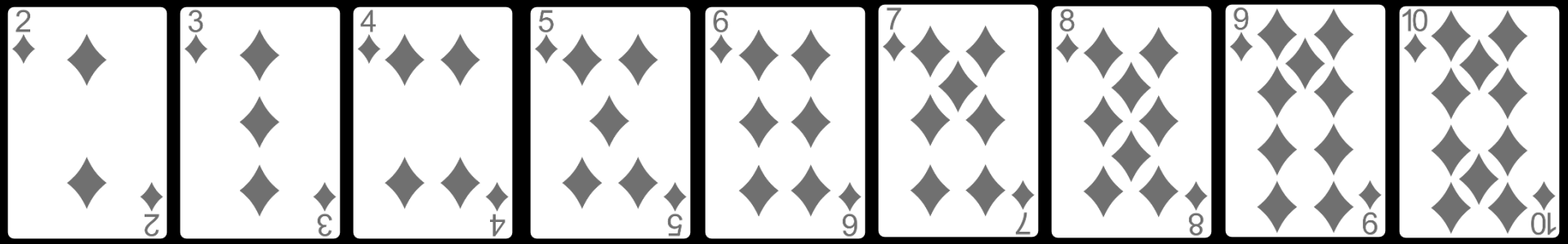
Selection Sort Visualization



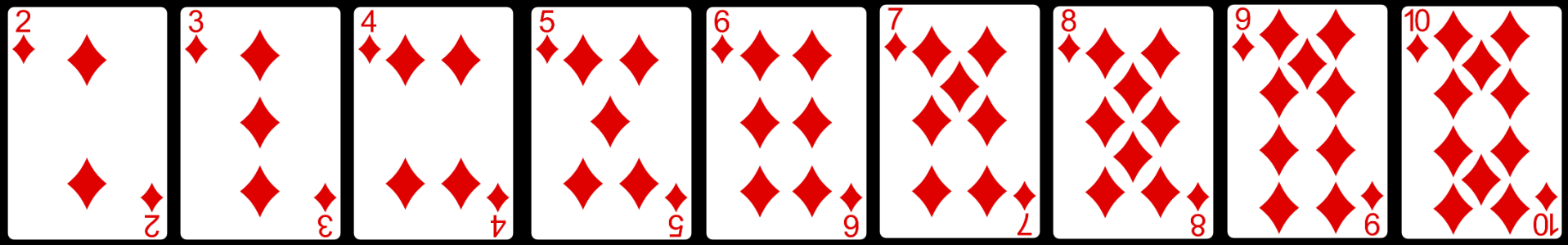
Selection Sort Visualization



Selection Sort Visualization

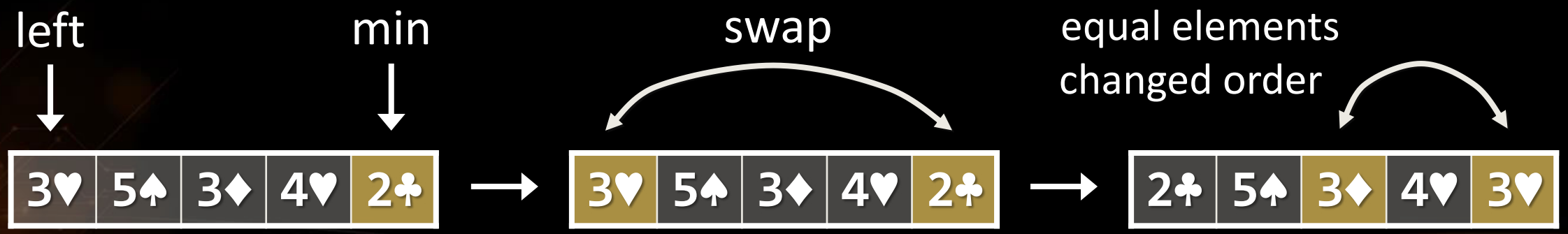


Selection Sort Visualization



Selection Sort: Why Unstable?

- Why the "selection sort" is **unstable**?
  1. Swaps the first element with the min element on the right
  2. Swaps the second element with the min element on the right
  3. …
- During the swaps equal elements can jump over each other



Selection Sort Code

```
for (int index = 0; index < collection.Length; index++)
{
  int min = index;
  for (int curr = index + 1; curr < collection.Length; curr++)
  {
    if (Less(collection[curr], collection[min]))
    {
       min = curr;
    }
  }
  Swap(collection, index, min);
}
```

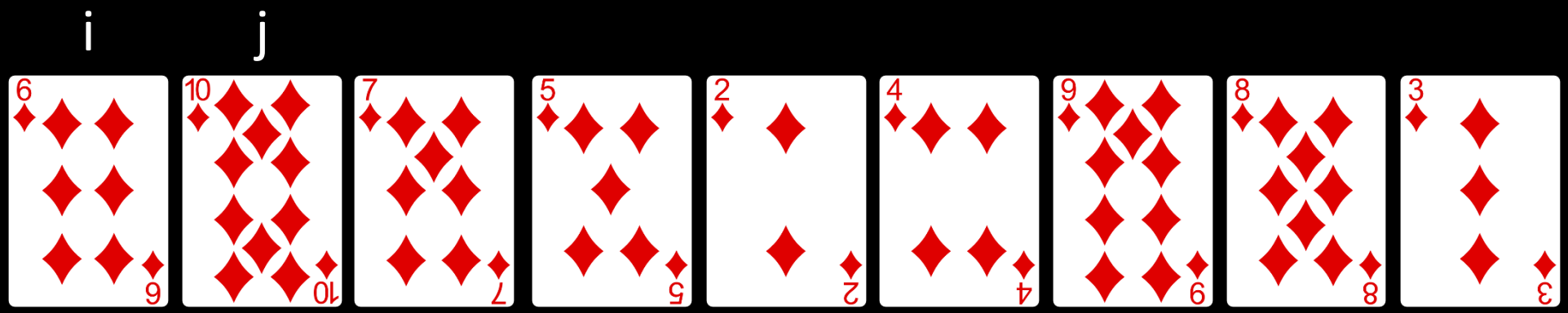
Comparison of Sorting Algorithms

Name	Best	Average	Worst	Memory	Stable	Method
SelectionSort	n^2	n^2	n^2	1	No	Selection

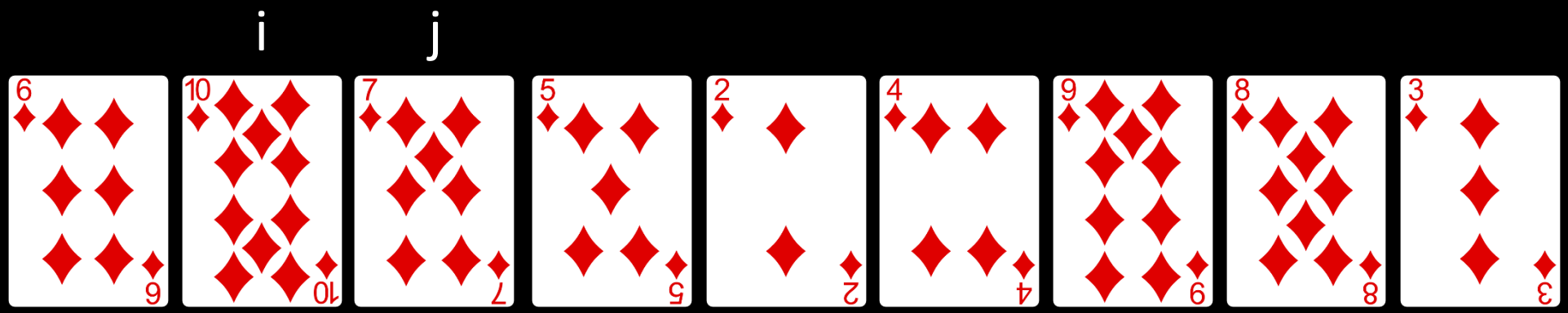
Bubble Sort

- Bubble sort – simple, but inefficient algorithm (visualize)
  - Swaps to neighbor elements when not in order until sorted
  - Memory: O(1)
  - Stable: Yes
  - Method: Exchanging

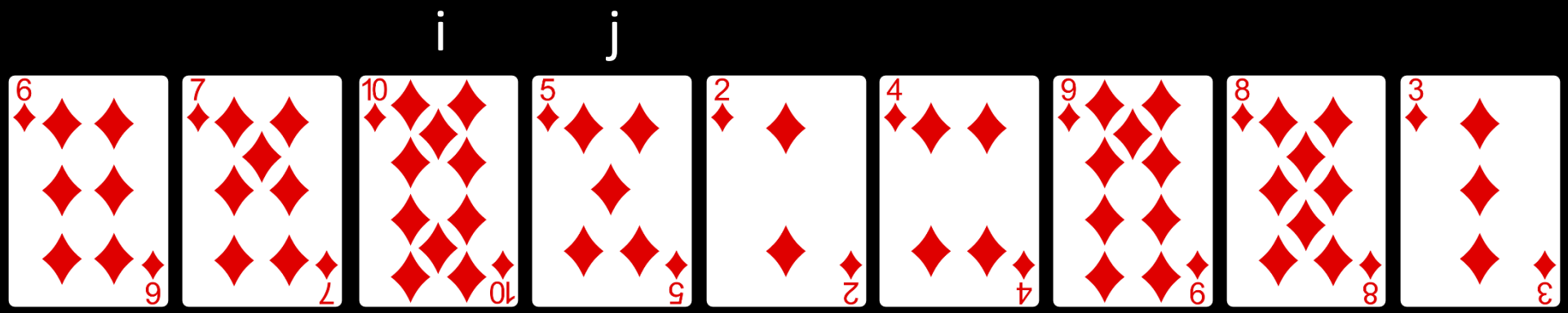
Bubble Sort Visualization



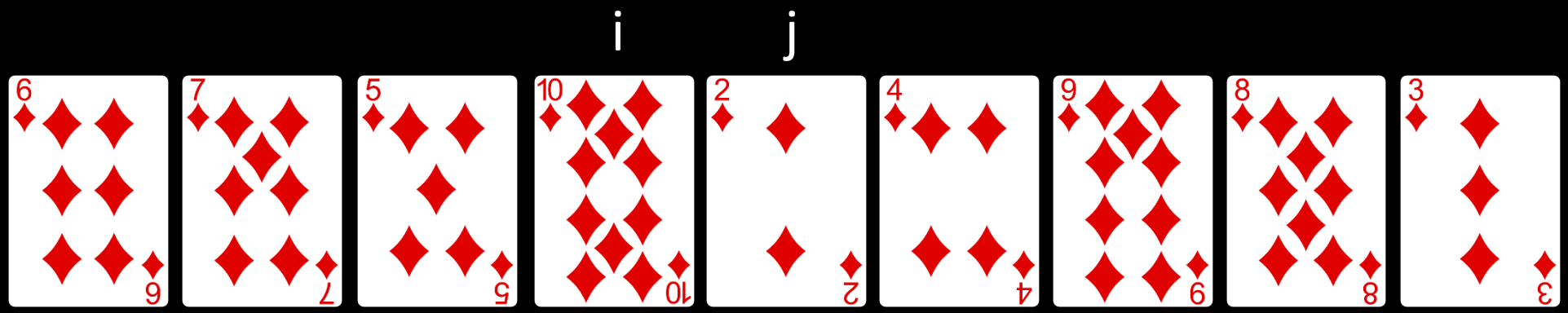
Bubble Sort Visualization



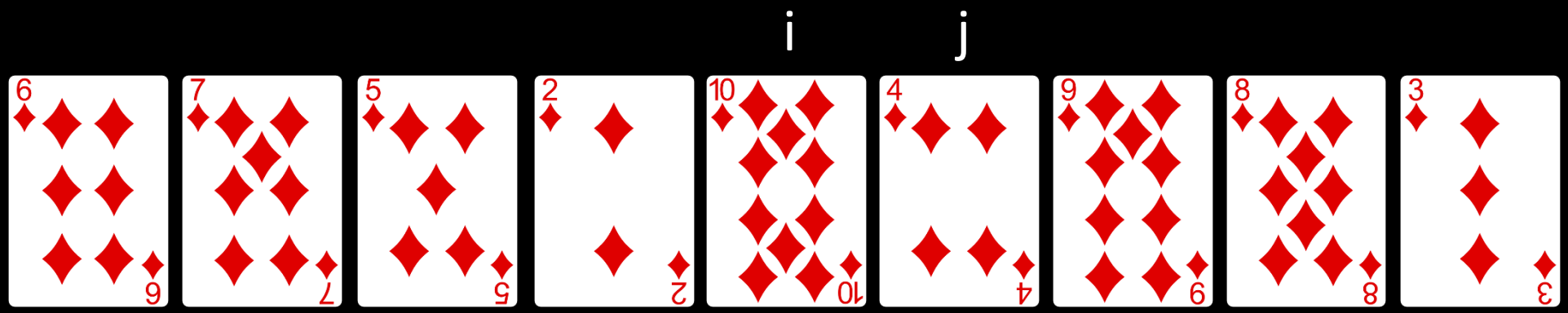
Bubble Sort Visualization



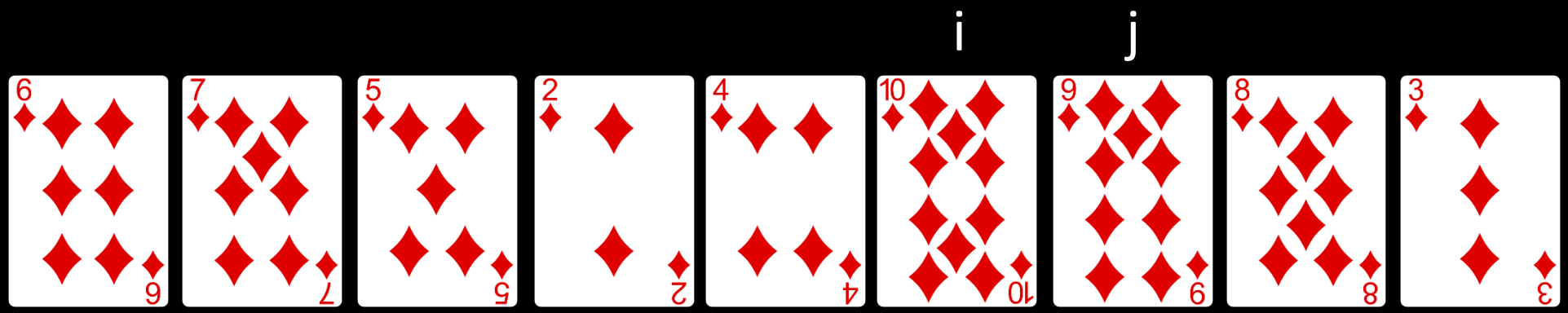
Bubble Sort Visualization



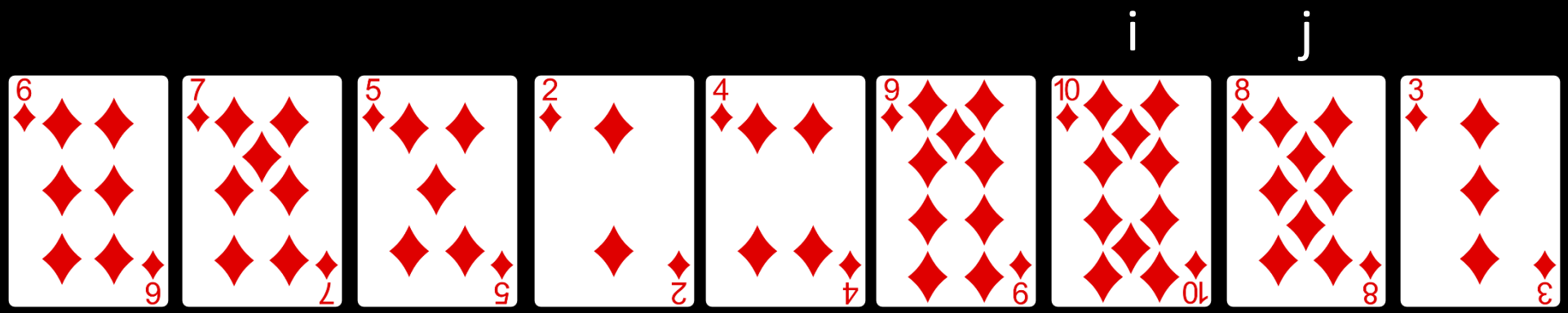
Bubble Sort Visualization



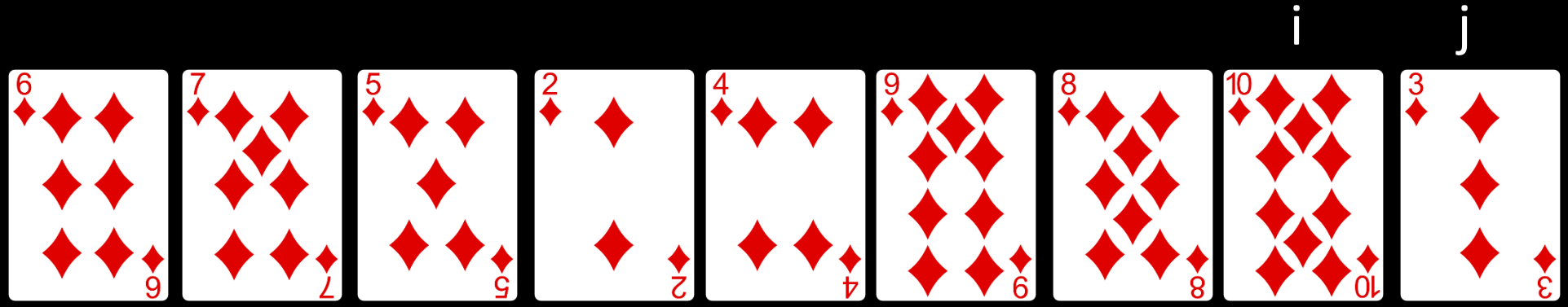
Bubble Sort Visualization



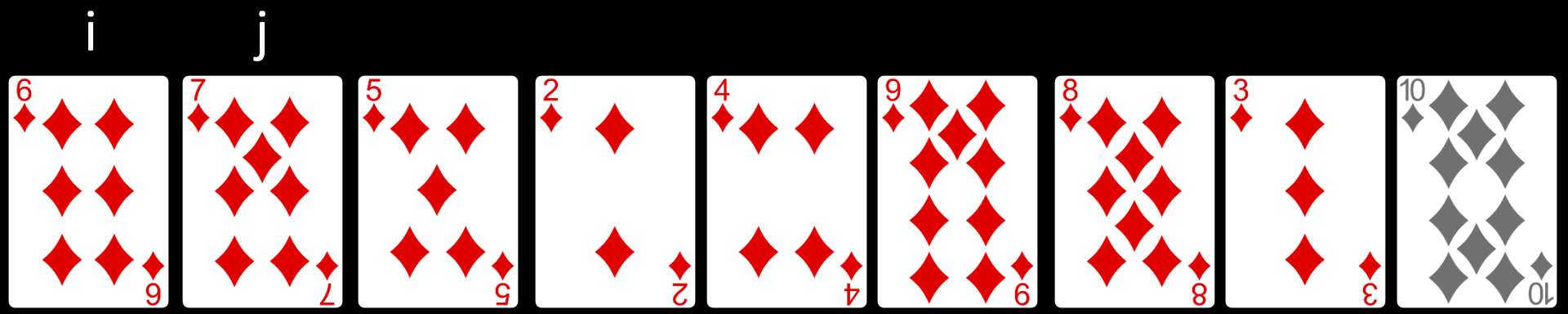
Bubble Sort Visualization



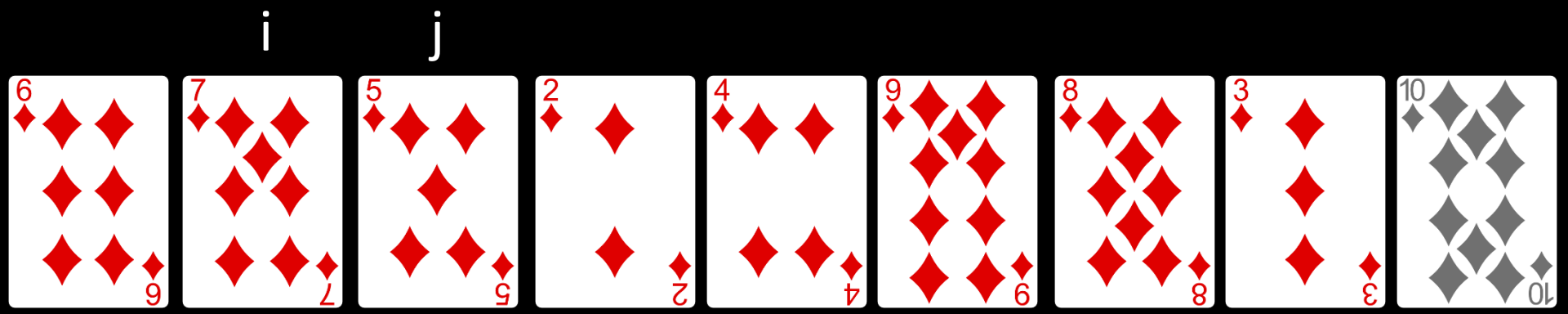
Bubble Sort Visualization



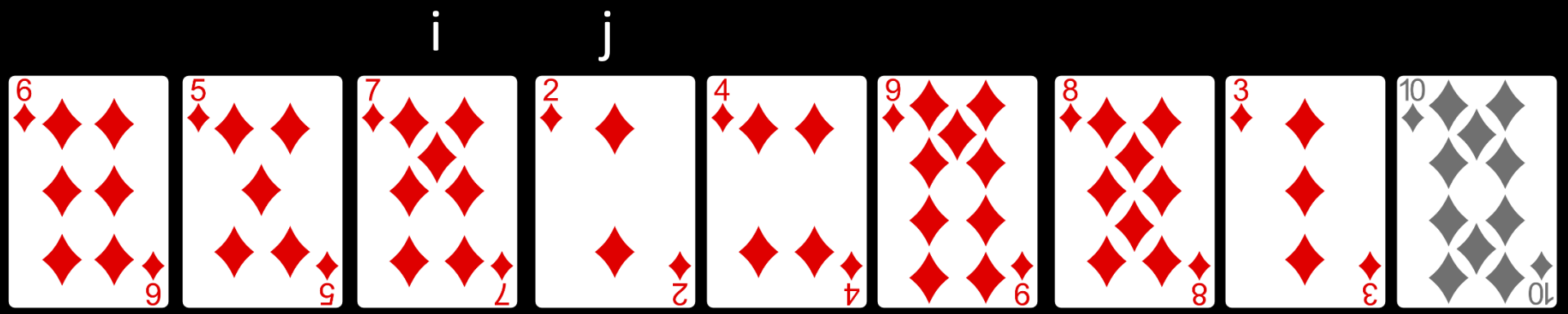
Bubble Sort Visualization



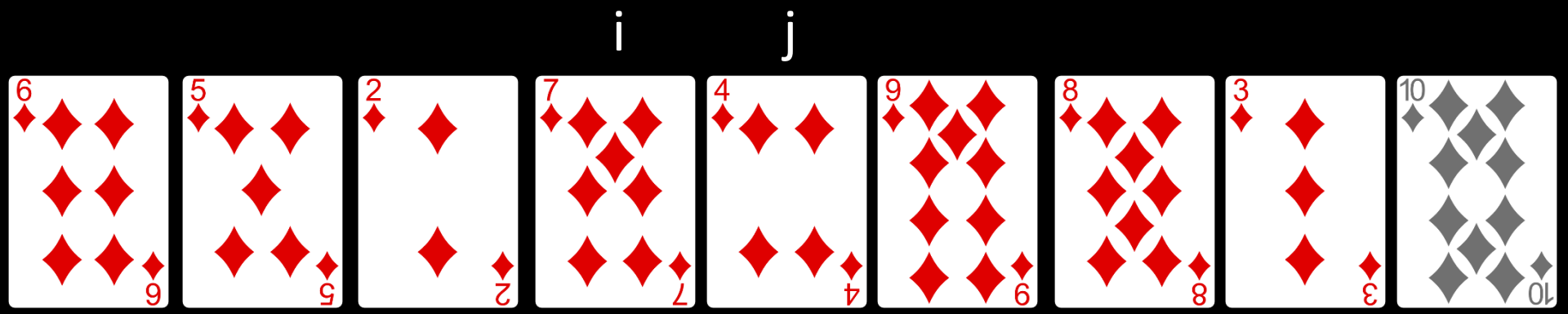
Bubble Sort Visualization



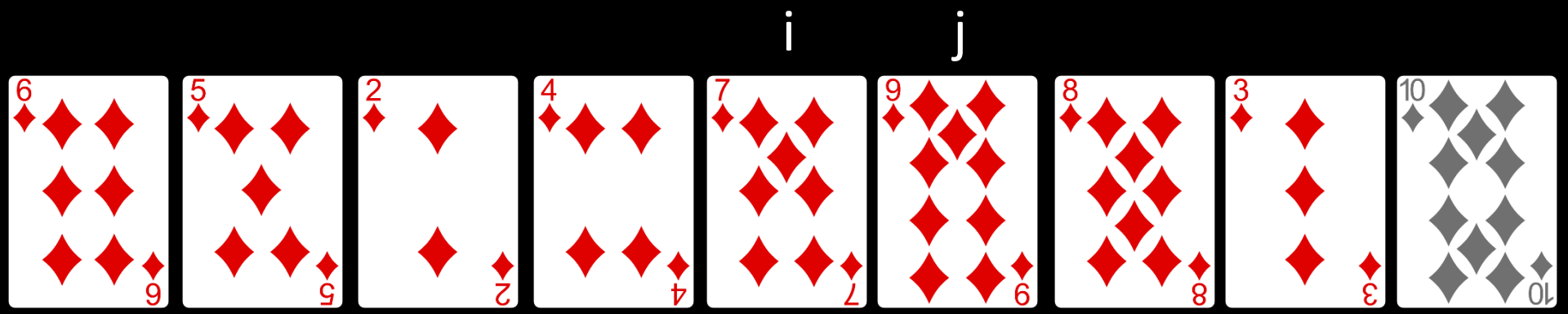
Bubble Sort Visualization



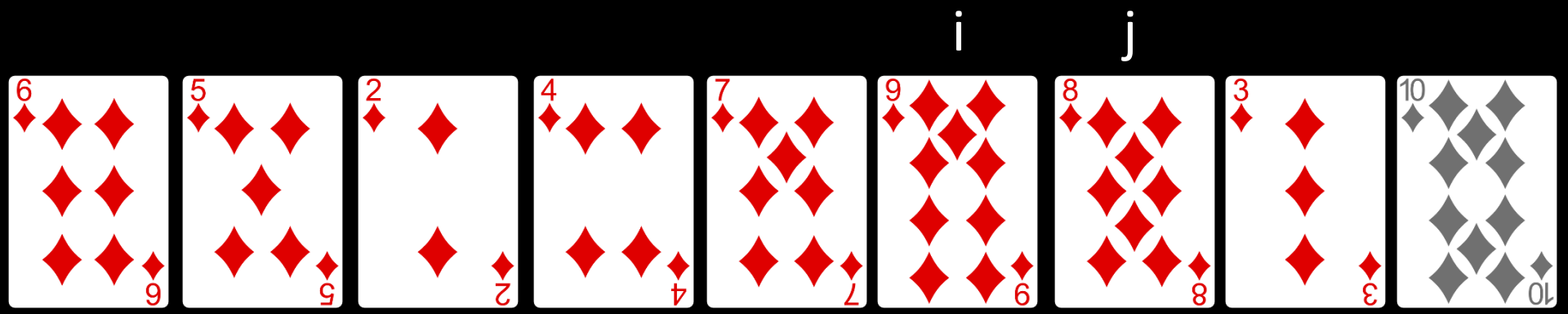
Bubble Sort Visualization



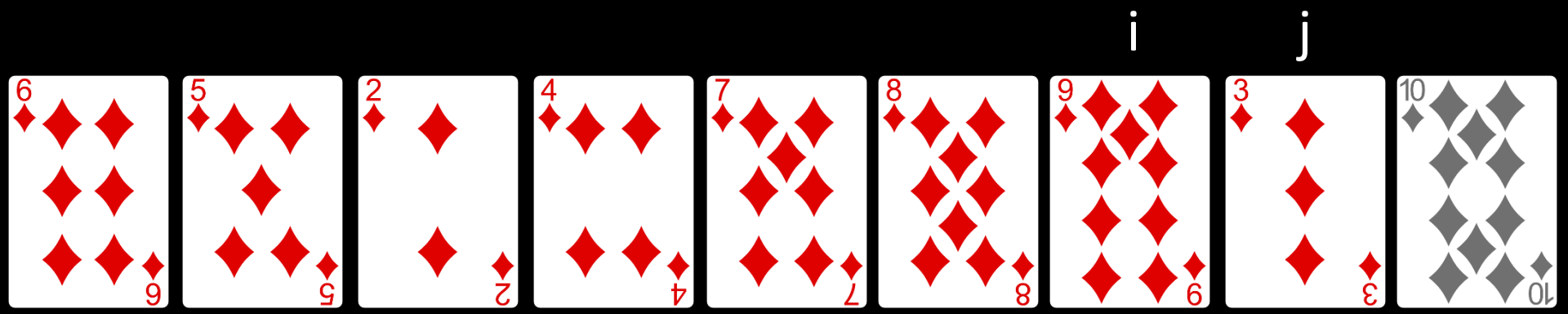
Bubble Sort Visualization



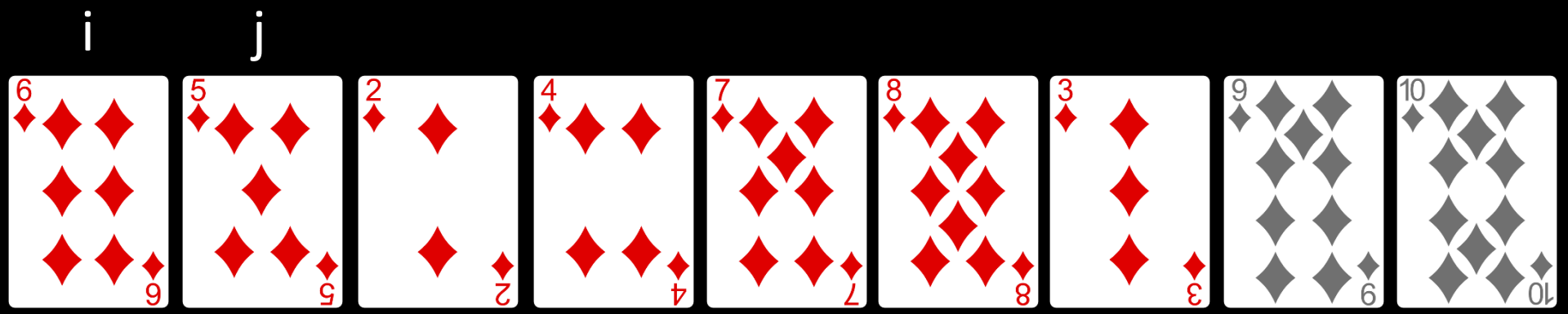
Bubble Sort Visualization



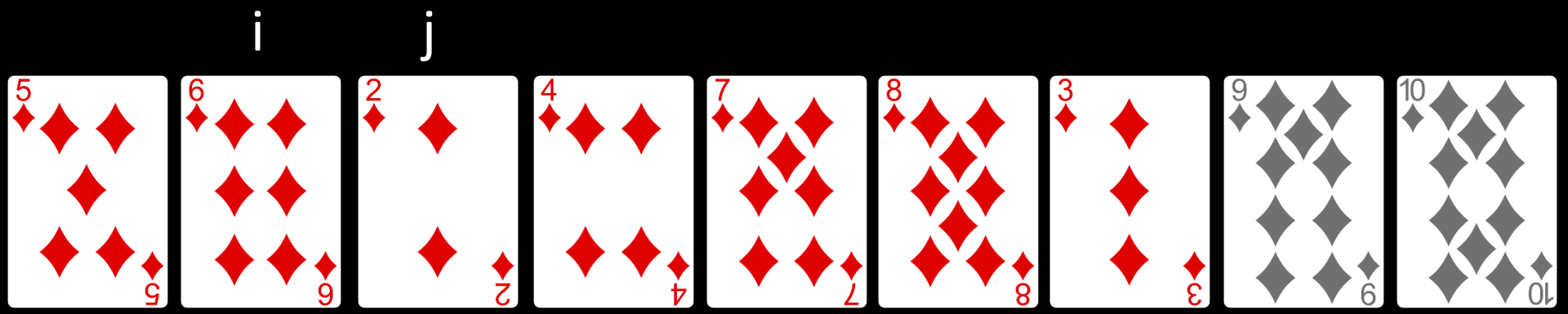
Bubble Sort Visualization



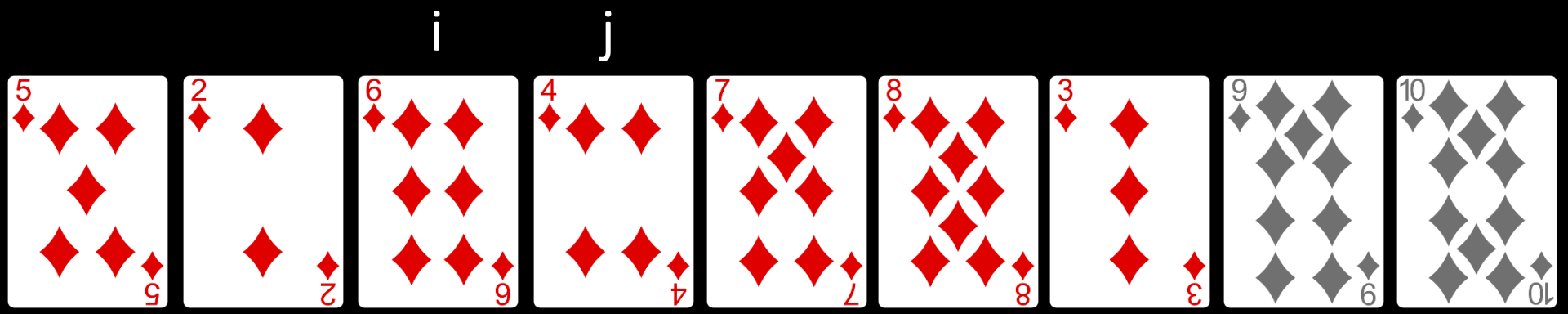
Bubble Sort Visualization



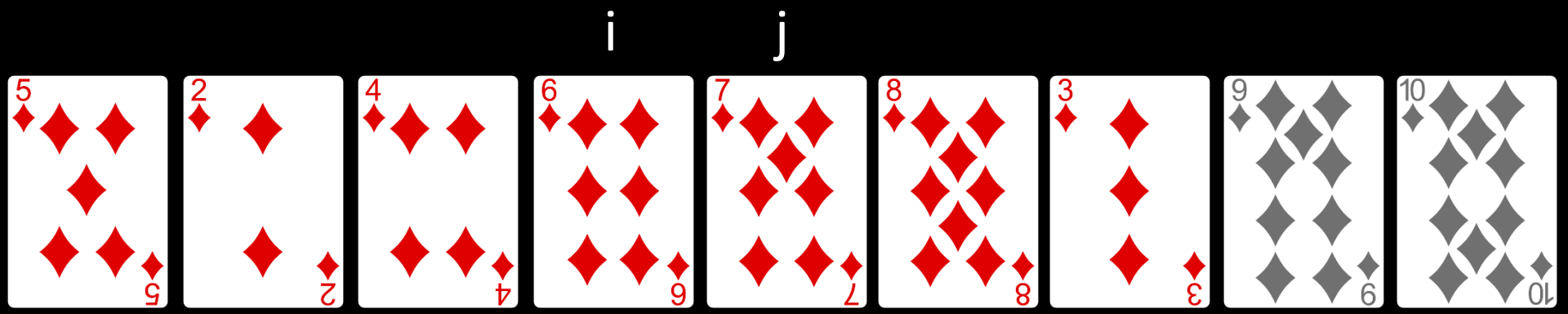
Bubble Sort Visualization



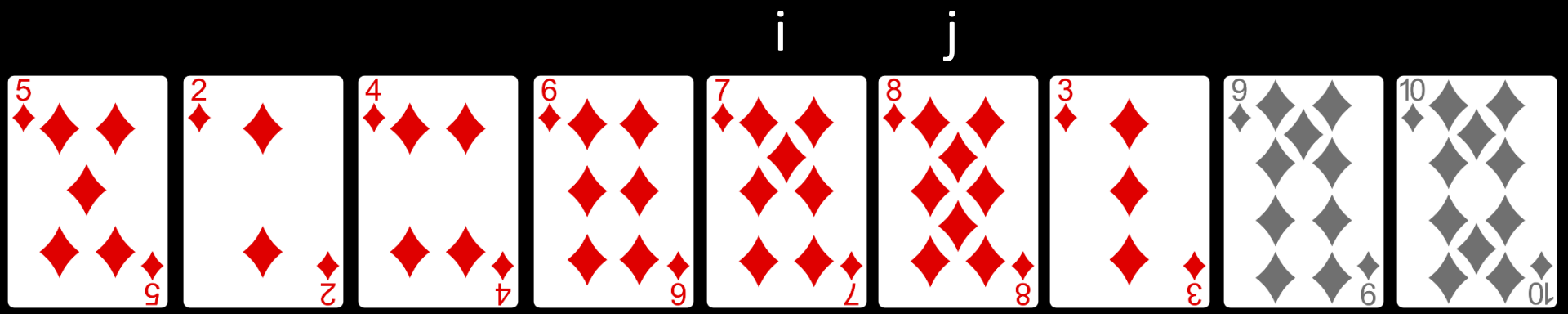
Bubble Sort Visualization



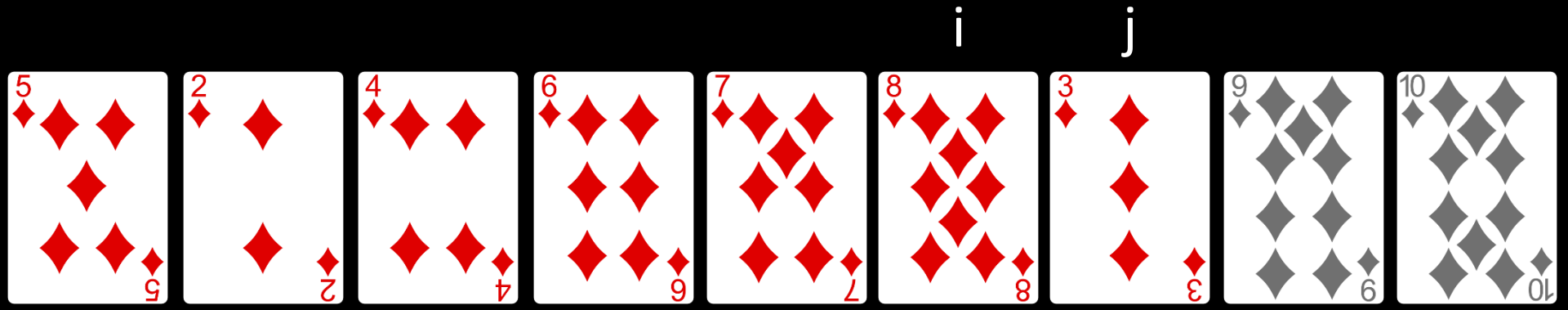
Bubble Sort Visualization



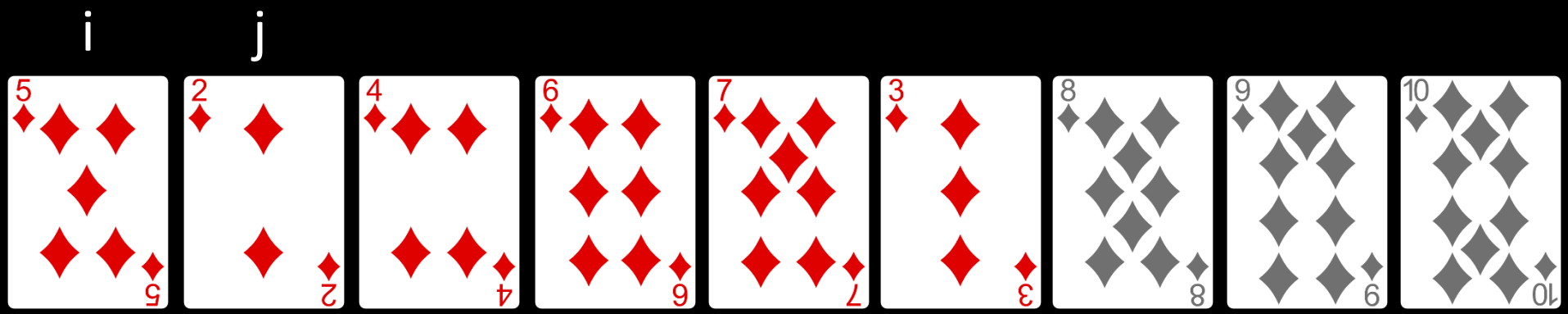
Bubble Sort Visualization



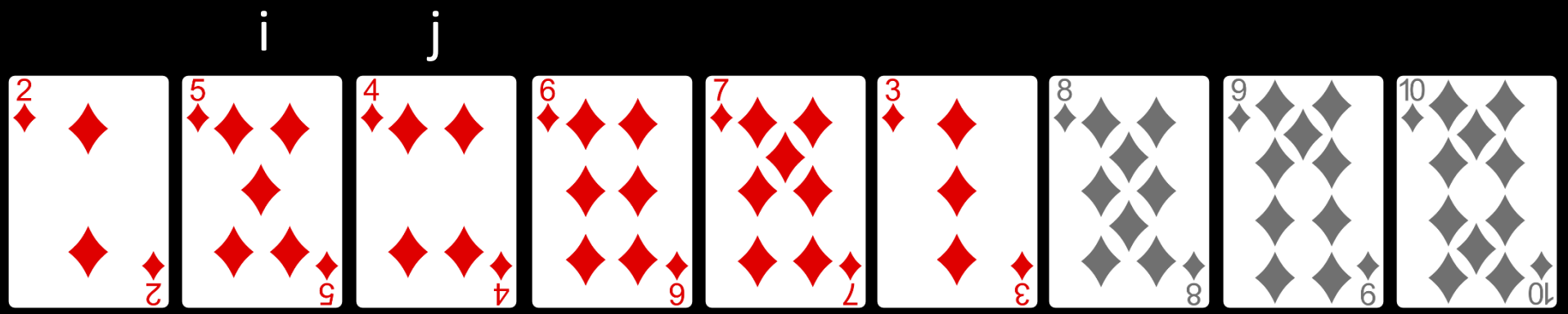
Bubble Sort Visualization



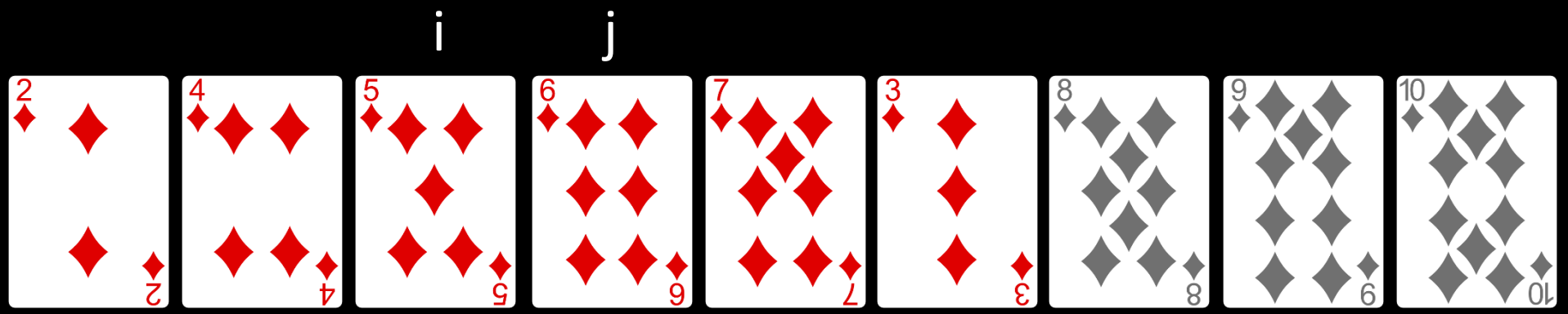
Bubble Sort Visualization



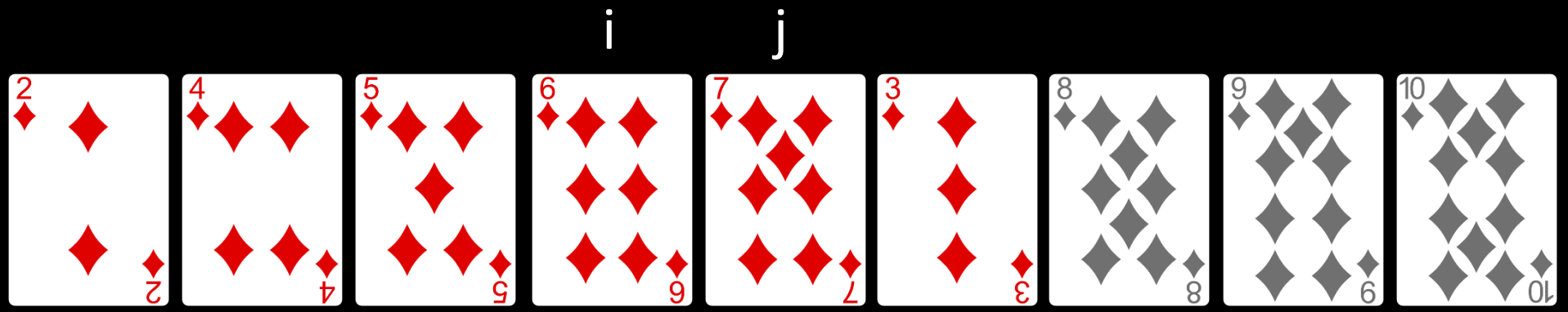
Bubble Sort Visualization



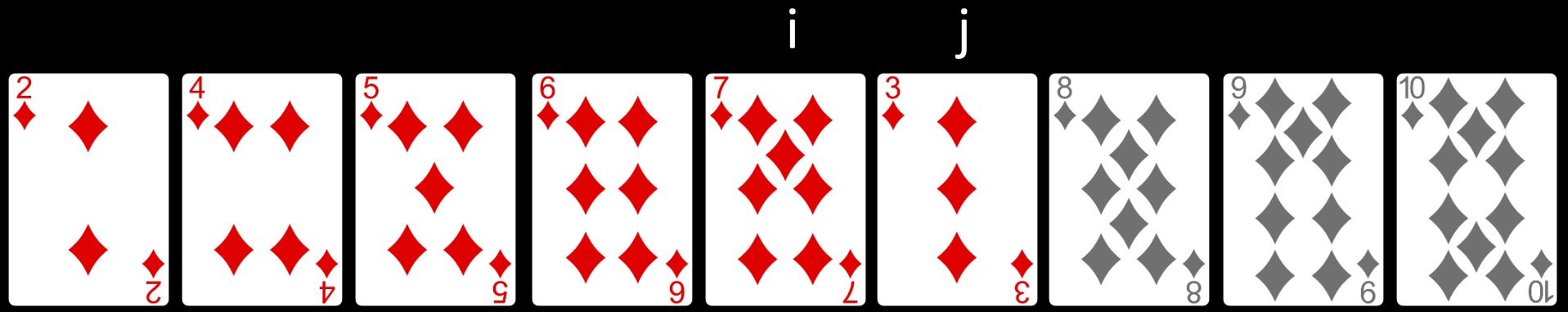
Bubble Sort Visualization



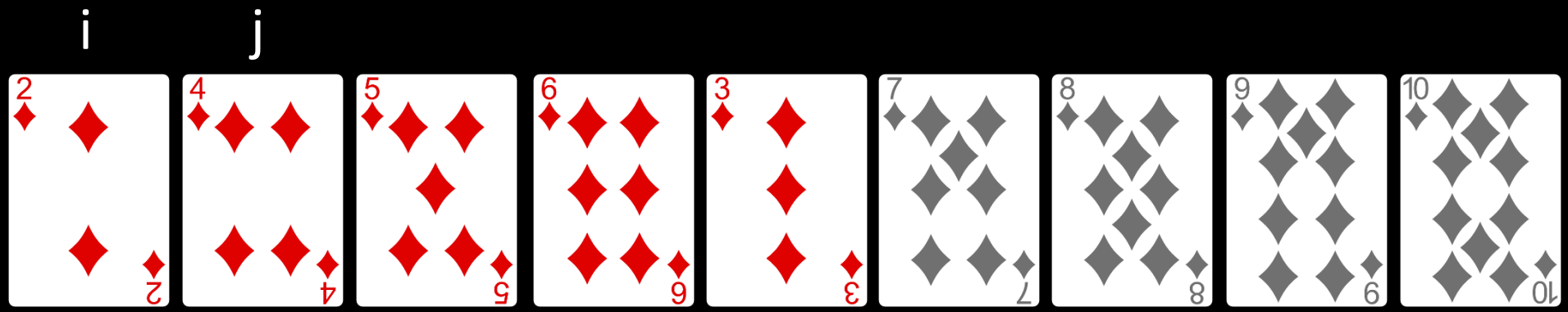
Bubble Sort Visualization



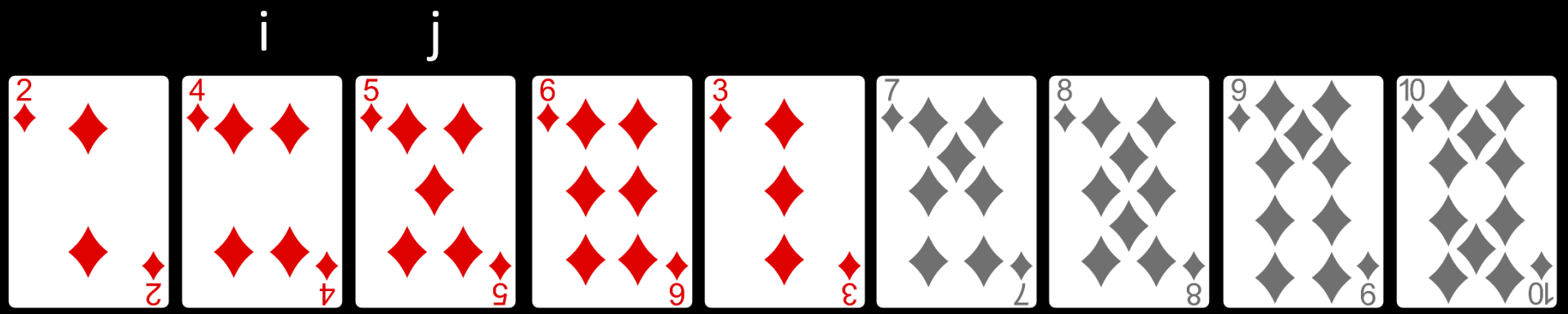
Bubble Sort Visualization



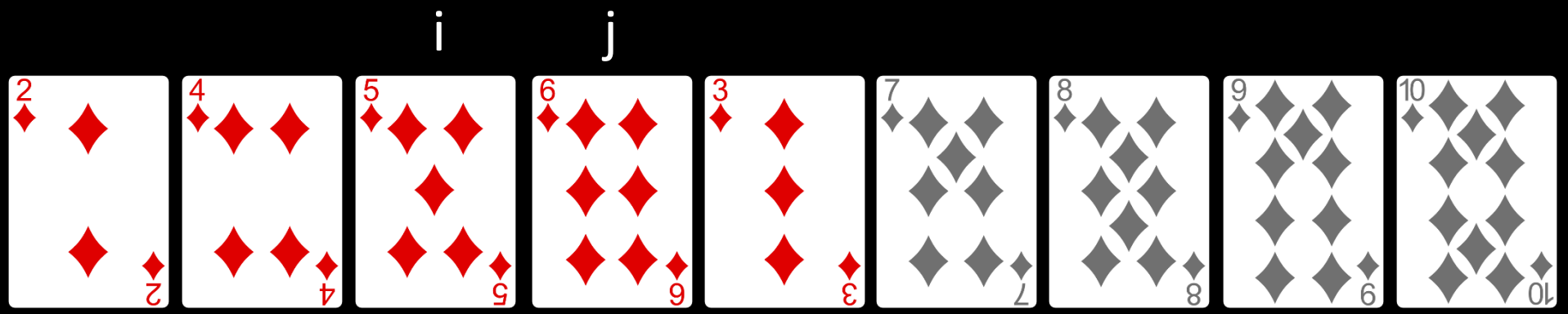
Bubble Sort Visualization



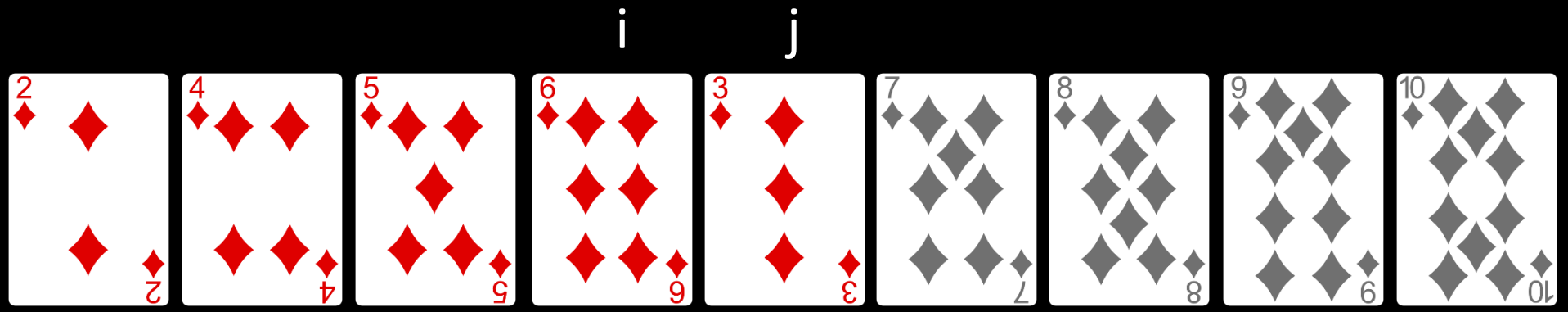
Bubble Sort Visualization



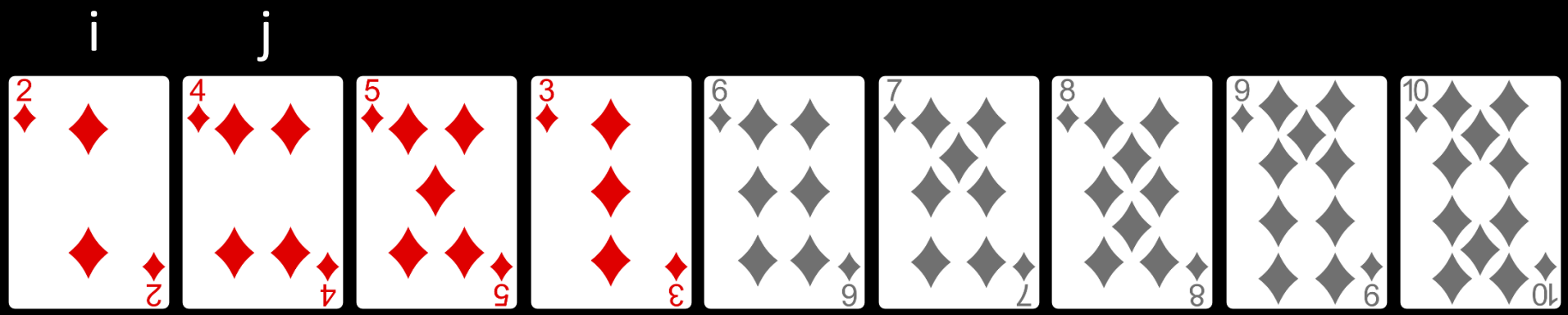
Bubble Sort Visualization



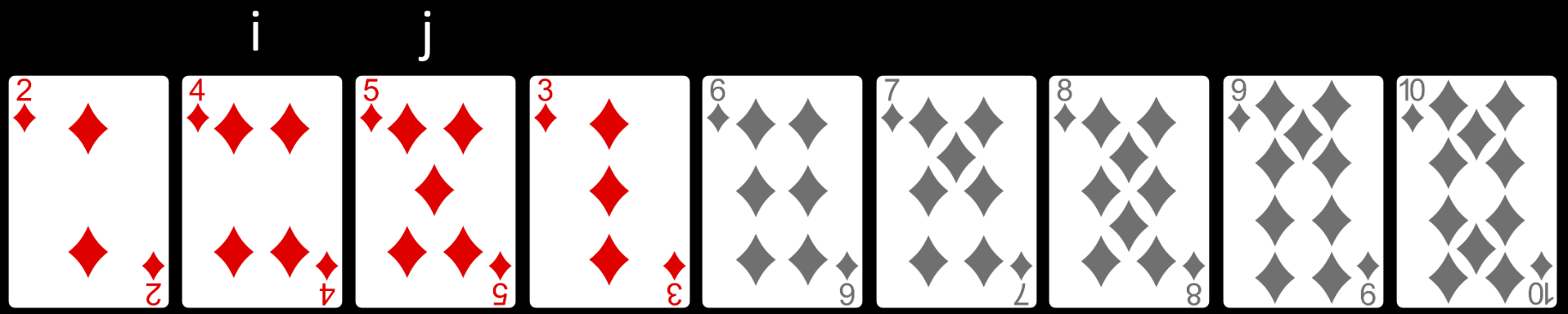
Bubble Sort Visualization



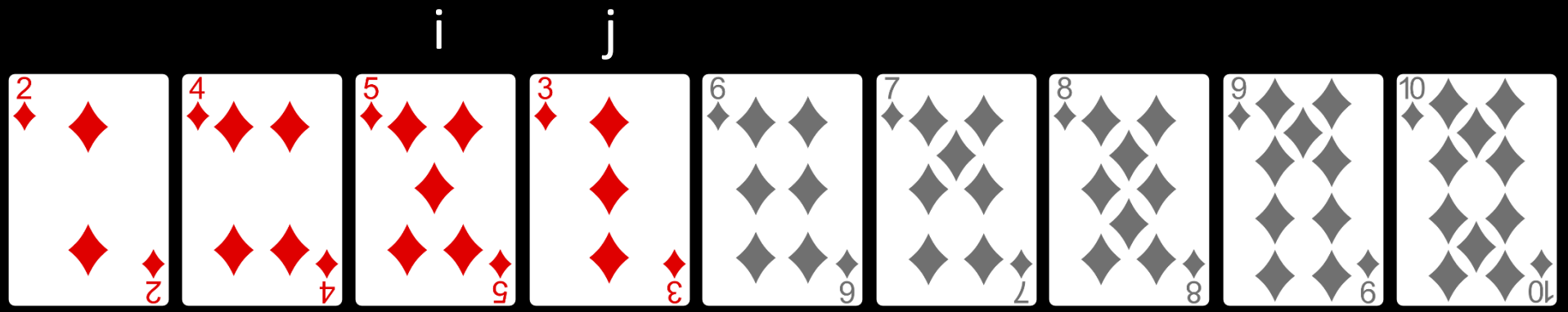
Bubble Sort Visualization



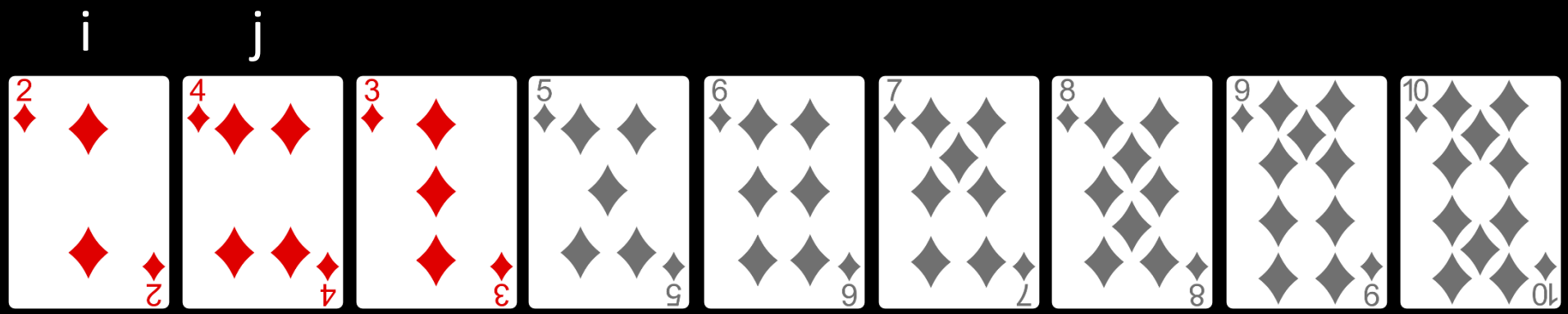
Bubble Sort Visualization



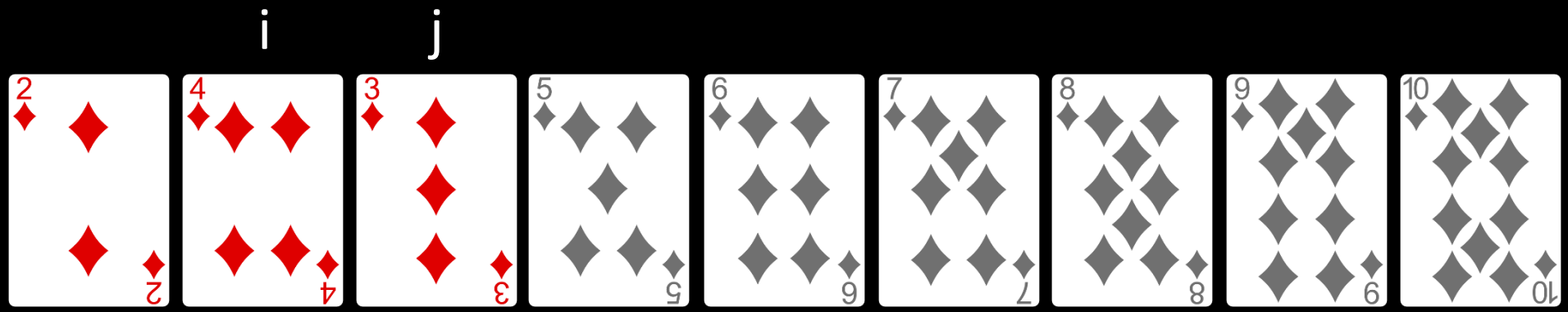
Bubble Sort Visualization



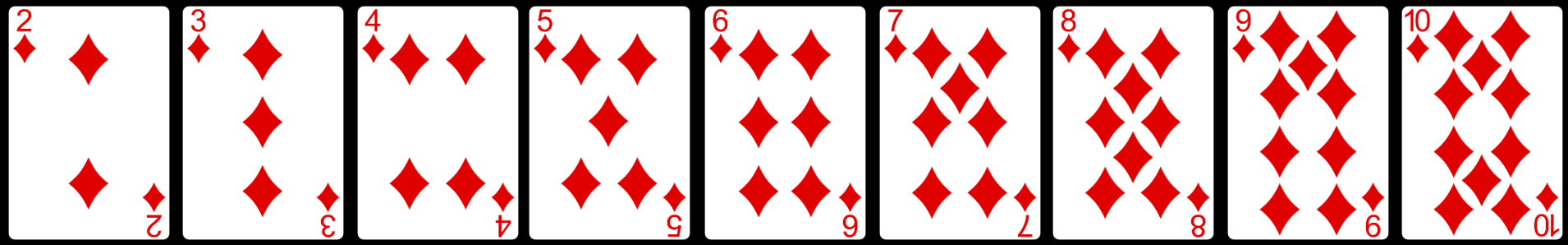
Bubble Sort Visualization



Bubble Sort Visualization



Bubble Sort Visualization



Comparison of Sorting Algorithms

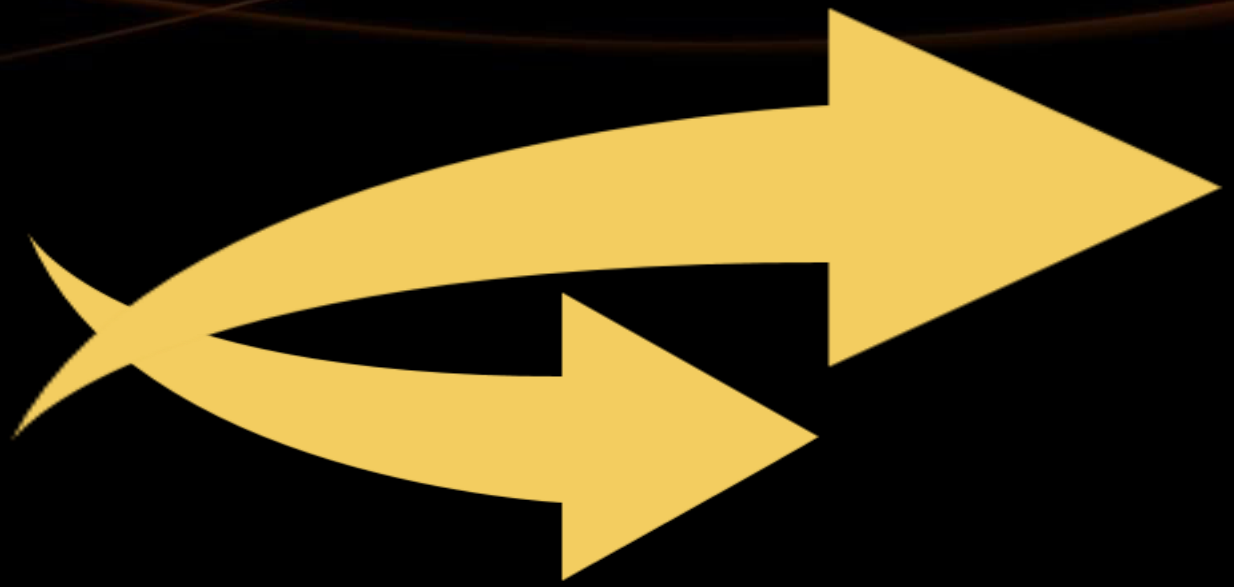
Name	Best	Average	Worst	Memory	Stable	Method
SelectionSort	n^2	n^2	n^2	1	No	Selection
BubbleSort	n	n^2	n^2	1	Yes	Exchanging

Insertion Sort

- Insertion sort – simple, but inefficient algorithm (visualize)
  - Move the first unsorted element left to its place
  - Memory: O(1)
  - Stable: Yes
  - Method: Insertion

Comparison of Sorting Algorithms

Name	Best	Average	Worst	Memory	Stable	Method
SelectionSort	n^2	n^2	n^2	1	No	Selection
BubbleSort	n	n^2	n^2	1	Yes	Exchanging
InsertionSort	n	n^2	n^2	1	Yes	Insertion



Shuffling

Fisher-Yates Shuffle

Shuffling

- Shuffling == randomizing the order of items in a collection
  - Generate a random permutation
- *The generation of random numbers is too important to be left to chance.* —Robert R. Coveyou

Fisher–Yates Shuffle Algorithm

```
public static void Shuffle<T>(T[] source)
{
  Random rnd = new Random();

  for (int i = 0; i < source.Length; i++)
  {
    // Exchange array[i] with random element in array[i … n-1]

    int r = i + rnd.Next(0, source.Length - i);

    T temp = source[i];
    source[i] = source[r];
    source[r] = temp;
  }
}
```

Shuffle algorithms: visualization

SOFTWARE UNIVERSITY FOUNDATION

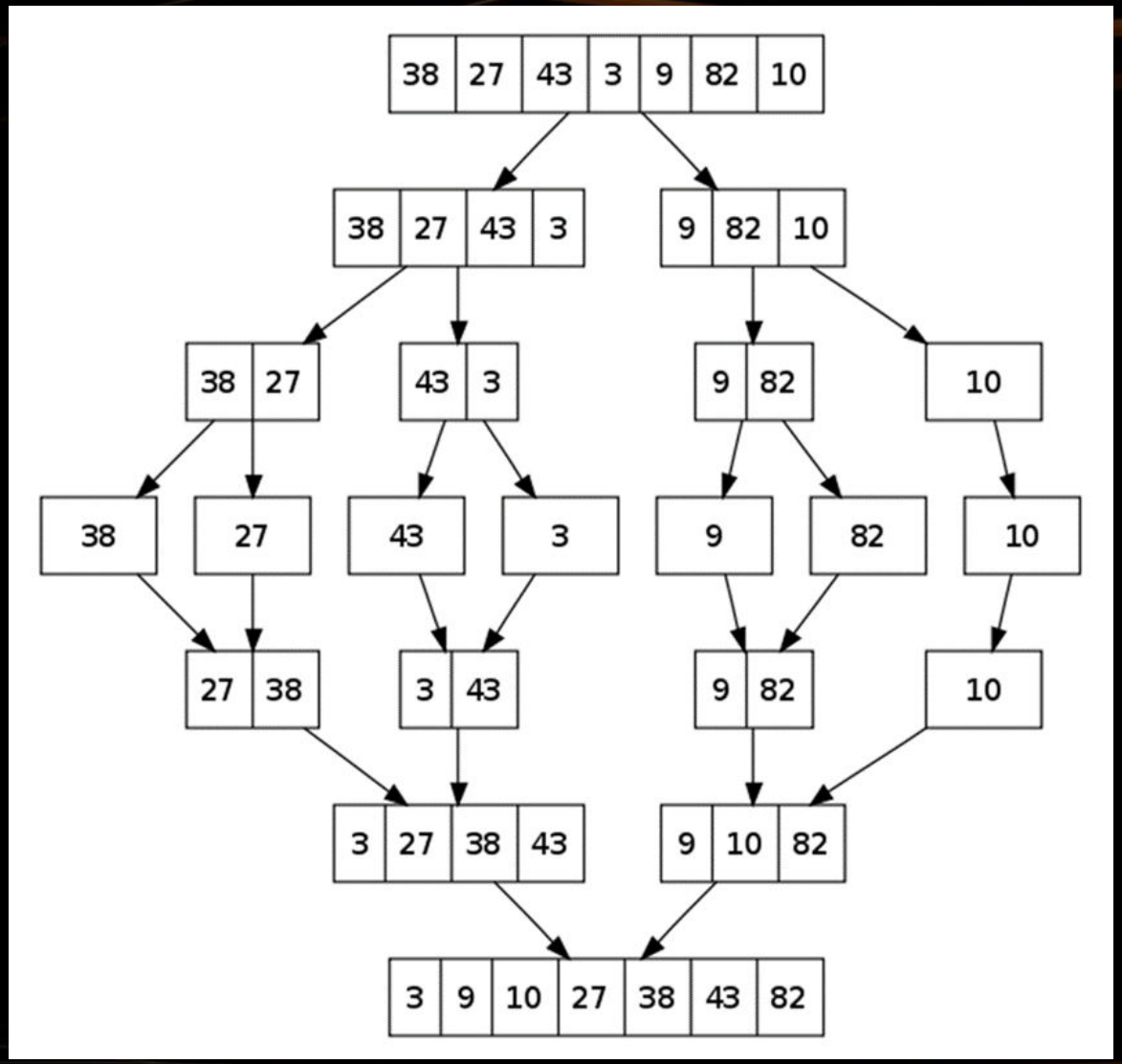
Advanced Sorting Algorithms

QuickSort, MergeSort, BucketSort

Merge Sort

- Merge sort is efficient sorting algorithm (visualize)
  1. Divide the list into sub-lists (typically 2 sub-lists)
  2. Sort each sub-list (recursively call merge-sort)
  3. Merge the sorted sub-lists into a single list
- Best, average and worst case: O(n*log(n))
- Memory:
  - Typically O(n)
  - With in-place merge can be O(1)
- Stable: Yes
- Method: Merging
- Highly parallelizable on multiple cores / machines → up to O(log(n))

Merge Sort: Hot It Works?

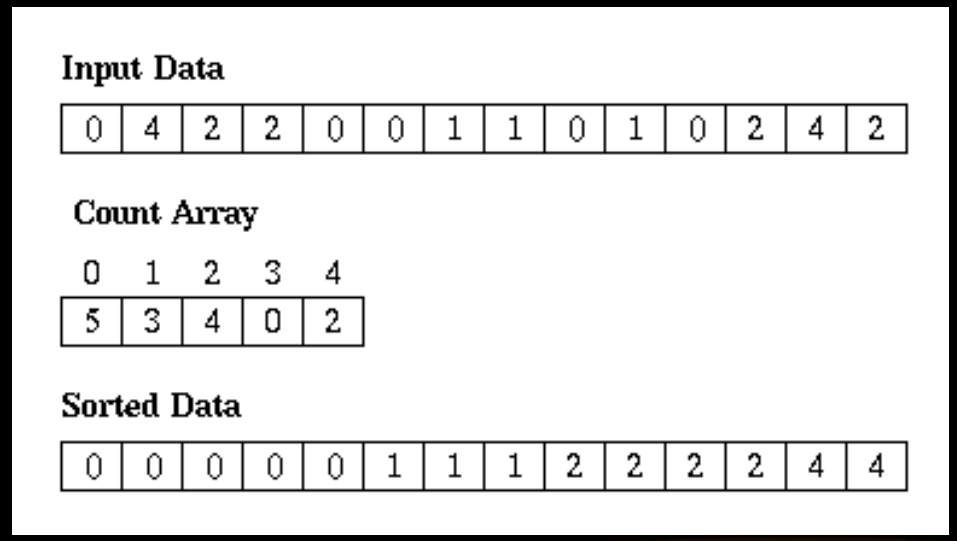


QuickSort

- QuickSort – efficient sorting algorithm (visualize)
  - Choose a pivot; move smaller elements left & larger right; sort left & right
  - Best & average case: O(n*log(n)); Worst: $O(n^2)$
  - Memory: O(log(n)) stack space (for recursion)
  - Stable: Depends
  - Method: Partitioning

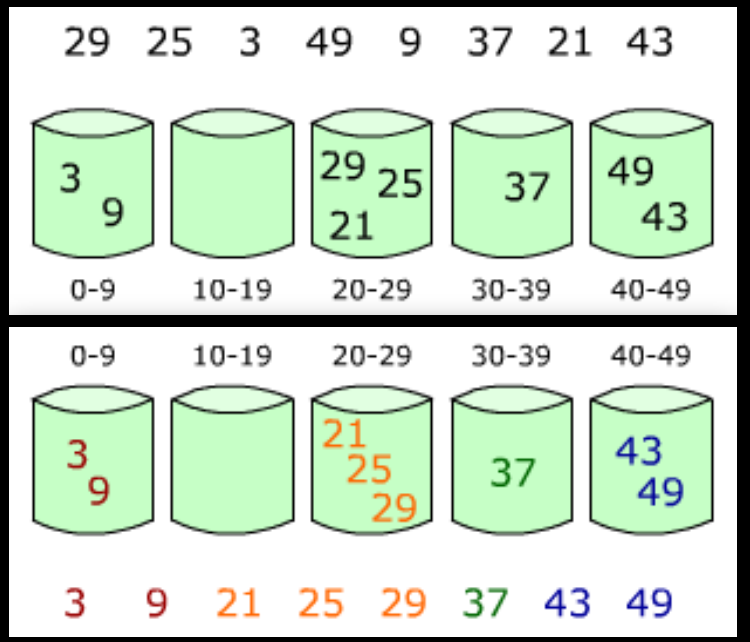
Counting Sort

- Counting sort is a very efficient sorting algorithm (visualize)
  - Sorts small integers by counting their occurrences
  - Not a comparison-based sort
- Best, average and worst case: O(n + k)
  - k is the range of numbers to be sorted
  - E.g. [-1000 ... 1000] → k = 2001
- Memory: O(n + k) Space: O(k)
- Stable: Yes
- Method: Counting



Bucket Sort

- Bucket sort partitions an array into a number of buckets
  - Each bucket is then sorted individually with a different algorithm
  - Not a comparison-based sort
- Average case: O(n + k)
  - k = the number of buckets
- Worst case: O(n * log n)
- Stable: Yes (depends on algorithm)
- Memory: O(n) – k buckets holding n elements totally



Comparison of Sorting Algorithms

Name	Best	Average	Worst	Memory	Stable	Method
SelectionSort	n^2	n^2	n^2	1	No	Selection
BubbleSort	n	n^2	n^2	1	Yes	Exchanging
InsertionSort	n	n^2	n^2	1	Yes	Insertion
QuickSort	n*log(n)	n*log(n)	n^2	log(n)	Depends	Partitioning
MergeSort	n*log(n)	n*log(n)	n*log(n)	1 (or n)	Yes	Merging
HeapSort	n*log(n)	n*log(n)	n*log(n)	1	No	Selection
BogoSort	n	n*n!	n*n!	1	No	Luck

SOFTWARE UNIVERSITY FOUNDATION

Searching Algorithms

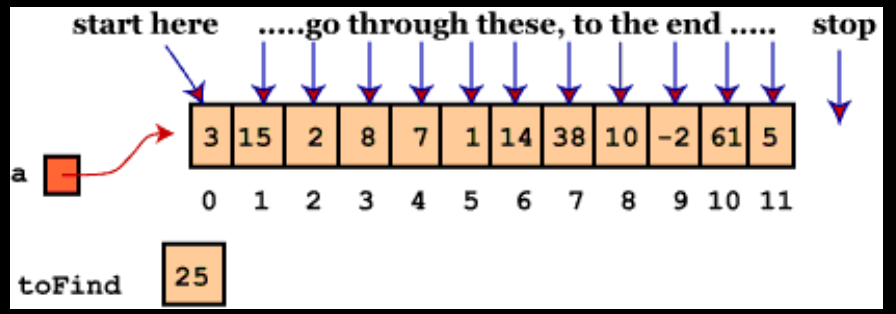
Linear, Binary and Interpolation

Search Algorithm

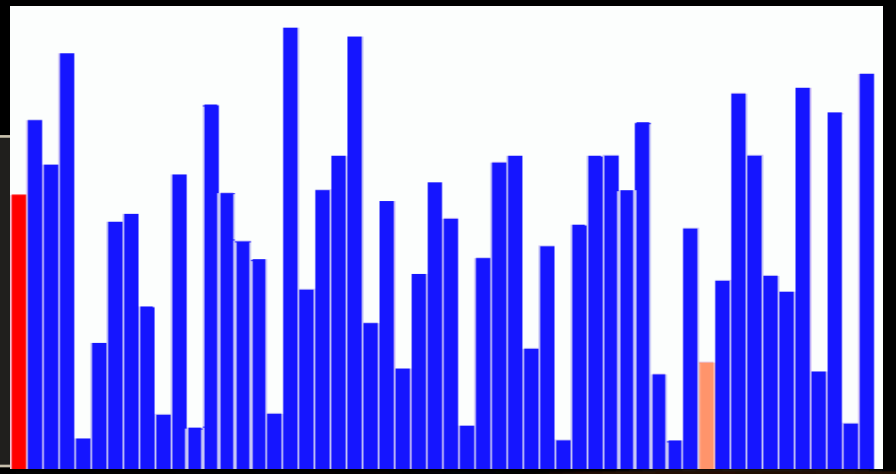
- Search algorithm == an algorithm for finding an item with specified properties among a collection of items
- Different types of searching algorithms
  - For virtual search spaces
    - Satisfy specific mathematical equations
    - Try to exploit partial knowledge about a structure
  - For sub-structures of a given structure
    - A graph, a string, a finite group
  - Search for the min / max of a function, etc.

Linear Search

- Linear search finds a particular value in a list (visualize)
  - Checking every one of the elements
  - One at a time, in sequence
  - Until the desired one is found
- Worst & average performance: O(n)

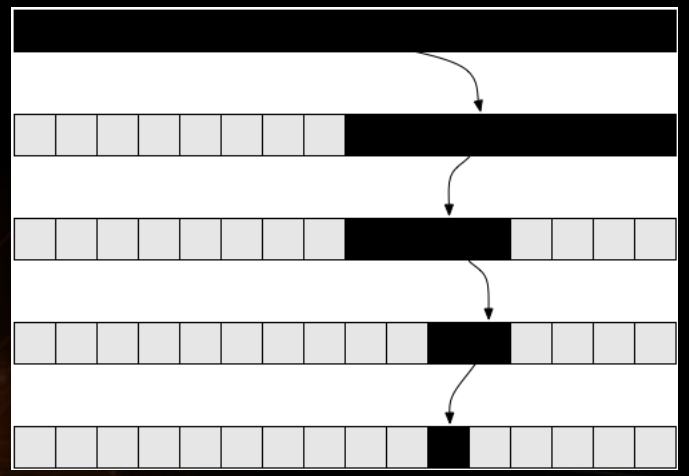


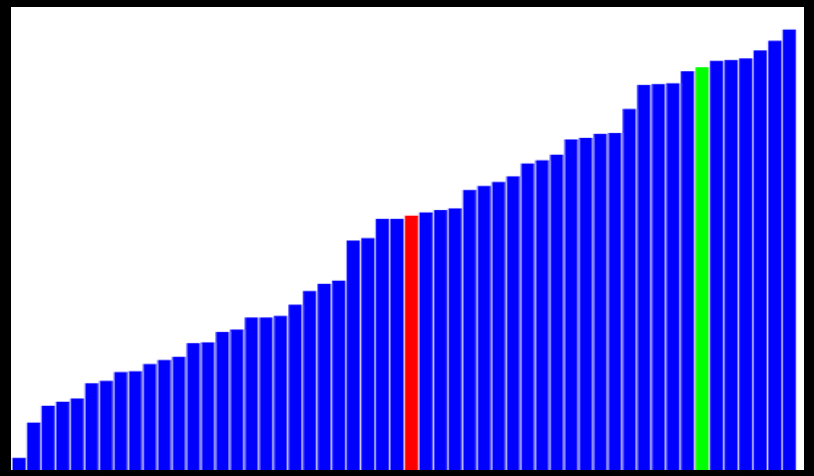

```
for each item in the list:
  if that item has the desired value,
    return the item's location
return nothing
```

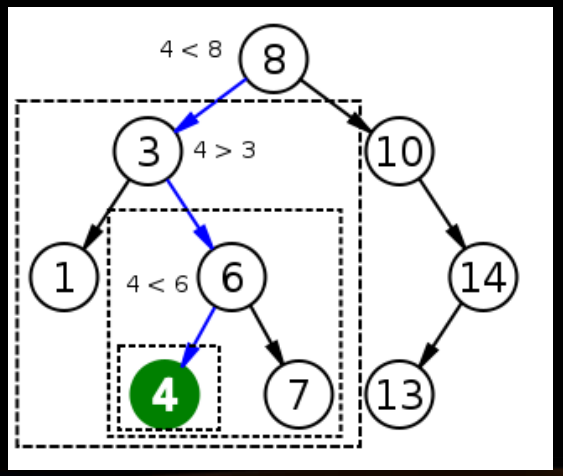


Binary Search

- Binary search finds an item within a ordered data structure
- At each step, compare the input with the middle element
  - The algorithm repeats its action to the left or right sub-structure
- Average performance: O(log(n))
- See the visualization







Binary Search (Recursive)

```
int BinarySearch(int arr[], int key, int start, int end) {
  if (end < start)
    return KEY_NOT_FOUND;
  else {
    int mid = (start + end) / 2;
    if (arr[mid] > key)
      return BinarySearch(arr, key, start, mid - 1);
    else if (arr[mid] < key)
      return BinarySearch(arr, key, mid + 1, end);
    else
      return mid;
  }
}
```

Binary Search (Iterative)

```
int BinarySearch(int arr[], int key, int start, int end) {
  while (end >= start) {
    int mid = (start + end) / 2;
    if (arr[mid] < key)
      start = mid + 1;
    else if (arr[mid] > key)
      end = mid - 1;
    else
      return mid;
  }
  return KEY_NOT_FOUND;
}
```

Interpolation Search

- Interpolation search == an algorithm for searching for a given key in an ordered indexed array
  - Parallels how humans search through a telephone book
  - Calculates where in the remaining search space the item might be
    - Binary search always chooses the middle element
    - Can we have a better hit, e.g. *Angel* in the phonebook should be at the start, not at the middle, OK?
- Average case: log(log(n)), Worst case: O(n)
- http://youtube.com/watch?v=l1ed_bTv7Hw

Interpolation Search – Sample Implementation


```
int InterpolationSearch(int[] sortedArray, int key) {
  int low = 0;
  int high = sortedArray.Length - 1;
  while (sortedArray[low] <= key && sortedArray[high] >= key) {
    int mid = low + ((key - sortedArray[low]) * (high - low))
      / (sortedArray[high] - sortedArray[low]);
    if (sortedArray[mid] < key)
      low = mid + 1;
    else if (sortedArray[mid] > key)
      high = mid - 1;
    else
      return mid;
  }
  if (sortedArray[low] == key) return low;
  else return KEY_NOT_FOUND;
}
```

Summary

- Slow sorting algorithms:
  - Selection sort, Bubble sort, Insertion sort
- Fast sorting algorithms:
  - Quick sort, Merge sort, etc.
  - How to choose the most appropriate algorithm?
    - http://stackoverflow.com/a/1934004
- Searching algorithms
  - Binary Search, Interpolation Search

Sorting and Searching Algorithms

Questions?

https://softuni.bg/opencourses/algorithms

License

- This course (slides, examples, labs, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Fundamentals of Computer Programming with C#" book by Svetlin Nakov & Co. under CC-BY-SA license
  - "Data Structures and Algorithms" course by Telerik Academy under CC-BY-NC-SA license

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - softuni.org
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

SOFTWARE UNIVERSITY

SOFTWARE UNIVERSITY FOUNDATION

FORUM