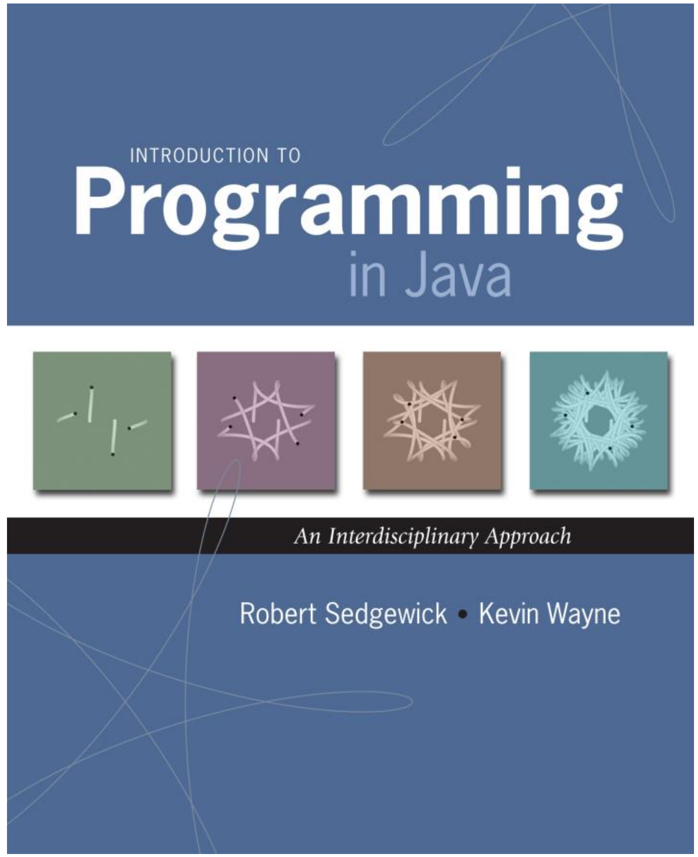
4.2 Sorting and Searching



Penn Engineering

Section 4.2

Searching

The problem:

Given a collection of data, determine if a query is contained in that collection.

This is a fundamentally important problem for a myriad of applications (from finding webpages to searching for fragments of DNA)

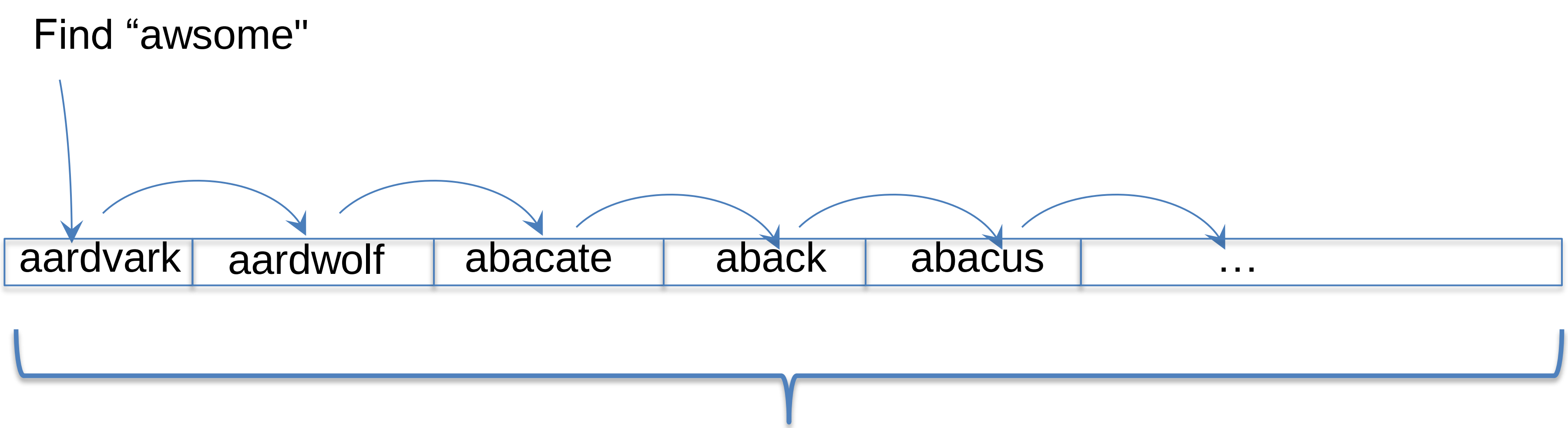
Penn Engineering

Motivating Example: Spellchecker!

Penn Engineering

Exhaustive (Linear) Search

– Systematically enumerate all possible values and compare to value being sought
– For an array, iterate from the beginning to the end, and test each item in the array



Array of all words in dictionary

Linear Search

Scan through array, looking for key.

- Search hit: return array index.
- Search miss: return -1.

```
public static int search(String key, String[] a) {
    int N = a.length;
    for (int i = 0; i < a.length; i++)
        if (a[i].compareTo(key) == 0)
            return i;
    return -1;
}
```

Penn Engineering

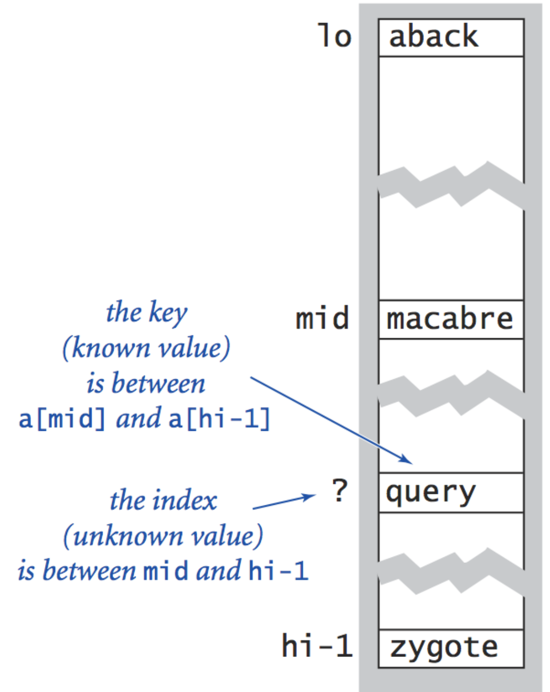
Binary Search

Quickly find an item (query) in a sorted list.

Examples: Dictionary, phone book, index, credit card numbers, ...

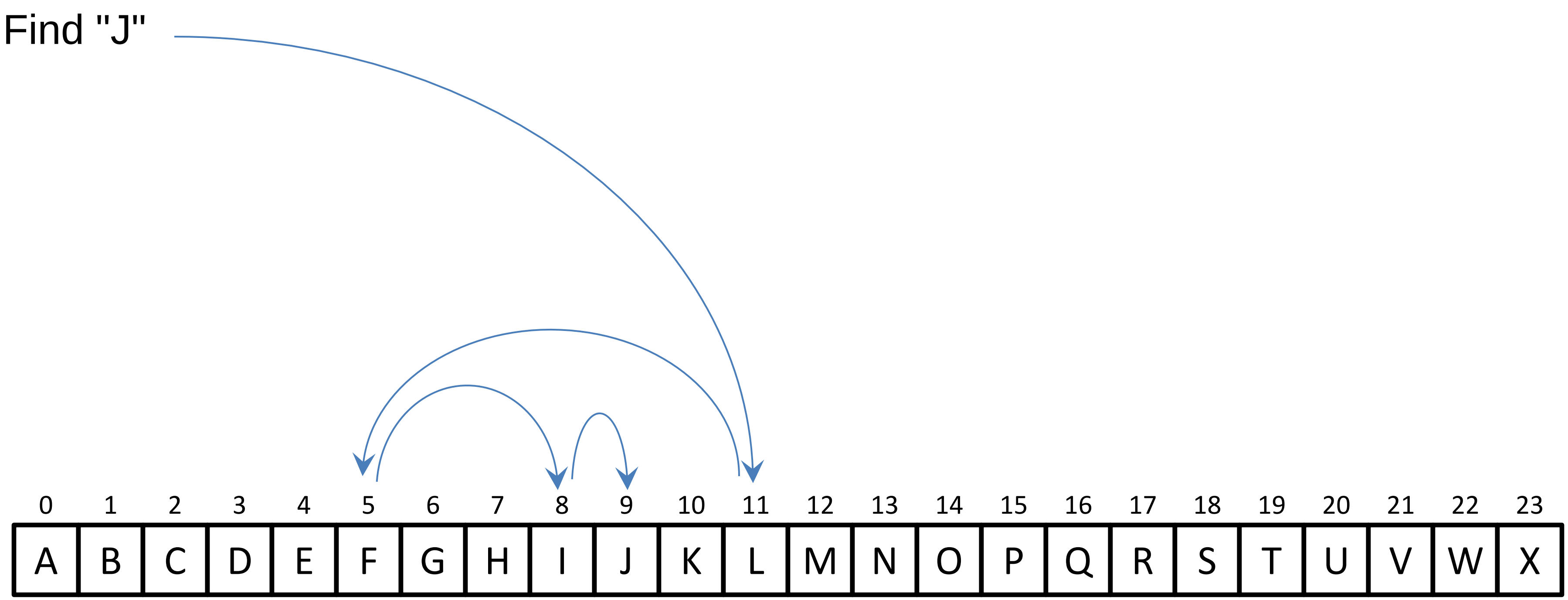
Binary Search:

- Examine the middle key
- If it matches, return its index
- Otherwise, search either the left or right half



Binary search in a sorted array (one step)

Binary Search



Binary Search Example

Find “awsome"

aardvark … macabre … Zyzzogeton

Binary Search Example

Find “awsome"

aardvark … macabre … Zyzzogeton

Binary Search Example

Find “awsome"

aardvark … fable … macabre

Binary Search Example

Find “awsome"

aardvark … fable … macabre

Binary Search Example

Find “awsome"

aardvark … catfish … fable

Binary Search Example

Find “awsome"

aardvark … catfish … fable

Binary Search Example

Find “awsome"

aardvark … beetle … catfish

Binary Search Example

Find “awsome"

aardvark … beetle … catfish

Binary Search Example

Find “awsome"

aardvark … awake … beetle

Binary Search Example

Find “awsome"

aardvark … awake $\boxed{\text{… beetle}}$

Binary Search Example

Find “awsome"

awaken … banjo … beetle

Binary Search Example

Find “awsome"

awaken … banjo … beetle

Binary Search Example

Find “awsome"

awaken … banjo … beetle

Repeat a few more times…

Binary Search Example

Find “awsome"

awry

Binary Search Example

Find “awsome"

awry

Return false

Binary Search

Invariant: Algorithm maintains **`a[lo] <= key < a[hi]`**

```
public static int search(String key, String[] a) {
   return search(key, a, 0, a.length);
}

public static int search(String key, String[] a, int lo, int hi) {
   if (hi <= lo) return -1;
   int mid = lo + (hi - lo) / 2;
   int cmp = a[mid].compareTo(key);
   if      (cmp > 0) return search(key, a, lo, mid);
   else if (cmp < 0) return search(key, a, mid+1, hi);
   else              return mid;
}
```

Penn Engineering

Binary Search

Analysis: Binary search in an array of size N

- One compare
- Binary search in array of size N/2

$$N \rightarrow N/2 \rightarrow N/4 \rightarrow N/8 \rightarrow \dots \rightarrow 1$$

Q. How many times can you divide by 2 until you reach 1?

A. $\log_2 N$

$$1$$
$$2 \rightarrow 1$$
$$4 \rightarrow 2 \rightarrow 1$$
$$8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

Penn Engineering

Spell-Checking Midsummer Night's Dream

- Exhaustive Search: **385,554 milliseconds** to check the entire text
- Binary Search: **104 milliseconds** to check the entire text
- Speedup: **3,707 times**!!!

Penn Engineering

Penn Engineering

Sitcom/Dictator Game

http://www.smalltime.com/Dictator

Sorting

- Sorting problem. Rearrange N items in ascending order.

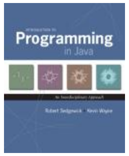
- Applications. Statistics, databases, data compression, bioinformatics, computer graphics, scientific computing, (too numerous to list), ...

Hauser	Hanley
Hong	Haskell
Hsu	Hauser
Hayes	Hayes
Haskell	Hong
Hanley	Hornet
Hornet	Hsu

Penn Engineering

Sorting

pentrust.org



Section 4.2

Sorting

pentrust.org

shanghaiscrap.org

Penn Engineering

Section 4.2

Sorting

pentrust.org

shanghaiscrap.org

De Beers

Section 4.2

Selection Sort

- Idea:
  - Find the smallest element in the array
  - Exchange it with the element in the first position
  - Find the second smallest element and exchange it with the element in the second position
  - Continue until the array is sorted

Example

8	4	6	9	2	3	1

25

Example

8	4	6	9	2	3	1

25

Example

8	4	6	9	2	3	1

25

Example

8	4	6	9	2	3	1

1	4	6	9	2	3	8

Example

8	4	6	9	2	3	1

1	4	6	9	2	3	8

25

Example

8	4	6	9	2	3	1

1	4	6	9	2	3	8

25

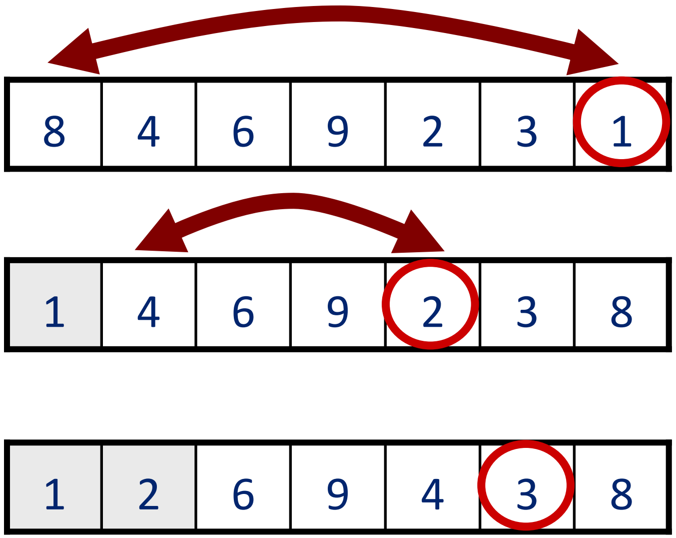
Example

8	4	6	9	2	3	1

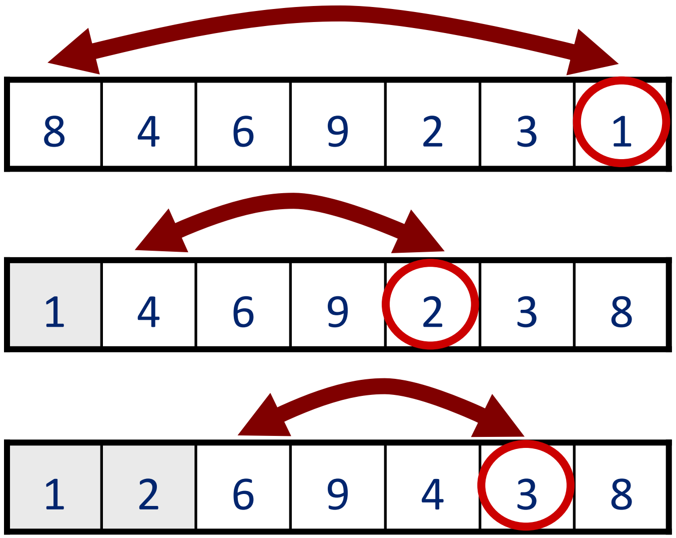
1	4	6	9	2	3	8

1	2	6	9	4	3	8

Example

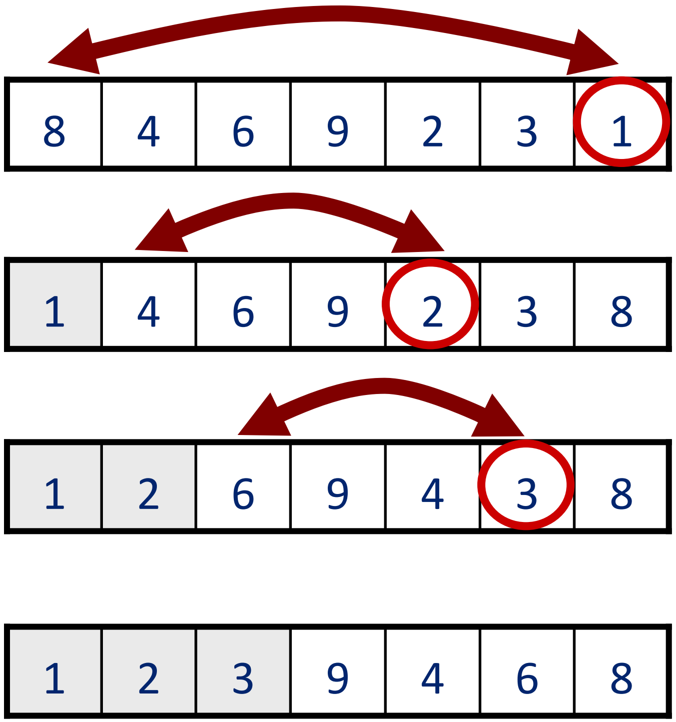


Example



25

Example



25

Example

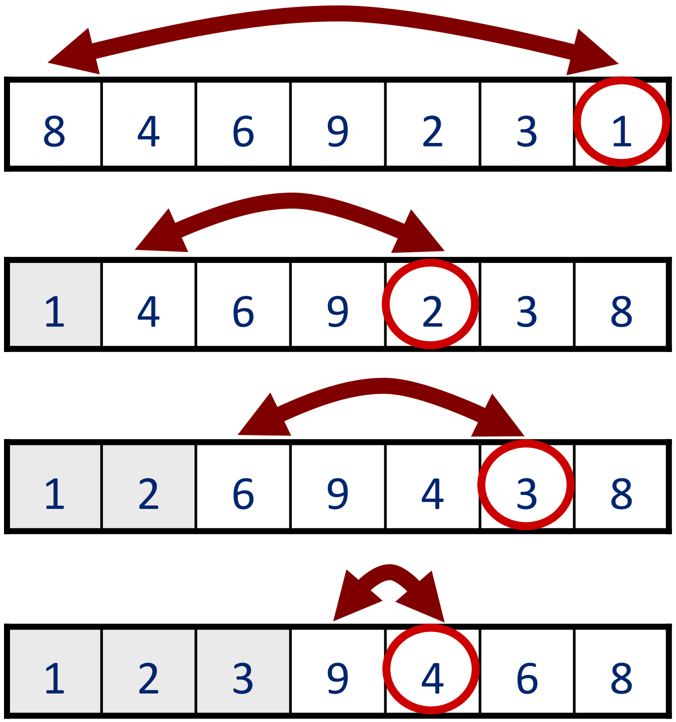
8	4	6	9	2	3	1

1	4	6	9	2	3	8

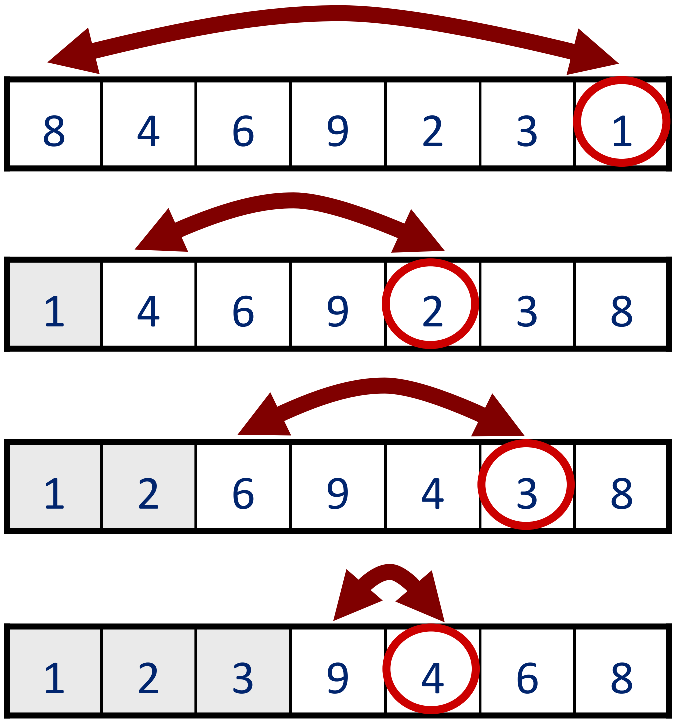
1	2	6	9	4	3	8

1	2	3	9	4	6	8

Example

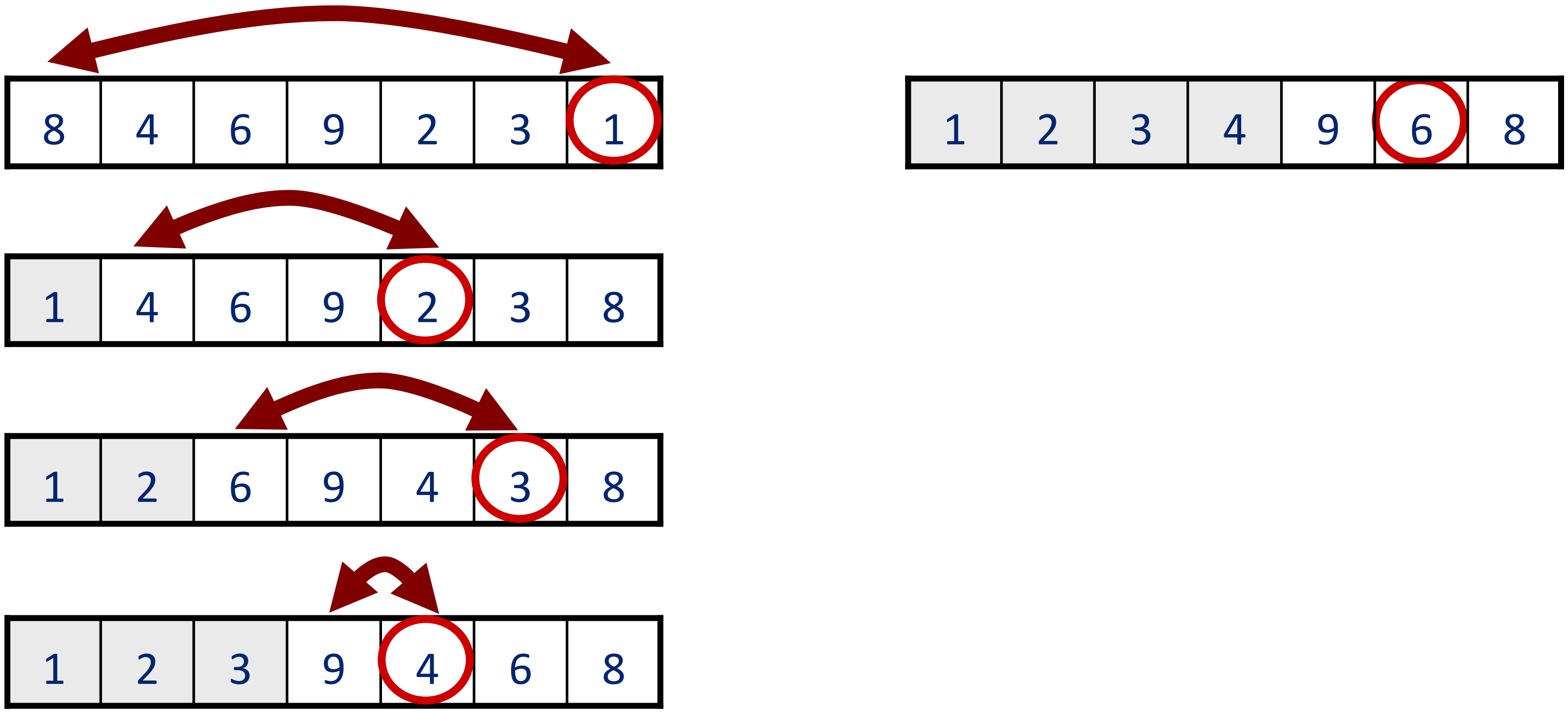


Example

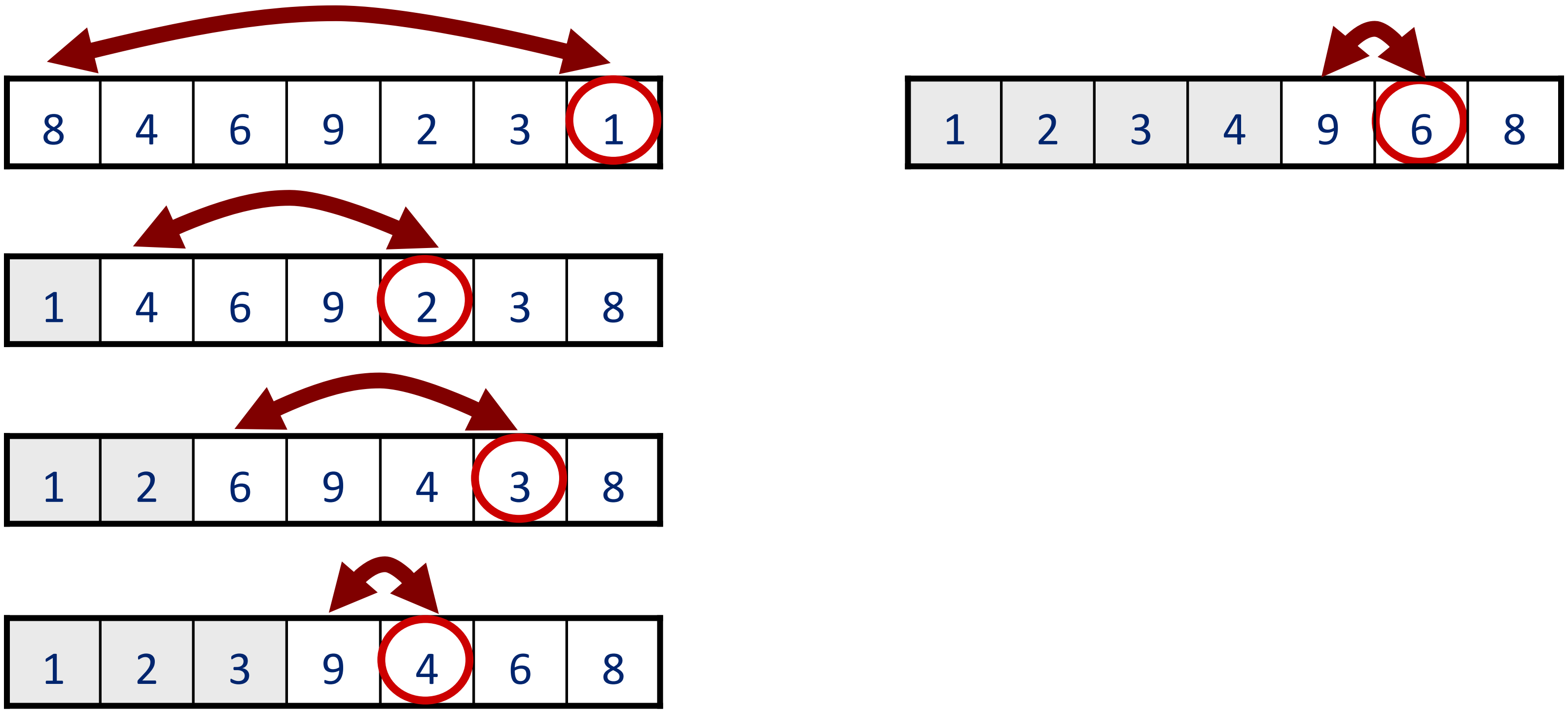


1	2	3	4	9	6	8

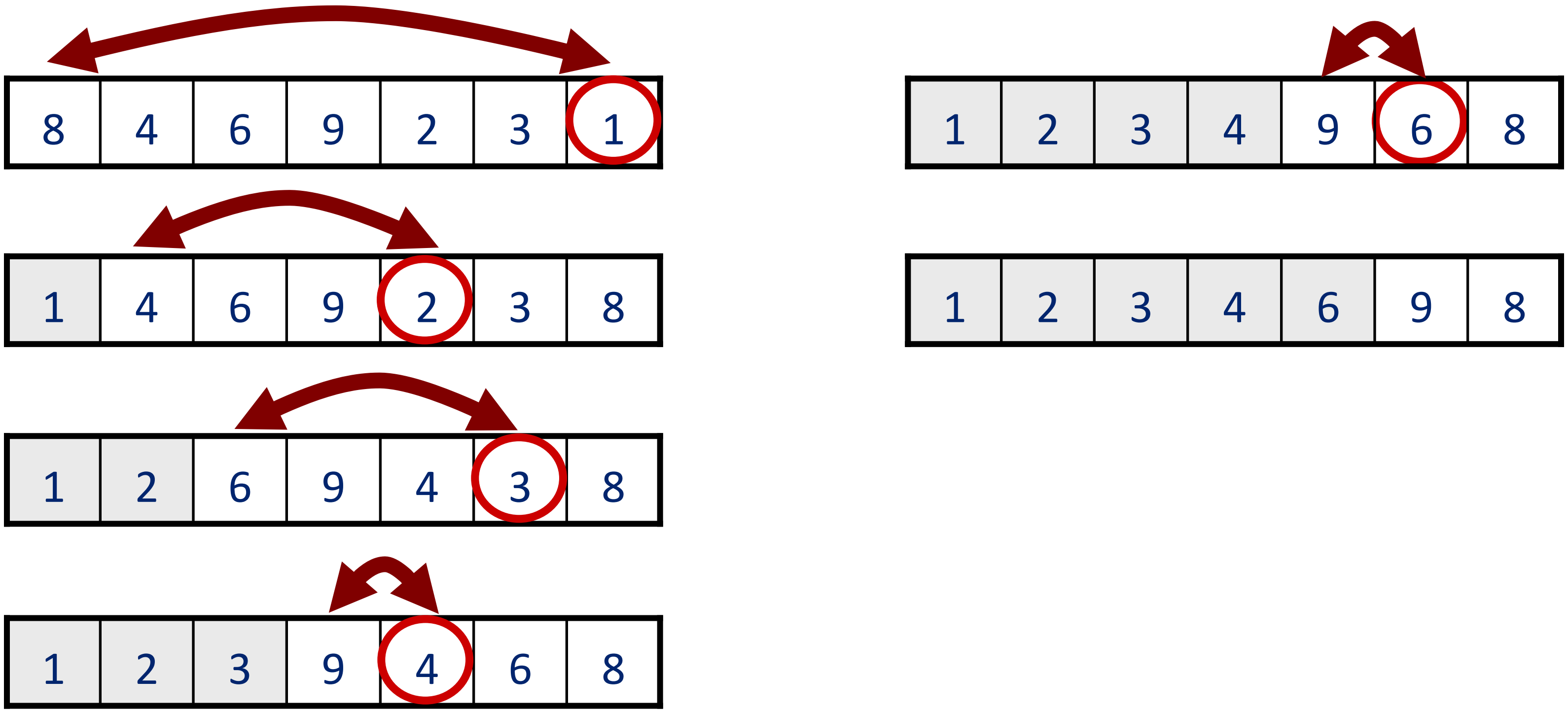
Example



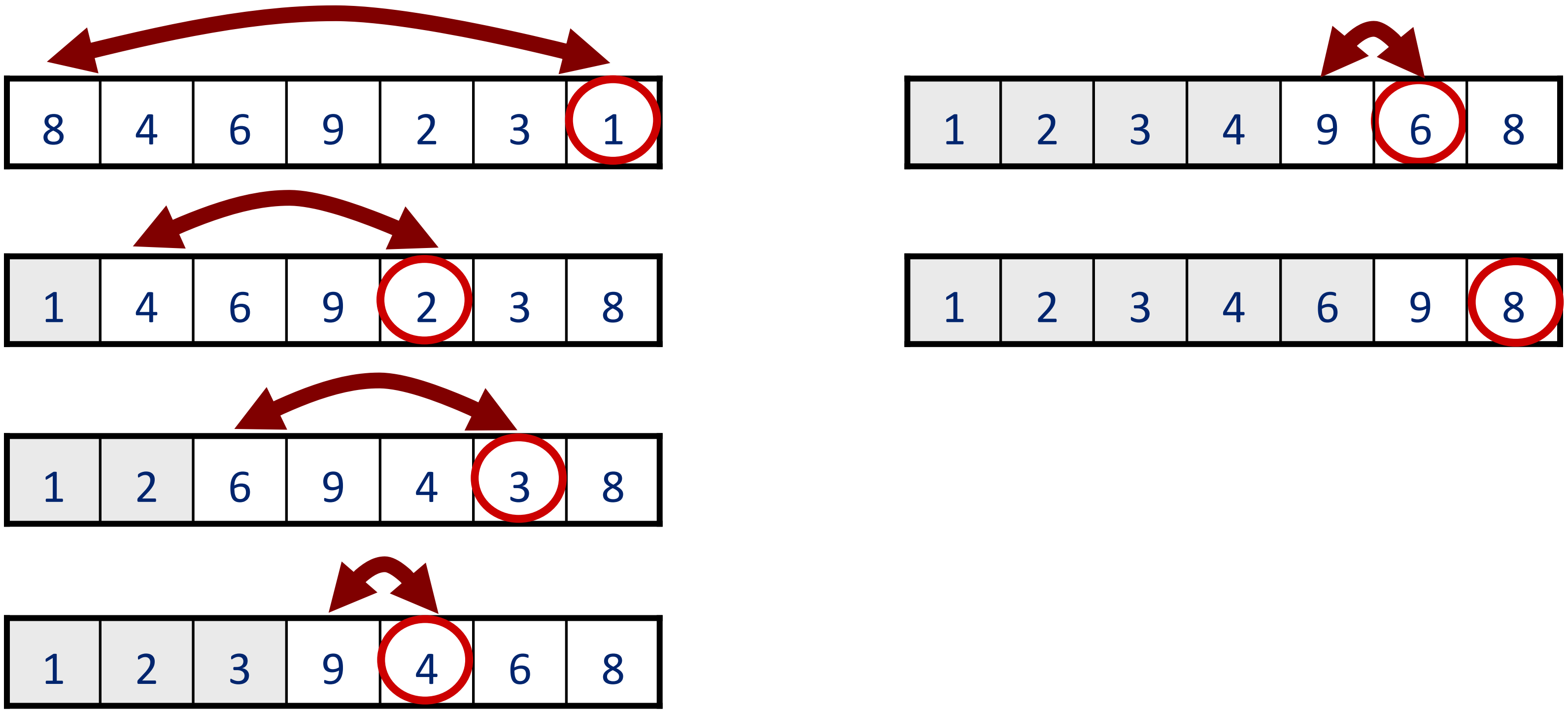
Example



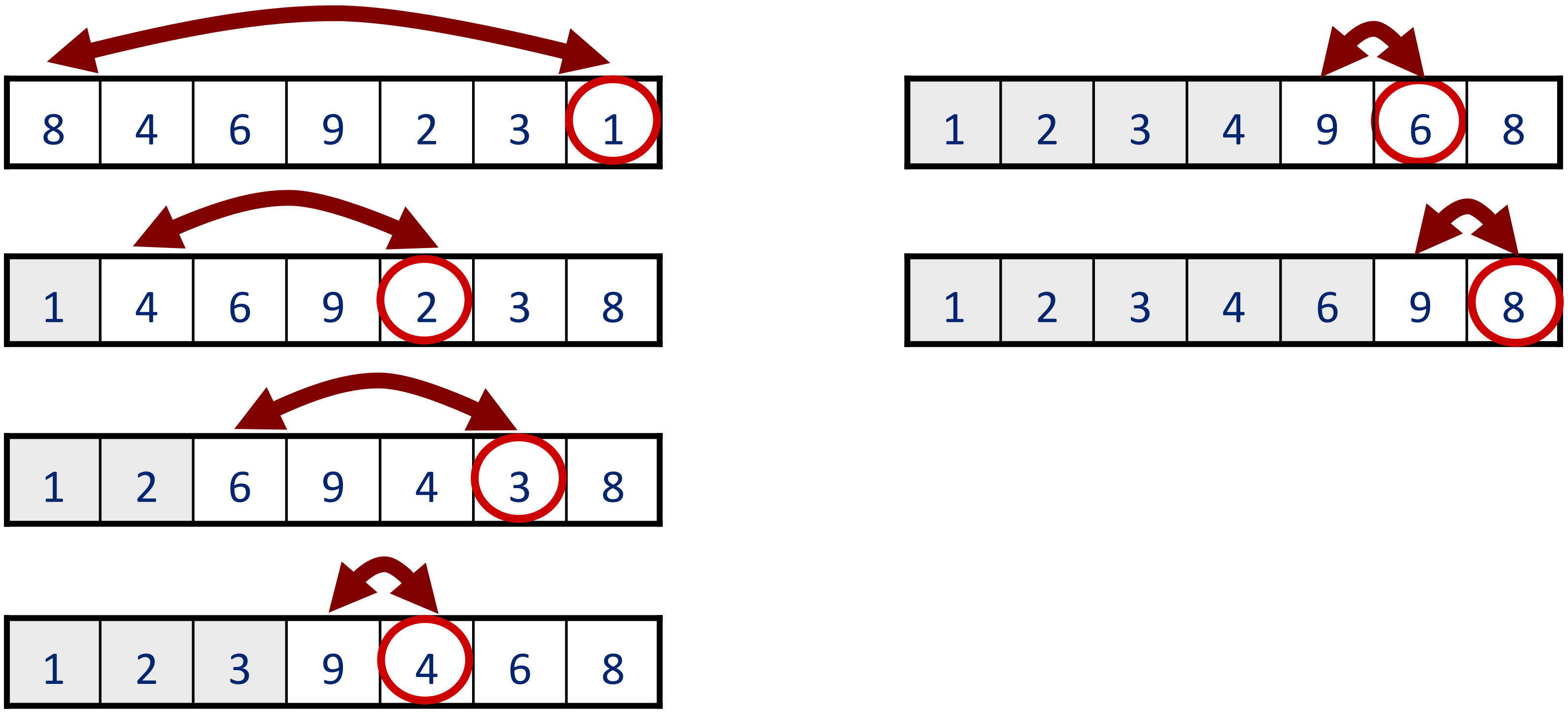
Example



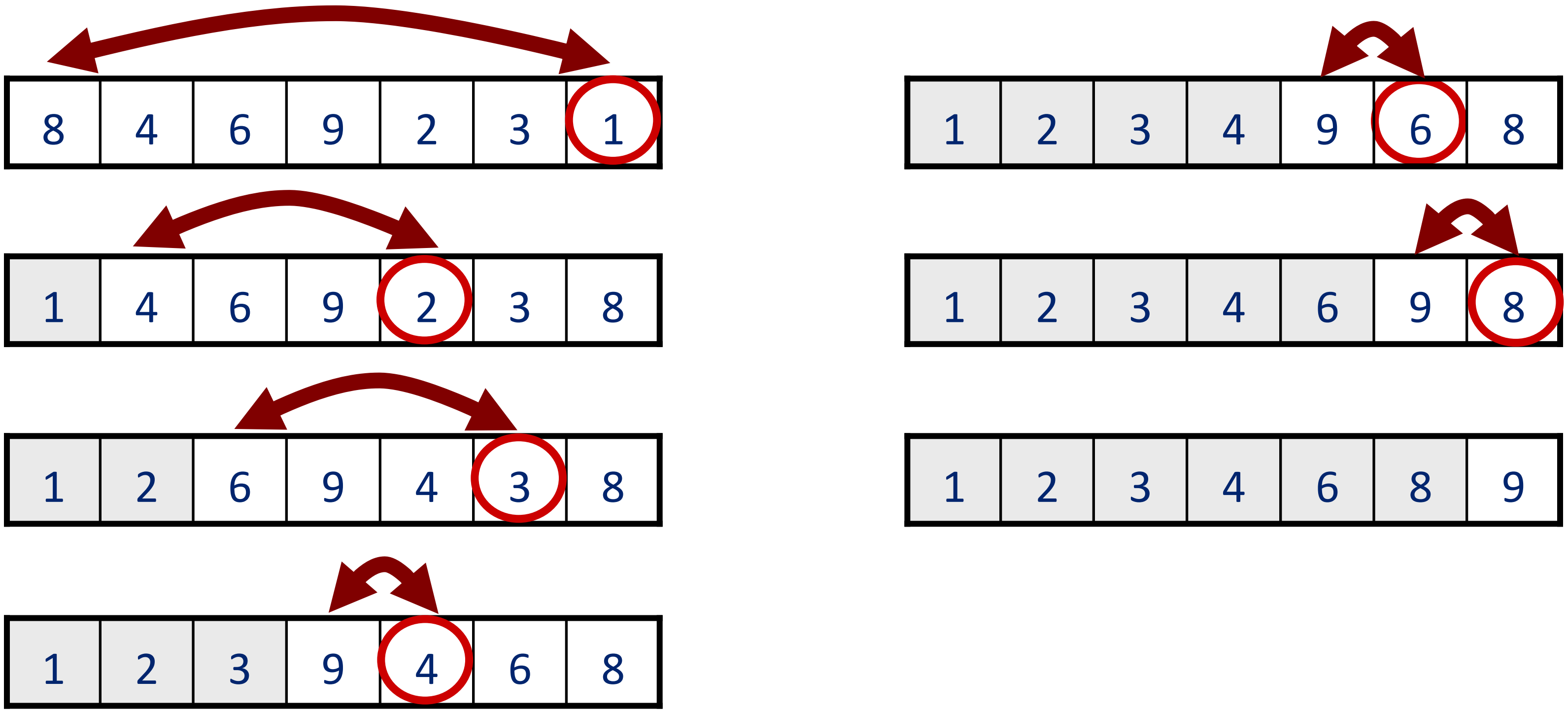
Example



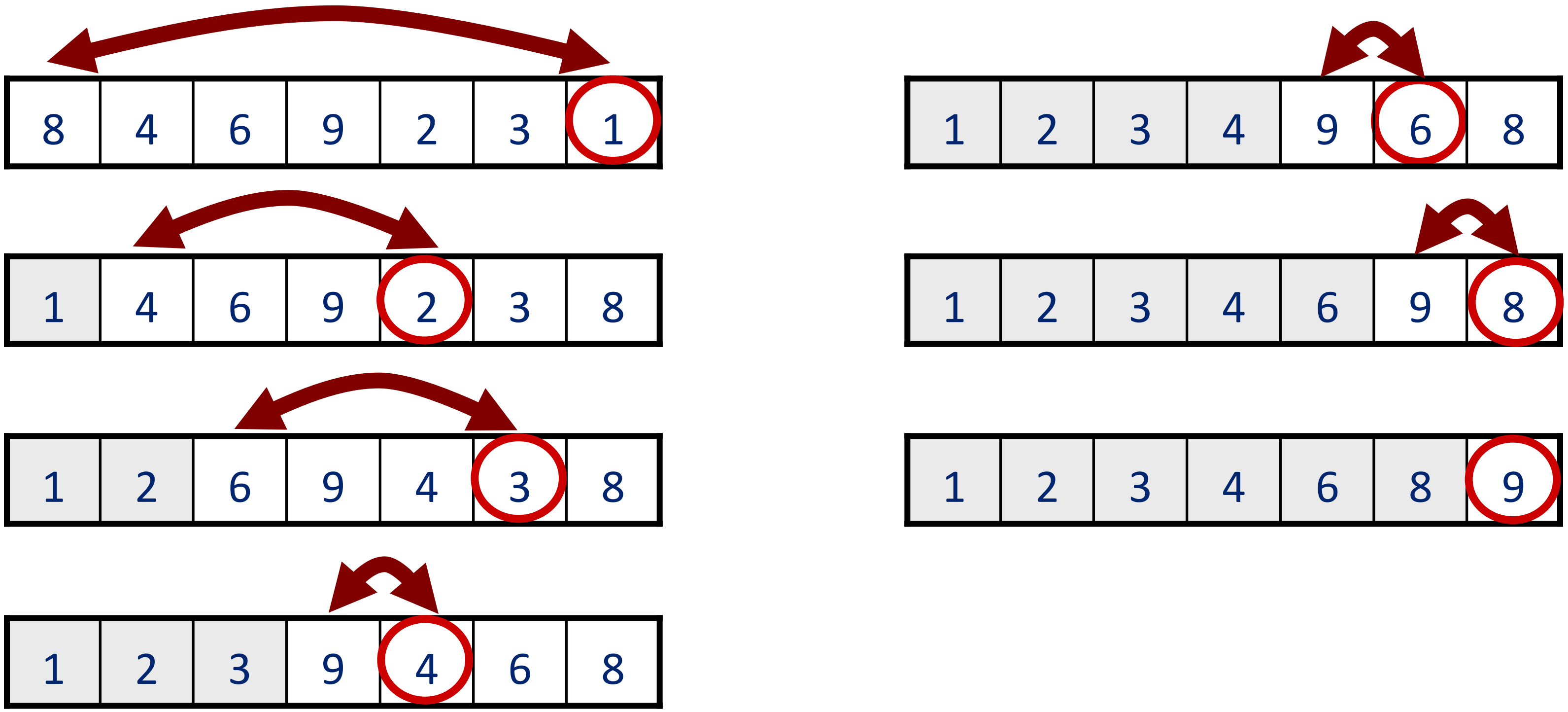
Example



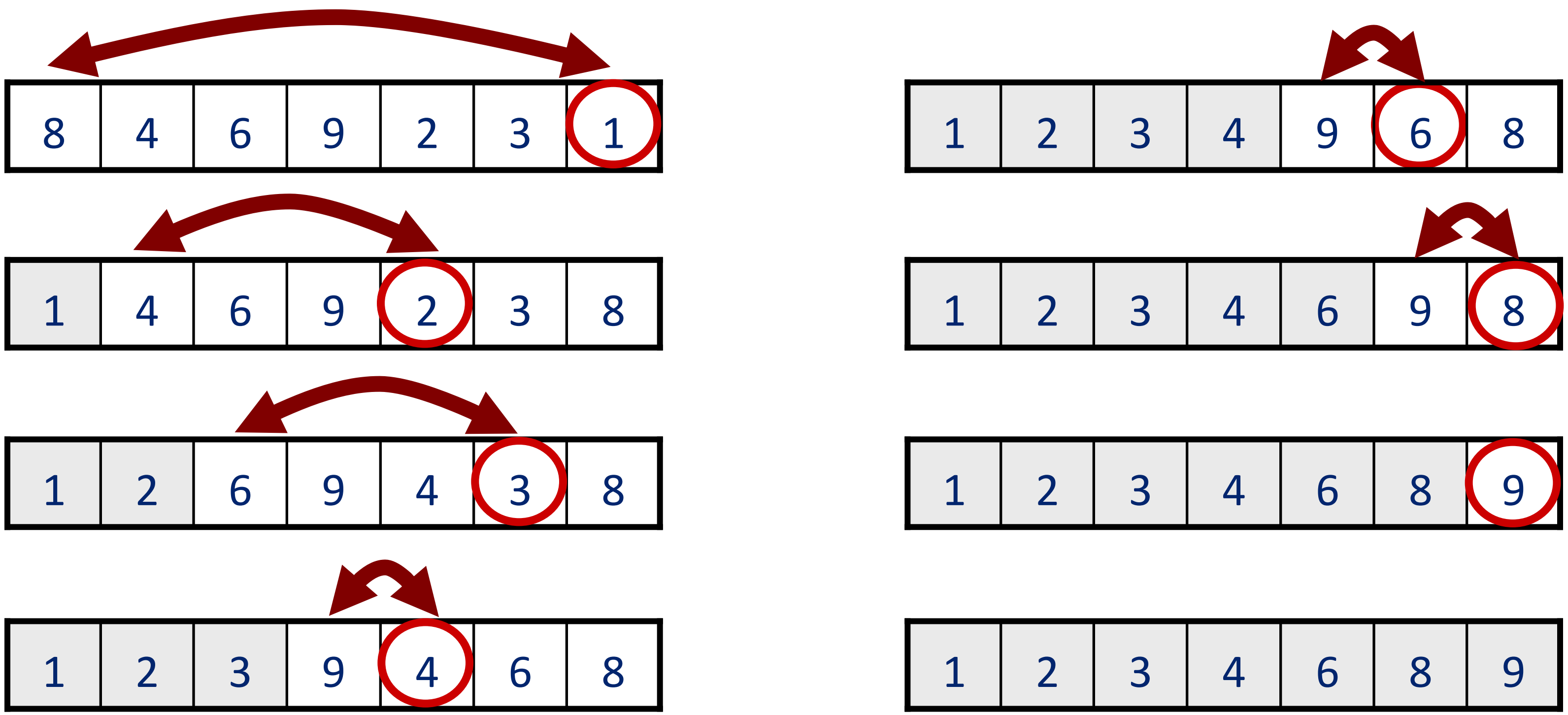
Example



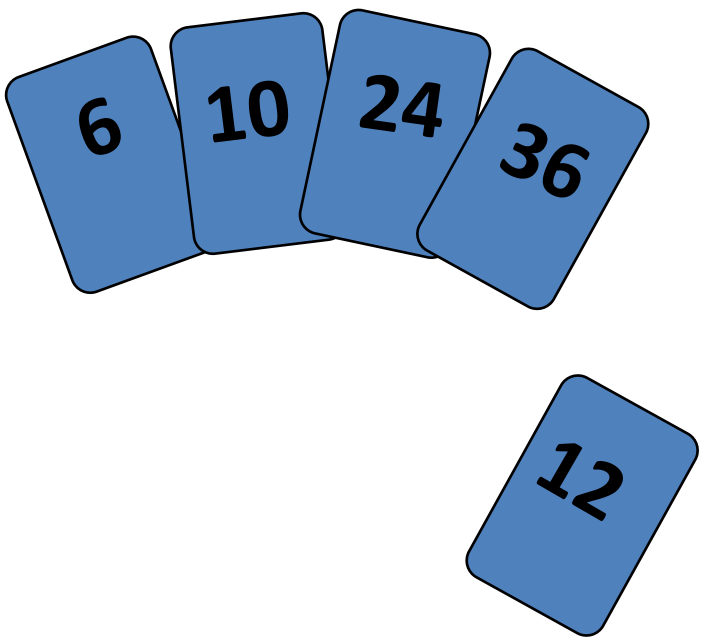
Example



Example

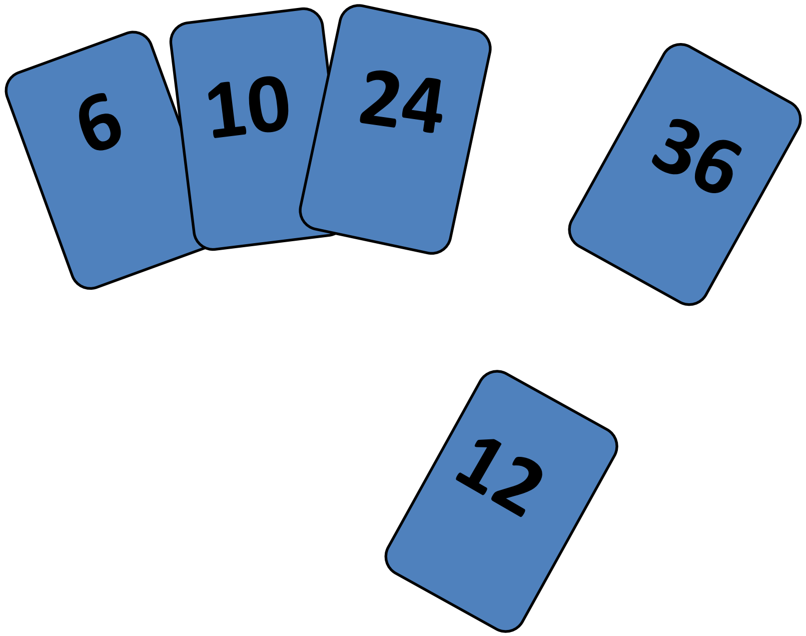

8 4 6 9 2 3 1
1 4 6 9 2 3 8
1 2 6 9 4 3 8
1 2 3 9 4 6 8
1 2 3 4 9 6 8
1 2 3 4 6 9 8
1 2 3 4 6 8 9
1 2 3 4 6 8 9

Insertion Sort

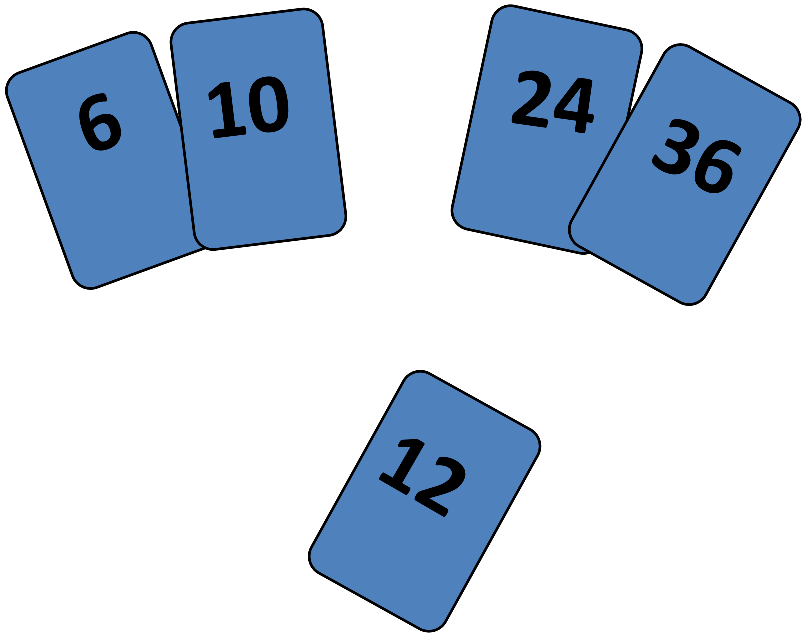


To insert 12, we need to make room for it by moving first 36 and then 24.

Insertion Sort

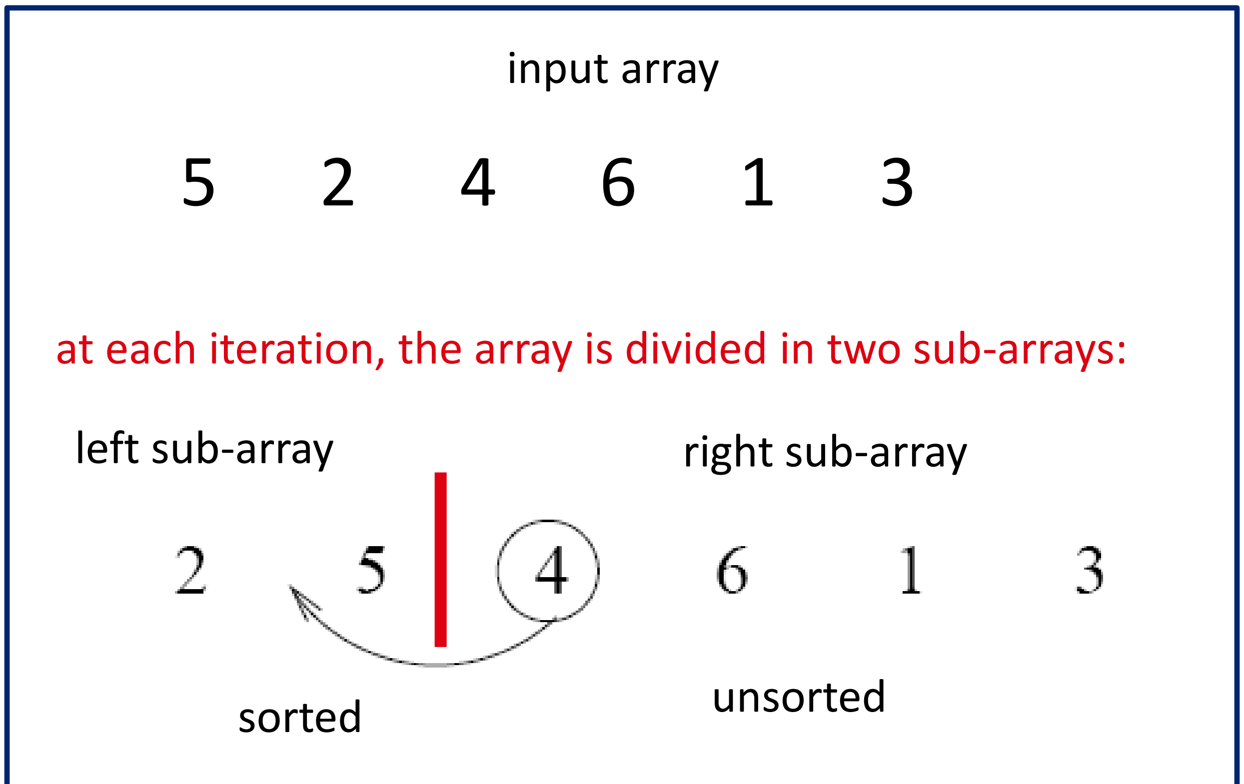


Insertion Sort



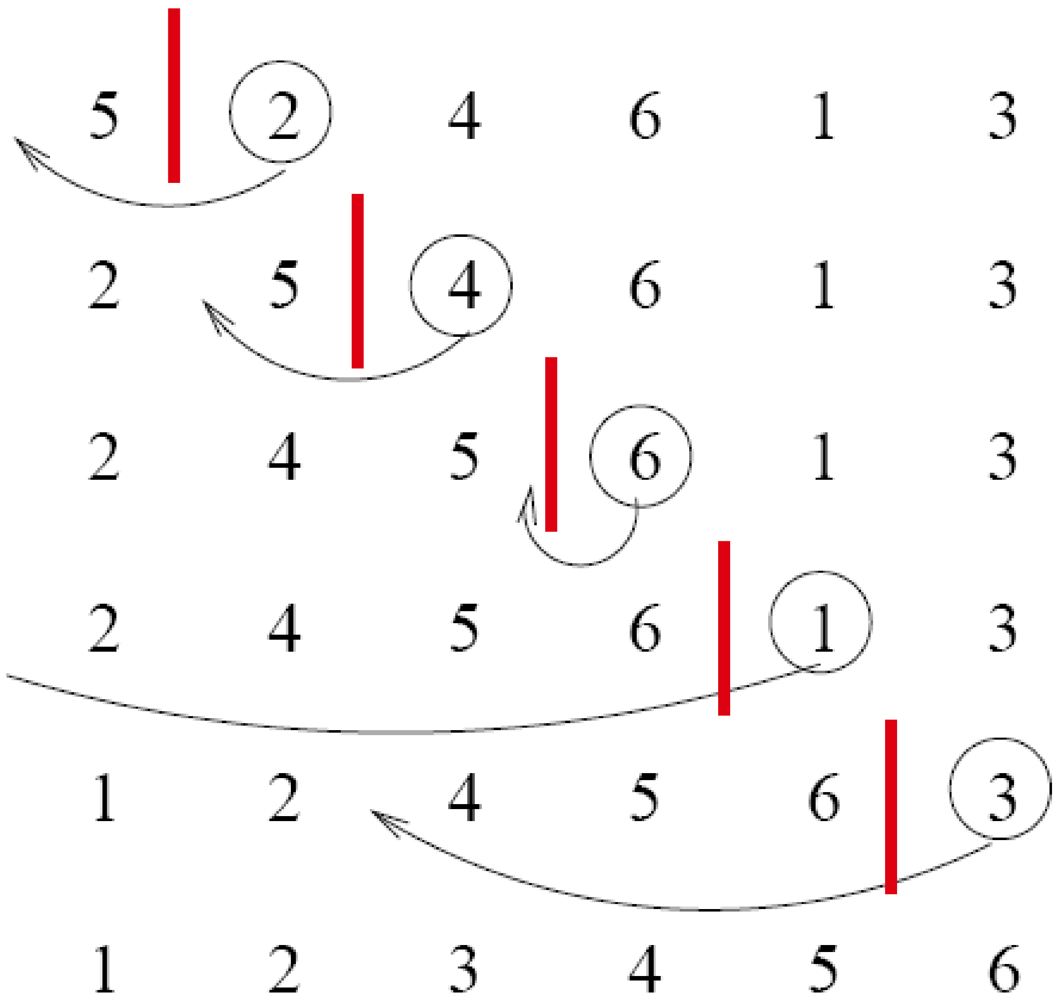
Insertion Sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange value with larger ones to left, one-by-one.



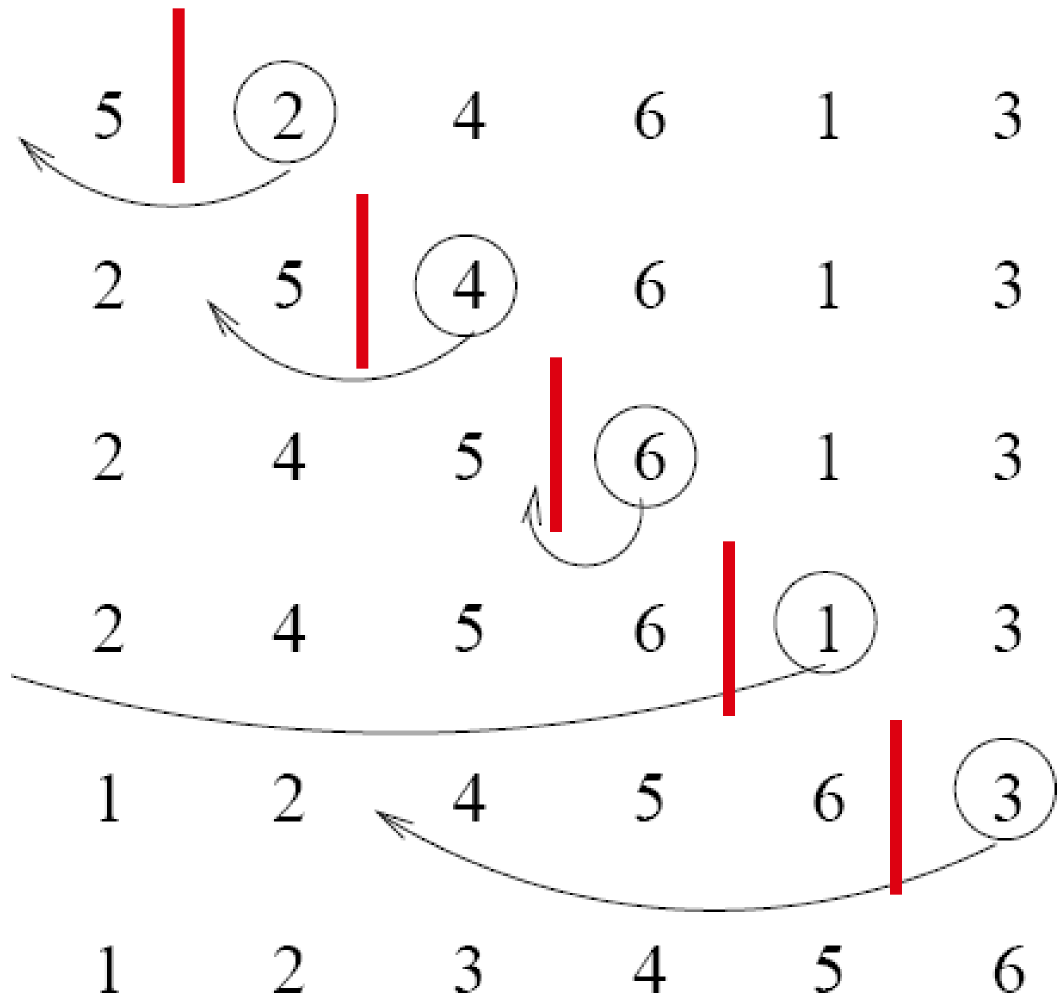
Penn Engineering

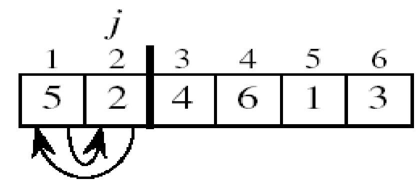
Insertion Sort



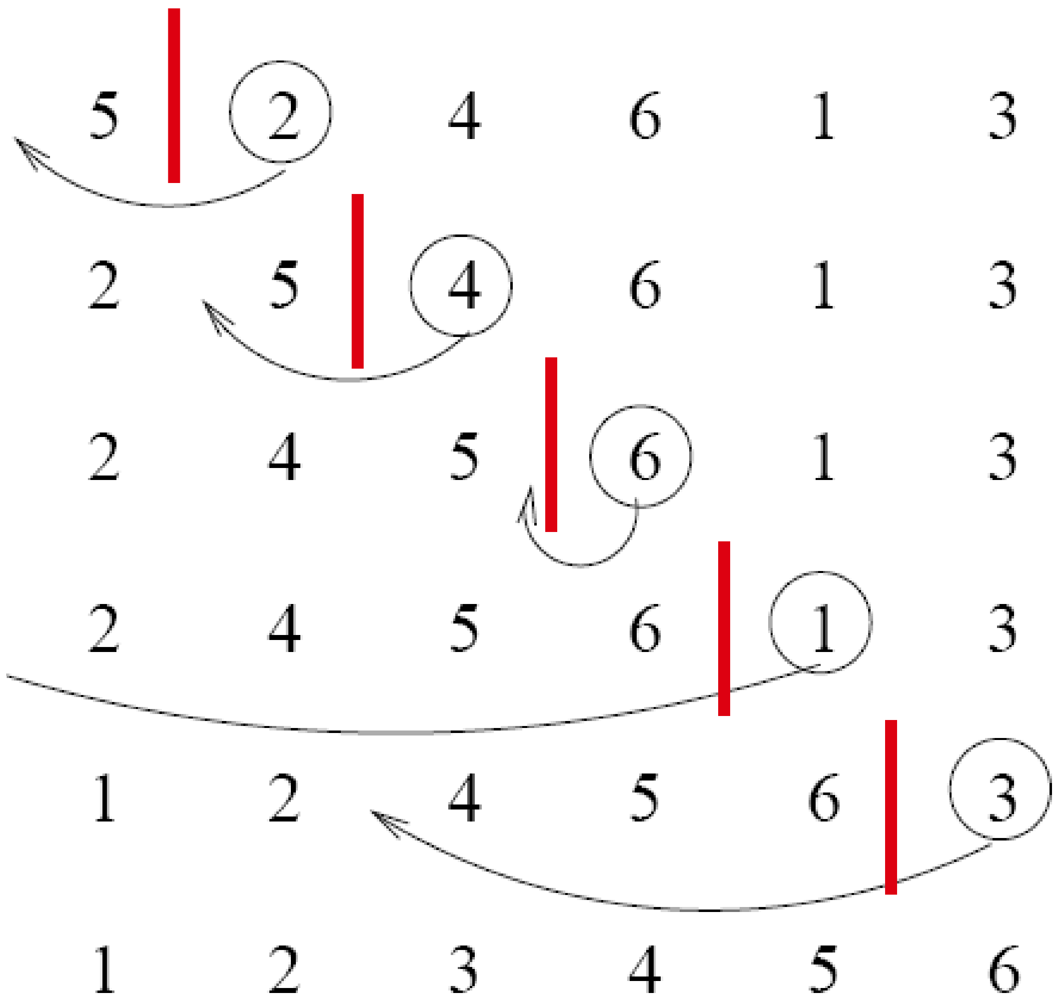
Penn Engineering

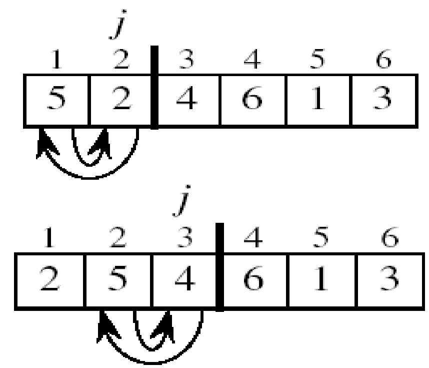
Insertion Sort



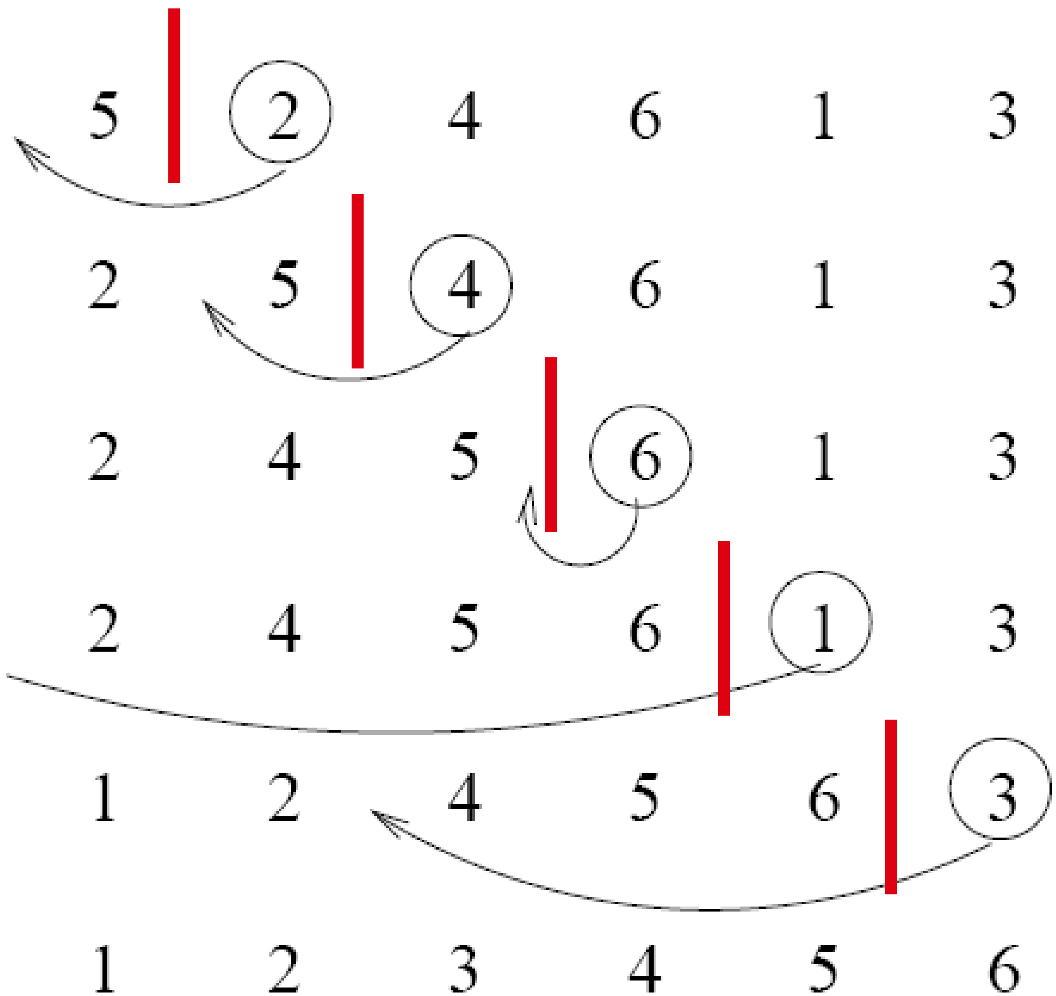


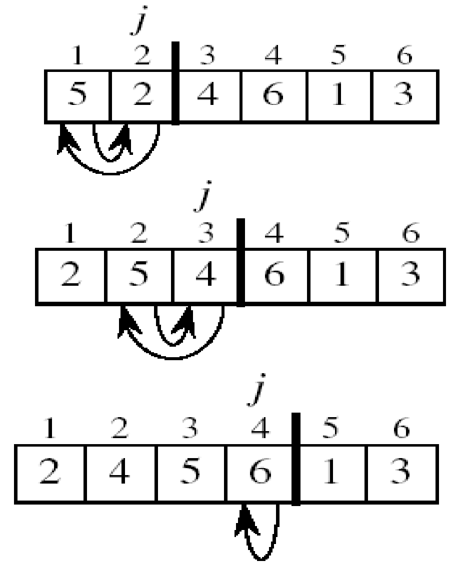
Insertion Sort



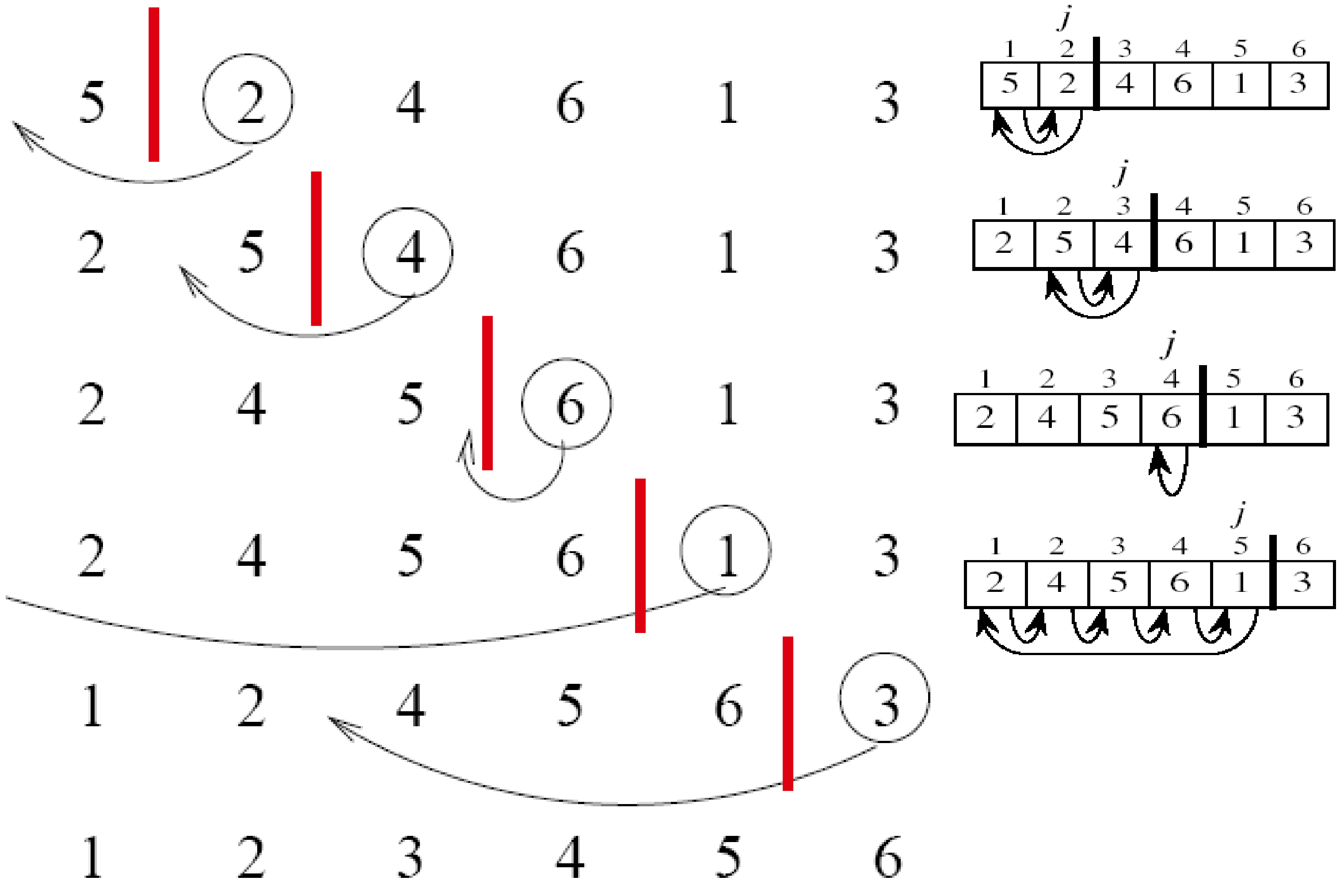


Insertion Sort

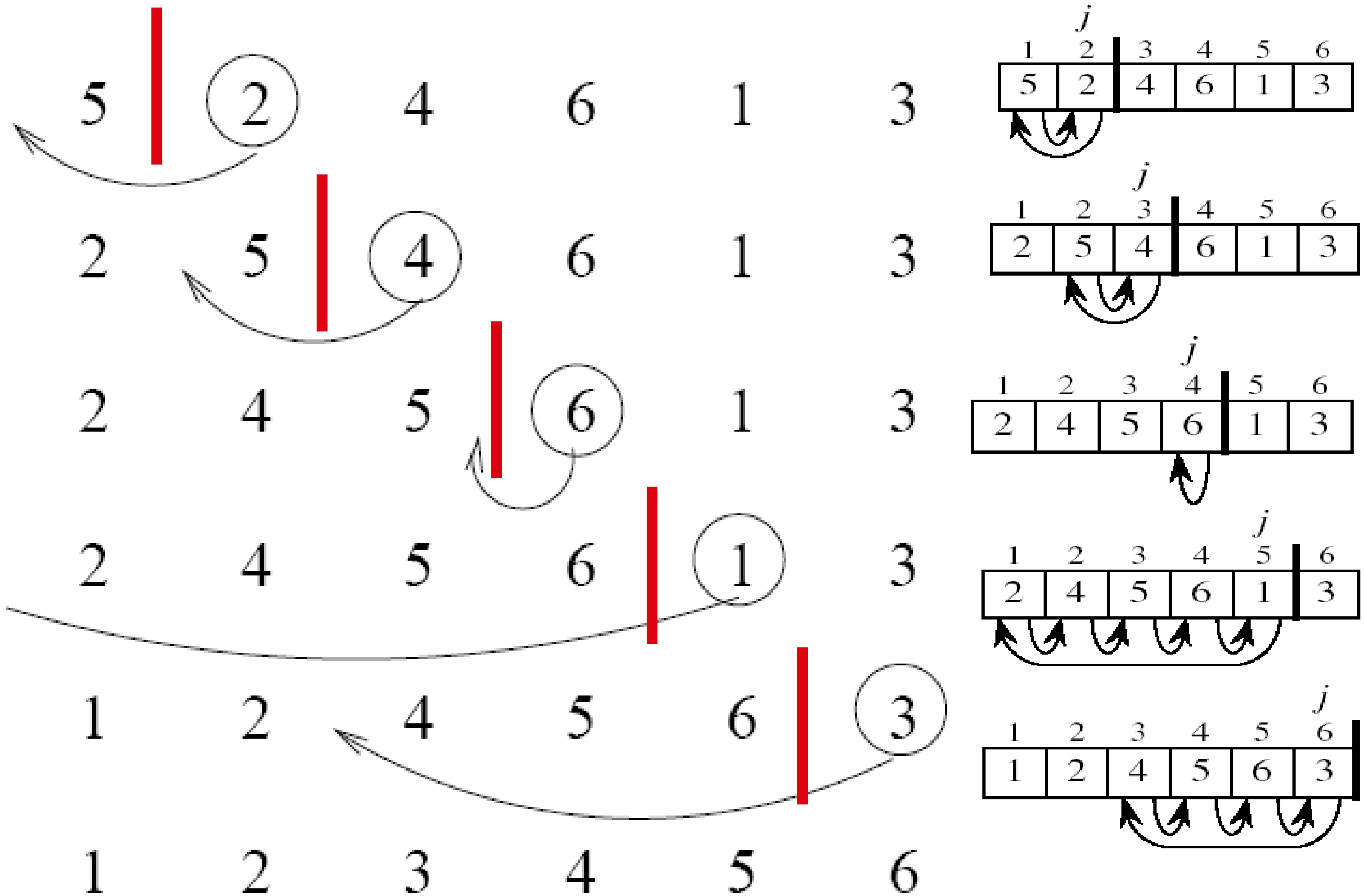




Insertion Sort



Insertion Sort



Insertion Sort

```java
public class Insertion {

   public static void sort(int[] a) {
      int N = a.length;
      for (int i = 1; i < N; i++)
         for (int j = i; j > 0; j--)
            if (a[j-1] > a[j])
               exch(a, j-1, j);
            else break;
   }

   private static void exch(int[] a, int i, int j) {
      int swap = a[i];
      a[i] = a[j];
      a[j] = swap;
   }
}
```

Penn Engineering

Insertion Sort: Call By Reference


```
public class Insertion {
    public static void sort(int[] a) {
        int N = a.length;
        for (int i = 1; i < N; i++)
            for (int j = i; j > 0; j--)
                if (a[j-1] > a[j])
                    exch(a, j-1, j);
                else break;
    }

    private static void exch(int[] b, int i, int j) {
        int swap = b[i];
        b[i] = b[j];
        b[j] = swap;
    }
}
```

b contains a copy of the address in a. Both point to the same contents.

Insertion Sort: Observation

Observe and tabulate running time for various values of N.

- Data source: N random numbers between 0 and 1.
- Machine: iMac Core i5 2.7GH, 12GB RAM.
- Timing: `System.currentTimeMillis()`.

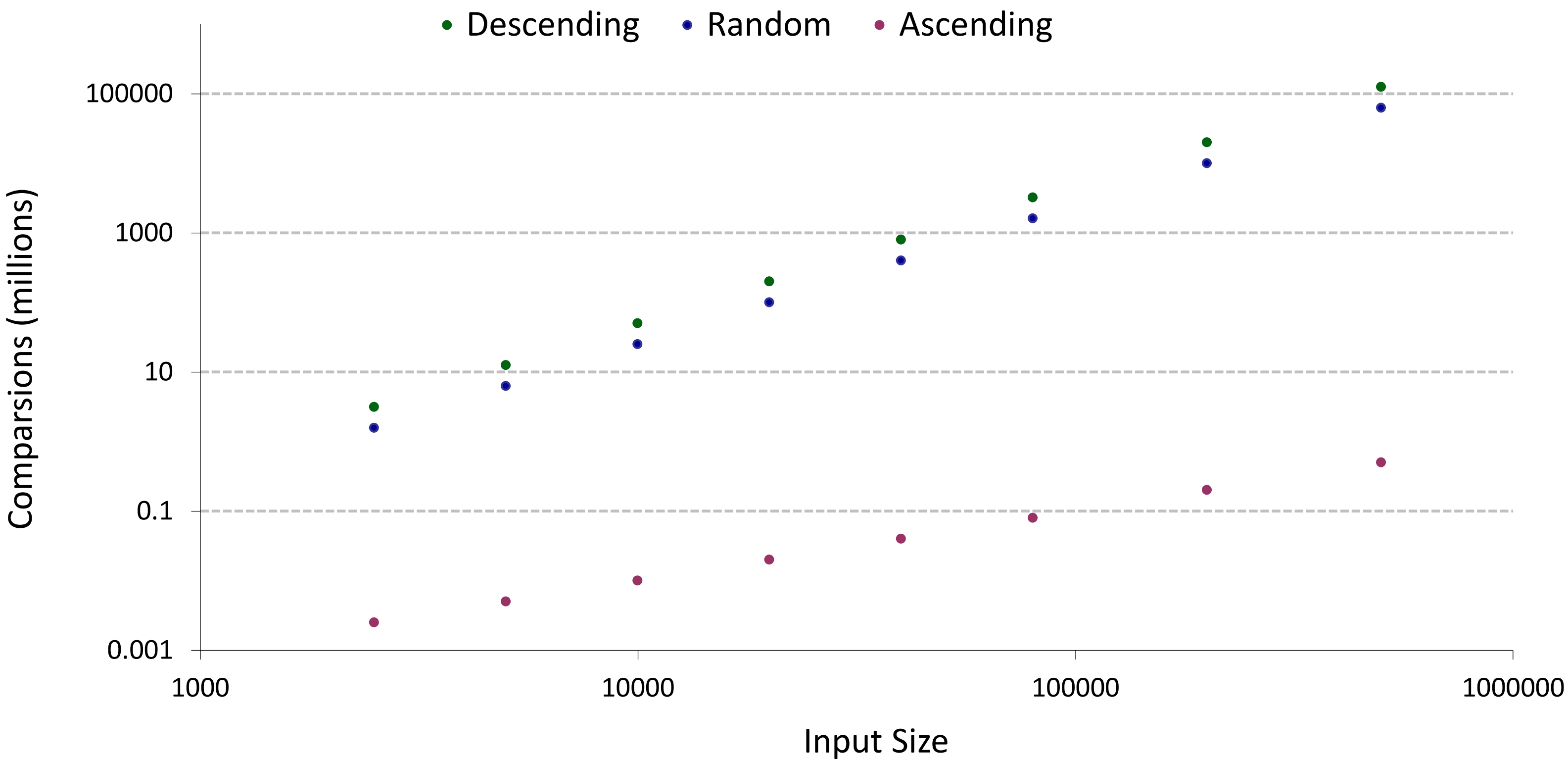
N	Comparisons	Time
5,000	6.2 million	0.016 seconds
10,000	25 million	0.063 seconds
20,000	99 million	0.211 seconds
40,000	400 million	0.79 seconds
80,000	1600 million	3.125 seconds

Penn Engineering

Empirical Analysis

Observation. Number of compares depends on input family.

- Descending: $\sim N^2 / 2$
- Random: $\sim N^2 / 4$
- Ascending: $\sim N$



Mathematical Analysis

Worst Case. (descending)

- Iteration i requires i comparisons.
- Total = $(0 + 1 + 2 + \ldots + N\text{-}1) \sim N^2 / 2$ compares.

E	F	G	H	I	J	D	C	B	A

i

Average Case. (random)

- Iteration i requires $i / 2$ comparisons on average.
- Total = $(0 + 1 + 2 + \ldots + N\text{-}1) / 2 \sim N^2 / 4$ compares

A	C	D	F	H	J	E	B	I	G

i

Penn Engineering

Mathematical Analysis

Worst Case. (descending)

- Iteration i requires i comparisons.
- Total = $(0 + 1 + 2 + \ldots + N\text{-}1) \sim N^2 / 2$ compares.

E	F	G	H	I	J	D	C	B	A

Say, "Quadratic," "N^2," "Order N^2," or "Oh of N^2."

Write, "$O(N^2)$"

Average Case. (random)

- Iteration i requires i / 2 comparisons on average.
- Total = $(0 + 1 + 2 + \ldots + N\text{-}1) / 2 \sim N^2 / 4$ compares

A	C	D	F	H	J	E	B	I	G

i

Penn Engineering

Sorting Challenge 1

- Q. A credit card company sorts 10 million customer account numbers, for use with binary search.

- Using insertion sort, what kind of computer is needed?

A. Toaster
B. Cell phone
C. Your laptop
D. Supercomputer
E. Google server farm

Insertion Sort: Lesson

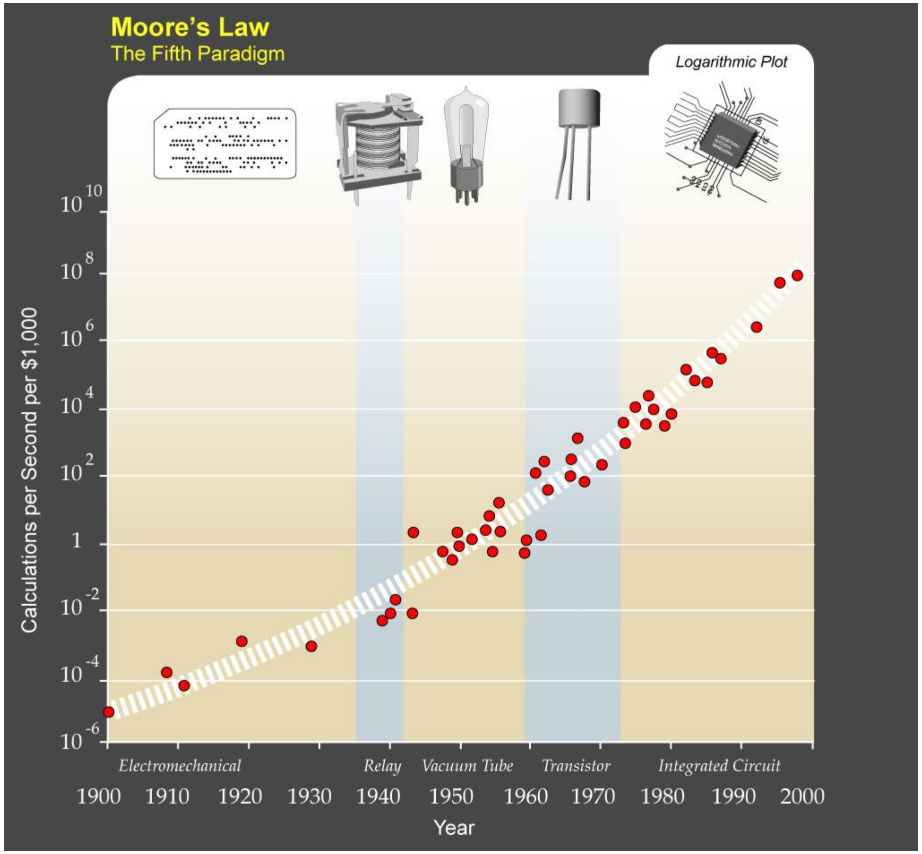
Lesson. Supercomputer can't rescue a bad algorithm.

Computer	Comparisons Per Second	Thousand	Million	Billion
laptop	10^7	instant	1 day	3 centuries
super	10^{12}	instant	1 second	2 weeks

Moore's Law

Moore's Law. Transistor density on a chip doubles every 2 years.

Variants. Memory, disk space, bandwidth, computing power/$.



http://en.wikipedia.org/wiki/Moore's_law

Moore's Law and Algorithms

Quadratic algorithms do not scale with technology.

- New computer may be 10x as fast.
- But, has 10x as much memory so problem may be 10x bigger.
- With quadratic algorithm, takes 10x as long!

> *"Software inefficiency can always outpace Moore's Law. Moore's Law isn't a match for our bad coding." – Jaron Lanier*

Lesson. Need linear or N log N algorithms to keep pace with Moore's law.