Introduction to Computer Science

Reichman University

Lecture 2-2

Conditional and Iterative Processing Part II

I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.

Lecture plan

- Conditional logic:

  `if`

  `switch`

- Strings

- Iterative logic:

  `while`

  ➡ `for`

  `do ... while`

The FOR statement

Example 1: Print something 100 times

```
public class ForDemo {
    public static void main(String[] args) {
```

```
% java ForDemo
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
...
```

Algorithm

```
for count = 0 , ..., 99
    print ("I will not argue...")
```

I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.

The FOR statement

Example 1: Print something 100 times

```java
public class ForDemo {
    public static void main(String[] args) {
        for (int count = 0; count < 100; count++) {
            System.out.println("I will not argue ...");
        }
    }
}
```

```
% java ForDemo
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
I will not argue ...
...
```

Algorithm

```
for $count$ = 0 , ..., 99
    print (“I will not argue...”)
```

The FOR statement

Example 2: Print the integers 0, 1, ..., 99

```
// Prints 0, 1, ..., 99
```

```
0
1
2
3
4
5
6
7
...
```

<u>Algorithm</u>

```
for i = 0 , ..., 99
    print (i)
```

The FOR statement

Example 2: Print the integers 0, 1, ..., 99

```java
// Prints 0, 1, ..., 99
for (int i = 0; i < 100; i++) {
    System.out.println(i);
}
```

```
0
1
2
3
4
5
6
7
...
```

Algorithm

```
for i = 0 , ..., 99
    print (i)
```

The FOR statement

Example 3: Print powers of two : $2^0, 2^1, 2^2, 2^3, \ldots, 2^N$

<u>Algorithm</u>

```
for $i = 0, \ldots, N$
    print($i$ , Math.power(2, $i$))
```

```
% java PowersOfTwo 6
0  1
1  2
2  4
3  8
4  16
5  32
6  64
```

The FOR statement

Example 3: Print powers of two : $2^0, 2^1, 2^2, 2^3, \ldots, 2^N$

`for` implementation

```java
// Prints the powers of 2 up to 2^N
for (int i = 0; i <= N; i++) {
    System.out.println(i + "  " + Math.pow(2,i));
}
```

Algorithm

```
for $i = 0 , \ldots, N$
    print($i$ , Math.power(2 , $i$))
```

Inefficient!

Do you see why?

```
% java PowersOfTwo 6
0  1
1  2
2  4
3  8
4  16
5  32
6  64
```

The FOR statement

Example 3: Print powers of two : $2^0, 2^1, 2^2, 2^3, \ldots, 2^N$

`for` implementation

```
// Prints the powers of 2 up to 2^N
```

<u>Algorithm</u>

```
$v = 1$
for $i = 0, \ldots, N$
    print($i$ , $v$)
    $v = 2 * v$
```

```
% java PowersOfTwo 6
0  1
1  2
2  4
3  8
4  16
5  32
6  64
```

The FOR statement

Example 3: Print powers of two : $2^0, 2^1, 2^2, 2^3, \ldots, 2^N$

`for` implementation

```
// Prints the powers of 2 up to 2^N
int v = 1;
for (int i = 0; i <= N; i++) {
    System.out.println(i + "  " + v);
    v = 2 * v;
}
```

<u>Algorithm</u>

```
$v = 1$
for $i = 0, \ldots, N$
    print($i$ , $v$)
    $v = 2 * v$
```

`while` implementation (equivalent)

```
// Prints the powers of 2 up to 2^N
int v = 1;
int i = 0;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

```
% java PowersOfTwo 6
0  1
1  2
2  4
3  8
4  16
5  32
6  64
```

<u>Best practice</u>: A for-loop is typically preferred over a while-loop.

The FOR statement

```
for (  int i = 0;   i <= N;   i++  ) {
    System.out.println(i + "  " + v);
    v = 2 * v;
}
```

The FOR statement

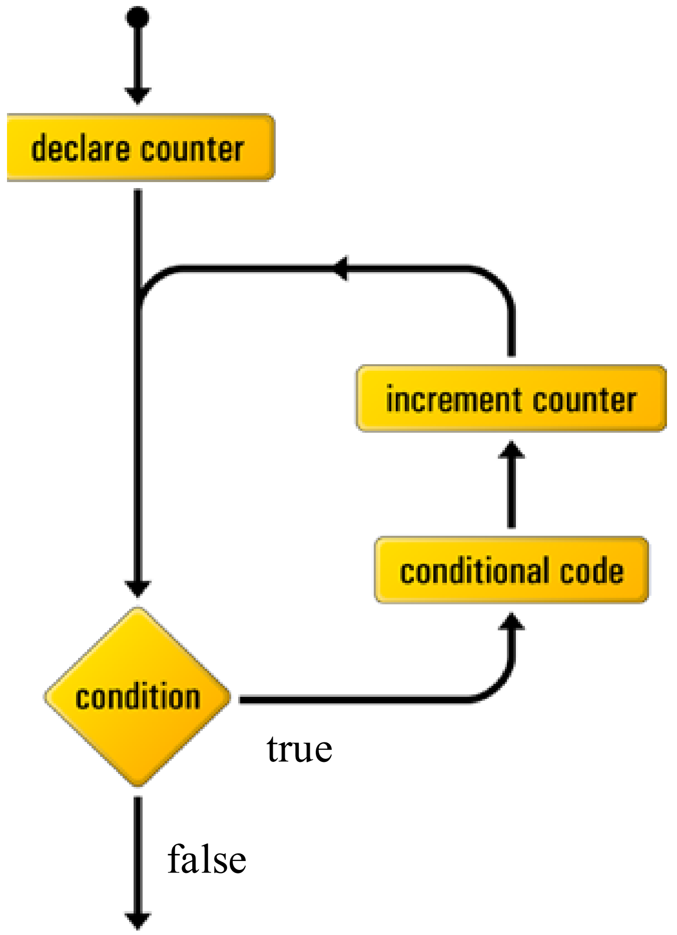
```
for ( int i = 0; i <= N; i++ ) {
    System.out.println(i + "  " + v);
    v = 2 * v;
}
```

declares a counter

checks condition

increments the counter

conditional code



The FOR statement

```
// Computes sum = 1 + 2 + 3 + ... + N
```

The FOR statement

```
// Computes sum = 1 + 2 + 3 + ... + N
int sum = 0;
for (int i = 1; i <= N; i++) {
    sum += i;   // shorthand of sum = sum + i
}
```

```
// Computes N!= 1 * 2 * 3 * ... * N
```

The FOR statement

```
// Computes sum = 1 + 2 + 3 + ... + N
int sum = 0;
for (int i = 1; i <= N; i++) {
    sum += i;   // shorthand of sum = sum + i
}
```

```
// Computes N!= 1 * 2 * 3 * ... * N
int factorial = 1;
for (int i = 1; i <= N; i++) {
    factorial *= i; // shorthand of factorial = factorial * i
}
```

```
// Computes sum = 1 + 1/2 + 1/3 + 1/4 + ... + 1/N
```

The FOR statement

```
// Computes sum = 1 + 2 + 3 + ... + N
int sum = 0;
for (int i = 1; i <= N; i++) {
    sum += i;
}
```

```
// Computes N!= 1 * 2 * 3 * ... * N
int factorial = 1;
for (int i = 1; i <= N; i++) {
    factorial *= i;
}
```

```
// Computes sum = 1 + 1/2 + 1/3 + 1/4 + ... + 1/N
double sum = 0.0;
for (int i = 1; i <= N; i++) {
    sum = sum + 1.0 / i;
}
```

Palindrome revisited, using FOR

```
public class Palindrome1 {
   public static void main(String[] args) {
         // Gets the string from the user:
         String s = args[0];
```

```
% java Palindrome1 adam
adam is not a palindrome

% java Palindrome1 madam
madam is a palindrome
```

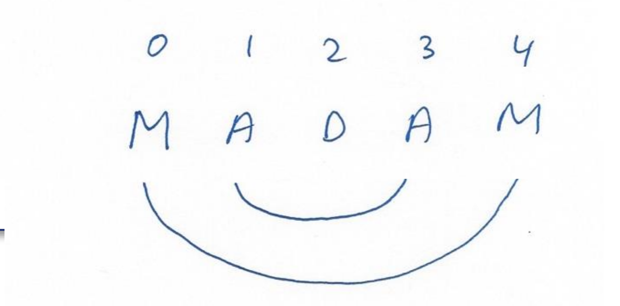
Algorithm

(string length = N)

Compare letters:

0 and 9
1 and 8
2 and 7
…
i and $N-1-i$

Stop when $i = N / 2$



Palindrome revisited, using FOR

```
public class Palindrome1 {
   public static void main(String[] args) {
        // Gets the string from the user:
        String s = args[0];
        boolean isPalindrome = true;
        int N = s.length();
        int mid = N / 2;
        for (int i = 0; i < mid; i++) {
            if (s.charAt(i) != s.charAt(N - 1 - i)) {
                isPalindrome = false;
                break; // Can be used to exit any for / while loop
            }
        }
        if (isPalindrome)
            System.out.println(s + " is a palindrome");
        else
            System.out.println(s + " is not a palindrome");
    }
}
```

```
% java Palindrome1 adam
adam is not a palindrome

% java Palindrome1 madam
madam is a palindrome
```

Algorithm

(string length = N)

Compare letters:

0 and 9

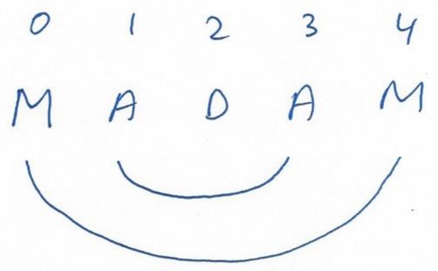
1 and 8

2 and 7

…

i and $N-1-i$

Stop when $i = N / 2$



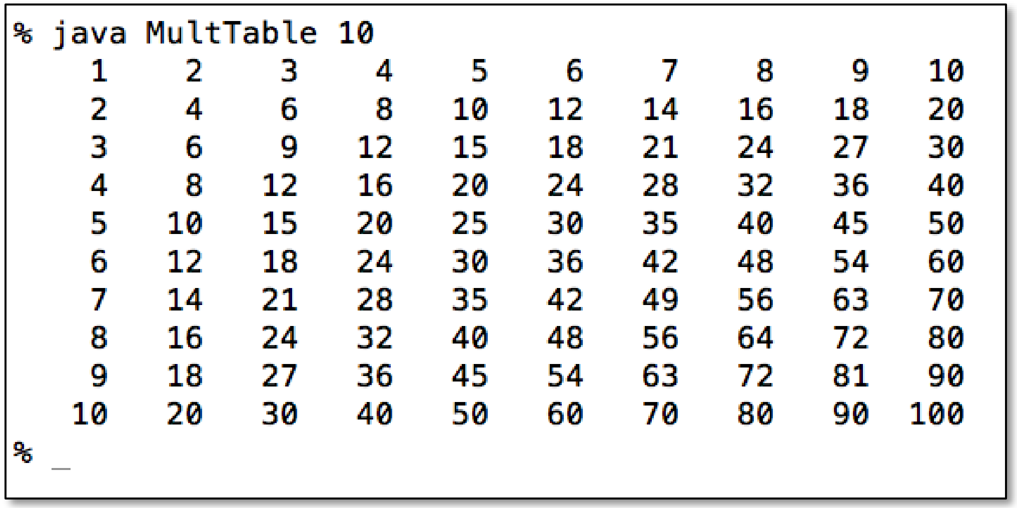
Nested for

Task: Print a multiplication table of size *N*

```
public class MultTable {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
```

Algorithm

```
for $i = 1$ , ..., $N$
    for $j = 1$ , ..., $N$
        print ($i * j$)
    newline
```



```
% java MultTable 10
    1    2    3    4    5    6    7    8    9   10
    2    4    6    8   10   12   14   16   18   20
    3    6    9   12   15   18   21   24   27   30
    4    8   12   16   20   24   28   32   36   40
    5   10   15   20   25   30   35   40   45   50
    6   12   18   24   30   36   42   48   54   60
    7   14   21   28   35   42   49   56   63   70
    8   16   24   32   40   48   56   64   72   80
    9   18   27   36   45   54   63   72   81   90
   10   20   30   40   50   60   70   80   90  100
% _
```

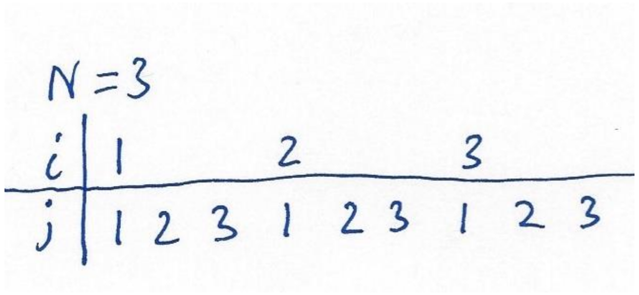
Nested for

Task: Print a multiplication table of size *N*

```
public class MultTable {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1 ; i <= N ; i++) {
            for (int j = 1 ; j <= N ; j++) {
                System.out.print("  " + i * j);
            }
            System.out.println();
        }
    }
}
```

Algorithm

```
for $i = 1 , ..., N$
    for $j = 1 , ..., N$
        print $(i * j)$
    newline
```



N=3

i	1			2			3		
j	1	2	3	1	2	3	1	2	3

```
% java MultTable 10
     1    2    3    4    5    6    7    8    9   10
     2    4    6    8   10   12   14   16   18   20
     3    6    9   12   15   18   21   24   27   30
     4    8   12   16   20   24   28   32   36   40
     5   10   15   20   25   30   35   40   45   50
     6   12   18   24   30   36   42   48   54   60
     7   14   21   28   35   42   49   56   63   70
     8   16   24   32   40   48   56   64   72   80
     9   18   27   36   45   54   63   72   81   90
    10   20   30   40   50   60   70   80   90  100
% _
```

Lecture plan

- Conditional logic:

  `if`

  `switch`

- Strings

- Iterative logic:

  `while`

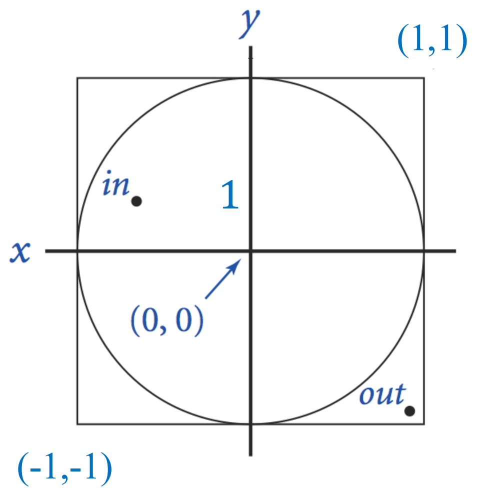
  `for`

  ➡ `do ... while`

Do-While (by example)

Task: Given: A unit circle ($r = 1$) centered at (0 , 0);

Generate a random point (x , y) inside the circle.



Algorithm (rejection method):

Generate a random point inside the unit square;

If the point is inside the unit circle, we're done;

Else, keep trying.

```
do {
    // Generates random x and y, each in [-1,1)
    x = 2.0 * Math.random() - 1.0;
    y = 2.0 * Math.random() - 1.0;
} while (x * x + y * y > 1.0);
// (x,y) is inside the circle
...
```

Geometry:

Circle outline equation: $x^2 + y^2 = r^2$

Rule: Point (x,y) is inside the circle

iff $x^2 + y^2 \leq r^2$

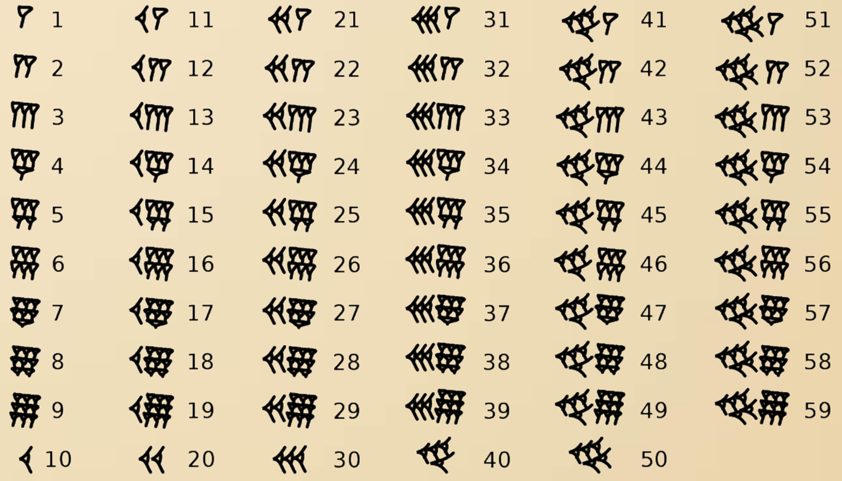
Technical observation

A `while` loop executes 0 or more iterations;

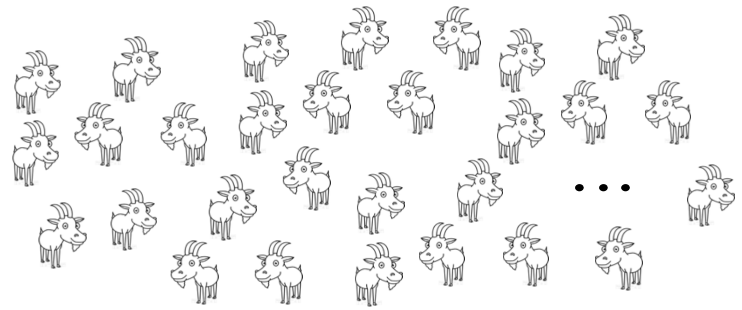
A `do-while` loop executes 1 or more iterations.

Lecture 2-2

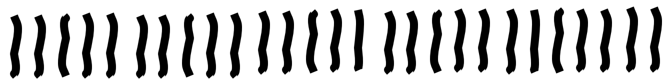
Numeral Systems


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
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

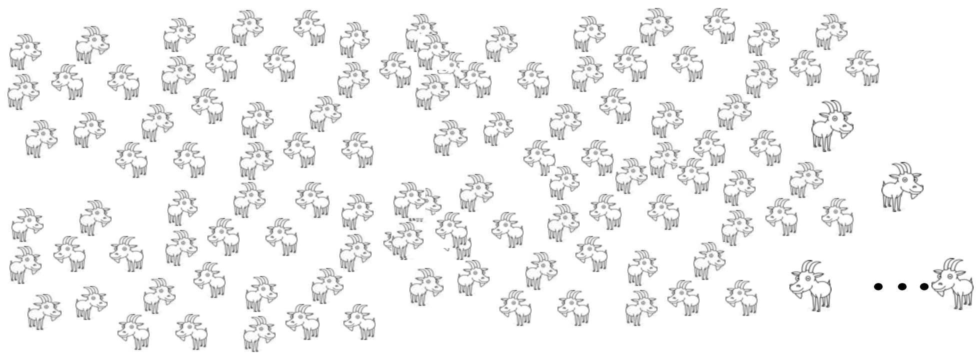
Numeral systems



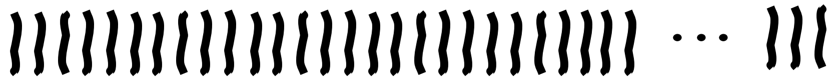
Twenty seven goats

Unary: 

Numeral systems

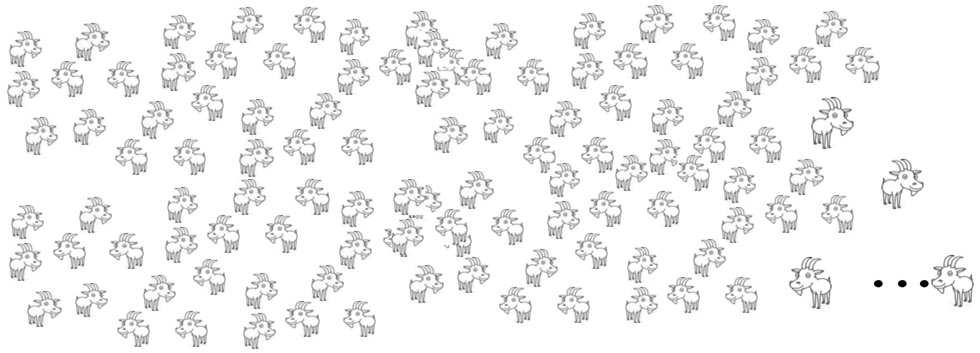


Seven thousands and fifty three goats

Unary: 

Oy…

Numeral systems



Seven thousands and fifty three goats

$$\begin{array}{cccc} \color{blue}{3} & \color{blue}{2} & \color{blue}{1} & \color{blue}{0} \\ 7 & 0 & 5 & 3 \end{array}$$

human friendly

$$\sum_{0}^{n-1} d_i \cdot 10^i = 7 \cdot 10^3 + 0 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^0 = 7053$$

(n is the number of digits in the numeral, d_i is the digit in position i)

$$\begin{array}{ccccccccccccc} \color{blue}{12} & \color{blue}{11} & \color{blue}{10} & & & \color{blue}{\cdot} & \color{blue}{\cdot} & \color{blue}{\cdot} & & \color{blue}{3} & \color{blue}{2} & \color{blue}{1} & \color{blue}{0} \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1_{bin} \end{array}$$

computer friendly

$$\sum_{0}^{n-1} d_i \cdot 2^i = 1 \cdot 2^{12} + 1 \cdot 2^{11} + 0 \cdot 2^{10} + \cdots + 1 \cdot 2^0 = 7053$$

Numeral systems

Decimal	Binary
0	0
1	1
2	1 0
3	1 1

Numeral systems

Decimal	Binary
0	0
1	1
2	1 0
3	1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Numeral systems

Decimal	Binary
0	0
1	1
2	1 0
3	1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1
...	...

Observations

Decimal system: Nothing special about it;

Binary system: Minimal, elegant, efficient.

Inside computers, *everything is represented in binary*.

Because humans prefer using the decimal system, computers have to work hard to convert …

- Binary → decimal
  (when showing numbers to humans)
- Decimal → binary
  (when getting numbers from humans).

Binary → Decimal

$$10011_{bin} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{dec}$$

A sequence of characters, representing `0`'s and `1`'s.

Algorithm

Input: a string of *N* `0` and `1` characters

Output: an integer, representing the decimal value coded by the input

Use a loop to iterate through the input string and compute the decimal value using the formula:

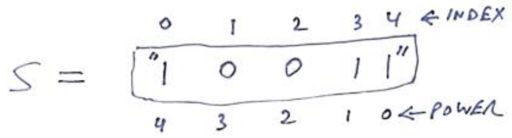
$$\sum_{0}^{N-1} d_i \cdot 2^i$$

```
% java BinToDec 10011
19

% java BinToDec 10000
16

% java BinToDec 101
5

% java BinToDec 1100110101110
6574

% java BinToDec 10
2

% java BinToDec 1
1

% java BinToDec 0
0
```

Binary → Decimal

$$10011_{bin} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{dec}$$

```
public class BinToDec {
   public static void main (String[] args) {
      String s = args[0];
      int N = s.length();
```



$$\sum_{0}^{N-1} d_i \cdot 2^i$$

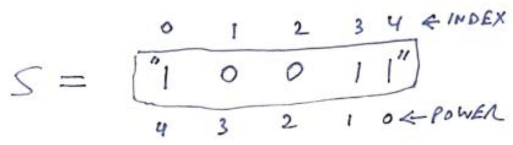
A sequence of characters, representing `0`'s and `1`'s.

```
% java BinToDec 10011
19

% java BinToDec 10000
16

% java BinToDec 101
5

% java BinToDec 1100110101110
6574

% java BinToDec 10
2

% java BinToDec 1
1

% java BinToDec 0
0
```

Binary → Decimal

$$10011_{bin} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{dec}$$

```java
public class BinToDec {
   public static void main (String[] args) {
      String s = args[0];
      int N = s.length();
      int val = 0;
      int power;
      for (int i = 0; i < N; i++) {
         if (s.charAt(i) == '1') {
            power = N - i - 1;
            val = val + (int) Math.pow(2, power);
         }
      }
      System.out.println(val);
   }
}
```



Inefficient!

$$\sum_{0}^{N-1} d_i \cdot 2^i$$

A sequence of characters, representing `0`'s and `1`'s.

```
% java BinToDec 10011
19

% java BinToDec 10000
16

% java BinToDec 101
5

% java BinToDec 1100110101110
6574

% java BinToDec 10
2

% java BinToDec 1
1

% java BinToDec 0
0
```

Binary → Decimal

$$10011_{bin} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{dec}$$

```
public class BinToDec {
   public static void main (String[] args) {
      String s = args[0];
      int N = s.length();
      int val = 0;
      int power;
      for (int i = 0; i < N; i++) {
         if (s.charAt(i) == '1') {
            power = N - i - 1;
            val = val + (int) Math.pow(2, power);
         }
      }
      System.out.println(val);
   }
}
```

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \leftarrow \text{INDEX} \\ S = & \boxed{\text{"}1 & 0 & 0 & 1 & 1\text{"}} \\ & 4 & 3 & 2 & 1 & 0 \leftarrow \text{POWER} \end{array}$$

Inefficient!

A sequence of characters, representing `0`'s and `1`'s.

```
% java BinToDec 10011
19

% java BinToDec 10000
16

% java BinToDec 101
5

% java BinToDec 1100110101110
6574

% java BinToDec 10
2

% java BinToDec 1
1

% java BinToDec 0
0
```

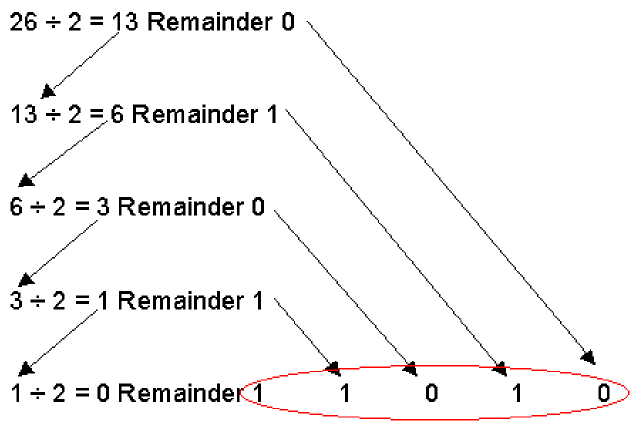
Self-study exercise

- Simulate using a trace table
- Find another algorithm that uses previously computed results without using `Math.pow()`.

Decimal → Binary

<u>Algorithm</u>

$26_{dec} = ?_{bin}$



Input: an integer value;

Output: a string of `0` and `1` characters.

Use a loop to iterate through the integer value and extract a binary digit in each iteration.

```java
public class DecToBin {
    public static void main (String[] args) {
        int x = Integer.parseInt(args[0]);
        String s = "";
        while (x > 0) {
            if ((x % 2) == 0) {
                s = "0" + s;
            }
            else {
                s = "1" + s;
            }
            x = x / 2;  // integer division
        }
        System.out.println(s);
    }
}
```

```
% java DecToBin 26
11010

% java DecToBin 16
10000

% java DecToBin 5
101

% java DecToBin 723462347
101011000111110010100011001011

% java DecToBin 2
10

% java DecToBin 1
1

% java DecToBin 0

%
```

<u>Bug:</u> the input `0` is not processed correctly

<u>Lesson:</u> always test edge cases!

Binary representation: Much more to learn (in other CS courses)

How to use binary numbers to represent:

- Nonnegative integers (done)
- Negative integers ("Two's Complement method")
- Real numbers ("Floating Point method")

How to use binary operations to efficiently implement:

- Addition
- Subtraction
- Multiplication
- Division
- Square root
- Power
- . . .

How to use binary numbers and operations to

- Process images
- Process sound
- Store, compress, validate, transmit... data

Lecture 2-2

Application Example: Gambling

the dream

the reality

Gambler's ruin

The game: In each coin flip you either win \$1, or lose \$1.

A gambler starts the game with \$500, with the goal of reaching \$2,500.

What are the chances to reach this goal?

Gambler's ruin

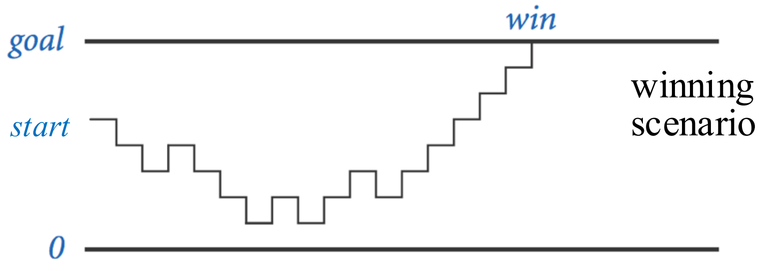
The game: In each coin flip you either win $1, or lose $1.

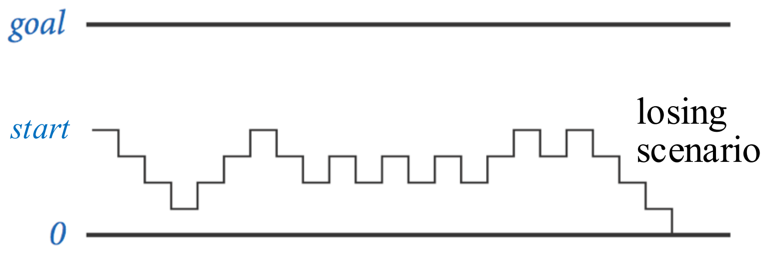
A gambler starts the game with $500, with the goal of reaching $2,500.

What are the chances to reach this goal?

Formally: a gambler starts the game with `cash = start` dollars, and a `goal`;

He plays $1 bets until `cash == 0` (loss) or `cash == goal` (win)





Gambler's ruin

The game: In each coin flip you either win $1, or lose $1.

A gambler starts the game with $500, with the goal of reaching $2,500.

What are the chances to reach this goal?

Formally: a gambler starts the game with `cash = start` dollars, and a `goal`;

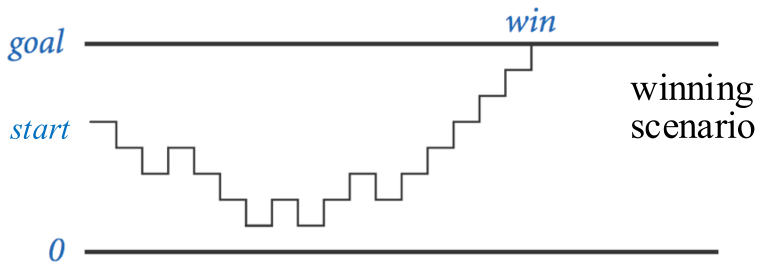
He plays $1 bets until `cash == 0` (loss) or `cash == goal` (win)

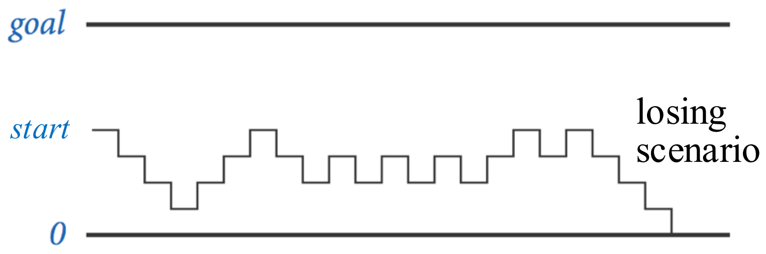
Statistics of interest

- What is the probability of reaching the goal?
- How many flips, on average, until a game ends?

Analysis strategy

- Flip a coin and see what happens
- Repeat many times
- Compute statistics





Monte Carlo simulation:
a method for generating numbers pseudo-randomly, and computing statistics on them;
Widely used in science, engineering, and gaming.

Gambling simulation

```
            start  goal    T
% java Gambler 500 2500 1000
percentage of winning games: 20%
average number of flips in a game: 1037755

% java Gambler 500 2500 1000
percentage of winning games: 21%
average number of flips in a game: 1039952

% java Gambler 500 2500 1000
percentage of winning games: 19%
average number of flips in a game: 966067
```

<u>`T`:</u>

Number of times we play the game (run the simulation);

In each game we
either win (`cash == goal`),
or lose (`cash == 0`)

<u>Note</u>

If `T` is large, the program can take a long time to run (minutes):

We are simulating 1,000 games, each involving an average of a million coin flips.

Gambling simulation

```
public class Gambler {
   public static void main(String[] args) {
      int start  = Integer.parseInt(args[0]);
      int goal   = Integer.parseInt(args[1]);
      int T      = Integer.parseInt(args[2]);
      int nFlips = 0;
      int nWins = 0;
```

start goal T

```
% java Gambler 500 2500 1000
percentage of winning games: 20%
average number of flips in a game: 1037755

% java Gambler 500 2500 1000
percentage of winning games: 21%
average number of flips in a game: 1039952

% java Gambler 500 2500 1000
percentage of winning games: 19%
average number of flips in a game: 966067
```

Gambling simulation

```
public class Gambler {
   public static void main(String[] args) {
      int start  = Integer.parseInt(args[0]);
      int goal   = Integer.parseInt(args[1]);
      int T      = Integer.parseInt(args[2]);
      int nFlips = 0;
      int nWins = 0;
      // repeats the experiment T times
      for (int t = 0; t < T; t++) {
            // runs one experiment
            int cash = start;
            while ((cash > 0) && (cash < goal)) {
                  // flips coin and update cash situation
                  if (Math.random() < 0.5) cash++;
                  else                     cash--;
                  nFlips++;
            }
            if (cash == goal) {
                nWins++;
            }
       } // loops back to start a new game

       System.out.println("percentage of winning games: " + 100 * nWins / T + "%");
       System.out.println("average number of flips in a game: " + nFlips / T);    }
}
```

```
              start   goal   T
% java Gambler 500 2500 1000
percentage of winning games: 20%
average number of flips in a game: 1037755

% java Gambler 500 2500 1000
percentage of winning games: 21%
average number of flips in a game: 1039952

% java Gambler 500 2500 1000
percentage of winning games: 19%
average number of flips in a game: 966067
```

Gambling simulation

```
                  start   goal  T
% java Gambler 500 2500 1000
percentage of winning games: 20%
average number of flips in a game: 1037755

% java Gambler 500 2500 1000
percentage of winning games: 21%
average number of flips in a game: 1039952

% java Gambler 500 2500 1000
percentage of winning games: 19%
average number of flips in a game: 966067
```

Results from probability theory:

Probability of winning = `start` / `goal`

Expected number of flips =

$$\texttt{start} \times (\texttt{goal} - \texttt{start})$$

Observations

- Theory and experiment lead to the same conclusion
- So... why simulate?
  - We use simulation to validate theory, or to help reach the theory
  - We use simulation for *generating pseudo-random events* from a given model
  - Examples: research, training, investing, playing, etc.
- In many cases a theoretical model is not available,
  and simulation is the only way to generate data and compute statistics.

Introduction to Computer Science
Reichman University

Lecture 2-2

Conditional and Iterative Processing
Part II

I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.