Unit 4: Iteration
Nested Loops

Adapted from:

1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum

Nested loops

- **nested loop**: A loop placed inside another loop.

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.print("*");
    }
    System.out.println();    // to end the line
}
```

- Output:

```
**********
**********
**********
**********
**********
```

- The outer loop repeats 5 times; the inner one 10 times.
  - "sets and reps" exercise analogy

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

- Output:

```
*
**
***
****
*****
```

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(i);
    }
    System.out.println();
}
```

- Output:

```
1
22
333
4444
55555
```

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

```
for (int i = 1; i <= 5; i++) {
    for (int j = i; j <= 5; j++) {
        System.out.print(i);
    }
    System.out.println();
}
```

- Output:

```
11111
2222
333
44
5
```

Common errors

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; i <= 10; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

Infinite loops!

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; i++) {
        System.out.print("*");
    }
    System.out.println();
}
```

Infinite loops!

Computational Complexity

Computational complexity of an algorithm is the amount of computational resources(e.g. math operations) needed to perform the algorithm.

For example, how many iterations does a for loop execute?

Some Examples

What's the total number of x++ operations?

```
int x = 0;
for(int i = 0; i < 10; i++){
      x++;
}
for(int j = 1; j <= 15; j++){
      x++;
}
```

Answer: 10 + 15 = 25

Some Examples

What’s the total number of x++ operations?

```
int x = 0;
for(int i = 0; i < 10; i++){
      for(int j = 0; j < 15; j++)
            x++;
}
```

Answer: 10 * 15 = 150

Some Examples

What's the total number of x++ operations **in terms of n**?

```
int x = 0;
for(int i = 0; i < n; i++){
      x++;
}
for(int j = 0; j < n; j++){
      x++;
}
```

Answer: $n + n = 2n$ (linear)

Some Examples

What's the total number of x++ operations **in terms of n**?

```
int x = 0;
for(int i = 0; i < n; i++){
      for(int j = 0; j < n; j++)
            x++;
}
```

Answer: n * n = n^2

Some Examples

What's the total number of operations in **terms of n**?

```
for(int i = 0; i < n; i++){
      for(int j = i; j < n; j++)
            x++;
}
```

Answer:

n + (n-1) + (n-2) + .. + 2 + 1 = n(n+1)/2 (quadratic)

where the equality is obtained from using the summation equation for an arithmetic series.

(For example, 1+2+3+4+…+100=100*101/2)

Exponential Complexity Problems

Algorithms that can be implemented with a polynomial time (linear, quadratic, cubic, etc…) complexity can be executed quickly on a modern processor. (Problems of this type belong to a class called P, Polynomial Time.)

However, there exists important and practical problems for which there exists no known polynomial time algorithm.

- E.g. given a set of integers, find a subset that sums to zero. A brute-force algorithm would try every possible subset. But there are 2^n different subsets. This is an example of an exponential time algorithm. If n is large, even the fastest computers would take too long.

Travelling Salesman(TSP)

Given a set of cities and paths connecting them, find the shortest path that visit each of them exactly once. This famous problem is known as the travelling salesman problem(TSP).

There is no known polynomial time algorithm that solves TSP. Solving TSP can, for example, lead to better transportation and bus routes.

TSP belongs to a class of related problems called NP(Non-deterministic Polynomial Time). None of these problems has a polynomial time solution. But if one does, then so do the rest.

Is P=NP?

- Is P=NP? In other words, can Travelling salesman and the other NP-complete problems be solved in polynomial time? Mathematicians believe that P is not equal to NP. No proof is known.

- This is one of 7 Millenium Problems. The Clay Mathematics Institute has offered a million dollar prize for solving any of them.

- Grigori Perelman solved one of the Millenium Problems, the Poincare Conjecture. He declined the million dollar prize as well as the Fields Medal, the equivalent of the Nobel Prize for Mathematics.
- https://medium.com/@phacks/how-grigori-perelman-solved-one-of-maths-greatest-mystery-89426275cb7

What if P=NP?

What if P=NP? Many very important problems in math, physics and engineering are currently intractable, i.e., solutions take exponential time. If P=NP, then there are efficient algorithms for solving them. This can lead to many advances in science.

But P=NP can have negative consequences.

For example, cryptography relies on certain problems being difficult. Public-key cryptography, a foundation for security applications including financial transactions over the internet, would be vulnerable if one can prove P=NP constructively.

Most mathematicians believe that P is not equal to NP.

Lab 1

Create a new repl on repl.it.

Implement a static void method called `printLadder` which accepts a nonnegative integer n and print a ladder shape of 1's consisting of n steps. For example:

```
printLadder(1);
1


printLadder(4);
1
11
111
1111
```

Lab 1

Implement a static void method called `multiplicationTable` which accepts a positive integer n and print the multiplication table starting at 1 up to and including n.(use tab to separate values "\t")

multiplicationTable(2);

```
1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
```

multiplicationTable(3);

```
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
```

References

1) CPJava Website
2) CPJava Google Classroom
3) CPJava repl.it Classroom
4) Runestone CSAwesome BUSHSCHOOL_CPJAVA Course
5) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp