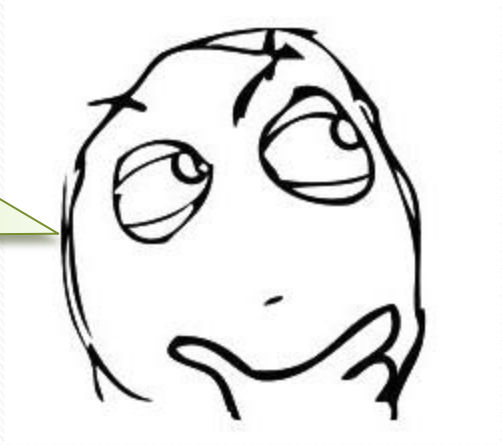
CSCI1530 Computer Principles and Java Programming

Tutorial 7
Problem Solving Technique : Brute Force Searching (Nested-loop and Branching)

Topics

- Problem solving technique : Enumeration
  - Basic idea
  - Two sample problems
  - Class exercise

This technique is also known as "brute-force search" or "exhaustive search".



Preliminaries

- Nested loops
  - For / while loops

```java
int i, j;
for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        System.out.println(i+" "+j);
    }
}
```

```java
i = 0;
while (i < 3) {
    j = 0;
    while (j < 3) {
        System.out.println(i+" "+j);
        j++;
    }
    i++;
}
```

```
Output - JavaApplication1 (run)
run:
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Conditional statements
  - If-(else) statements
  - Boolean operators

Example 1

- There are two positive integers, A and B
- You know that:
  - $A - B \geq 5$
  - $A \times B \leq 6$
- Question: A = ? B = ?

- One possible solution: Try all $1 \leq (A, B) \leq 6$
  - A = 1, B = 1 : First condition violated; try next
  - A = 1, B = 2 : …
- This may not be very clever, but is easy to code

Enumeration

- "Generate and test"
  - Generate all possible solutions
  - Test them one by one
  - If the conditions are satisfied, we have found our answer
  - If not, try the next one

How to generate all possibilities?

How to formulate the conditions?

Example 1

```
public static void main(String[] args) {
    int A, B, no_of_sol = 0;
    for (A = 1; A <= 6; A++) {
        for (B = 1; B <= 6; B++) {
            if ((A-B >= 5) && (A*B <= 6)) {
                System.out.println(A+" "+B);
                no_of_sol++;
            }
        }
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

Generate

Test

Answer: A = 6, B = 1

```
Output - JavaApplication1 (run)
run:
6 1
1 answer(s) found.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Example 1

- You may also use while loop

```
public static void main(String[] args) {
    int A = 1, B, no_of_sol = 0;
    while (A <= 6) {
        B = 1;
        while (B <= 6) {
            if ((A-B >= 5) && (A*B <= 6)) {
                System.out.println(A+" "+B);
                no_of_sol++;
            }
            B++;
        }
        A++;
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

Generate

Test

Example 2

- Five divers, A, B, C, D and E, join a diving competition and are asked to predict the results
  - A: I will be the 3rd and B will be the 2nd
  - B: I will be the 2nd and E will be the 4th
  - C: I will be the 1st and D will be the 2nd
  - D: I will be the 3rd and C will be the 5th
  - E: I will be the 4th and A will be the 1st

- It turns out that the everyone's prediction is only half right (so one guess is correct and the other is wrong)
- Question: What is the real ranking?

Problem taken from *Fundamentals of Programming*, Wenhu Wu

Example 2

- Generate all possibilities
  - Five for loops
- Test for the conditions
  - Take A's prediction as an example
    - ((A == 3) && (B != 2)) || ((A != 3) && (B == 2))
  - Alternatively, use exclusive-OR:
    - (A == 3) ^ (B == 2)

Not all parentheses are necessary, but with them we can see the order more easily.

XOR returns true when exactly one side holds.

- *One more constraint:* two divers cannot share the same rank ("All different")
  - This is a bit harder… well, at least for now

Example 2

```
public static void main(String[] args) {
    int A, B, C, D, E, no_of_sol = 0;
    for (A = 1; A <= 5; A++) {
        for (B = 1; B <= 5; B++) {
            if (B != A) {
                for (C = 1; C <= 5; C++) {
                    if ((C != A) && (C != B)) {
                        for (D = 1; D <= 5; D++) {
                            if ((D != A) && (D != B) && (D != C)) {
                                for (E = 1; E <= 5; E++) {
                                    if ((E != A) && (E != B) && (E != C) && (E != D)) {
                                        if (((A == 3) ^ (B == 2)) && ((B == 2) ^ (E == 4)) &&
                                            ((C == 1) ^ (D == 2)) && ((D == 3) ^ (C == 5)) &&
                                            ((E == 4) ^ (A == 1))) {
                                            System.out.println(A+" "+B+" "+C+" "+D+" "+E);
                                            no_of_sol++;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

Generate

Test

Example 2

```
public static void main(String[] args) {
    int A, B, C, D, E, no_of_sol = 0;
    for (A = 1; A <= 5; A++) {
        for (B = 1; B <= 5; B++) {
            if (B != A) {
                for (C = 1; C <= 5; C++) {
                    if ((C != A) && (C != B)) {
                        for (D = 1; D <= 5; D++) {
                            if ((D != A) && (D != B) && (D != C)) {
                                for (E = 1; E <= 5; E++) {
                                    if ((E != A) && (E != B) && (E != C) && (E != D)) {
                                        if (((A == 3) ^ (B == 2)) && ((B == 2) ^ (E == 4)) &&
                                            ((C == 1) ^ (D == 2)) && ((D == 3) ^ (C == 5)) &&
                                            ((E == 4) ^ (A == 1))) {
                                            System.out.println(A+" "+B+" "+C+" "+D+" "+E);
                                            no_of_sol++;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

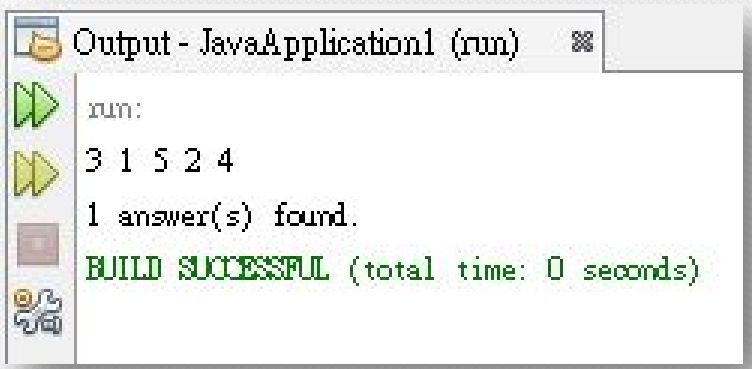
"All different"

Alternatively,
set the constraint using
sum and product, e.g.:
A+B+C+D+E == 15 &&
A*B*C*D*E == 120

Example 2

- Answer:
  - A: 3rd
  - B: 1st
  - C: 5th
  - D: 2nd
  - E: 4th

A: **I will be the 3rd** and B will be the 2nd
B: I will be the 2nd and **E will be the 4th**
C: I will be the 1st and **D will be the 2nd**
D: I will be the 3rd and **C will be the 5th**
E: **I will be the 4th** and A will be the 1st



Example 2

- Don't want the long conditional statement?
  - Use nested if
  - Use a temporary Boolean variable
  - Use a failure count

```
if ((A == 3) ^ (B == 2)) {
    if ((B == 2) ^ (E == 4)) {
        if ((C == 1) ^ (D == 2)) {
            if ((D == 3) ^ (C == 5)) {
                if ((E == 4) ^ (A == 1)) {
```

```
boolean flag = ((A == 3) ^ (B == 2));
flag &= ((B == 2) ^ (E == 4));
flag &= ((C == 1) ^ (D == 2));
flag &= ((D == 3) ^ (C == 5));
flag &= ((E == 4) ^ (A == 1));
if (flag) {
```

- Can you rewrite the code using while loop?

Class exercise

- After a nuclear power station accident, the authorities received four different accounts / reports of the explosion order:
  - Reporter A: “Plant 3 was the second and Plant 1 was the third.”
  - Witness B: “Plant 1 was the second and Plant 3 was the fourth.”
  - Witness C: “Plant 2 was the second and Plant 4 was the third.”
  - Rescuer D: “Plant 3 was the fourth and Plant 2 was the first.”
- However, everyone had only got one statement correct (and thus the other statement was wrong)

Problem designed by Michael

Class exercise

- Write a Java program to determine the full picture of the accident
- Your program should exhaustively try all the ordering combinations and output like this:

```
Explosion order of Plant 1: 3
Explosion order of Plant 2: …
…
```

- There is only one correct answer