Project 4
Search Based Testing
Task

Generate test inputs that achieve full branch coverage.
public class Example {
    int a;
    static void m(int x, int y) {
        if (x + y > 10) {
            a = 1;
        }
    }
}
Genetic Algorithm

- See lecture on search-based testing
Start

- Choose target to cover.
- Generate start population.
Example

Target to cover:
Condition in if evaluates to false

Start Population:
- 13,4
- 7,8
- 2,10
Compute Fitness

• Fitness:
  
  approach level + normalized branch distance
Approach Level

- Number of control dependent edges between goal and chosen path
- Approach = Number of dependent nodes - number of executed nodes
public class Example {
    int a;
    static void m(int x, int y) {
        if (x + y > 10) {
            a = 1;
        }
    }
}

Inputs:
13,4  7,8  2,10

Approach Level: 0
Distance

Branch Distance

- Critical branch = branch where control flow diverged from reaching target
- Distance to branch = distance to predicate being true / false
- Distance metric for logical formulas
- E.g. distance from true - false = 1

Table 1: Distance metrics

<table>
<thead>
<tr>
<th>Construct</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = b$</td>
<td>$a - b = 0 ? 0 : \text{abs}(a - b) + k$</td>
</tr>
<tr>
<td>$a \neq b$</td>
<td>$a - b \neq 0 ? 0 : k$</td>
</tr>
<tr>
<td>$a &lt; b$</td>
<td>$a - b &lt; 0 ? 0 : (a - b) + k$</td>
</tr>
<tr>
<td>$a \leq b$</td>
<td>$a - b \leq 0 ? 0 : (a - b) + k$</td>
</tr>
<tr>
<td>$a &gt; b$</td>
<td>$b - a &lt; 0 ? 0 : (b - a) + k$</td>
</tr>
<tr>
<td>$a \geq b$</td>
<td>$b - a \leq 0 ? 0 : (b - a) + k$</td>
</tr>
<tr>
<td>boolean</td>
<td>$\text{true}? 0 : k$</td>
</tr>
<tr>
<td>$a &amp;&amp; b$</td>
<td>$\text{distance}(a) + \text{distance}(b)$</td>
</tr>
<tr>
<td>$a | b$</td>
<td>$\min(\text{distance}(a), \text{distance}(b))$</td>
</tr>
<tr>
<td>$!a$</td>
<td>Move inward and propagate, e.g. $!(a &gt; b)$ becomes $a \leq b$ and $!(a &amp;&amp; b)$ becomes $!a | !b$.</td>
</tr>
</tbody>
</table>
public class Example {
    int a;
    static void m(int x, int y) {
        if (x + y > 10) {
            a = 1;
        }
    }
}
Instrumentation

```java
static void m(int x, int y) {
    traceDist(10 - (x+y) < 0 ? 0 : (10-(x+y)) + k, 0);
    traceDist((x+y) - 10 <= 0 ? 0 : (x+y-10) + k, 1);
    if (x + y > 10) {
        a = 1;
    }
}
```
<table>
<thead>
<tr>
<th>Inputs:</th>
<th>Fitness:</th>
</tr>
</thead>
<tbody>
<tr>
<td>13,4</td>
<td>0 + 8/9</td>
</tr>
<tr>
<td>7,8</td>
<td>0+6/7</td>
</tr>
<tr>
<td>2,10</td>
<td>0+3/4</td>
</tr>
</tbody>
</table>
Elitism

• Keep best chromosomes. For next generation.

• In example: 2,10
Tournament Selection

- \( N = \) Tournament size
- Select \( N \) individuals randomly
- Best of the \( N \) individuals is selected
- Tournament size defines selective pressure
- A worse individual can win with a given probability
Example

Tournament Size: 2

13,4
0 + 8/9

2,10
0+3/4
Crossover

3, 8

6, 10

0 0 1 0 1 0 1 0

0 1 1 1 1 0 0 0
Mutation

2, 2

0 0 1 0 0 0 1 0
Stopping condition

• Maximum number of iterations reached.
• Target is covered. (fitness is 0)
• Write out generated inputs to csv-file.
• Else repeat the steps (elitism, crossover, mutation).