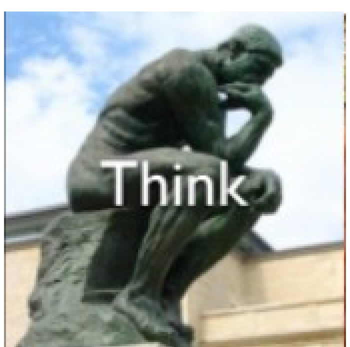
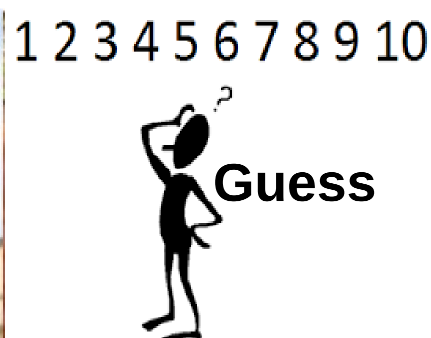
Topics for this Lecture

- Automatic Test Generation Approaches
  - ~~Random Testing~~ **(Guess)**
  - Search Based Software Testing √√ **(Search)**
  - Constraint-Based(symbolic execution) Testing **(Deduce)**





Oregon State University

Problem Definition

```
float divide(float x, float y){
  if (y == 0) {
   // Example handling of this error. Writing a message to stderr, and exiting with failure.
      fprintf(stderr, "Division by zero! Aborting...\n");
      return EXIT_FAILURE; /* indicate failure.*/
   }
      return x / y;
}
```

Oregon State University

Problem Definition

```
double divide(float x, float y){
  if (y == 0) {
   // Example handling of this error. Writing a message to stderr, and exiting with failure.
     fprintf(stderr, "Division by zero! Aborting...\n");
     return EXIT_FAILURE;/* indicate failure.*/
   }
     return x / y;
}
```

Test Case#1	Test Case #2

void testcase1(){
  float r=**divide**(3.0,2.0);
  assert(r == **1.5**);
}
	void testcase2(){
	  float r=**divide**(0.0,0.0);
	  assert (r == **EXIT_FAILURE**);
	  exit(EXIT_FAILURE); /* indicate failure.*/
	}

Oregon State University

Problems

1. How to generate the input data?

2. How to generate the assertions and detect if there is a bug?

We can manually generate some inputs. (Think)

Test Case#1	Test Case #2

void testcase1(){
  float r=**divide**(3.0,2.0);
  assert(r == **1.5**);
}
	void testcase2(){
	  float r=**divide**(0.0,0.0);
	  assert (r == **EXIT_FAILURE**);
	  exit(EXIT_FAILURE); /* indicate failure.*/
	}

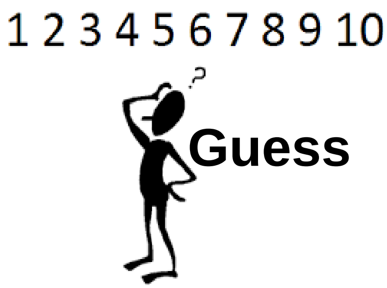
Oregon State University

Random Testing (RT)

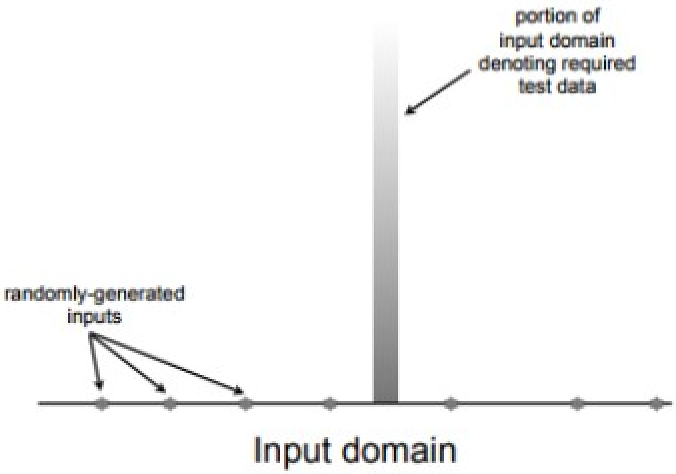
1. Write code to randomly choose a method

2.Generate random values for each parameter for the chosen method.

3.Execute/invoke the method.



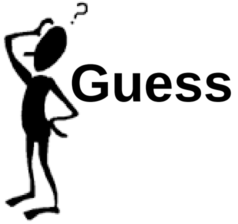
- **Problems:**

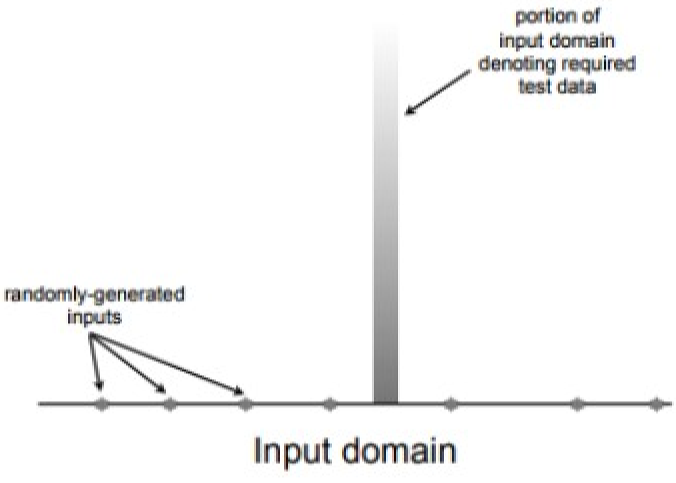


P. McMinn, "Search-based software testing: Past, present and future," 2011

Can we be **smarter** than Random Testing (RT)?

1 2 3 4 5 6 7 8 9 10





P. McMinn, "Search-based software testing: Past, present and future," 2011

Oregon State University

Search Based Software Testing (SBST)

- The problem of test data generation is **converted** to a optimization/search problem, and search algorithms such as Hill Climbing (**HC)** or Genetic Algorithm **(GA)** are used to find the best test data.

- What to optimize? e.g., code coverage (i.e., branch coverage)

Search Based Software Testing (SBST)

- There are three requirements that need to be fulfilled in order to apply a search optimization technique.

1. **Search Space**: The set of all possible inputs for a program under test. e.g., divide (float x, float y)

2. **Search operators**. How to modify the tests to explore the search space. e.g., remove a method call, add a random method call, modify a parameter (e.g., + or -1 for integer values), etc.

3. **Fitness Function** (**FF**). Evaluate how good a solution (i.e., test case) is and guides the search toward better solutions.

Oregon State University

Fitness Function (Branch Distance)

- **Fitness Function**: it **estimates** how **close** a candidate input (i.e., **test case**) is to **satisfy** a **coverage goal**. A common FF is Branch Distance.

- **Branch Distance:** how **near** the input was to **executing** the **required** branch

- Assume you want to solve

  if (y==0){/* this our target branch */}

- You have test values 1 and 100

- Neither of them solve the y==0. In other word, both values do not cover the target branch (i.e., the true if statement) because only 0 solves the constraint of the target branch .

- But, 1 is closer than 100 to 0

Oregon State University

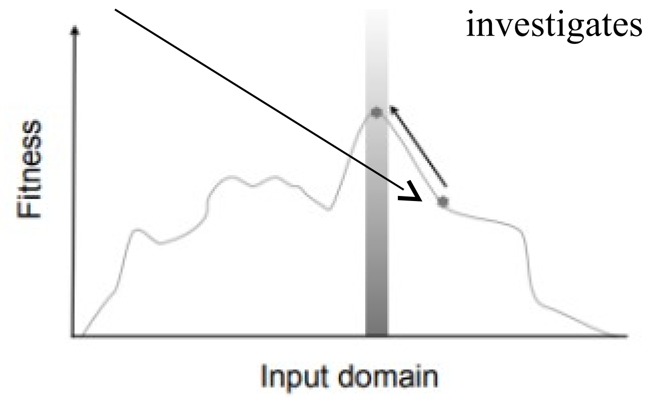
Local Search: Hill Climbing (HC)

- **local search algorithms aim to improve one individual by exploring its neighbors.**

1- Hill Climbing (HC) usually starts with a random input value

2- Then it considers the set of near neighbors to this input value

3- If a fitter neighbor is found, HC moves to it and again it investigates its neighbors



(space of all possible solutions)

P. McMinn, "Search-based software testing: Past, present and future," 2011

Oregon State University

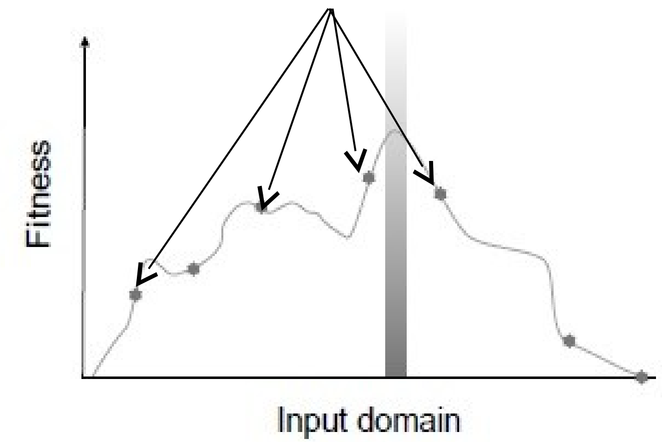
HC example

- Assume that you want to solve
  - if (**y**==0){/* this our target branch */}
- Starting with random values **y**=10
- The **branch distance (BD)** is |**y**-0| = |10-0|=10
- Neighborhood +-1 : **y**=9 and **y**=11
  - The **BD** is |**y**-0| = |9-0| and |11-0| which is 9 and 11
  - 9 is closer to 0 → moving to **y**=9 (new value), and discard 11
- Repeat for **y**=9, i.e., +-1: **y**=8 and **y**=10
  - The **BD** is |**y**-0| = |8-0| and |10-0| which is 8 and 10
  - 8 is closer to 0 → moving to **y**=8 (new value) and discard 10
- Repeat until the solution y=0

Oregon State University

Genetic Algorithm (Global Search)

- **Global** search, sampling **many points** in the search space at once
- **Global** search is more effective than **local** search, but less efficient, as it is more costly.
- The most commonly applied global search algorithms is a ***Genetic Algorithm*** (GA).
- A GA tries to imitate the natural processes of evolution
- Genetic Algorithms are inspired by Darwinian evolution and the concept of survival of the fittest.
- **Evolution** is change in the heritable characteristics of biological populations over successive generations

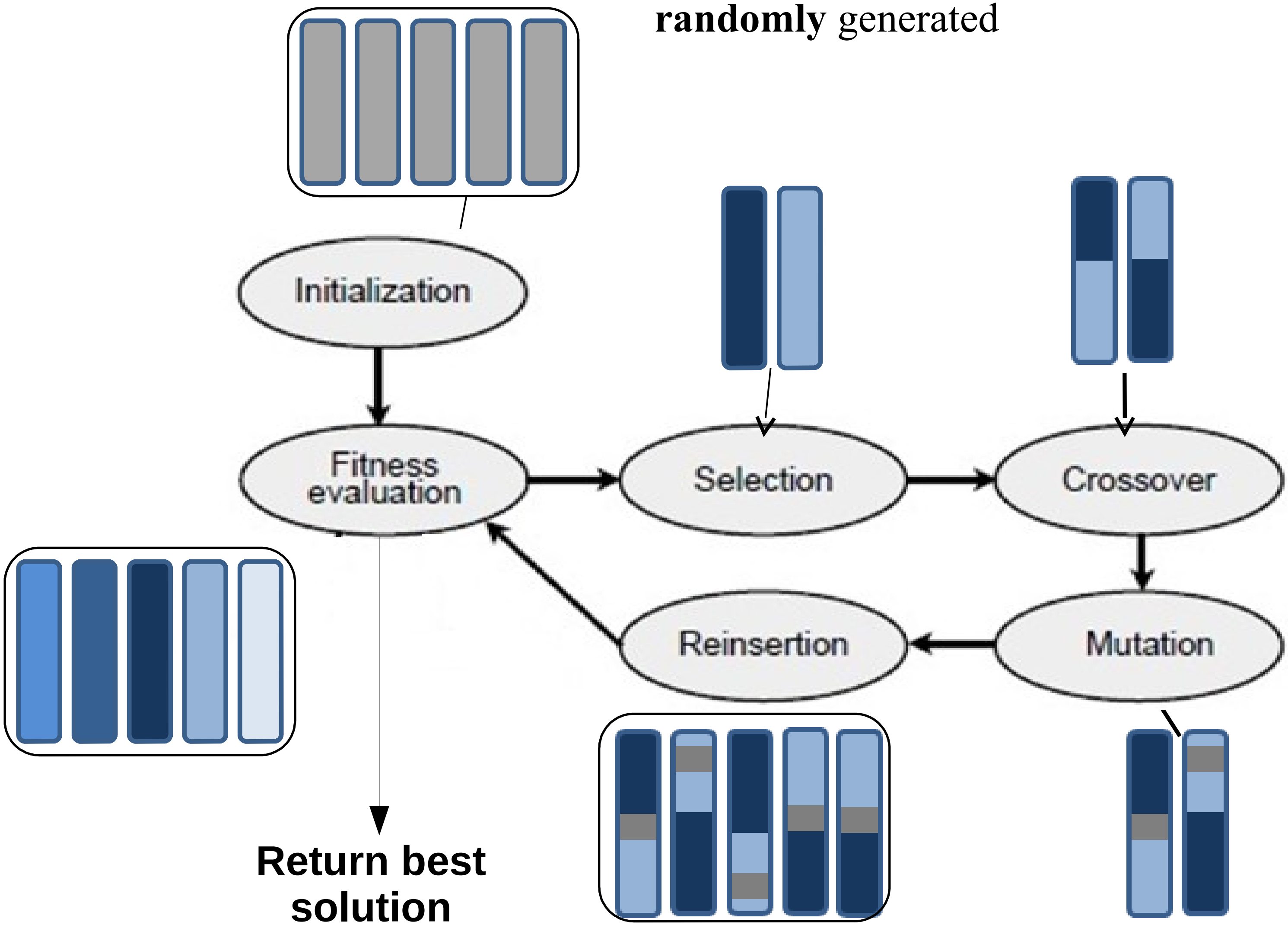


P. McMinn, "Search-based software testing: Past, present and future," 2011

Oregon State University

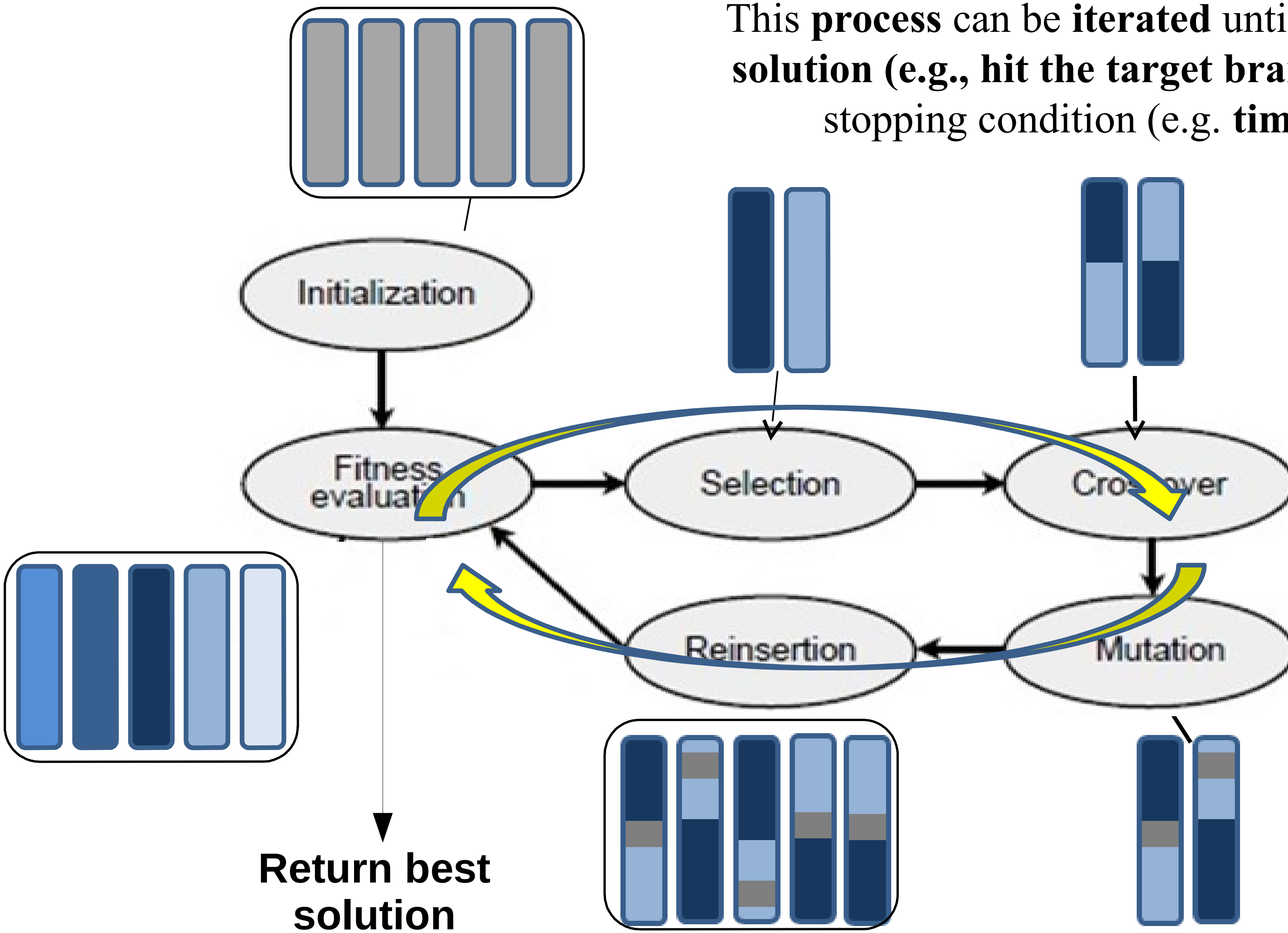
Global Search (Genetic Algorithm (GA))

- **Initial** population **(i.e., values/test cases)** is **randomly** generated



Global Search (Genetic Algorithm (GA))

This **process** can be **iterated** until either an **optimal solution (e.g., hit the target branch)** is found, or a stopping condition (e.g. **timeout**) holds.



GA vs HC

- Assume that you want to solve
  if (x==3 && x*y=72){/* this our target branch */}
- We have two constraints to solve which are x=3 and x*y=72
- Lets say that you start point is x=9 and y=8
- +1 or -1 on x or y would not improve your solution and HC get stuck.
- Having a population (more than one solution), GA can overcome these local problems.

Oregon State University

Some automatic SBST input generation tools for Java

- **Java**
  - **EvoSuite** tool:(http://www.evosuite.org/)

  - **AVMf** tool: (http://avmframework.org/)

- **C Language**
  - **AUSTIN** tool (https://github.com/kiranlak/austin-sbst)

- They are freely available for use

References:

Noraini, Mohd Razali, and John Geraghty. "Genetic algorithm performance with different selection strategies in solving TSP." (2011).

https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)