Code Hunt Hint System

Daniel Perelman\textsuperscript{1}

Sumit Gulwani\textsuperscript{2} \hspace{1cm} Dan Grossman\textsuperscript{1} \hspace{1cm} Judith Bishop\textsuperscript{2}

\textsuperscript{1}University of Washington

\textsuperscript{2}Microsoft Research

February 9, 2015
Code Hunt programming game
Code Hunt programming game

SELECT SECTOR

00	TRAINING
01	ARITHMETIC
02	LOOPS
03	LOOPS 2
04	CONDITIONALS
05	CONDITIONALS 2
06	STRINGS
07	STRINGS 2
08	NESTED LOOPS
09	1D ARRAYS
10	JAGGED ARRAYS
11	ARRAYS 2
12	SEARCH SORT
13	CRYPTERS
14	PUZZLES
Code Hunt programming game
Code Hunt programming game

Discover the arithmetic operation applied to 'x'.

```java
public class Program {
    public static int Puzzle(int x) {
        return 0;
    }
}
```
Code Hunt programming game

Discover the arithmetic operation applied to 'x'.

```java
public class Program {
    public static int Puzzle(int x) {
        return 0;
    }
}
```

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>EXPECTED RESULT</th>
<th>YOUR RESULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>Mismatch</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Mismatch</td>
</tr>
</tbody>
</table>

LEVEL: 00.02
Discover the arithmetic operation applied to 'x'.

```java
public class Program {
    public static int Puzzle(int x) {
        return 1;
    }
}
```
Code Hunt programming game

Discover the arithmetic operation applied to 'x'.

```java
public class Program {
    public static int Puzzle(int x) {
        if (x == -1) {
            return 0;
        } else if (x == 0) {
            return 1;
        } else if (x == 1) {
            return 2;
        } else {
            return 0;
        }
    }
}
```

<table>
<thead>
<tr>
<th>X</th>
<th>EXPECTED RESULT</th>
<th>YOUR RESULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>Mismatch</td>
</tr>
</tbody>
</table>
Discover the arithmetic operation applied to 'x'.

```java
public class Program {
    public static int Puzzle(int x) {
        return x+1;
    }
}
```
You repaired and captured the code fragment.

**SKILL RATING:**

you wrote elegant code!

**TOTAL SCORE:** 6

KEEP TRYING

NEXT

LEVEL SELECT
More difficult level
More difficult level

```java
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        return lowerBound * upperBound;
    }
}
```
The image shows a coding challenge from the website "Code Hunt," titled "More difficult level." The current code snippet is:

```java
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        return 40320;
    }
}
```

The table lists various test cases and outputs:

<table>
<thead>
<tr>
<th>LOWERBOUND</th>
<th>UPPERBOUND</th>
<th>EXPECTED RESULT</th>
<th>YOUR RESULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>40320</td>
<td>40320</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>518918400</td>
<td>40320</td>
<td>Mismatch</td>
</tr>
</tbody>
</table>

A note is provided: "The expression 40320 is rarely used to solve this level."
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        return lowerBound * upperBound;
    }
}

You may find a loop useful on this level.
More difficult level

```java
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        int r = 1;
        for(int i = lowerBound; i < upperBound; i++)
            r *= i;
        return r;
    }
}
```

<table>
<thead>
<tr>
<th>LOWERBOUND</th>
<th>UPPERBOUND</th>
<th>EXPECTED RESULT</th>
<th>YOUR RESULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>40320</td>
<td>5040</td>
<td>Mismatch</td>
</tr>
<tr>
<td>16</td>
<td>22</td>
<td>859541760</td>
<td>39070080</td>
<td>Mismatch</td>
</tr>
</tbody>
</table>

Looking good. Look at line 4 to capture the code.
More difficult level

```java
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        int r = 1;
        for(int i = lowerBound; i < upperBound; i++)
        r *= i;
        return r;
    }
}
```
More difficult level

```
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        int r = 1;
        for (int i = lowerBound; i <= upperBound; i++)
            r *= i;
        return r;
    }
}
```
Kinds of hints

- Line hints: “Look at line 4 to capture the code.”
- Positive recommendation hints: “You may find a loop useful on this level.”
- Negative recommendation hints: “The expression 40320 is rarely used to solve this level.”
Line hints

- Generated using program synthesis
- Solve level based on user’s attempt, return lines that differ
Line hints

- Generated using program synthesis
- Solve level based on user’s attempt, return lines that differ
- Can only give such hints when the user is close to a solution
  - Otherwise, the hint would have to say “change everything”. 
Are line hints useful?

user needs a hint

user almost solved level
Are line hints useful? Yes, a little.

- Compute hints, only show to randomly selected users.

Summary: majority of users don't need help, but a statistically significant minority do go faster. But the help only helped users complete the level faster; almost all of them completed it either way.
Are line hints useful? Yes, a little.

- Compute hints, only show to randomly selected users.

<table>
<thead>
<tr>
<th></th>
<th>show hint</th>
<th>hide hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved within 1 attempt</td>
<td>83 %</td>
<td>57%</td>
</tr>
<tr>
<td>solved eventually</td>
<td>98.4%</td>
<td>98%</td>
</tr>
</tbody>
</table>

- Summary: majority of users don’t need help, but a statistically significant minority do go faster.
Are line hints useful? Yes, a little.

- Compute hints, only show to randomly selected users.

<table>
<thead>
<tr>
<th></th>
<th>show hint</th>
<th>hide hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved within 1 attempt</td>
<td>83 %</td>
<td>57%</td>
</tr>
<tr>
<td>solved eventually</td>
<td>98.4%</td>
<td>98%</td>
</tr>
</tbody>
</table>

- Summary: majority of users don’t need help, but a statistically significant minority do go faster.

- But the help only helped users complete the level faster; almost all of them completed it either way.
How are line hints generated?

- Requires an aside on TDS synthesizer.
Code Hunt workflow

- User
- Test cases
- User modifies program
- Program
- Done?
- Generate new test case
- Win game
Synthesizer plays Code Hunt

- intro CS DSL
- test cases
  - sequence
  - synthesize
  - change
  - program
- done?
  - yes
  - win game
- no
  - generate new test case
Comparison to human players for entire levels

Completion times comparable to human players... use for hints?
Code Hunt line hint workflow

Diagram:
- User
- Spec
- Synthesize change
- User program
- Program

Legend:
- Code Hunt line hint workflow
- User
- Spec
- Synthesize change
- User program
- Program
Code Hunt line hint workflow

1. User program
2. Program
3. Generate hint
4. Done?
5. Yes
6. No
7. Test cases sequence
8. Synthesize change
9. Generate new test case (Pex)

Flows:
- User program -> Program
- Program -> Generate hint
- Done? yes -> Generate hint
- Generate new test case -> Pex
- Pex -> Test cases sequence
- Test cases sequence -> Synthesize change
- Synthesize change -> Program
- Program -> Done?

Flow options:
- Generate new test case (Pex)
Code Hunt line hint workflow

1. DSL for puzzle
2. Generate test cases
3. Test cases sequence
4. Synthesize change
5. Program
6. User program
7. Generate hint
8. Determine if done?
   - Yes: End
   - No: Generate new test case

Diagram:
- Pex logo
- Flowchart showing the steps and decision points.
Modifying program

- Replace single subexpression covered by failing test cases

- **Where** to modify program

- **What** to replace with
Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        int r = 1;
        for(int i = lowerBound; i < upperBound; i++)
            r *= i;
        return r;
    }
}
Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2    public static int Puzzle(int lowerBound, int upperBound) {
3        int r = 1;
4        for(int i = lowerBound; i < upperBound; i++)
5            r *= i;
6        return r;
7    }
8}

Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2   public static int Puzzle(int lowerBound, int upperBound) {
3       int r = 1;
4       for(int i = lowerBound; i < upperBound; i++)
5           r *= i;
6       return r;
7   }
8 }

38 / 65
Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2    public static int Puzzle(int lowerBound, int upperBound) {
3        int r = 1;
4        for(int i = lowerBound; i < upperBound; i++)
5            r *= i;
6        return r;
7    }
8 }

Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)

```java
public class Program {
    public static int Puzzle(int lowerBound, int upperBound) {
        int r = 1;
        for(int i = lowerBound; i < upperBound; i++)
            r *= i;
        return r;
    }
}
```
Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }
// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2    public static int Puzzle(int lowerBound, int upperBound) {
3        int r = 1;
4        for(int i = lowerBound; i < upperBound; i++)
5            r *= i;
6        return r;
7    }
8 }

Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

44 / 65
Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1  public class Program {
2       public static int Puzzle(int lowerBound,
3               int upperBound) {
4           int r = 1;
5           for(int i = lowerBound; i < upperBound; i++)
6               r *= i;
7           return r;
8       }
9  }
10
Where to modify: all subexpressions

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

What to put there

- DSL defines space of expressions
  - Obtained from other users’ solutions

- Prefer expressions found in user’s attempt
// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

Complication: multiple possible hints

```
// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = upperBound;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }
```
Complication: multiple possible hints

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i < upperBound; i++)
5             r *= i;
6         return r * upperBound;
7     }
8 }
Complication: multiple possible hints

// Puzzle(1, 8) == 40320 (not 5040)
// Puzzle(16, 22) == 859541760 (not 39070080)
1 public class Program {
2     public static int Puzzle(int lowerBound, int upperBound) {
3         int r = 1;
4         for(int i = lowerBound; i <= upperBound; i++)
5             r *= i;
6         return r;
7     }
8 }

Solution: choose smallest change

- Get multiple solutions from synthesizer.
- Compute edit distance in characters for each and take the minimum.
Solution: choose smallest change

- Get multiple solutions from synthesizer.
- Compute edit distance in characters for each and take the minimum.
- Downside: can’t return a hint as soon as the first one is computed.
Complication: single line programs

- Single line attempts are common, especially on earlier levels.
- User could add newlines to get more detail.
Solution: more detail than just line number

Discover the arithmetic operation applied to 'x'.

```java
public class Program {
    public static int Puzzle(int x) {
        return x + 11;
    }
}
```

<table>
<thead>
<tr>
<th>X</th>
<th>EXPECTED RESULT</th>
<th>YOUR RESULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>@</td>
<td>Getting close. Look at numbers on line 4 to capture the code.</td>
<td>11</td>
<td>Mismatch</td>
</tr>
</tbody>
</table>
Solution: more detail than just line number

- If line hint covers entire program, give kind of expression like “number” or “variable”.
Recommendation hints

- Positive recommendation hints:
  “You may find a loop useful on this level.”

- Negative recommendation hints:
  “The expression 40320 is rarely used to solve this level.”
How to generate recommendation hints

- Use data!
- Have lots of attempts for which we know both if they are correct and which concepts (e.g. “loops” or “the expression 40320”) they contain.
How to generate recommendation hints

- For every concept, count users that use it by whether it appears in a solution.
- If often tried but does not end up in solutions, warn users of that concept away from it.
- If often appears in solutions, suggest it to users if no other hint is available.
Evaluation

- Ran experiment where hints were disabled for some users.
  - “Always”: always show hints
  - “Never”: never show hints
  - “Sometimes”: give hints on some levels but not others
- Will show data on one-day period after user started playing
Effect on number of levels won

Always hint
Sometimes hint
Never hint
Effect on how long users play

- Always hint
- Sometimes hint
- Never hint

Prop. of users vs. # of hours played
Hint kind by edit distance

![Histogram showing the number of attempts for different hint kinds and edit distances. The x-axis represents the edit distance categories (<30, 30-39, 40-49, ..., >100), and the y-axis represents the number of attempts. The bars are color-coded: purple for None, blue for helpful, yellow for unhelpful, red for tds, and grey for tds-wholeMethod.](image)
Hint kind by edit distance
Questions
Backup slides
The Greek (for recommendation hints)

Constants: \( a = 10, z = 1, \tau = 0.75 \)

Adjusted counts: \( c' = c + a, \bar{c}' = \bar{c} + a, n' = c' + \bar{c}' \)

Estimated probability: \( p = \frac{c'}{n'} \)

Standard deviation: \( \sigma = \sqrt{\frac{p(1-p)}{n'}} \)

Check for significance: \( p - z\sigma > \tau \)