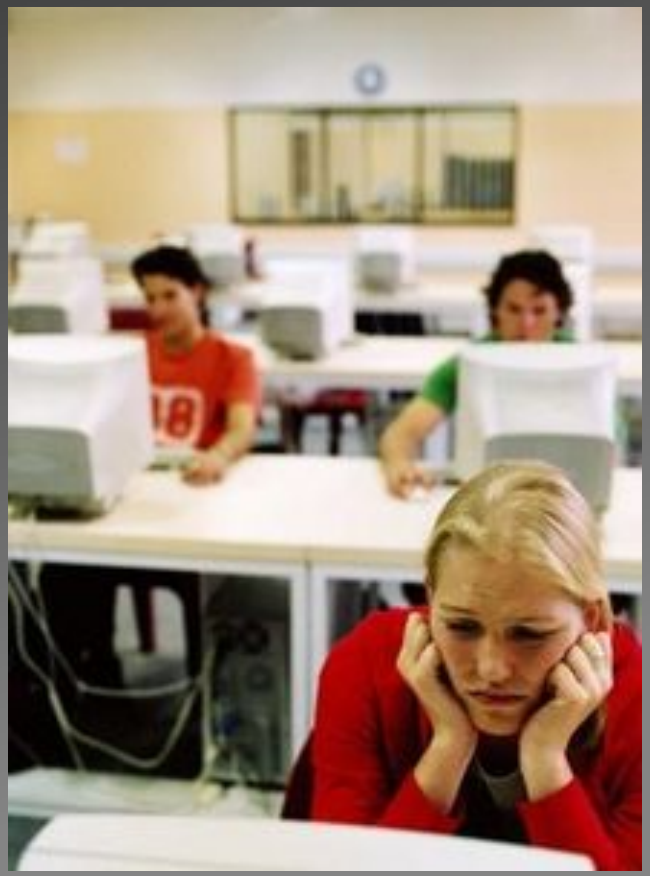
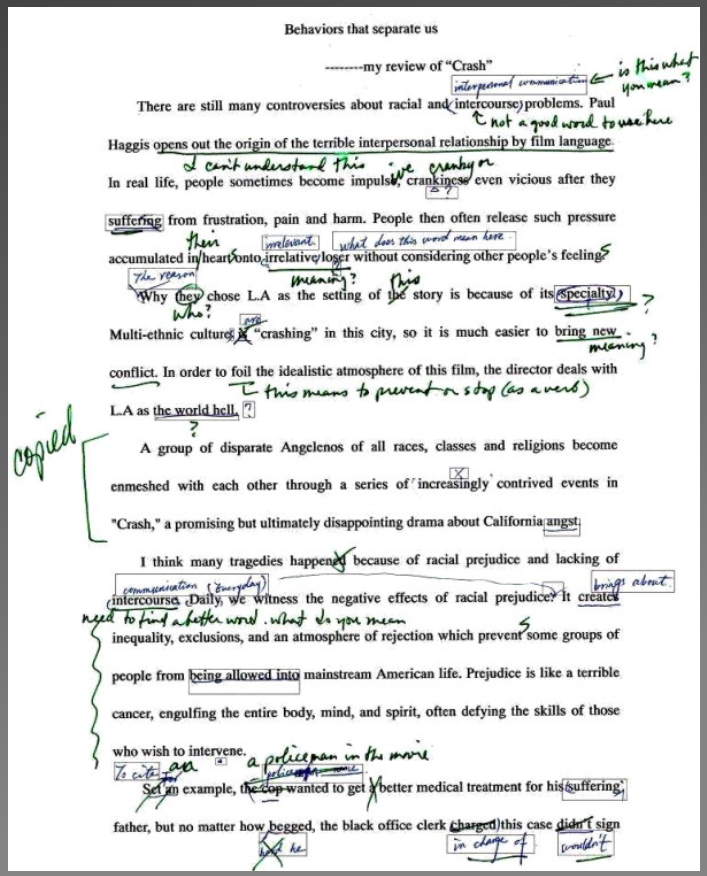
Automatic Generation of Programming Feedback: A Data-Driven Approach

Kelly Rivers and Ken Koedinger

Programming is Hard



The Importance of Feedback



Behaviors that separate us

--------my review of "Crash"

There are still many controversies about racial and intercourse problems. Paul Haggis opens out the origin of the terrible interpersonal relationship by film language. In real life, people sometimes become impulsive, crankiness even vicious after they suffering from frustration, pain and harm. People then often release such pressure accumulated in heart onto irrelative loser without considering other people's feeling. Why they chose L.A as the setting of the story is because of its specialty. Multi-ethnic cultures, "crashing" in this city, so it is much easier to bring new conflict. In order to foil the idealistic atmosphere of this film, the director deals with L.A as the world hell.

A group of disparate Angelenos of all races, classes and religions become enmeshed with each other through a series of increasingly contrived events in "Crash," a promising but ultimately disappointing drama about California angst.

I think many tragedies happened because of racial prejudice and lacking of intercourse. Daily, we witness the negative effects of racial prejudice. It created inequality, exclusions, and an atmosphere of rejection which prevent some groups of people from being allowed into mainstream American life. Prejudice is like a terrible cancer, engulfing the entire body, mind, and spirit, often defying the skills of those who wish to intervene.

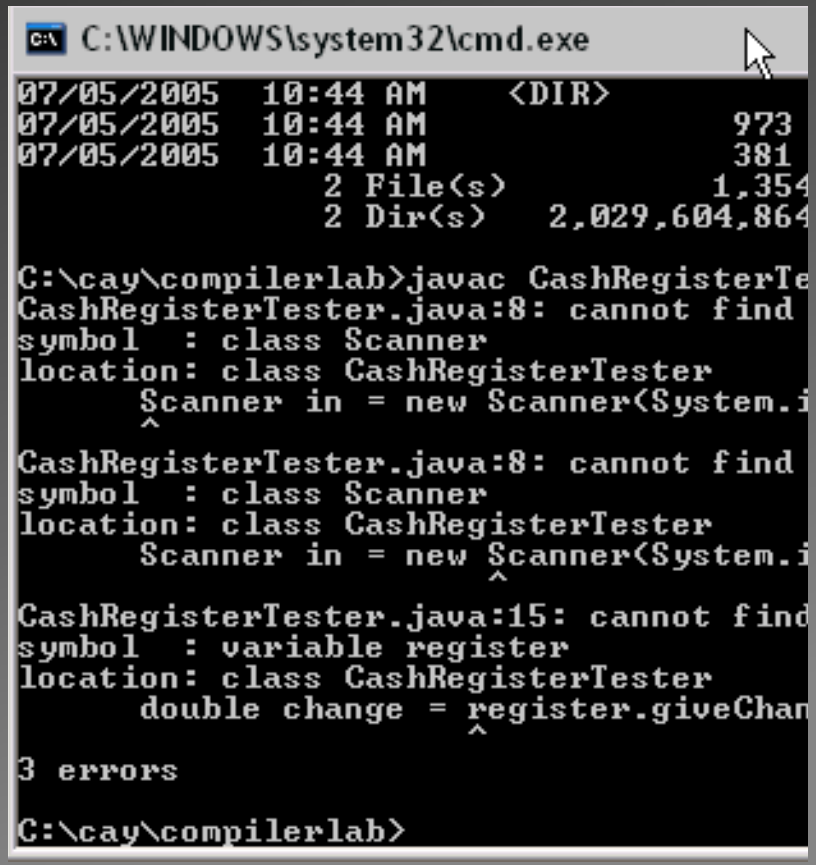
Set an example, the cop wanted to get better medical treatment for his suffering father, but no matter how begged, the black office clerk charged this case didn't sign

Corbett & Anderson, 1991

But This Takes Time!

Research Question:
Can We Make it Automatically?

What We Currently Have



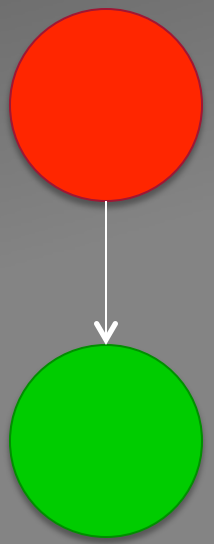
And more…

- Knowledge Modeling
  - Syntax Patterns
  - Plans/Templates/Clichés
  - Error models (Singh et al, 2013)

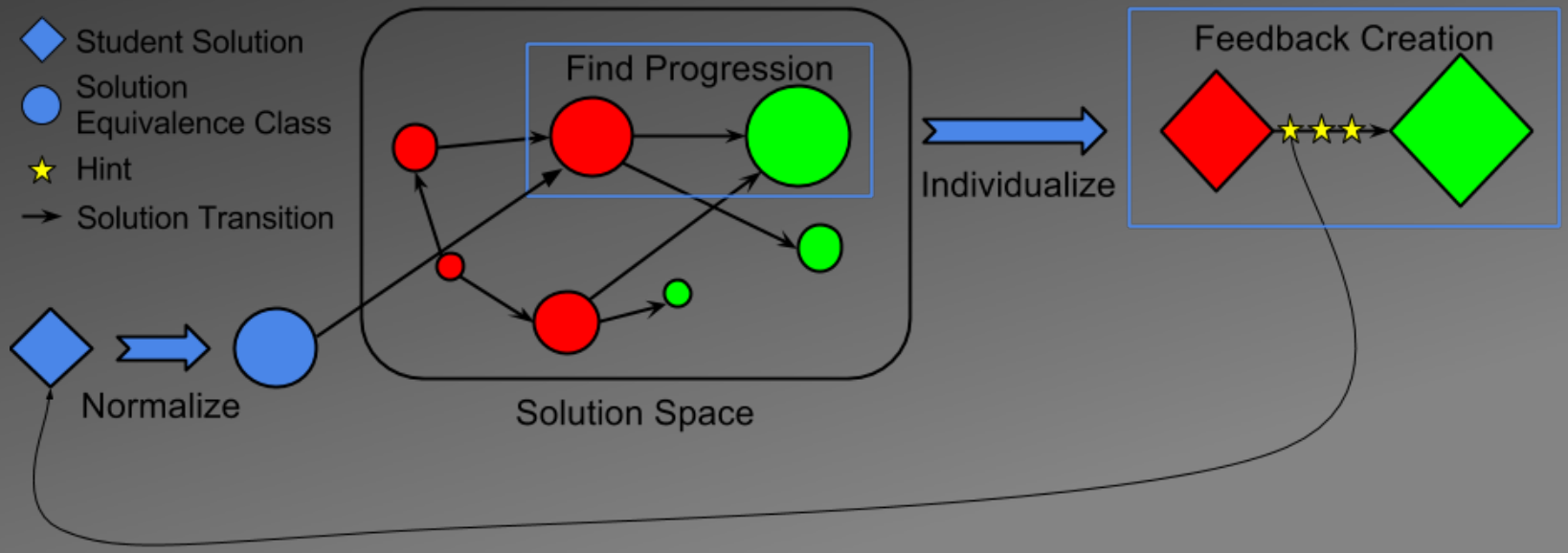
The Solution Space

Each **node** is a solution state that a student might reach, incorrect or correct

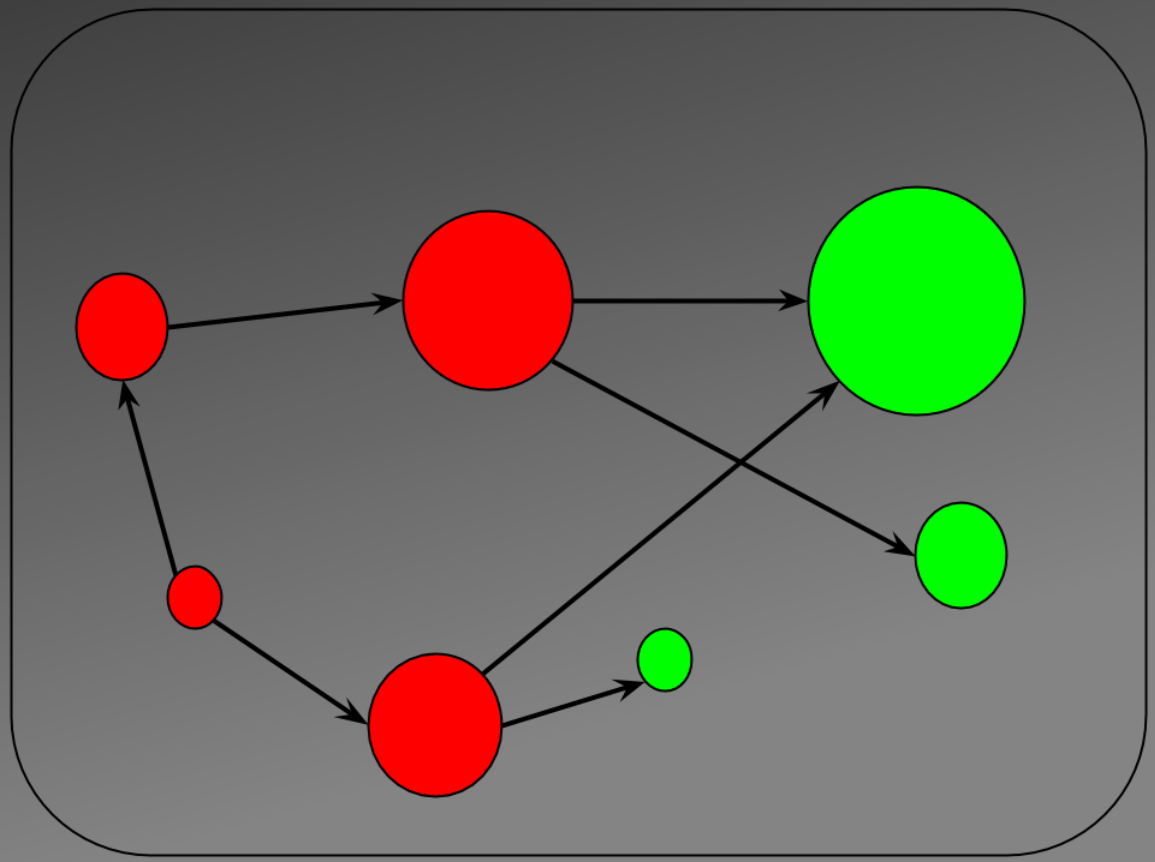
Each **edge** is the transition between the first state and the second, the edits made to the solution

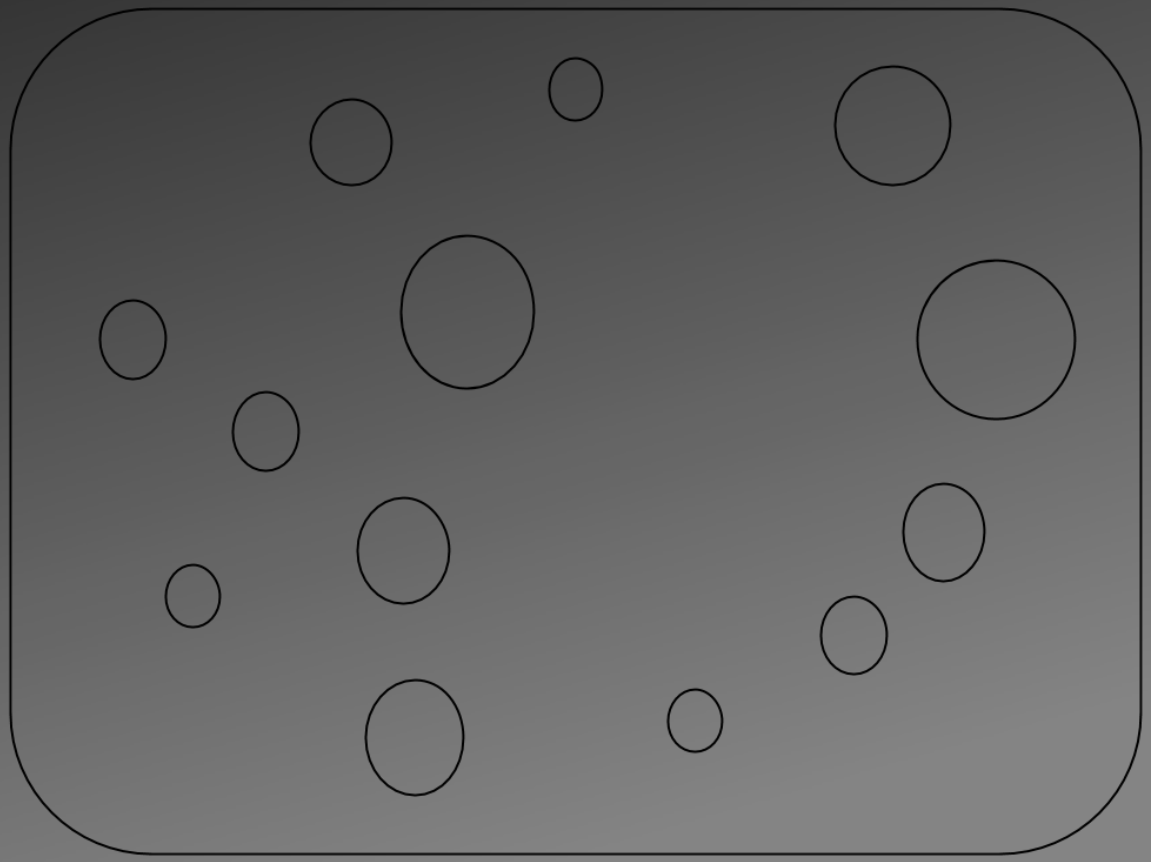


Our Approach

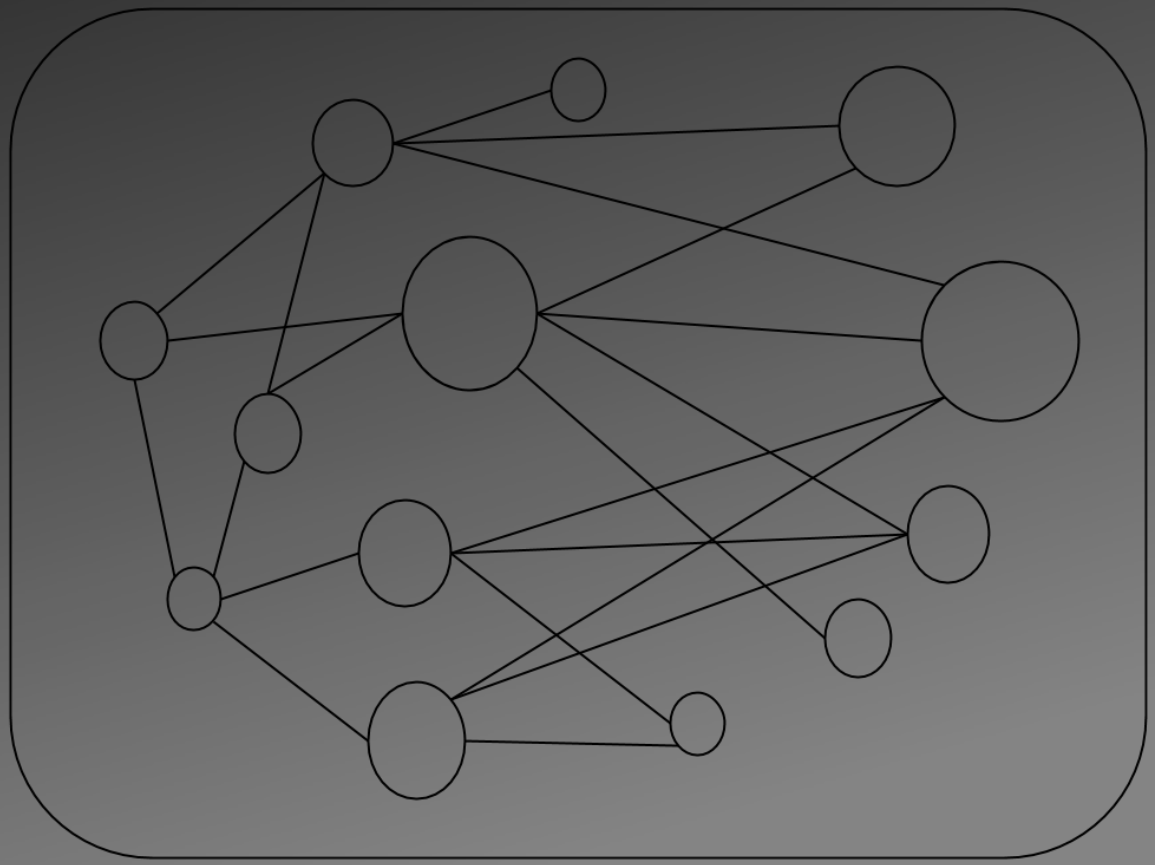


Step 0: Solution Space Setup





We start with a collection of solution states.



The programming solution space for a given problem is **very** crowded.

Program Normalization

- Remove dead code and comments
- Variable propagation
- Commutative Expression Ordering
- Anonymize variable names

Rivers, K., and Koedinger, K. (2012). A Canonicalizing Model for Building Programming Tutors. In Proceedings of the 11th International Conference on Intelligent Tutoring Systems (pp. 591-593).

Program Normalization

Original Student Program

```
def findPattern(s, pattern, startIndex):
    # This should return the location
    #of the pattern
    l = len(s)
    for i in range(l):
        if (findPatternAtIndex(s, pattern,
            startIndex + i) == True):
            return i + startIndex
    # return ??
```

Normalized Version

```
def findPattern(s, pattern, startIndex):
    l = len(s)
    for i in range(l):
        if (findPatternAtIndex(s, pattern,
            startIndex + i) == True):
            return i + startIndex
```

Program Normalization

Original Student Program

```
def findPattern(s, pattern, startIndex):
    l = len(s)
    for i in range(l):
        if (findPatternAtIndex(s, pattern,
            startIndex + i) == True):
            return i + startIndex
```

Normalized Version

```
def findPattern(s, pattern, startIndex):
    for i in range(len(s)):
        if (findPatternAtIndex(s, pattern,
            startIndex + i) == True):
            return i + startIndex
```

Program Normalization

Original Student Program

```
def findPattern(s, pattern, startIndex):
    for i in range(len(s)):
        if (findPatternAtIndex(s, pattern,
            startIndex + i) == True):
            return i + startIndex
```

Normalized Version

```
def findPattern(s, pattern, startIndex):
    for i in range(len(s)):
        if findPatternAtIndex(s, pattern,
            startIndex + i):
            return i + startIndex
```

Program Normalization

Original Student Program

```
def findPattern(s, pattern, startIndex):
    for i in range(len(s)):
        if findPatternAtIndex(s, pattern,
           startIndex + i):
            return i + startIndex
```

Normalized Version

```
def findPattern(s, pattern, startIndex):
    for i in range(len(s)):
        if findPatternAtIndex(s, pattern,
           startIndex + i):
            return startIndex + i
```

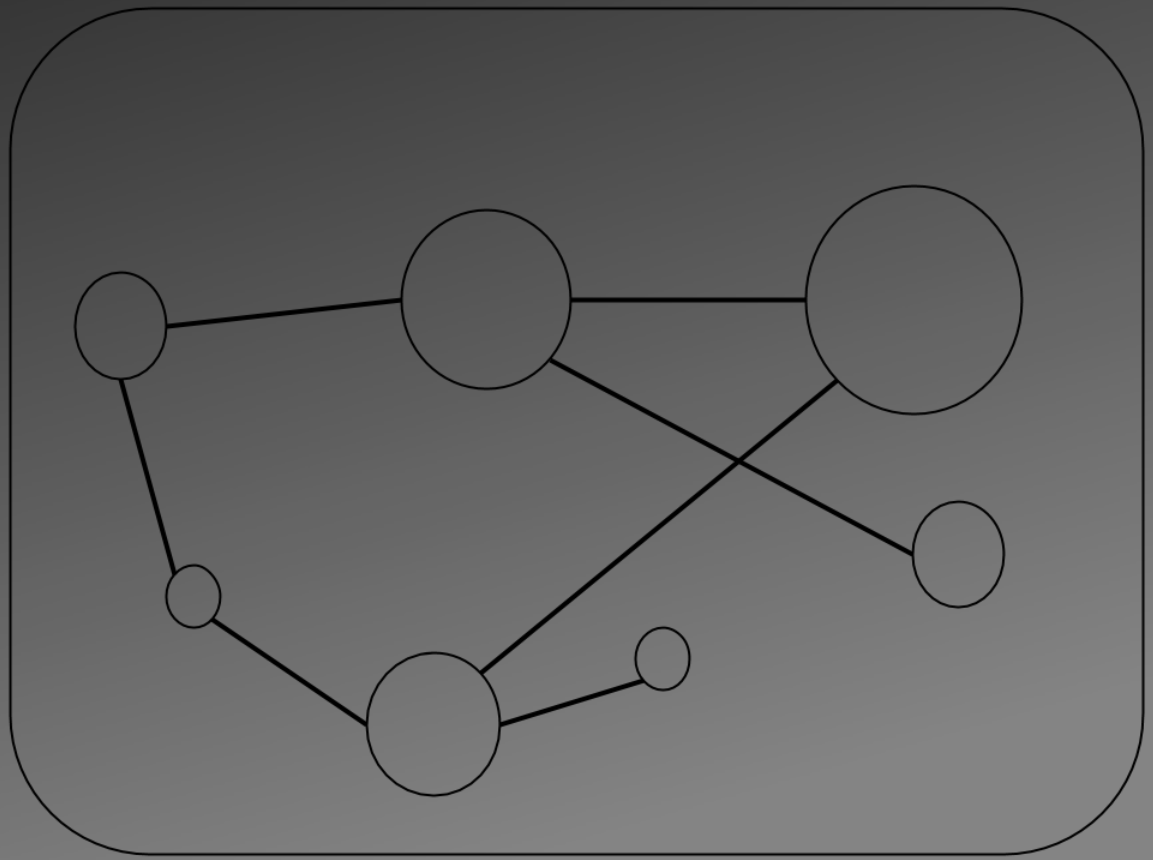
Program Normalization

Original Student Program

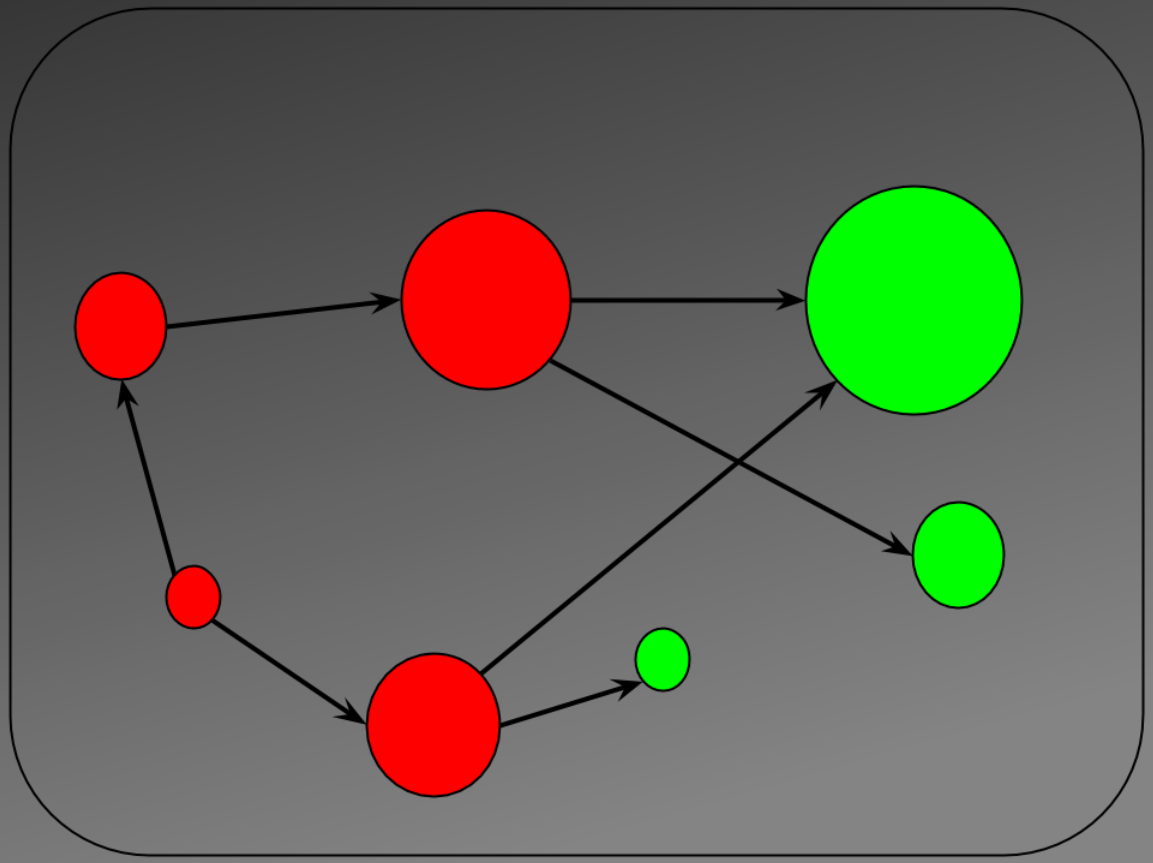
```
def findPattern(s, pattern, startIndex):
    for i in range(len(s)):
        if findPatternAtIndex(s, pattern,
          startIndex + i):
            return startIndex + i
```

Normalized Version

```
def findPattern(v0, v1, v2):
    for v3 in range(len(v0)):
        if findPatternAtIndex(v0, v1, v2 + v3):
            return v2 + v3
```

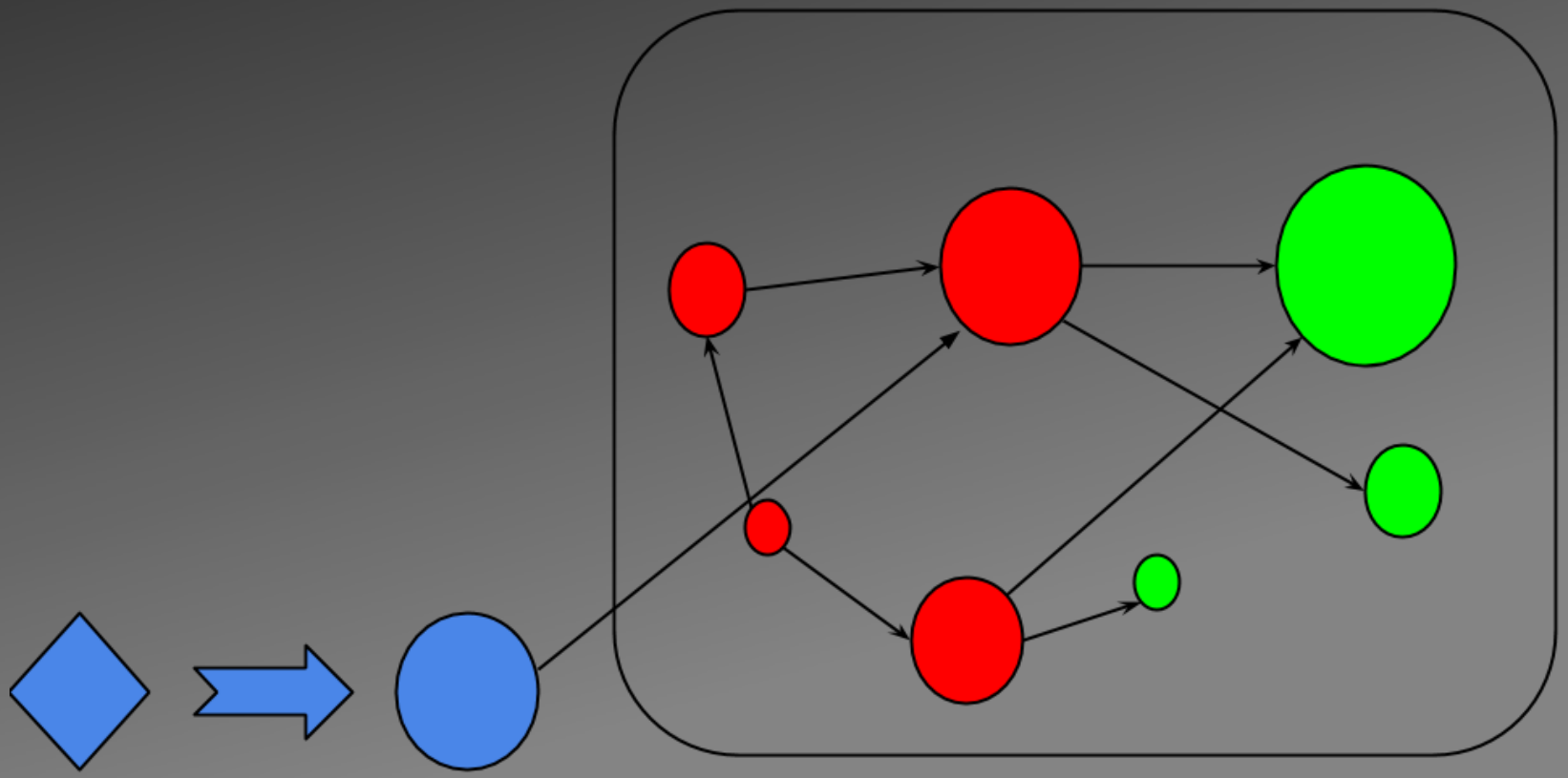


Once normalized, the solution space has a more reasonable scope, and some common states are evident

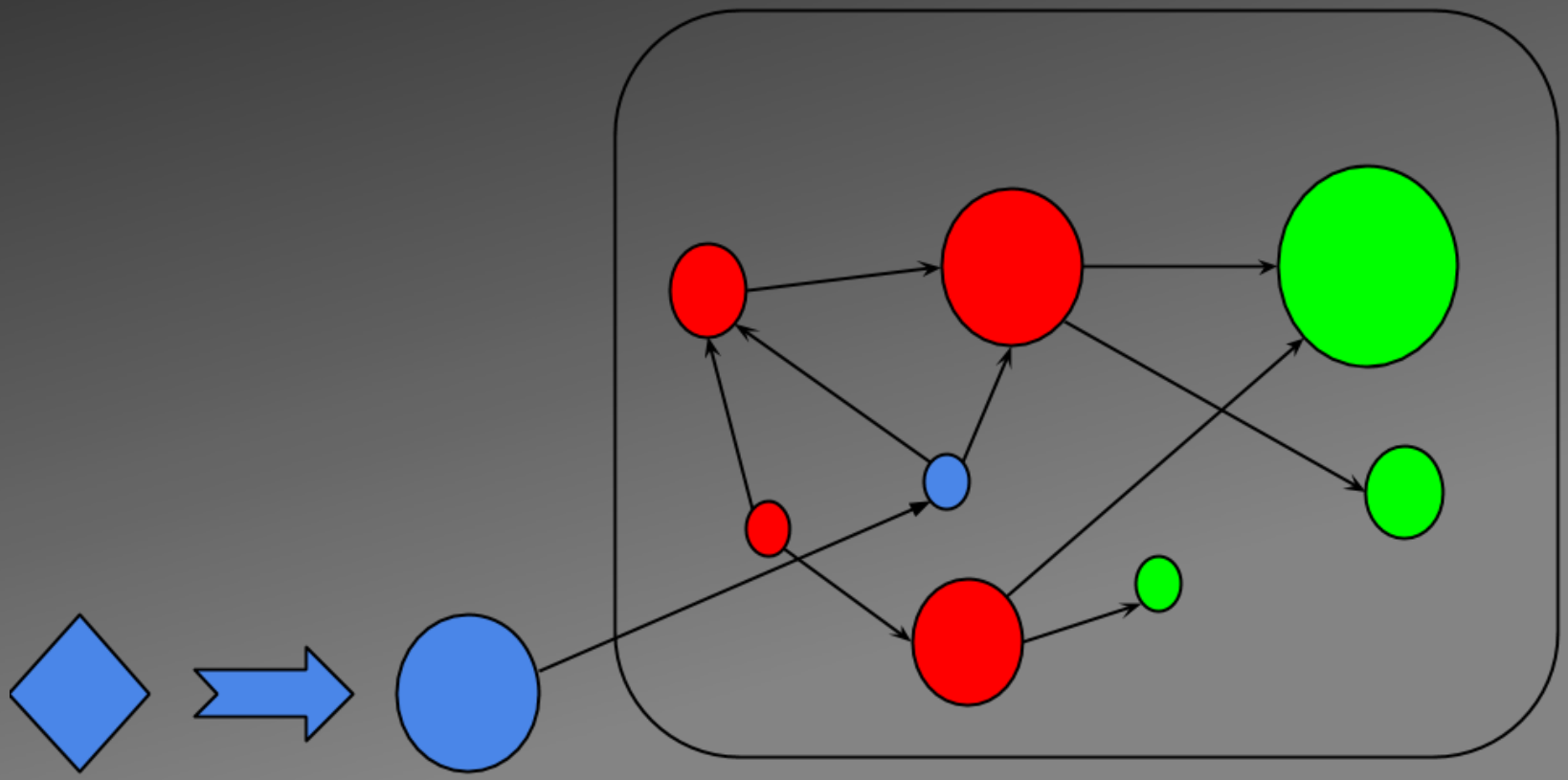


When state correctness is added, common paths can be found as well.

Step 1: Find Optimal Learning Progression



Insert New State



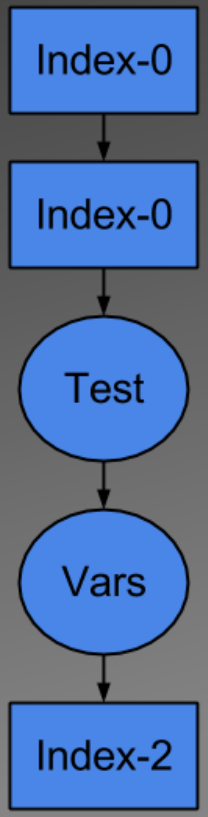
Finding the Best Path

- Distance function
  - Tree edits
  - Levenshtein string distance
  - Feature vectors
- Chains of actions
  - Sequence of states to closest correct

Step 2: State Transition - Edits

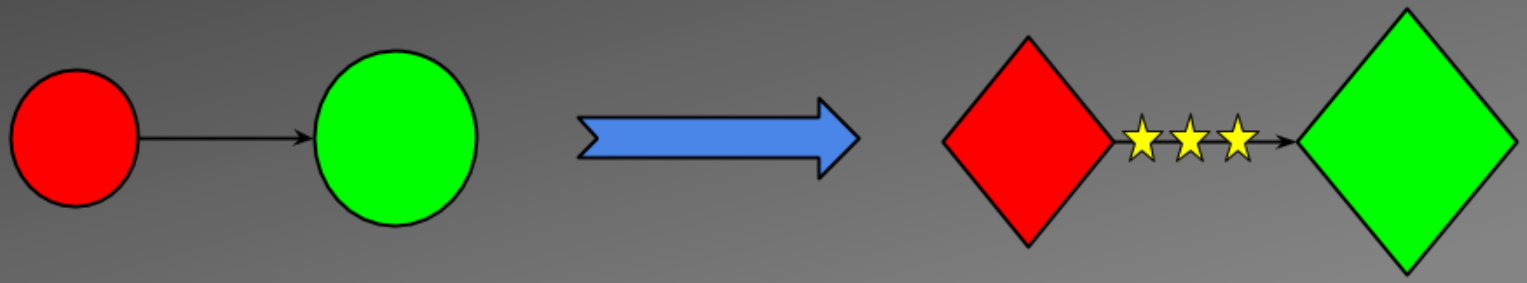
- Deletions: ([code lines], [])
  - Semantically unnecessary code
- Changes: ([code fragment], [code fragment])
  - Switching from one version to another
- Additions: ([], [code lines])
  - Missing a step in the solution

State Transition - Trace




```
def findPattern(v0, v1, v2):
    for v3 in range(len(v0)):
        if findPatternAtIndex(v0, v1, v3):
            return v2 + v3
    return -1
```

Step 3: Generating Individual Feedback



Levels of Hints

- Location: What line needs to be changed?
  - Make a change in line 26.
- Content: Which code fragment is wrong?
  - Change v3 in line 26.
- Edit: What is the correct code?
  - Replace v3 with v2+v3 in line 26.

The Feedback Doesn't Match

```
23 def findPattern(s, pattern, startIndex):
24     l = len(s)
25     for i in range(l):
26         if findPatternAtIndex(s, pattern, startIndex + i) == True:
27             return i
28     return -1
```

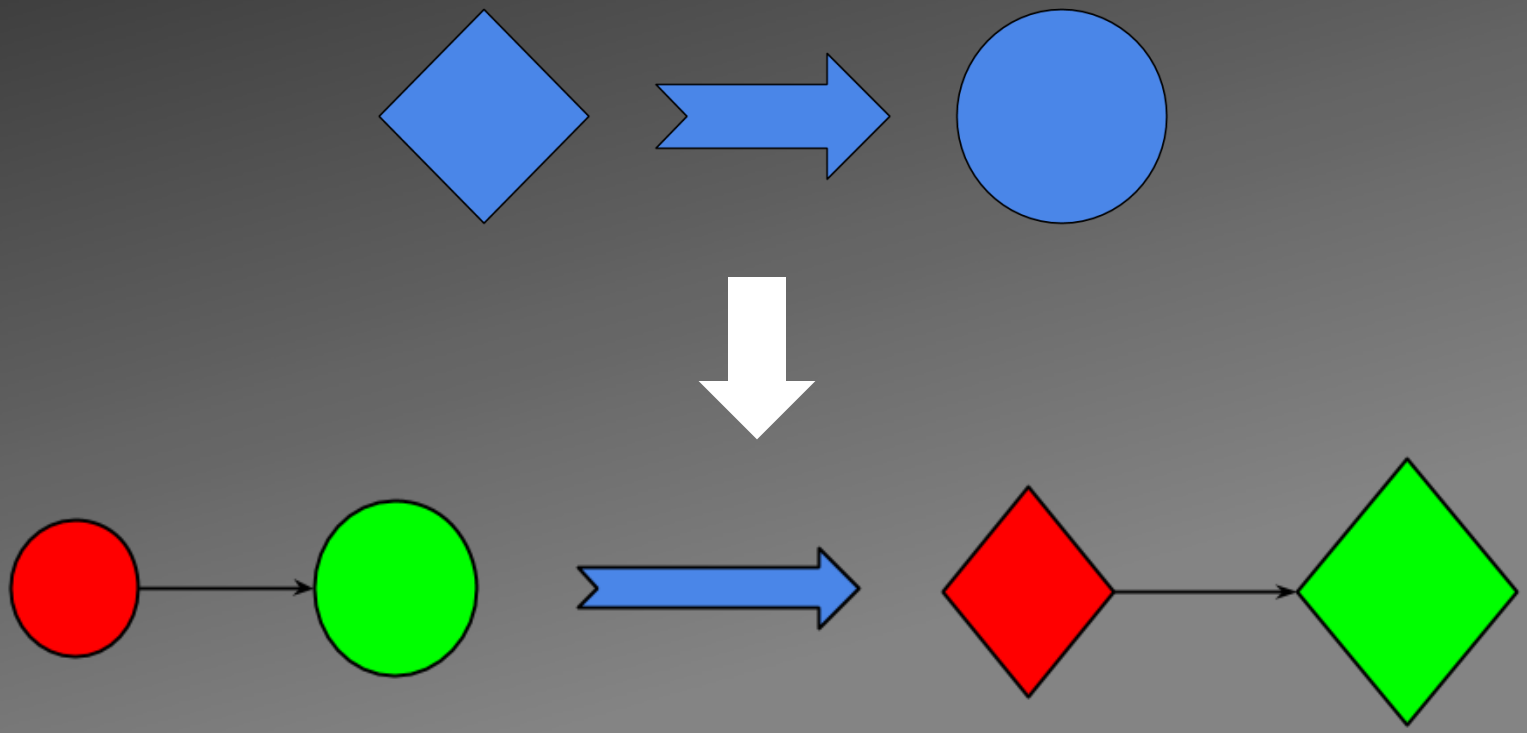
Replace v3 with v2+ v3 in line 26.

But it's Normalized!

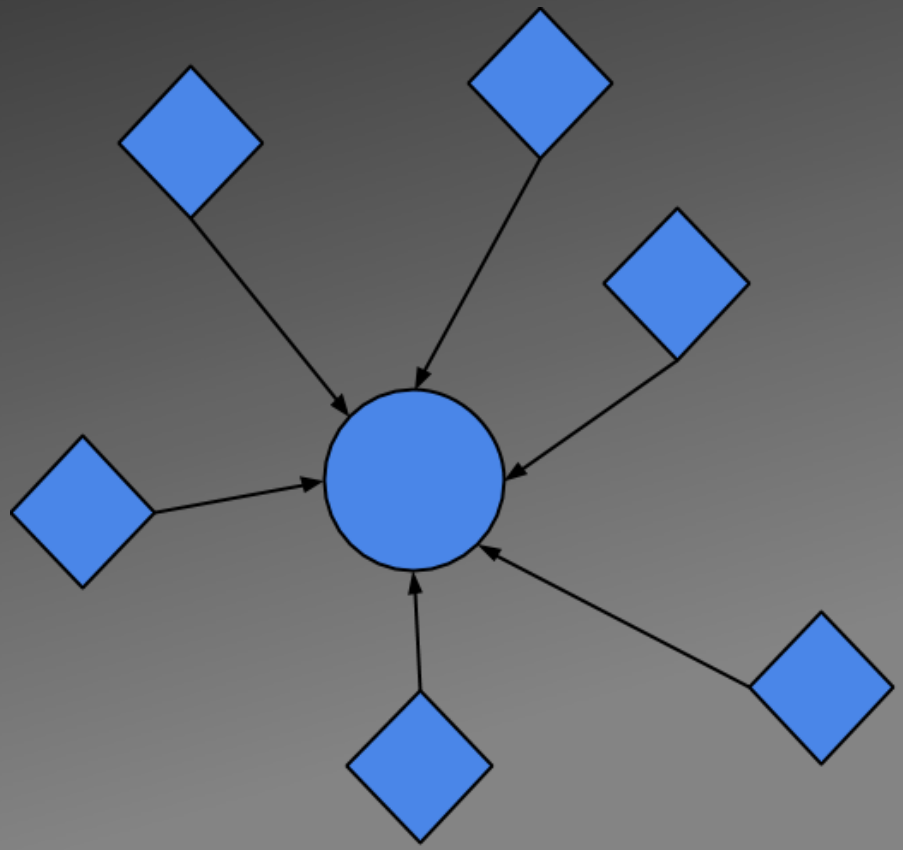
```
23 def findPattern(v0, v1, v2):
24    for v3 in range(len(v0)):
25       if findPatternAtIndex(v0, v1, v2 + v3):
26          return v3
27    return -1
```

Replace v3 with v2+ v3 in line 26.

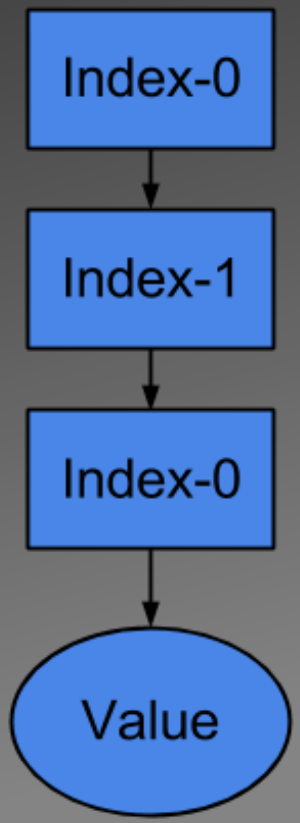
Undo the Transformations



Many to One

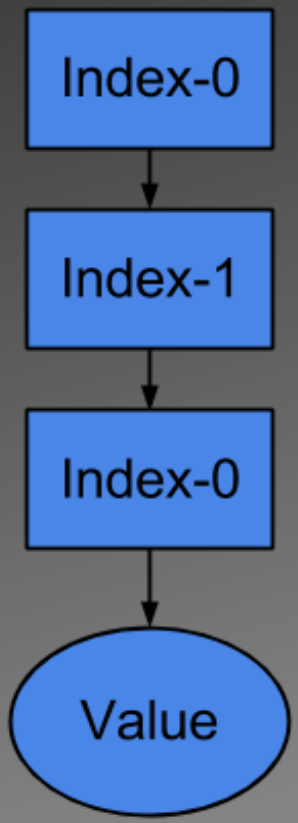


Unrolling the Trace




```
def findPattern(v0, v1, v2):
    for v3 in range(len(v0)):
        if findPatternAtIndex(v0, v1, v2 + v3):
            return v3
    return -1
```

Unrolling the Trace



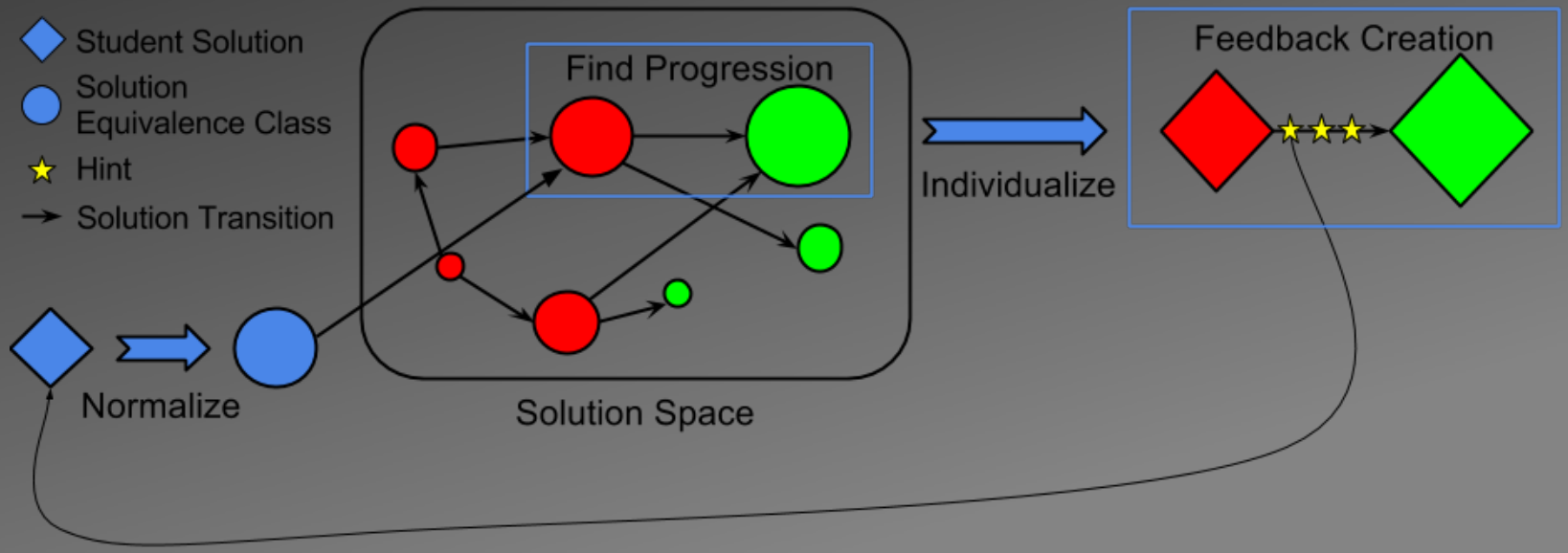

```
def findPattern(s, pattern, startIndex):
    for i in range(len(s)):
        if findPatternAtIndex(s, pattern, startIndex + i) == True:
            return i
    return -1
```

Mapping the Transformations?

- Deleted lines
- Extra code
- Reordered expressions

- How?

Overview


Student Solution
Solution Equivalence Class
Hint
Solution Transition
Normalize
Find Progression
Solution Space
Individualize
Feedback Creation

Let’s try it!

So What?

Research Question

- Automatically generate feedback

Research Question

- Automatically generate feedback
- … in order to make programming less painful for novices

How to Measure?

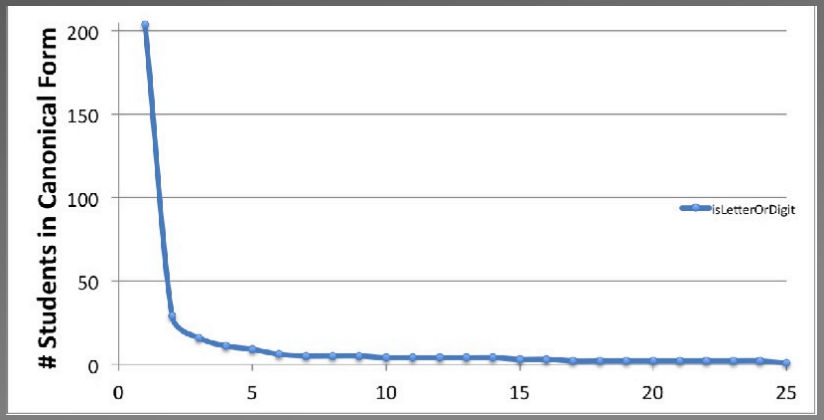
- Rate relevance of messages
  - Relation to test results
  - Targeted solution
- Test with real students!
  - Fall 2013

For students, we hope…

- Help them squash 'impossible' bugs
- Recommend how correct solutions can become better

For teachers, we hope…

- Help them target struggling students
- Discover missing knowledge components



Discussion

- Limitation: reliance on previously collected data
- Learning to debug: when do we stop giving feedback?
- Open-ended problems: how to approach?

Next Steps

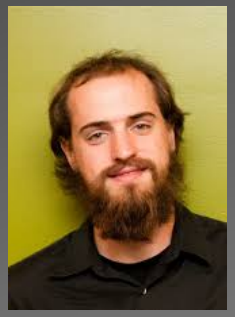
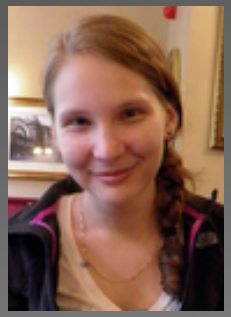
- Generalize for many teachers…
- … and other languages…
- … and even other domains?

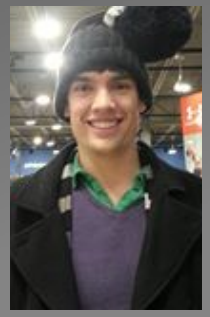
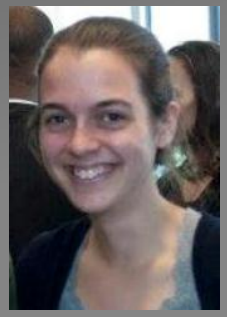
Questions?







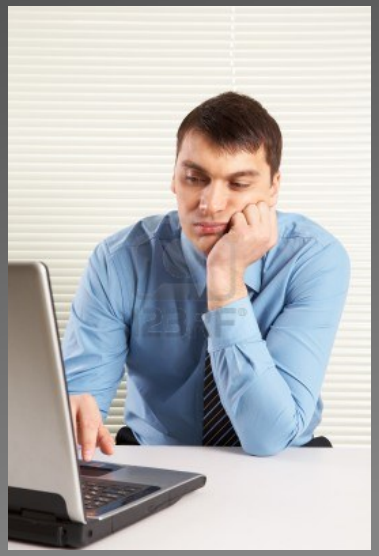




This work was supported in part by Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (# R305B090023).

Time on Task

Stoppers

Tinkerers



Movers

Perkins & Martin, 1986; Jadud, 2006