How do Learners Use Scratch Paper when Working on Dynamic Programming Problems?

Zihan Wu[†], Jonathan Liu[‡], Erica Goodwin[‡], Diana Franklin[‡]

[†]University of Michigan [‡]University of Chicago

Introduction & Background

Overview: Analysis of scratch paper from students solving DP problems can provide insight for how students commonly interact with examples and formulas when attempting to design DP algorithms. This analysis can help with building a tool that scaffolds learning this paradigm.

Related Work

- Effective use of scratch paper for code tracing has shown to be helpful in CS1.
- Dynamic Programming has shown to be one of the most difficult parts of algorithms courses.
  - Visualization tools have been developed to help with visualizing running a DP algorithm, but none that directly target the *design* of the algorithm.

Methods

Collection: Scratch paper scans were collected from a separate think-aloud study in which students solved DP algorithm design problems with verbal guidance from the interviewer [1]. Scratch paper was collected for 18 students who together solved 27 problems.

Analysis: Scratch Paper scans were parsed for common usage patterns to develop a codebook. Once developed, authors independently coded all scratch paper for these usage. The codebook and frequency of codes is shown on the right.

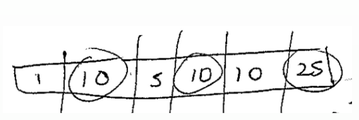
Future Steps

Tool Development: We are developing a tool that scaffolds practicing DP algorithm design problems by providing an example input and acting as augmented scratch paper. We aim to ensure that the tool not only provides the affordances that students naturally utilize, as found in this poster's work, but also makes more clear how example inputs and outputs can be used to design subproblems and recurrences for Dynamic Programming algorithms.

[1] Jonathan Liu et al. 2025. Student Utilization of Metacognitive Strategies in Solving Dynamic Programming Problems. SIGCSE '25.

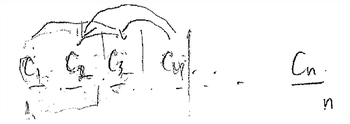
Results

Input Representation



Example input value
Student created example input with specific values

(57% problems, 73% learners)



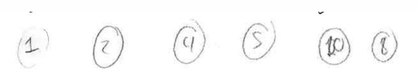
Mathematical example
Student used mathematical symbols as examples

(54% problems, 67% learners)

(A picture demonstrating distances between gas stations)

Abstract example
Student drew abstract graphical representations or used symbols not necessarily representing the relationship between values, e.g. A, B, C…

(15% problems, 23% learners)

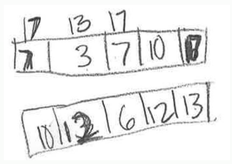


(A student used circles to represent "coins" in a problem, when they did not use circles for any other problems)

Realistic representation
Student used graphic representations to make the examples look more realistic in the context of the problem

(7% problems, 10% learners)

Using the Example

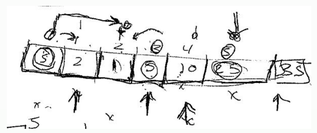


Multiple examples
Student created more than one example in one problem

(39% problems, 50% learners)

Filling in a table
Student used 2-D table structures to organize calculation with the example

(39% problems, 53% learners)

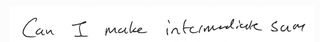


(A student used arrows, circles, and other symbols to "animate" the examples)

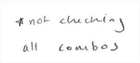
Interaction with Examples
Student used symbols to interact with individual terms of the example, animating the calculation process or relationships between the items

(72% problems, 90% learners)

Others



A student wrote question to themselves: "Can I make intermediate sum"



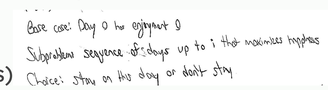
A student made notes: "not checking all combos"

Written thoughts
Student wrote comments or thoughts in natural language

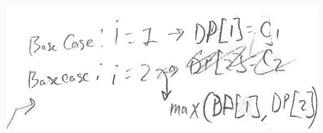
(50% problems, 63% learners)

Decomposition by steps
Student wrote the problem decomposition for DP problems: base case, subproblem, and recurrence

(22% problems, 27% learners)



Process for Developing the Formula

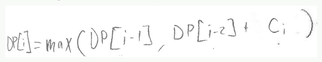


Base case
Student attempted to write out base cases

(50% problems, 67% learners)

Formula in math
Student attempted to write out mathematical representations of formulas

(24% problems, 73% learners)

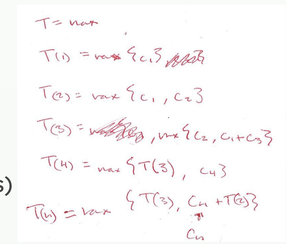


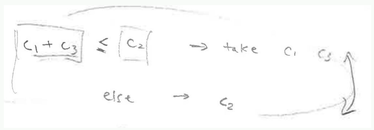
Formula in words
Student used natural language to express formula

(72% problems, 37% learners)

Mathematical Induction
Student attempted to build a recurrence from a sequence of consecutive cases

(15% problems, 23% learners)





Written logic with control flow
Student wrote pseudo code or written logic that contains control flow (e.g. if-else, for, while)

(57% problems, 73% learners)