

Division Strategies - Strategic Trials

Compiled by: Theodore M. Savich

April 1, 2025

This is a sharing division strategy. With sharing division problems, the number of items in each group is unknown, while the number of groups and the total number of items are both known.

$$\boxed{\text{Number of groups}} \times \boxed{\text{Unknown Number of items in each group}} = \boxed{\text{Total number of items}}$$

Transcript

Video from Carpenter et al. (1999). Strategy descriptions and examples adapted from Hackenberg (2025).

- **Teacher:** Mrs. Carpenter made 56 cupcakes for a birthday party. She has eight boxes to carry the cupcakes to his party. How many cupcakes should she put in each box if she wants to put the same number of cupcakes in each box?
- **Student:** [inaudible] Put seven in. Seven.
- **Teacher:** I can tell just tell you did that. Thank you very much, Victoria.

This strategy is more sophisticated than Dealing by Ones because it involves selecting an initial, reasonable group size, testing it, and then logically refining that choice as needed.

Description of Strategic Trials:

Begin with an initial trial number for the items per group. **Utilize a multiplication strategy** to calculate the total number of items and verify it against the given total. Adjust your trial number upward or downward as necessary, and recalculate until you arrive at the correct result.

Notation and Visual Representations for Strategic Trials: Use clear notation and diagrams to illustrate the equal groups multiplication strategy you have chosen.

For example, second-grade student Victoria was tasked with determining how many cupcakes should be placed in each of 8 boxes, given a total of 56 cupcakes. She initially assumed 8 cupcakes per box and employed a doubling method to compute the total:

$$8 + 8 = 16$$

$$16 + 16 = 32$$

$$32 + 32 = 64$$

Seeing that 64 exceeded the given total, she then tried 6 cupcakes per box:

$$6 + 6 = 12$$

$$12 + 12 = 24$$

$$24 + 24 = 48$$

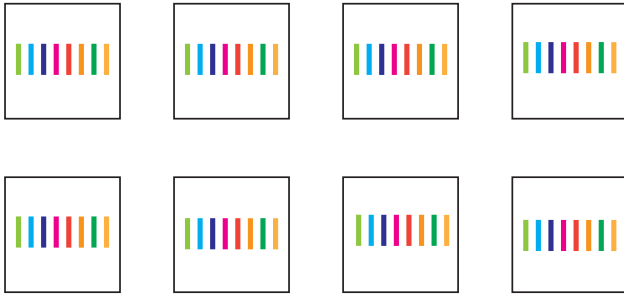
Realizing 48 was too low, Victoria understood she was estimating the number of cupcakes per box. After trying 8 (which was too high) and 6 (which was too low), she decided to test 7 cupcakes per box:

$$7 + 7 = 14$$

$$14 + 14 = 28$$

$$28 + 28 = 56 \quad (\text{using her addition strategy})$$

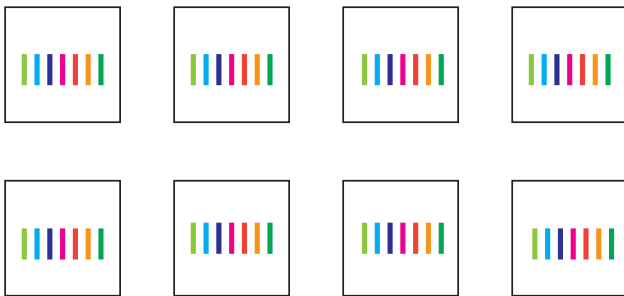
Eight 8s = 64



Eight 6s = 48



Eight 7s = 56



She concluded that each box should contain 7 cupcakes. In class, we highlighted that her method was not merely “trial and error,” but a thoughtful process of strategic adjustment. When the initial guess was too high, she adjusted downward, and when it was too low, she adjusted upward. This iterative process is a hallmark of strategic trials.

Strategic Trials

Strategy Overview

Strategic Trials involves testing different grouping configurations to find the correct division outcome. This strategy is iterative and relies on trial-and-error to determine the appropriate number of groups or the group size required for division.

Automaton Design

We design a **Pushdown Automaton (PDA)** that systematically:

1. Attempts a trial grouping by pushing a trial marker T and assigning a set of elements.
2. Checks whether the trial group meets the required size.
3. Adjusts the trial group if the size is incorrect.
4. Upon a correct trial, confirms the group by pushing a group identifier G and then outputs the final grouping.

Automaton Tuple

The PDA is defined as the 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_{0/accept}, \#, F)$$

where:

- $Q = \{q_{0/accept}, q_{trial}, q_{check}, q_{adjust}, q_{output}\}$ is the set of states. (Here, $q_{0/accept}$ serves as both the start and the accepting state.)
- $\Sigma = \{E\}$ is the input alphabet (with E representing an element).
- $\Gamma = \{\#, T, G\}$ is the stack alphabet:
 - $\#$ is the bottom-of-stack marker.
 - T represents a trial grouping.
 - G represents a confirmed group.
- $q_{0/accept}$ is the start (and accept) state.
- $\#$ is the initial stack symbol.
- $F = \{q_{0/accept}\}$ is the set of accepting states.

State Transition Table

Current State	Input Symbol	Stack Top	Next State	Stack Operation	Description
$q_{0/accept}$	ε	—	q_{trial}	Push $\#$	Initialize
q_{trial}	ε	any	q_{check}	Push T ; assign a trial group	Attempt trial
q_{check}	ε	any	q_{output}	(If trial correct: push G)	Trial correct
q_{check}	ε	any	q_{adjust}	—	Trial incorrect
q_{adjust}	ε	any	q_{trial}	Adjust trial	Modify trial group
q_{output}	ε	any	$q_{0/accept}$	Count G 's	Output final grouping

Automaton Behavior

1. Initialization:

- Start in $q_{0/accept}$, push $\#$ onto the stack.
- Transition to q_{trial} to begin the trial process.

2. Attempting a Trial:

- In q_{trial} , push T to represent a trial group and assign a set of elements to it.
- Transition to q_{check} .

3. Checking the Trial:

- In q_{check} , evaluate if the trial group meets the required size.
- If the trial is correct, push a confirmed group G and transition to q_{output} .
- If the trial is incorrect, transition to q_{adjust} .

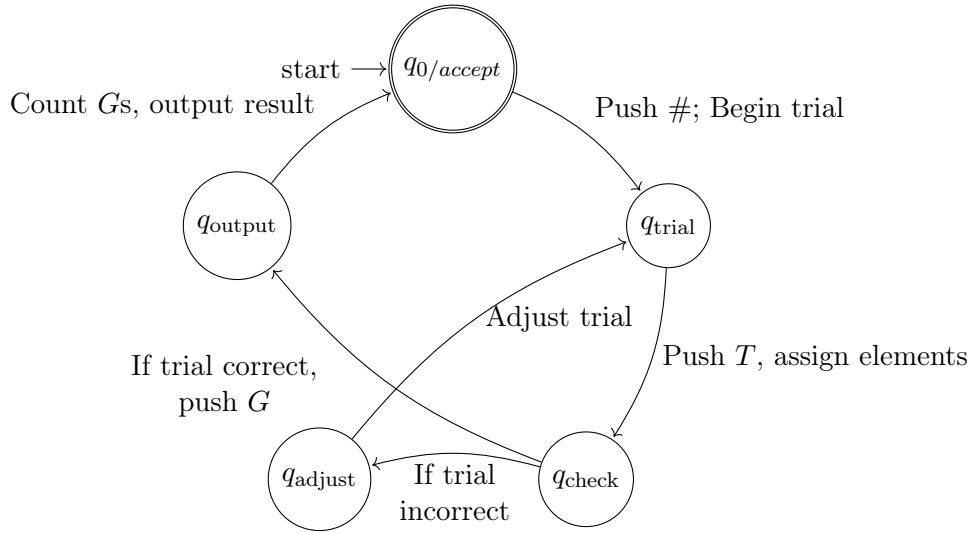
4. Adjusting the Trial:

- In q_{adjust} , modify the trial group size (by adding or removing elements).
- Return to q_{trial} to try again.

5. Outputting the Result:

- In q_{output} , count the number of confirmed groups (G symbols) on the stack.
- Output the final grouping and transition back to q_0/accept (the merged start/accept state).

Circular PDA Diagram



Example Execution

Problem: Divide 24 items into groups of 8 using strategic trials.

1. Start:

- The initial stack contains: # followed by 24 E symbols.

2. Trial 1:

- In q_{trial} , a trial group of 7 elements is attempted (push T , assign 7 E symbols).
- In q_{check} , the trial is evaluated: $7 \neq 8$, so transition to q_{adjust} .

3. Adjust Trial:

- In q_{adjust} , the trial is modified (e.g., increase group size to 8).
- Return to q_{trial} for a new attempt.

4. Trial 2:

- In q_{trial} , attempt a trial group of 8 elements.
- In q_{check} , the trial is correct ($8 = 8$); a confirmed group G is pushed.

5. Repeat:

- Continue trials until all 24 items are grouped.
- Final output: 3 groups of 8.

Iterative Handling of Trials

The PDA iteratively attempts different group sizes, adjusting the trial configuration as needed based on feedback from the check phase. This iterative process continues until the correct grouping is achieved, ensuring an accurate division.

HTML Implementation

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Division: Strategic Trials</title>
5   <style>
6     body { font-family: sans-serif; }
7     .container { max-width: 800px; margin: 10px auto; padding: 10px;}
8     .control-section, .trials-section, .result-section {
9       margin-bottom: 20px; padding: 10px; border: 1px solid #eee;
10      background-color: #f9f9f9; border-radius: 5px;
11    }
12    label { margin-right: 5px;}
13    input[type=number] { width: 60px; margin-right: 15px;}
14    button { padding: 5px 10px; font-size: 1em; margin-right: 5px; }
15    #statusMessage { color: #e65c00; font-weight: bold; margin-left: 15px;}
16
17    .trial-visualization {
18      margin-top: 15px;
19      padding-top: 10px;
20      border-top: 1px dashed #ccc;
21    }
22    .group-container { /* Container for all groups in a trial */
23      display: flex;
24      flex-wrap: wrap; /* Allow groups to wrap */
25      gap: 10px; /* Space between groups */
26      margin-bottom: 5px;
27    }
28    .group-box {
29      display: inline-block; /* Display groups inline */
30      border: 1px solid #999;
31      padding: 4px;
32      background-color: #e8f4ff;
33      min-width: 40px; /* Minimum width */
34      text-align: center; /* Center items */
35    }
36    .group-box-label { font-size: 0.8em; color: #555; margin-bottom: 3px; display:
37      block;}
38    .item-block {
39      display: inline-block; /* Items side-by-side */
40      width: 8px; height: 8px; margin: 1px; /* Smaller items */
41      background-color: #6495ED; /* Cornflower blue */
42      border: 1px solid #444;
43    }
44    .trial-summary { font-weight: bold; margin-top: 5px; }
45    .trial-correct { color: darkgreen; }
46    .trial-incorrect { color: darkred; }
47    #finalResultValue { font-size: 1.5em; font-weight: bold; color: darkgreen; }
48  </style>
49 </head>
50 <body>
51 <div class="container">
```

```

52 <h1>Division Strategies - Strategic Trials</h1>
53
54 <div class="control-section">
55   <label for="stratTotalInput">Total Items:</label>
56   <input type="number" id="stratTotalInput" value="56" min="1"> <!-- Example -->
57   <label for="stratGroupsInput">Number of Groups:</label>
58   <input type="number" id="stratGroupsInput" value="8" min="1"> <!-- Example -->
59   <button onclick="setupTrialSimulation()">Set Up / Reset</button>
60   <button onclick="performNextTrial()" id="trialBtn" disabled>Perform Next Trial</button>
61   <span id="statusMessage"></span>
62 </div>
63
64 <div class="trials-section">
65   <strong>Trials:</strong>
66   <div id="trialsDisplay">
67     <!-- Trial visualizations will be added here -->
68   </div>
69 </div>
70
71 <div class="result-section">
72   <strong>Result (Items per group):</strong> <span id="finalResultValue">?</span>
73 </div>
74
75
76 <script>
77   // --- Simulation State Variables ---
78   let totalItems = 0;
79   let numGroups = 0;
80   let currentTrialSize = -1; // -1 indicates simulation not started or needs initial
      guess
81   let attempts = []; // Stores history: { trialSize: number, trialResult: number,
      outcome: string }
82   let finalGroupSize = null; // The correct answer when found
83   let isTrialComplete = true;
84
85   // --- DOM Element References ---
86   const totalInput = document.getElementById("stratTotalInput");
87   const groupsInput = document.getElementById("stratGroupsInput");
88   const finalResultValueSpan = document.getElementById("finalResultValue");
89   const trialsDisplay = document.getElementById("trialsDisplay");
90   const trialBtn = document.getElementById("trialBtn");
91   const statusMessage = document.getElementById("statusMessage");
92
93   // --- Simulation Functions ---
94   function setupTrialSimulation() {
95     totalItems = parseInt(totalInput.value);
96     numGroups = parseInt(groupsInput.value);
97
98     if (isNaN(totalItems) || isNaN(numGroups) || numGroups <= 0 || totalItems < 0)
99     {
100       statusMessage.textContent = "Please enter valid positive numbers (Groups > 0).";
      trialBtn.disabled = true;
    }
  }

```



```

101         isTrialComplete = true;
102         finalResultValueSpan.textContent = "?";
103         trialsDisplay.innerHTML = ""; // Clear previous trials
104         return;
105     }
106
107     // Make the first guess intentionally off (e.g., +/- 1 or 2 from rough
108     // estimate)
109     let roughEstimate = Math.max(1, Math.round(totalItems / numGroups)); // Ensure
110     // guess is at least 1
111     let randomOffset = Math.random() < 0.5 ? (roughEstimate > 1 ? -1 : 1) : 1; //
112     // Offset by +/- 1
113     currentTrialSize = roughEstimate + randomOffset;
114     // Ensure guess isn't accidentally correct if estimate was close
115     if (currentTrialSize * numGroups === totalItems && currentTrialSize > 1) {
116         currentTrialSize--; // Adjust if first guess happens to be right
117     }
118     if (currentTrialSize <= 0) currentTrialSize = 1; // Ensure guess is at least
119     // 1
120
121     attempts = []; // Clear history
122     finalGroupSize = null;
123     isTrialComplete = false;
124
125     statusMessage.textContent = 'Ready. Initial trial guess: ${currentTrialSize}
126     items per group.';
127     finalResultValueSpan.textContent = "?";
128     trialsDisplay.innerHTML = ""; // Clear previous trials visually
129     trialBtn.disabled = false;
130 }
131
132 function performNextTrial() {
133     if (isTrialComplete) {
134         statusMessage.textContent = "Found correct group size! Press Reset to start
135         again.";
136         trialBtn.disabled = true;
137         return;
138     }
139
140     statusMessage.textContent = 'Trying ${currentTrialSize} items per group...';
141
142     // 1. Multiply to get trial total
143     const trialResult = currentTrialSize * numGroups;
144
145     // 2. Check against actual total
146     let outcome = "";
147     let outcomeClass = "";
148     if (trialResult === totalItems) {
149         outcome = "Correct!";
150         outcomeClass = "trial-correct";
151         finalGroupSize = currentTrialSize;
152         isTrialComplete = true;
153         trialBtn.disabled = true; // Disable button once correct

```

```

149         statusMessage.textContent = 'Found correct group size: ${finalGroupSize
            }!';
150         finalResultValueSpan.textContent = finalGroupSize;
151     } else if (trialResult < totalItems) {
152         outcome = 'Too Low (${trialResult} < ${totalItems})';
153         outcomeClass = "trial-incorrect";
154     } else { // trialResult > totalItems
155         outcome = 'Too High (${trialResult} > ${totalItems})';
156         outcomeClass = "trial-incorrect";
157     }
158
159     // 3. Store attempt
160     attempts.push({
161         trialSize: currentTrialSize,
162         trialResult: trialResult,
163         outcome: outcome,
164         outcomeClass: outcomeClass
165     });
166
167     // 4. Draw this attempt
168     drawTrialVisualization(currentTrialSize, numGroups, trialResult, outcome,
        outcomeClass);
169
170
171     // 5. Adjust for next trial (if not correct)
172     if (!isTrialComplete) {
173         if (trialResult < totalItems) {
174             // Increase guess (could be smarter, e.g., based on how far off)
175             currentTrialSize++;
176         } else {
177             // Decrease guess
178             currentTrialSize--;
179             if (currentTrialSize <= 0) currentTrialSize = 1; // Don't guess 0 or
                negative
180         }
181         statusMessage.textContent += ' Adjusting guess to ${currentTrialSize}.';
182     }
183 }
184
185 function drawTrialVisualization(trialSize, groups, result, outcome, outcomeClass)
    {
186     const trialDiv = document.createElement('div');
187     trialDiv.className = 'trial-visualization';
188
189     const groupContainer = document.createElement('div');
190     groupContainer.className = 'group-container';
191
192     for (let g = 0; g < groups; g++) {
193         const groupBox = document.createElement("div");
194         groupBox.className = "group-box";
195         // groupBox.innerHTML = '<span class="group-box-label">Group ${g + 1}</span
            >'; // Optional label
196
197         // Arrange items within the box (e.g., simple horizontal flow)

```

```

198     let itemsHtml = '';
199     let itemsPerRow = Math.max(5, Math.ceil(Math.sqrt(trialSize))); // Simple
    layout heuristic
200     for(let i = 0; i < trialSize; i++) {
201         itemsHtml += '<span class="item-block"></span>';
202         if ((i + 1) % itemsPerRow === 0) itemsHtml += '<br>'; // Add line
    break
203     }
204     groupBox.innerHTML += itemsHtml;
205     groupContainer.appendChild(groupBox);
206 }
207 trialDiv.appendChild(groupContainer);
208
209     const summary = document.createElement('div');
210     summary.className = 'trial-summary';
211     summary.innerHTML = `Trial: ${groups} groups  ${trialSize} items/group = ${
    result}. <span class="${outcomeClass}">${outcome}</span>`;
212     trialDiv.appendChild(summary);
213
214
215     trialsDisplay.appendChild(trialDiv);
216     trialsDisplay.scrollTop = trialsDisplay.scrollHeight; // Scroll to bottom
217 }
218
219
220 // --- Helper SVG/Typeset Functions (Not needed for this block viz) ---
221 function typesetMath() { /* Placeholder */ }
222
223 // --- Initialize ---
224 setupTrialSimulation(); // Initialize state on load
225
226
227 </script>
228
229 <!-- New button for viewing PDF documentation -->
230 <button onclick="openPdfViewer()">Want to learn more about this strategy? Click here.</
    button>
231
232 <script>
233     function openPdfViewer() {
234         // Opens the PDF documentation for the strategy.
235         window.open('../SMR_DIV_Strategic_Trials.pdf', '_blank');
236     }
237 </script>
238
239 </div> <!-- End Container -->
240 </body>
241 </html>

```

References

- Carpenter, T. P., Fennema, E., Franke, M. L., Levi, L., & Empson, S. B. (1999). *Children's mathematics: Cognitively guided instruction* [Includes supplementary material: Children's mathematics: Cognitively guided instruction – videotape logs]. Heinemann; The National Council of Teachers of Mathematics, Inc.
- Hackenberg, A. (2025). *Course notes* [Unpublished course notes].