Recitation-2

Divide & Conquer

Hamed Shahbazi

Practice Questions

Question -1

Practice Question 1

Problem: Show that all horses in a set **S** of any size are in the same color !

Practice Question 1

Proof by Induction:

- **Base:** For **S** of size **K=1** the horse is in the same color like

  {white }

- **Inductive Hypothesis:** For **S** of size **K=n** we assume all horses are in the same color like

  $\{white_1, white_2, \ldots white_k \}$

  or

  $\{blue_1, blue_2, \ldots blue_k \}$

Practice Question 1

- **Inductive Step:** Partition set **S** of size **K=n+1** into sets **S1** = {1,…n} and **S2**={2,…n+1} of size k=n.

Since **S1** and **S2** are of size **n** thus **hypothesis** says **S1**'s horses are in same color like **color1** and **S2**'s horses are also in same color like **color2**.

Since **S1** and **S2** overlap thus

color1 = color2

therefore for k=n+1 all are in same color.

∎

Practice Question 1

But the Induction is Wrong

Overlapping assumption doesn't hold for k =2

$\{1,2\} = \{1\}, \{2\}$

Question -2

Practice Question 2

Input: Sorted Distinct Integers **A[1…n]**

Output: Check if **A[i] = i** for any **i** in time **O(log(n))**

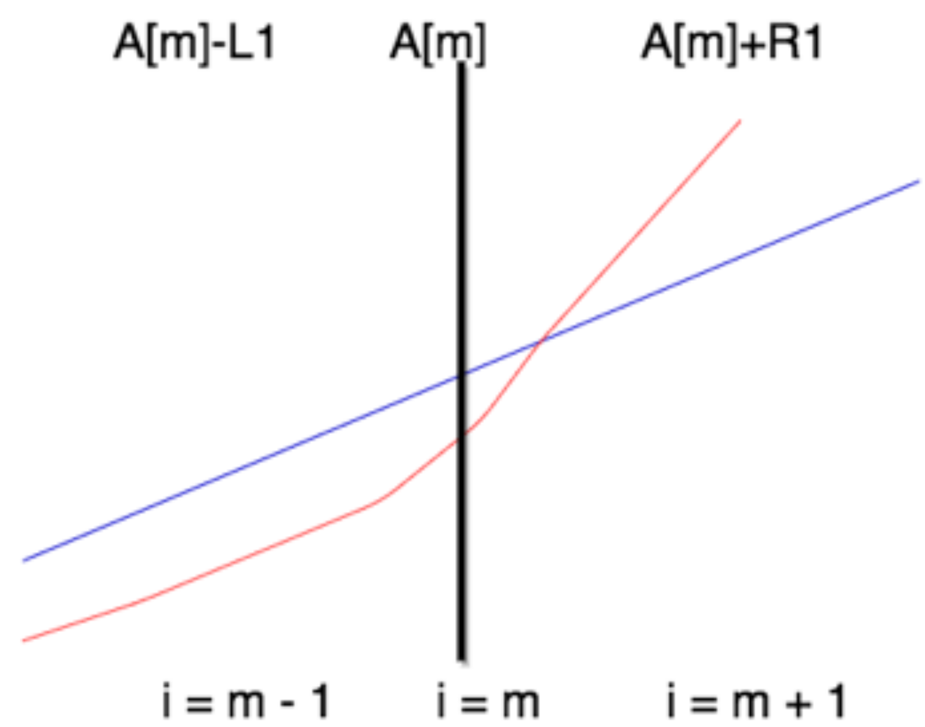
Practice Question 2

Hint:

1. We use properties of array **A**; **sorted**, **distinct** and **integer**

2. Get rid of **almost half** of **A** at constant number of comparisons

Practice Question 2



For any index m

A[m] = m return true
A[m] > m the **right side** is hopeless because the rate of increase for **A[i]** is **equal or higher** than **i**; therefore index **i** never reaches **A[i] at the right side**

A[m] < m the **left side** is hopeless because the rate of decrease for **A[i]** is **equal or higher** than **i**; therefore **i** can never reach **A[i]** in the left side

Conclusion: We can get rid of one side of **m** with one comparison

Practice Question 2

Set **$m = n/2$** to get **$O(\log(n))$**

$$\mathbf{T(n) = T(n/2) + O(1) = O(\log n)}$$

Practice Question 2

$\texttt{algo}(A[1,\ldots,n])$

1. *if* $n == 1$, *return (A[n]==n)*
2. $m = \lceil \frac{n}{2} \rceil$
3. *if* $A[m] = m$ *return true*
4. *else if* $A[m] > m$, *return* $\texttt{algo}(A[1,\ldots,m-1])$
5. *else return* $\texttt{algo}(A[m+1,\ldots,n])$

Practice Question 2

Proof by Induction:

Base: **n= 1** is correct

Hypothesis: **n= k** is correct

Inductive Step: n= k+1:

1. If **A[n/2] = n/2** the algorithm finds answer correctly in in O(log n)

2. If **A[n/2] > n/2** then for each j > n/2 we have
   A[j] >= A[n/2] + (j-n/2) > n/2 + j - n/2 **> j**

3. If **A[n/2] < n/2** then for each j < n/2 we have
   A[j] <= A[n/2] - (n/2-j) < n/2 + j - n/2 **< j**

Question -3

Practice Question 3

Input: A[1..n], B[1..n] **Sorted**
Output: Median of combined A, B with size 2n in **O(log n)**

Practice Question 3

Hint: When we like **O(log n)**, we need to
to get rid of about **O(n/2)** at each constant comparisons

Practice Question 3

Definition: Median of an array is a point at position **m** i.e **Med = A[m]** such that almost **half** of A is **less** than **A[m]** and other **half** is **greater** than **A[m]**

Let m_a and m_b be the medians of A and B:

$$A[1...n] = [A_L, m_a, A_R]$$

$$B[1...n] = [B_L, m_b, B_R]$$

Practice Question 3

Observation: We can remove **equal** number of elements from **two sides of the median**

Example:

A = 1, 12, 4, 9, 3, 16, 28, 19, 13, 21, 18, 24, 70

1, 12, 4, 9, 3, 13, 16, 28, 19, 21, 24, 70, 18

12, 4, 9, 16, 19, 24, 70

12, 16, 24

Practice Question 3

$A_L < m_a < A_R$

$B_L < m_b < B_R$

Practice Question 3

Case 1: $m_a = m_b$

$\{A_L, B_L\} < \{m_b\ m_a\} < \{B_R, A_R\}$

Overall median must be between m_a and m_b since there are **n/2 + 1** element before m_a and **n/2 + 1** after m_b.
Thus we can remove equal number of elements from both sides.

$m_b\ m_a$

Therefore **$m_a = m_b$ = overall median**

Practice Question 3

Case 2: $m_a > m_b$

$$\{ B_L \} < m_b < m_a < \{ A_R \}$$

A_L B_R

- **Overall median** must be smaller than $\mathbf{m_a}$ because $|A_L| + |B_L| + 1_{(mb)}$ = n/2 + n/2 + 1 = n + 1 elements are less than $\mathbf{m_a}$
- **Overall median** must be bigger than $\mathbf{m_b}$: because $|B_R| + |A_R| + 1_{(ma)}$ = n/2 + n/2 + 1 = n + 1 elements are bigger than $\mathbf{m_b}$

Practice Question 3

Since overall median is between $\mathbf{m_b}$ and $\mathbf{m_a}$
we can remove equal number of elements
from both sides of median
which are $\mathbf{B_L}$, $\mathbf{A_R}$, $\mathbf{m_a}$ and $\mathbf{m_b}$.

Thus: median(A, B) = median($\mathbf{A_L}$, $\mathbf{B_R}$)

Practice Question 3

Case 3: $m_a < m_b$

$\{ A_L \} < m_a < mb < \{ B_R \}$

B_L A_R

- Overall median must be smaller than $\mathbf{m_b}$
- Overall median must be bigger than $\mathbf{m_a}$

Since overall median is between $\mathbf{m_a}$ and $\mathbf{m_b}$, we can remove whatever
number of elements from both sides which are
$\mathbf{A_L}$, $\mathbf{B_R}$, $\mathbf{m_a}$ and $\mathbf{m_b}$.

Thus: median(A, B) = median(A_R, B_L)

Practice Question 3

$$
\begin{array}{l}
\textbf{function } median2(a, b) \\
\quad if\ n \le 2 \\
\qquad explicitly\ find\ the\ median\ and\ return\ it \\
\quad compute\ the\ median\ of\ a\ and\ b\text{: } m_1\ and\ m_2\ respectively \\
\quad if\ m_1 == m_2\text{:} \\
\qquad return\ m_1 \\
\quad else\ if\ m_1 > m_2\text{:} \\
\qquad return\ median2(a_L, b_R) \\
\quad else\text{:} \\
\qquad return\ median2(a_R, b_L)
\end{array}
$$

T(n) = T(n/2) + O(1) = O(log n)

Question -4

Practice Question 4

Input: A[1..n] **Unimodal**
Output: Find index p in [1..n] which is the modality point with complexity **O(log n)**

For **[1,2,5,9,7,3]** the modality point is **p = 4**

Hint: Again for **O(log n)** we need to get rid of half of array at each constant comparisons

Practice Question 4

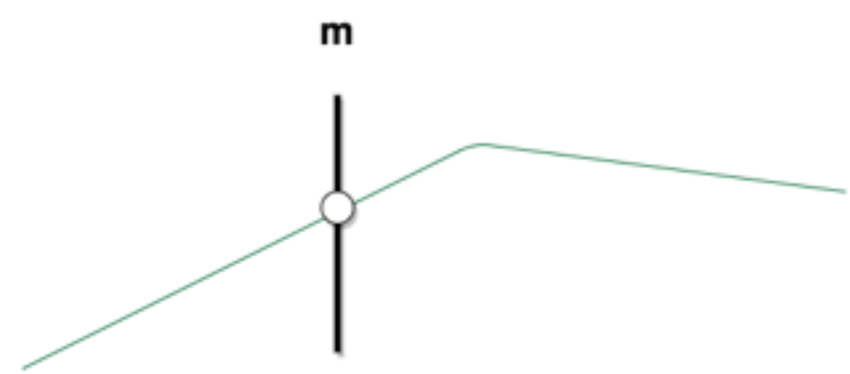
For each point at index **m** there are three cases:

- **A[m+1] < A[m] :**

Turning point **t** is between **[1, m]**

- **A[m+1] > A[m] :**

Turning point is between **(m, n]**



- **|A| = 1 return A[1]**

Practice Question 4

$\texttt{algo}(A[1, \ldots, n])$

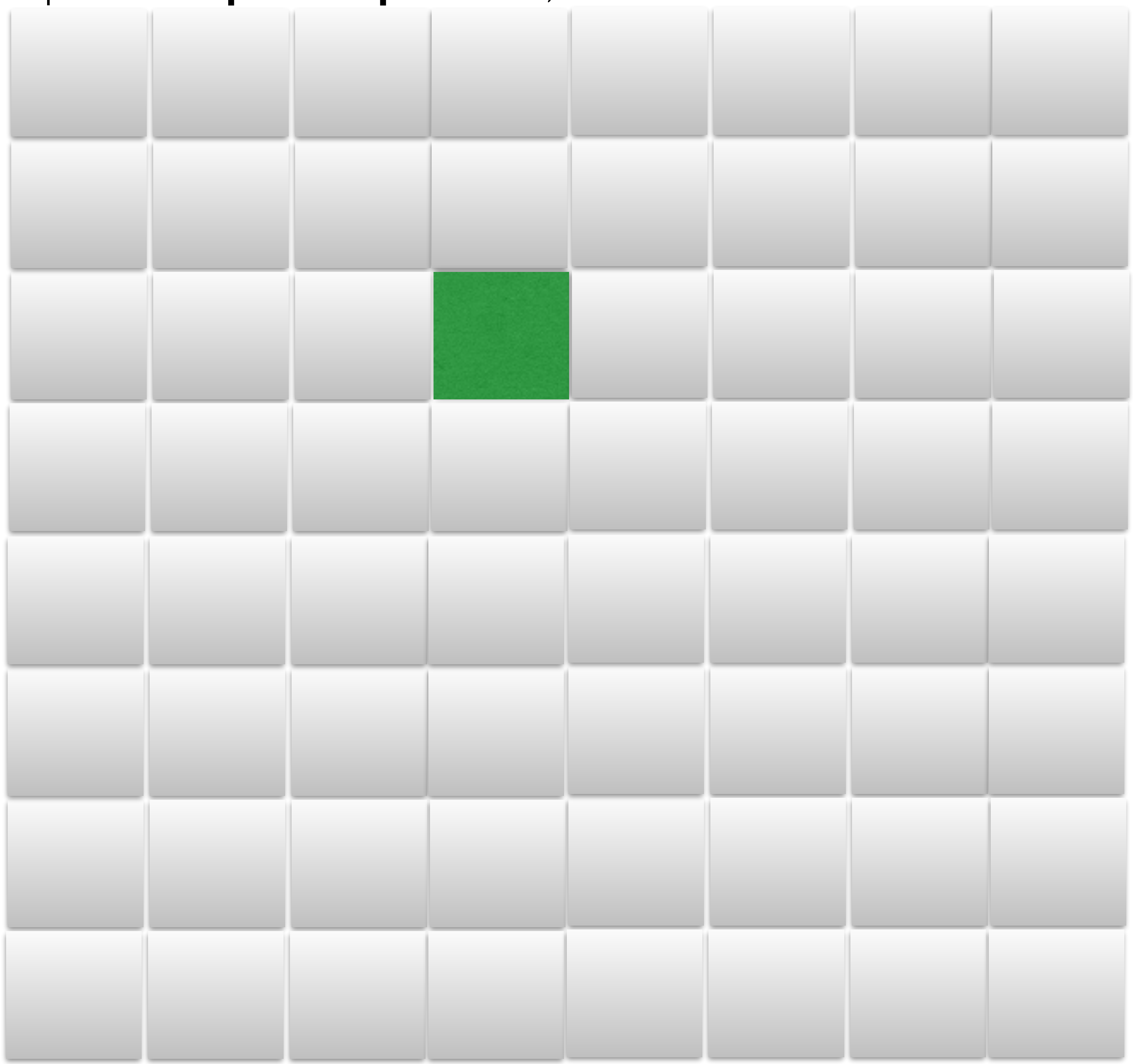
1. *if* $n == 1$, *return* **A[n]**
2. $m = \frac{n}{2}$
3. *if* $A[m] < A[m+1]$ *return* $\texttt{algo}(A[m+1, \ldots, n])$
5. *else return* $\texttt{algo}(A[1, \ldots, m])$

We choose **m = n/2** to get **O(log n)**

Question -5

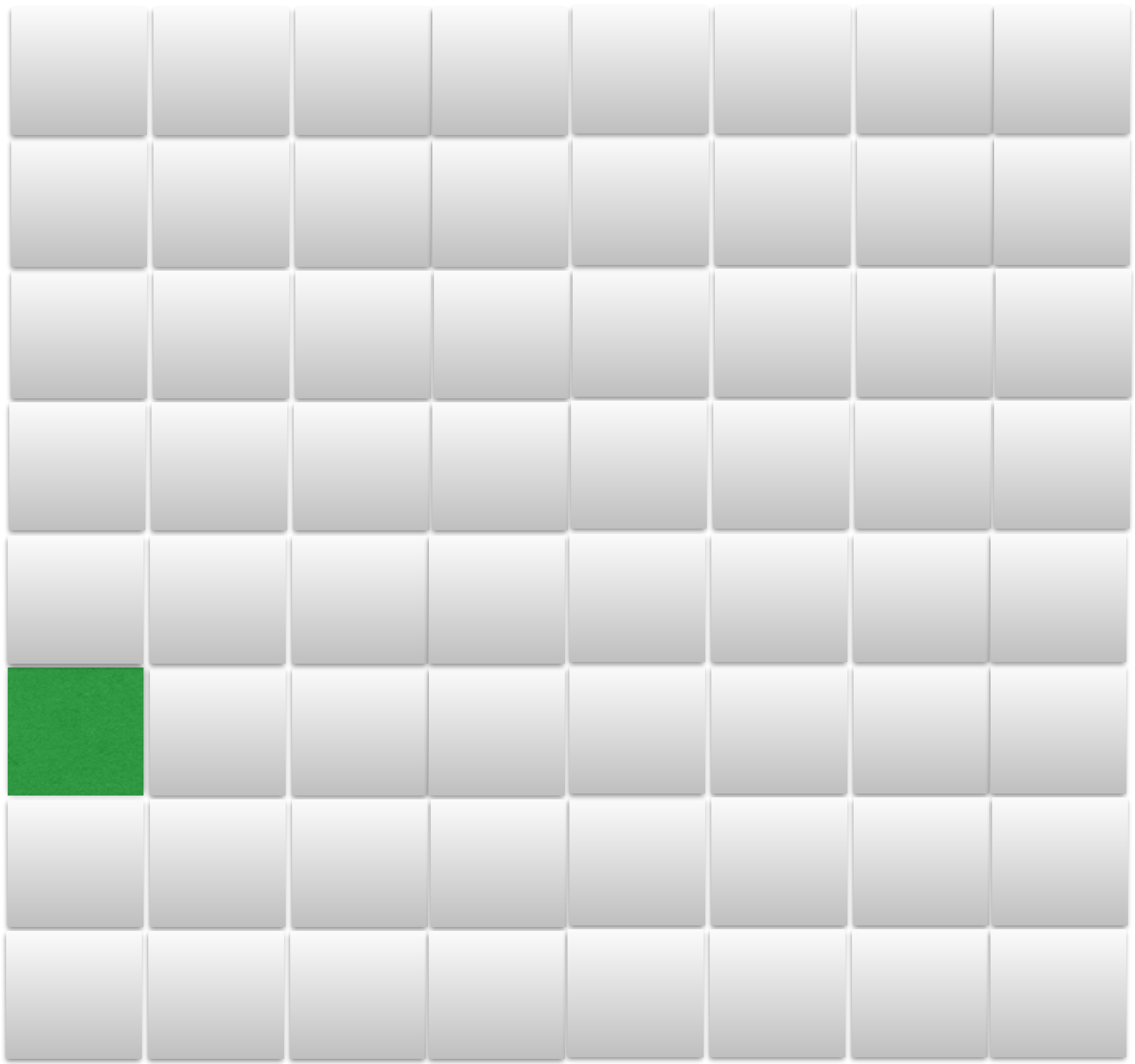
Practice Question 5

Tromino Puzzle : For a board of $2^{n+1} * 2^{n+1}$ with **one missing** square **at optional position**, fill the board with **L** tiles.



Practice Question 5

Missing square can be anywhere

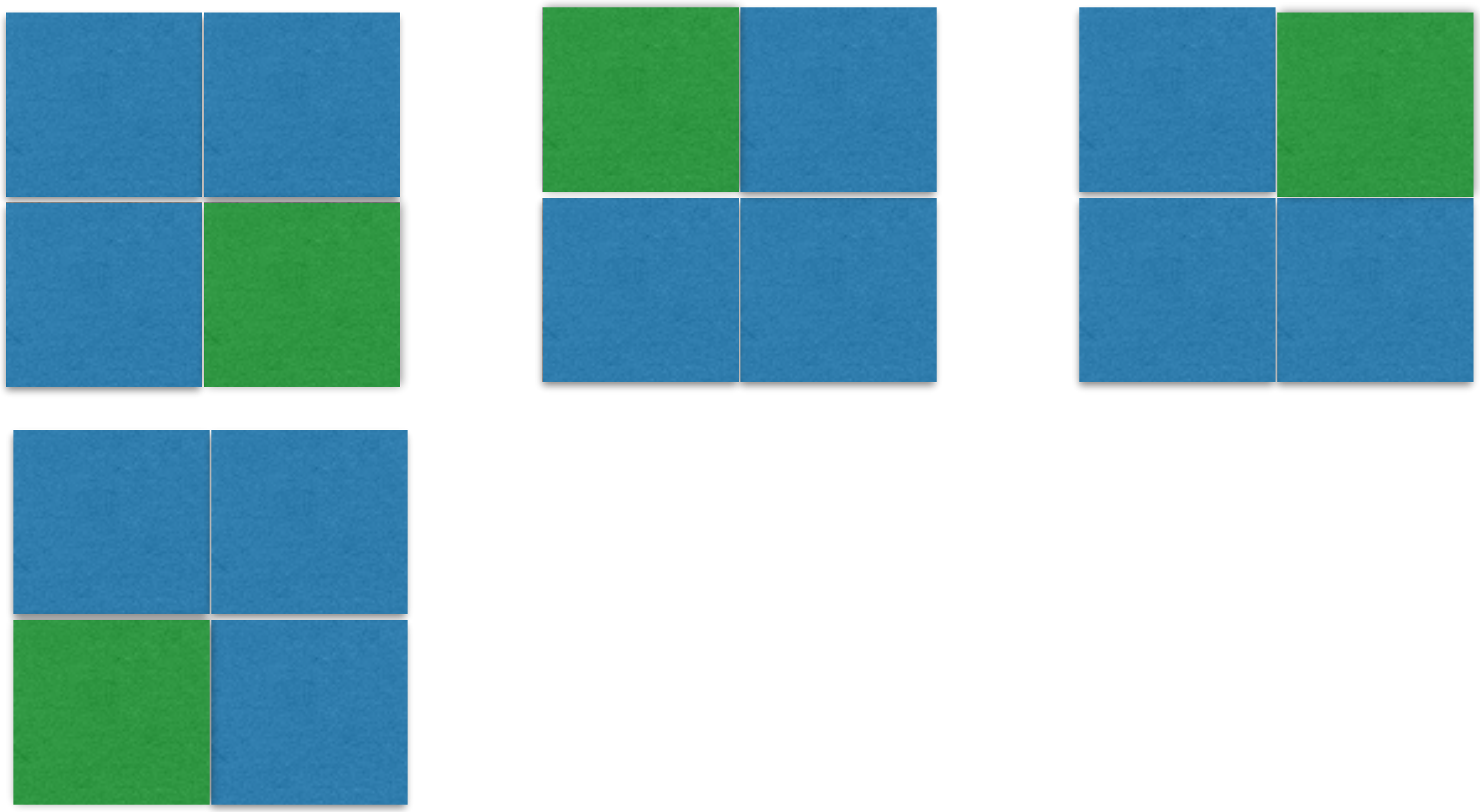


Practice Question 5

Divide & conquer solution

Practice Question 5

Base Case : For board of $2^1 * 2^1$ we can solve regardless of the missing square

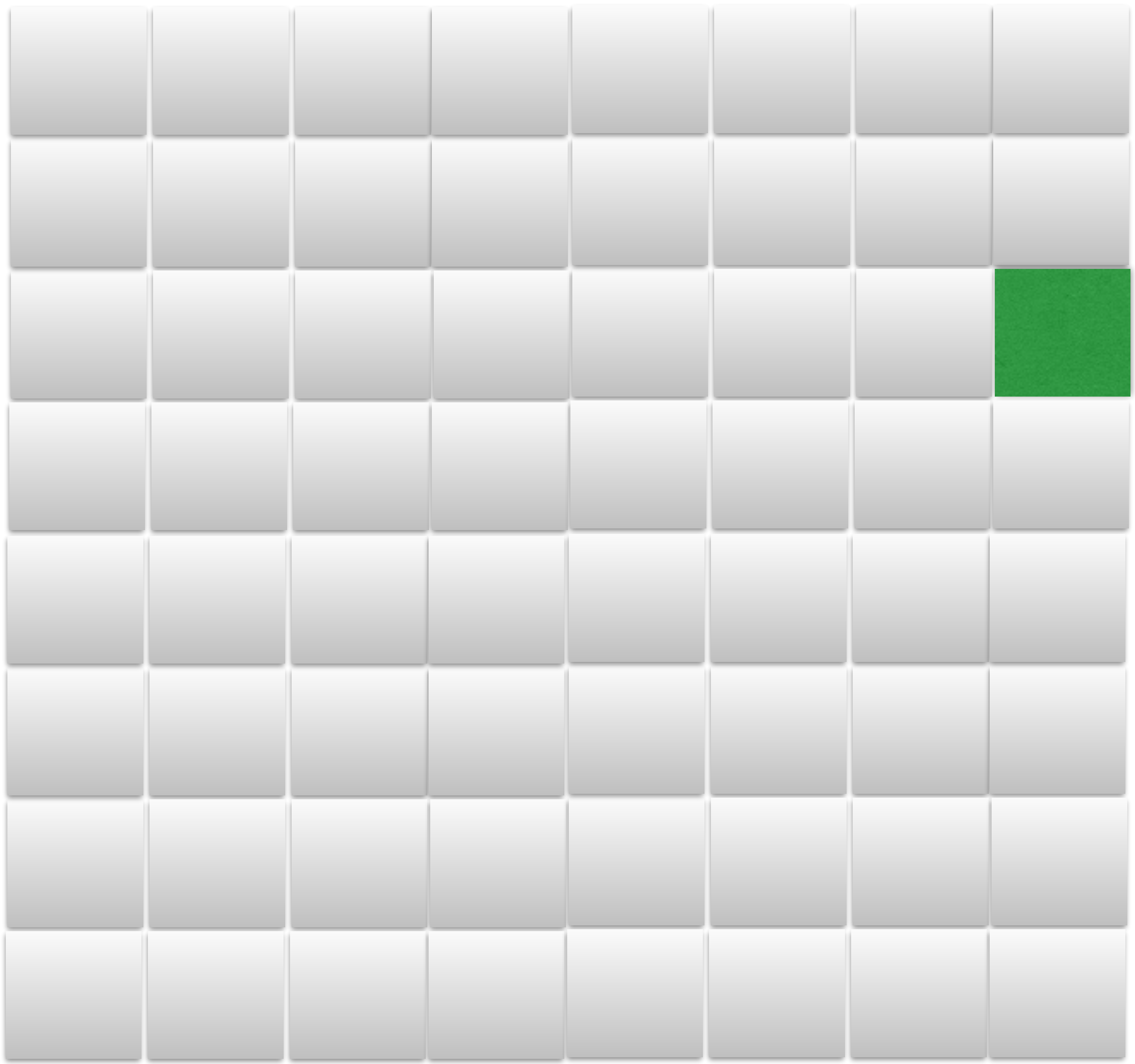


Practice Question 5

Hypothesis : We can solve for $2^n * 2^n$ with any optional missing square

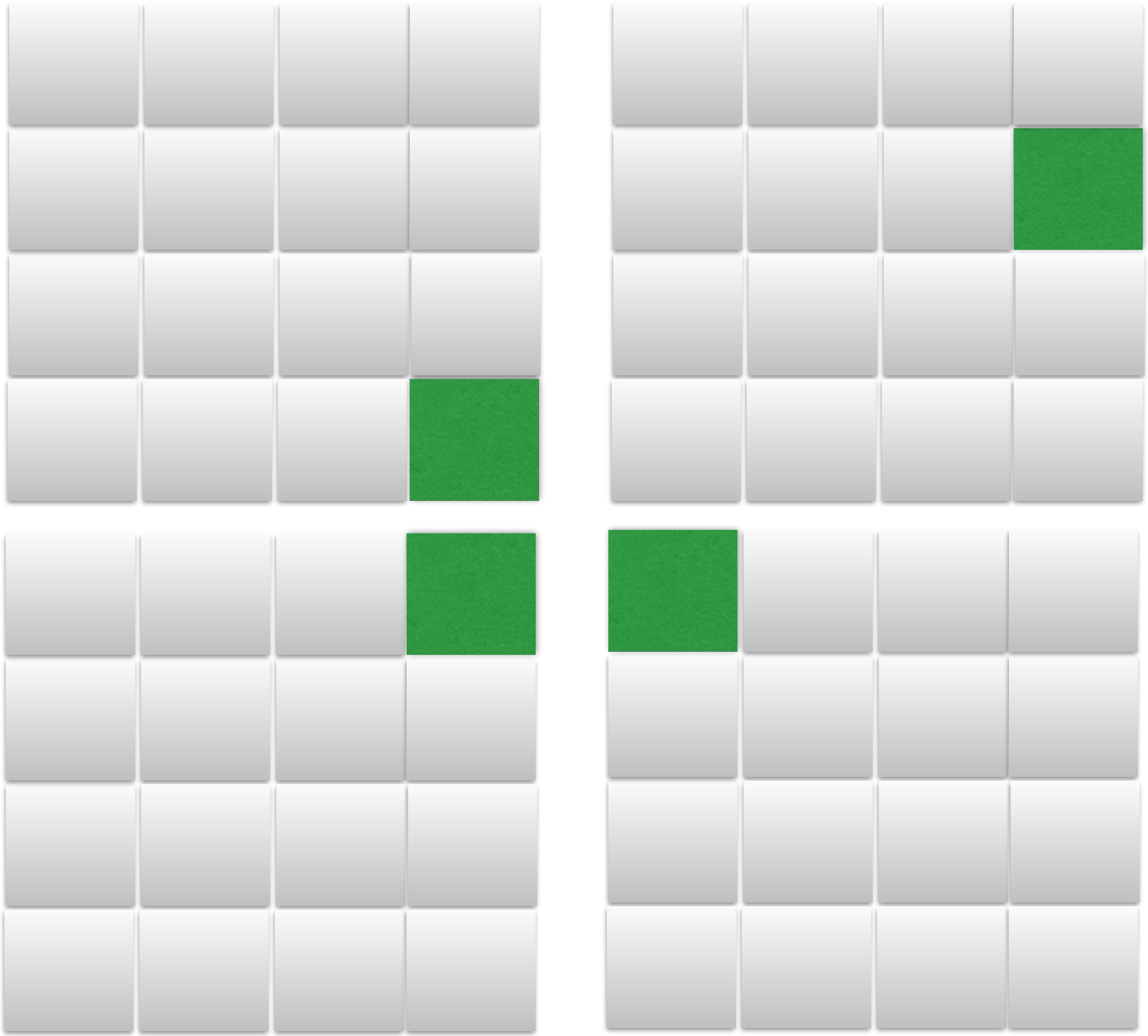
Practice Question 5

Let the missing square be



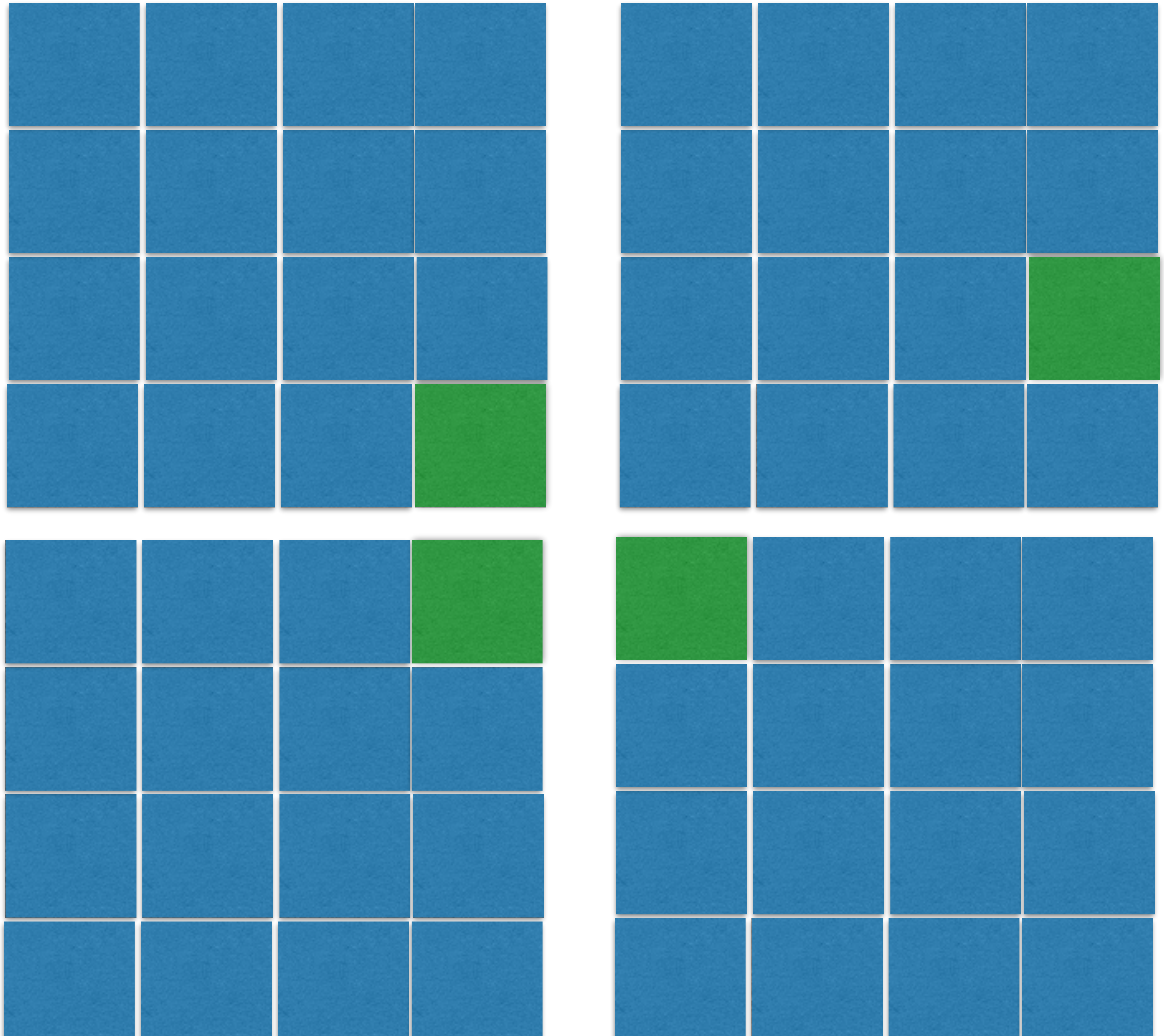
Practice Question 5

Induction Step: For $2^{n+1} * 2^{n+1}$ generate 4 quartiles of $2^n * 2^n$ as subproblems



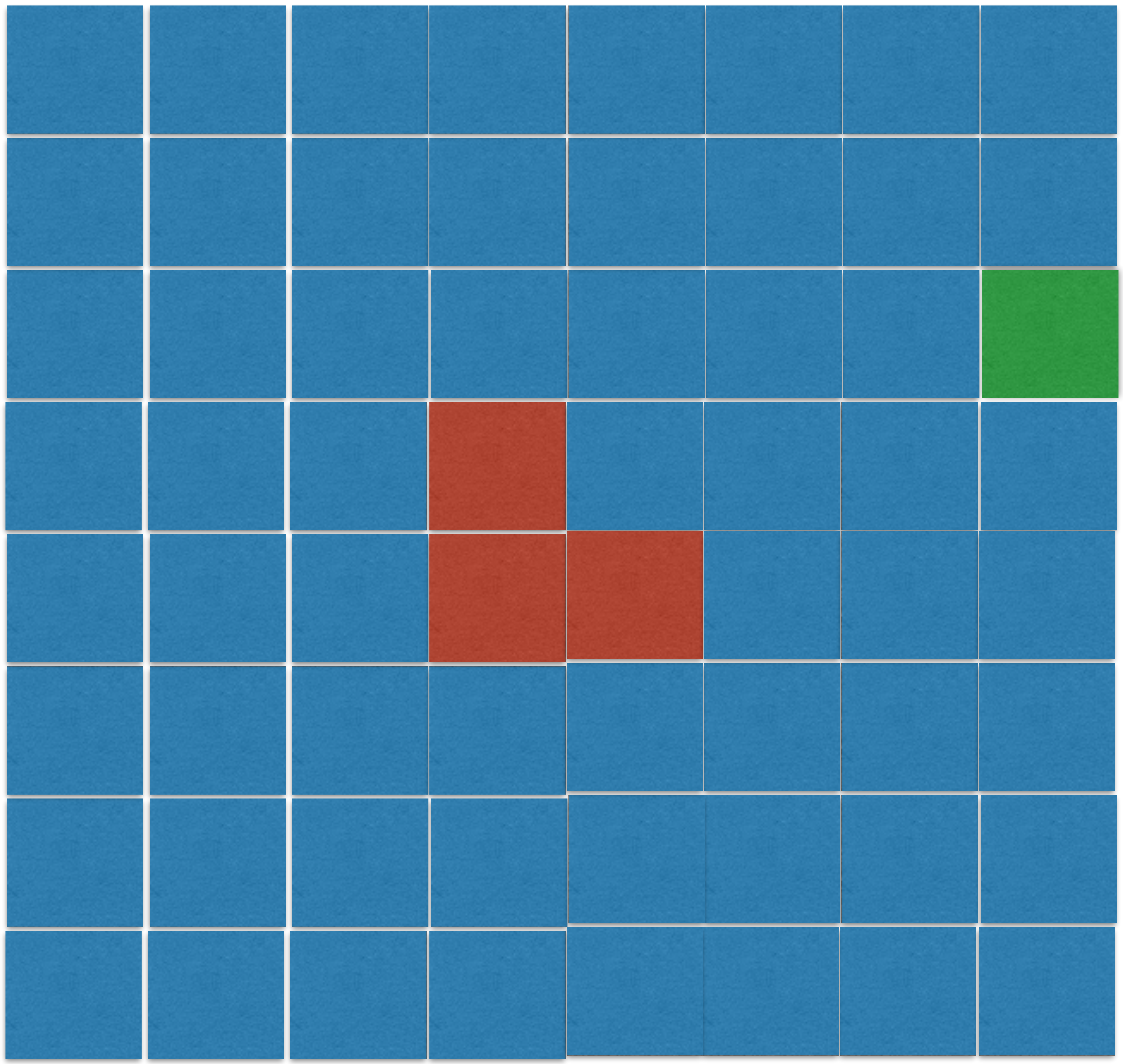
Practice Question 5

Step 2: Solve subproblems



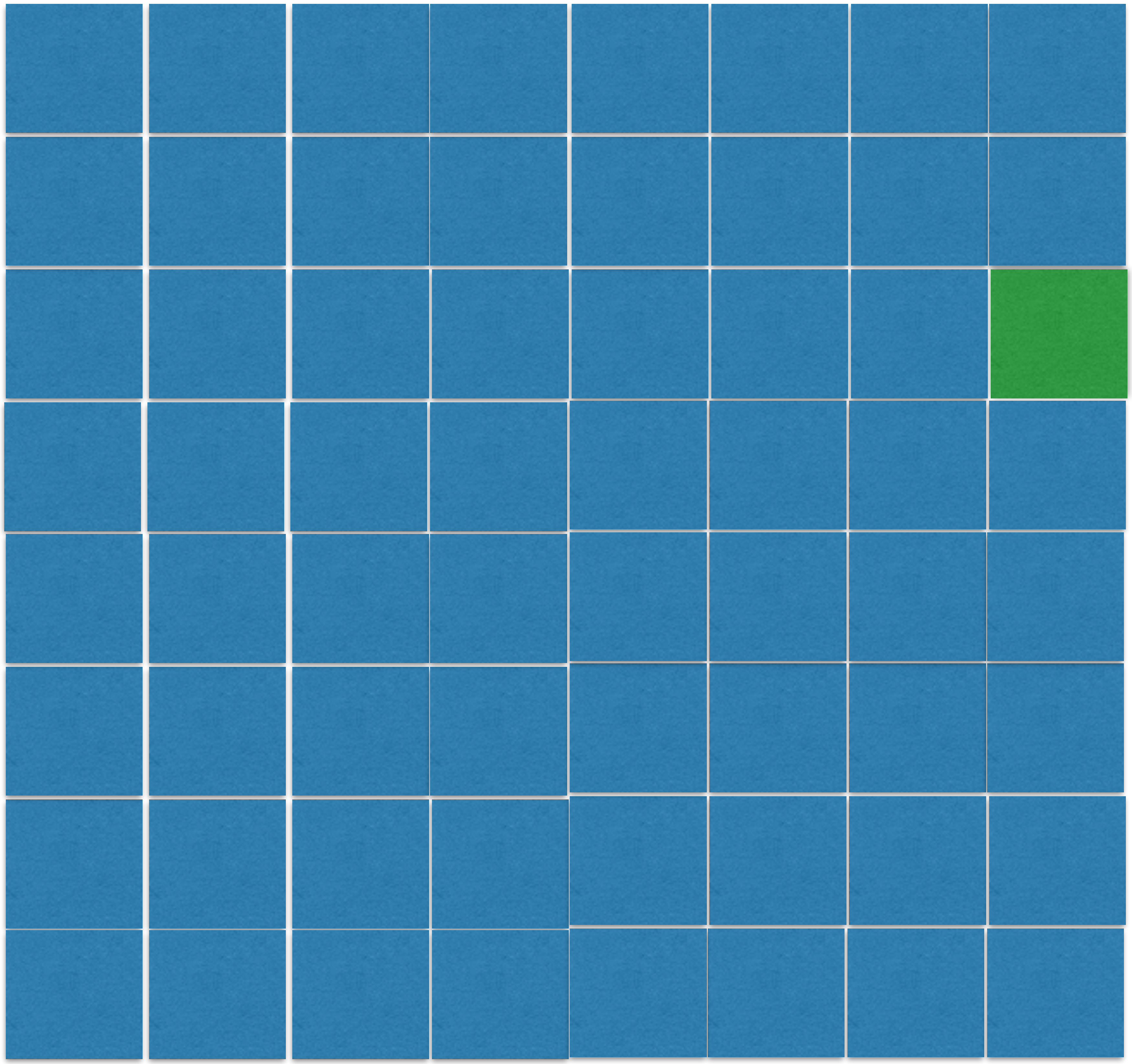
Practice Question 5

Step 3: Put a tile **L** as follows



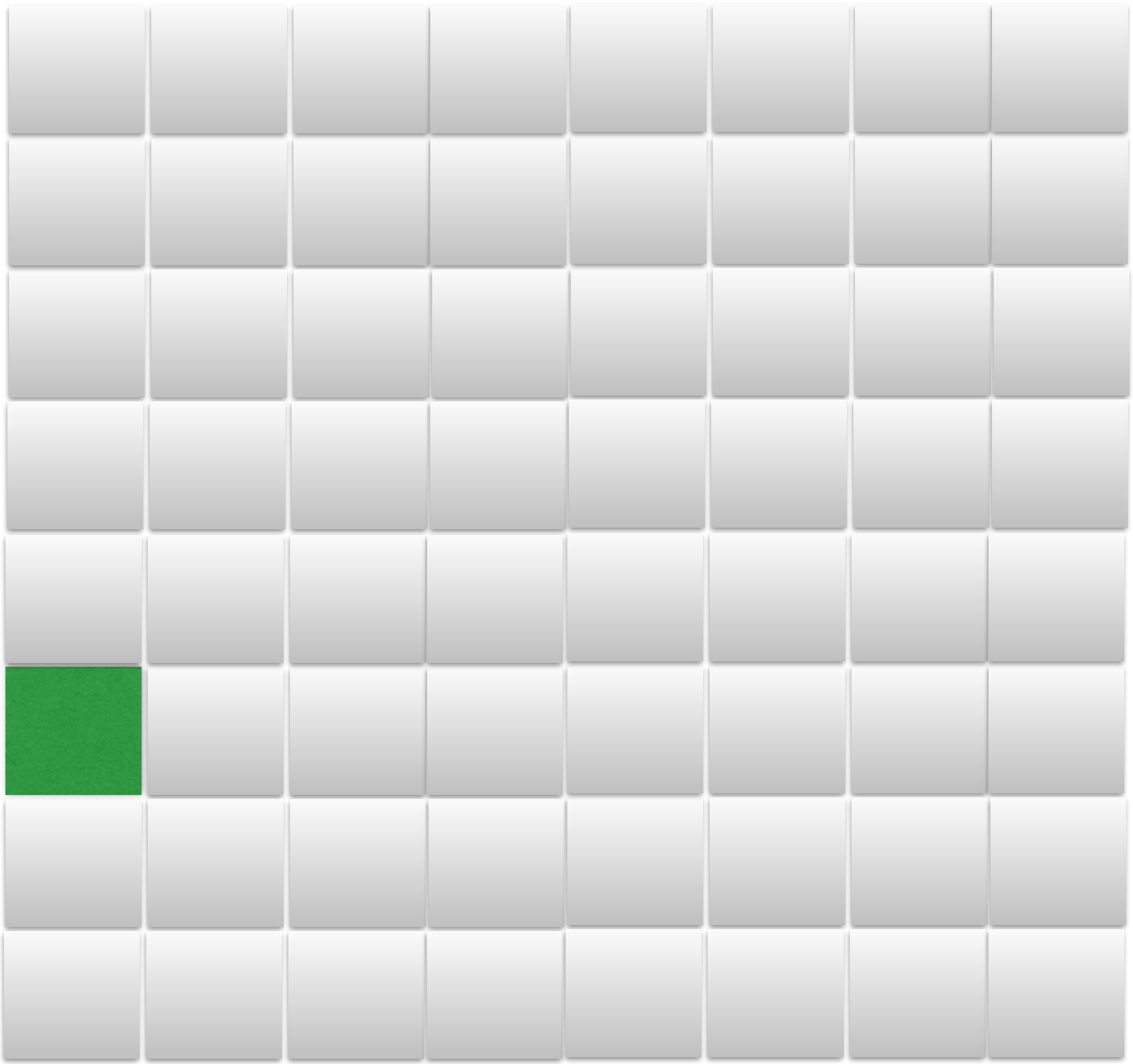
Practice Question 5

Step 4: return



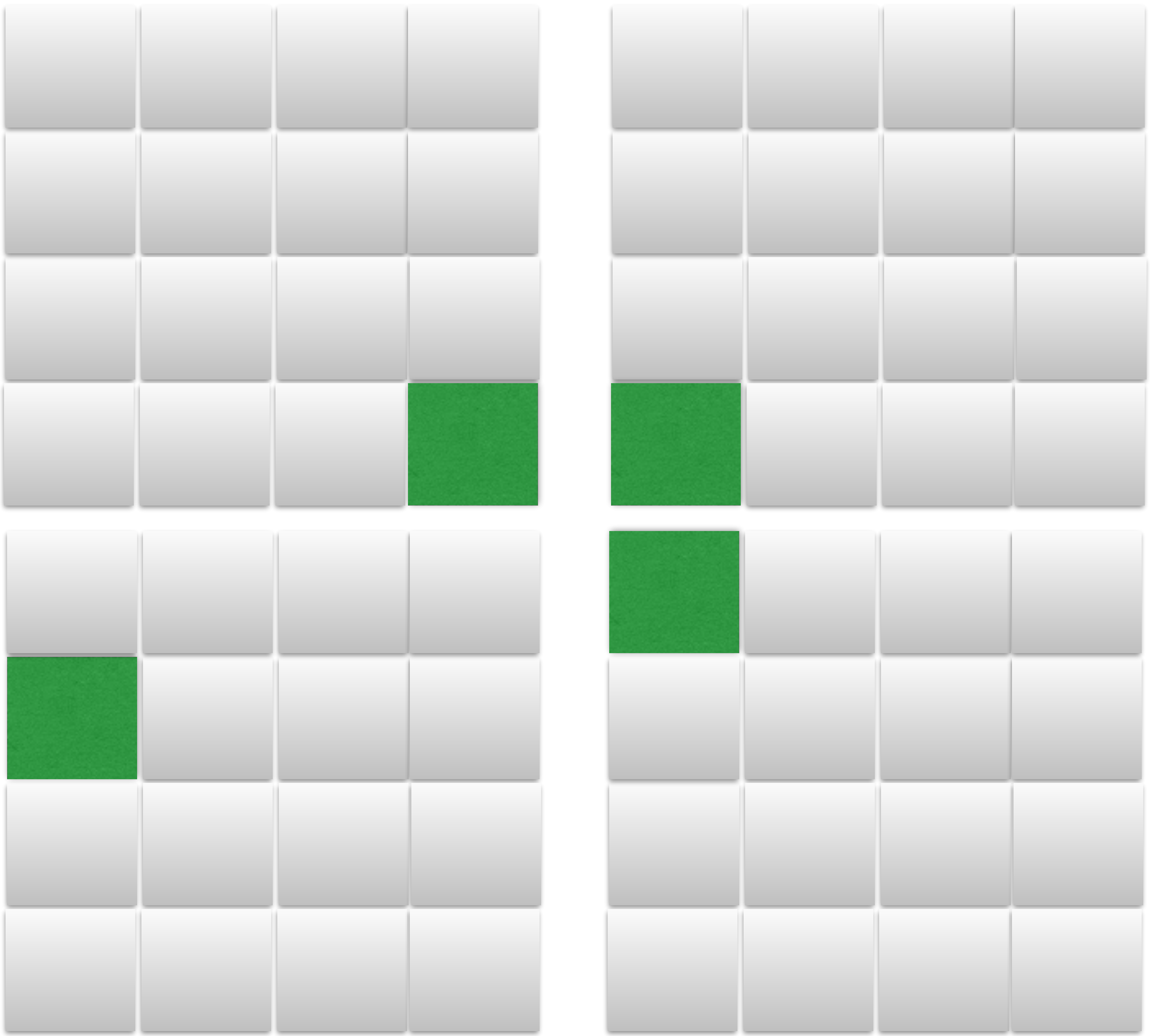
Practice Question 5

Another example



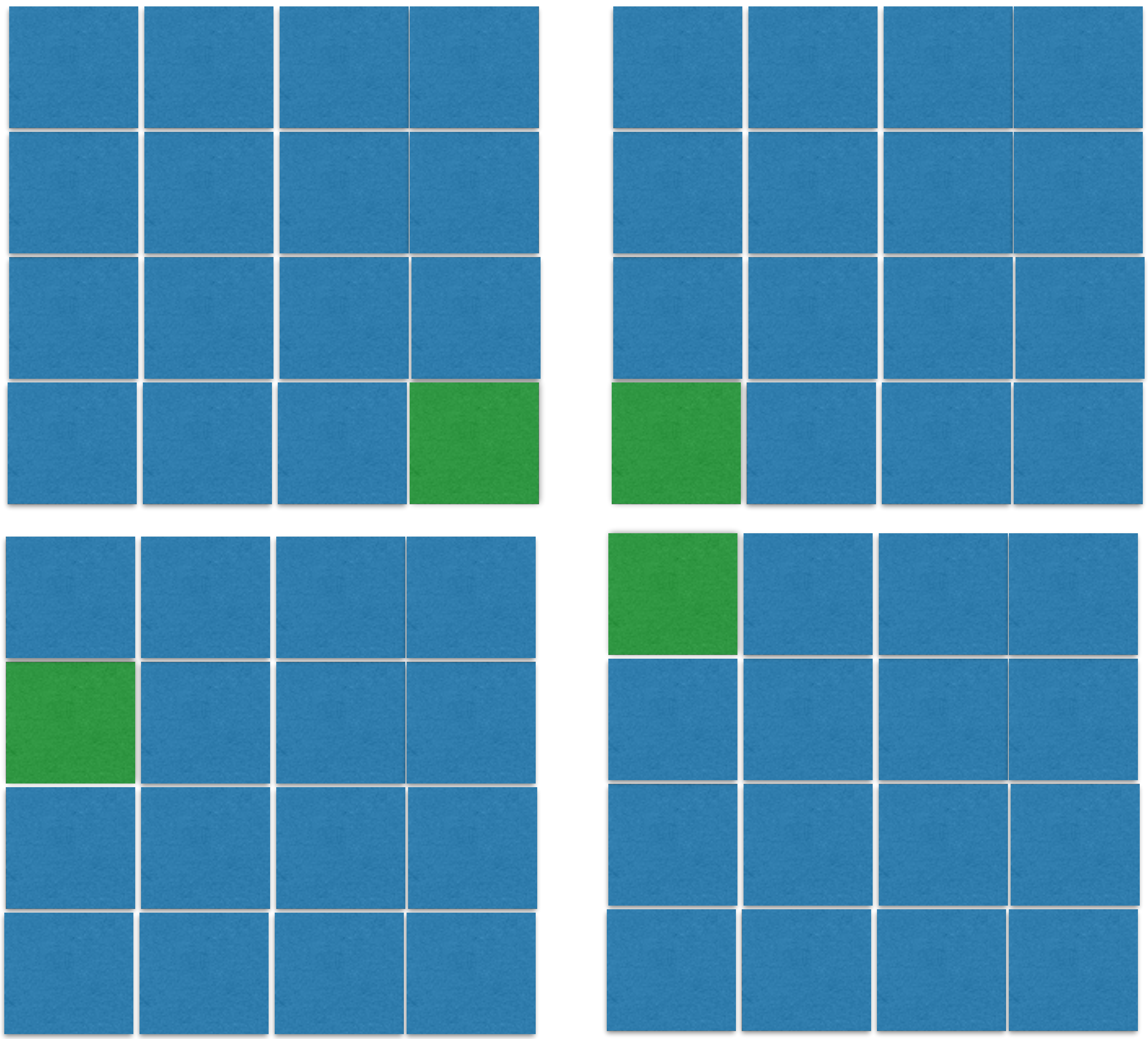
Practice Question 5

generate 4 subproblems



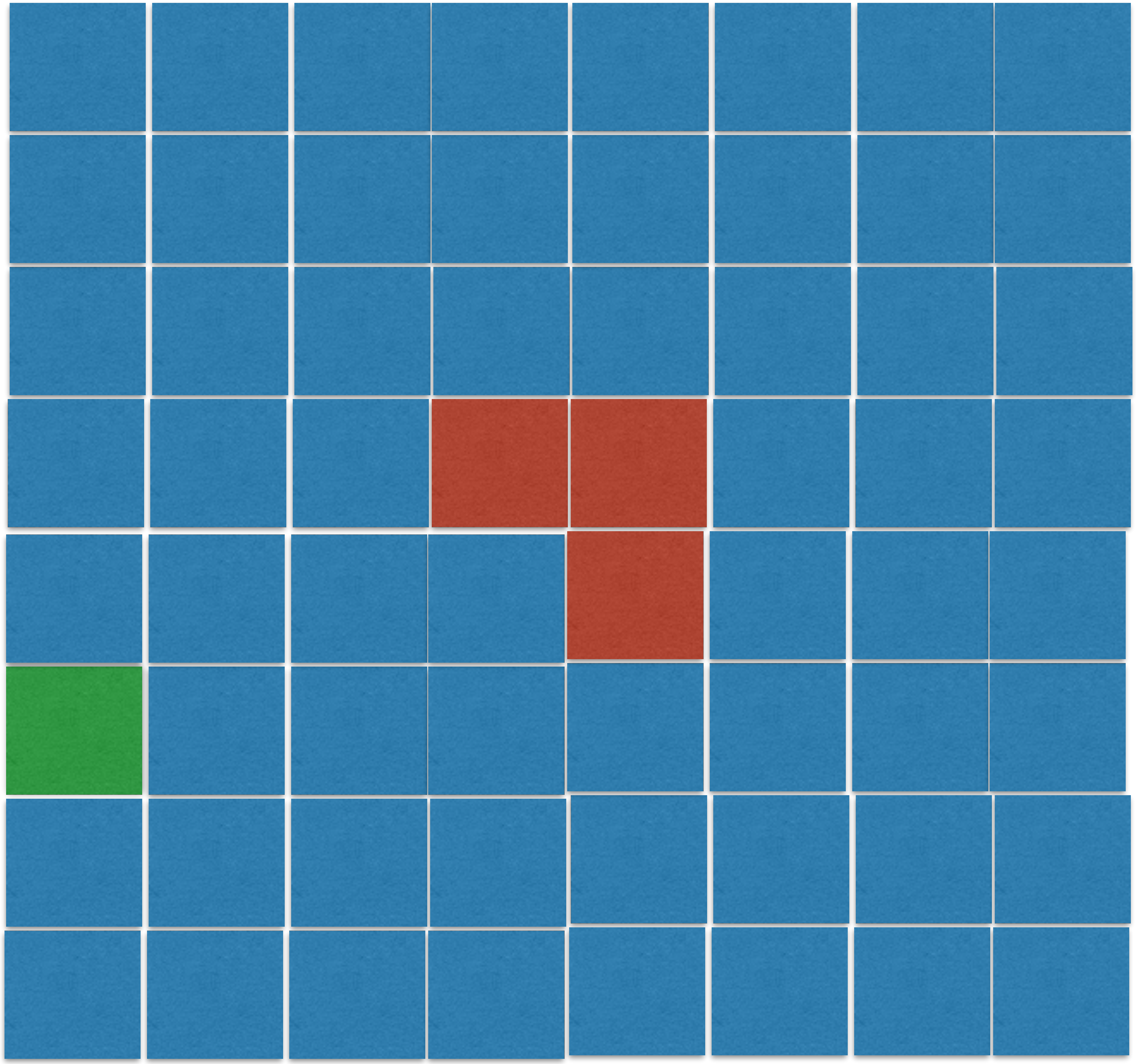
Practice Question 5

Solve subproblems



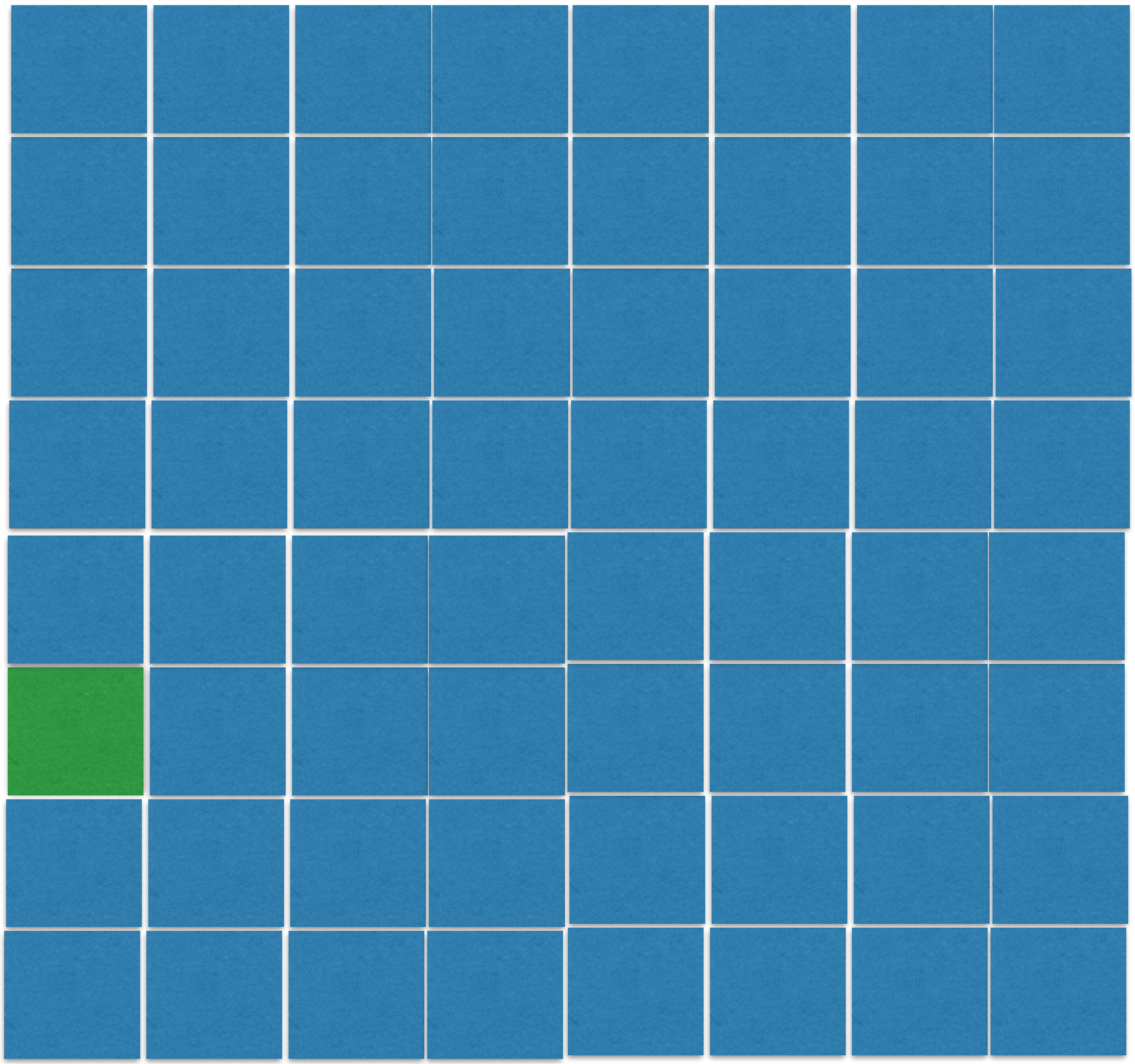
Practice Question 5

Add new L tile



Practice Question 5

return



The position of new tile with respect to the missing square is important

$$T(n) = 4\,T(n/4) + O(1)$$

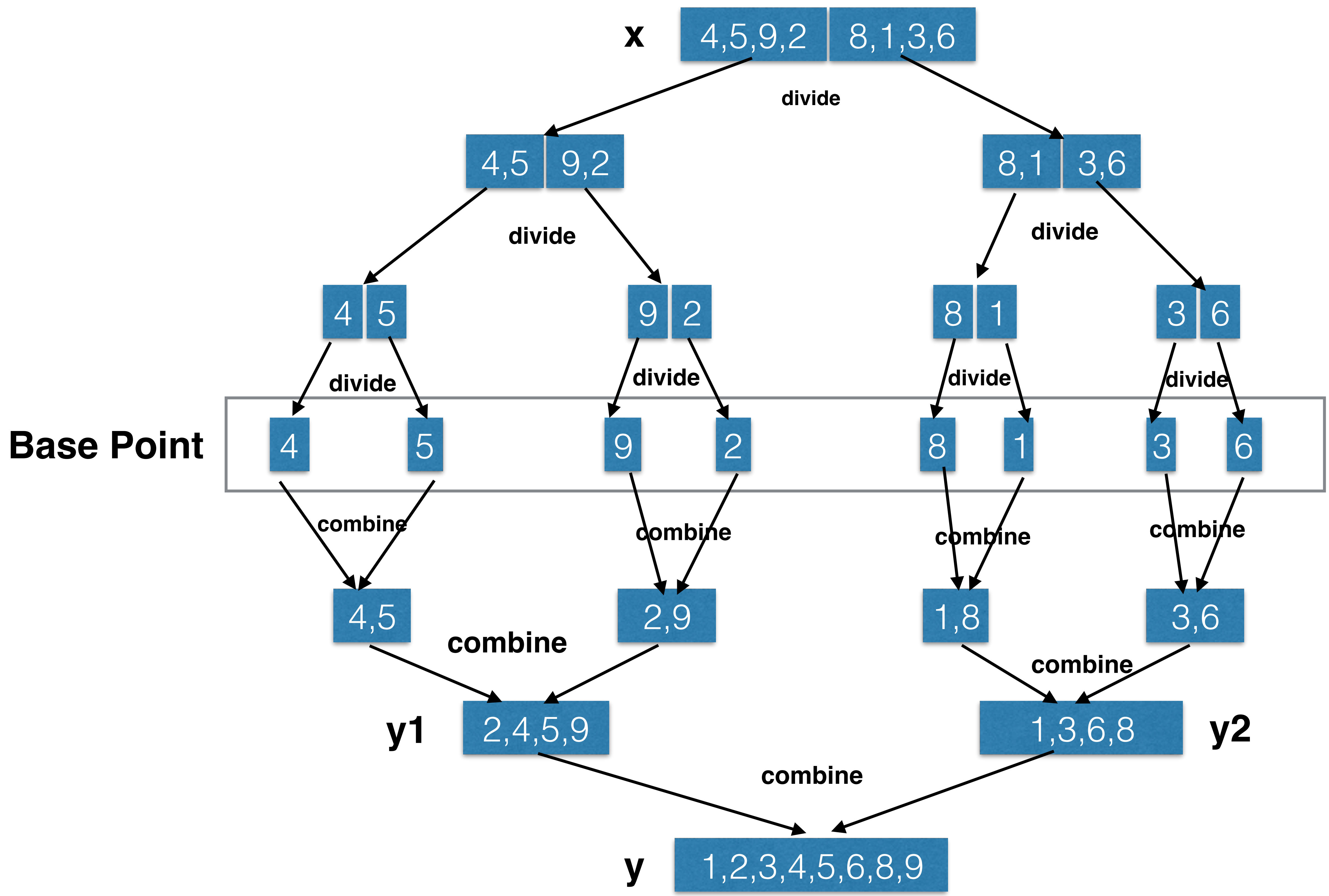
Other Materials

Example: Merge Sort

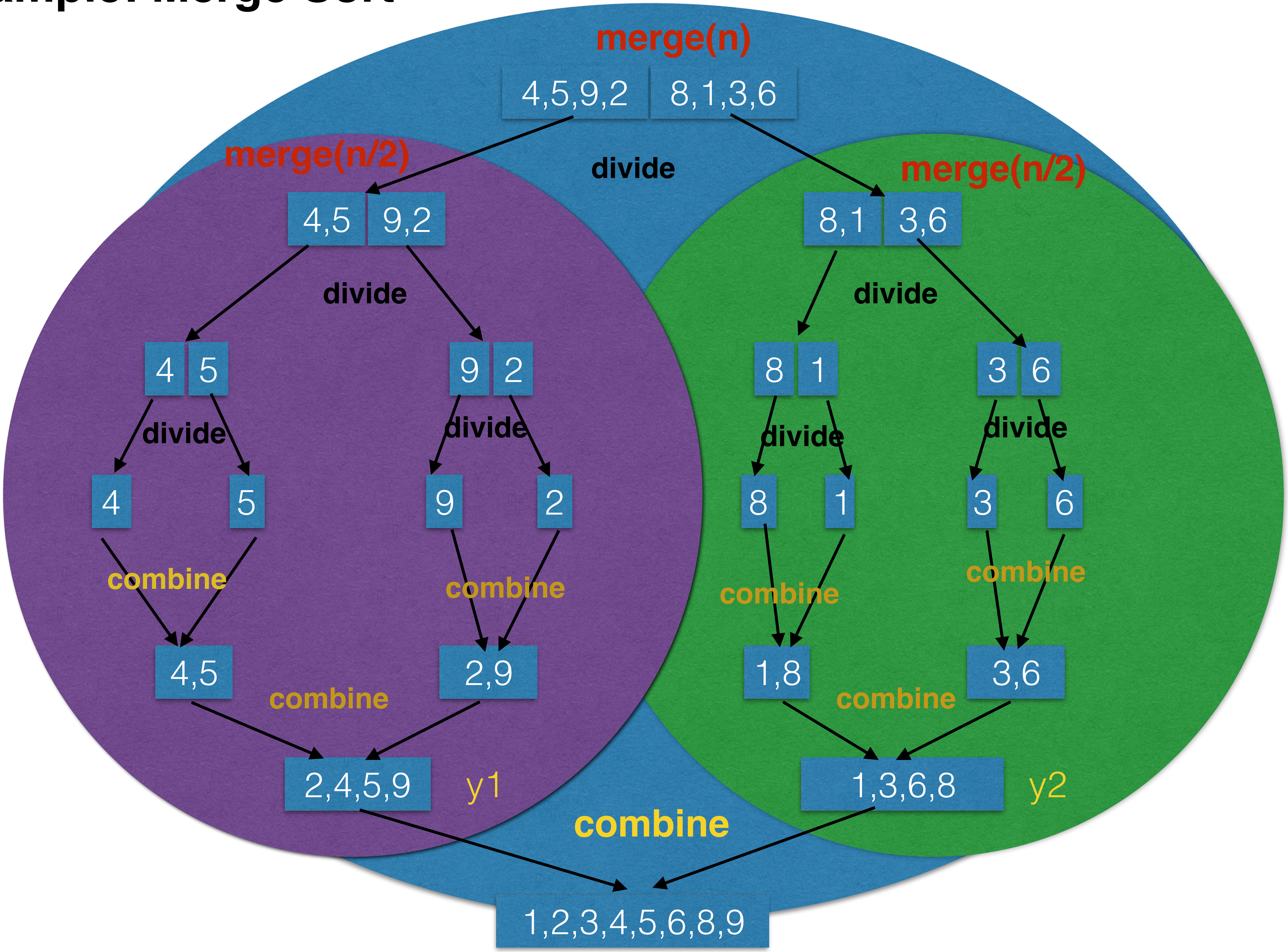
```
Merge(x):
    If len(x) = 1:
        return x
    y1= Merge(x[1..n/2])
    y2= Merge(x[n/2+1,...n])
    y= Combine( y1, y2)
    return y
```

$T(n) = 2\, T(n/2) + O(n)$

Example: Merge Sort



Example: Merge Sort



Merge Sort: Time Complexity:

T(n) = T(n/2) + T(n/2) + T(combine_n)

=

T(n/4) + T(n/4) + T(combine_n/2)

+

T(n/4) + T(n/4) + T(combine_n/2)

+

T(combine_n)

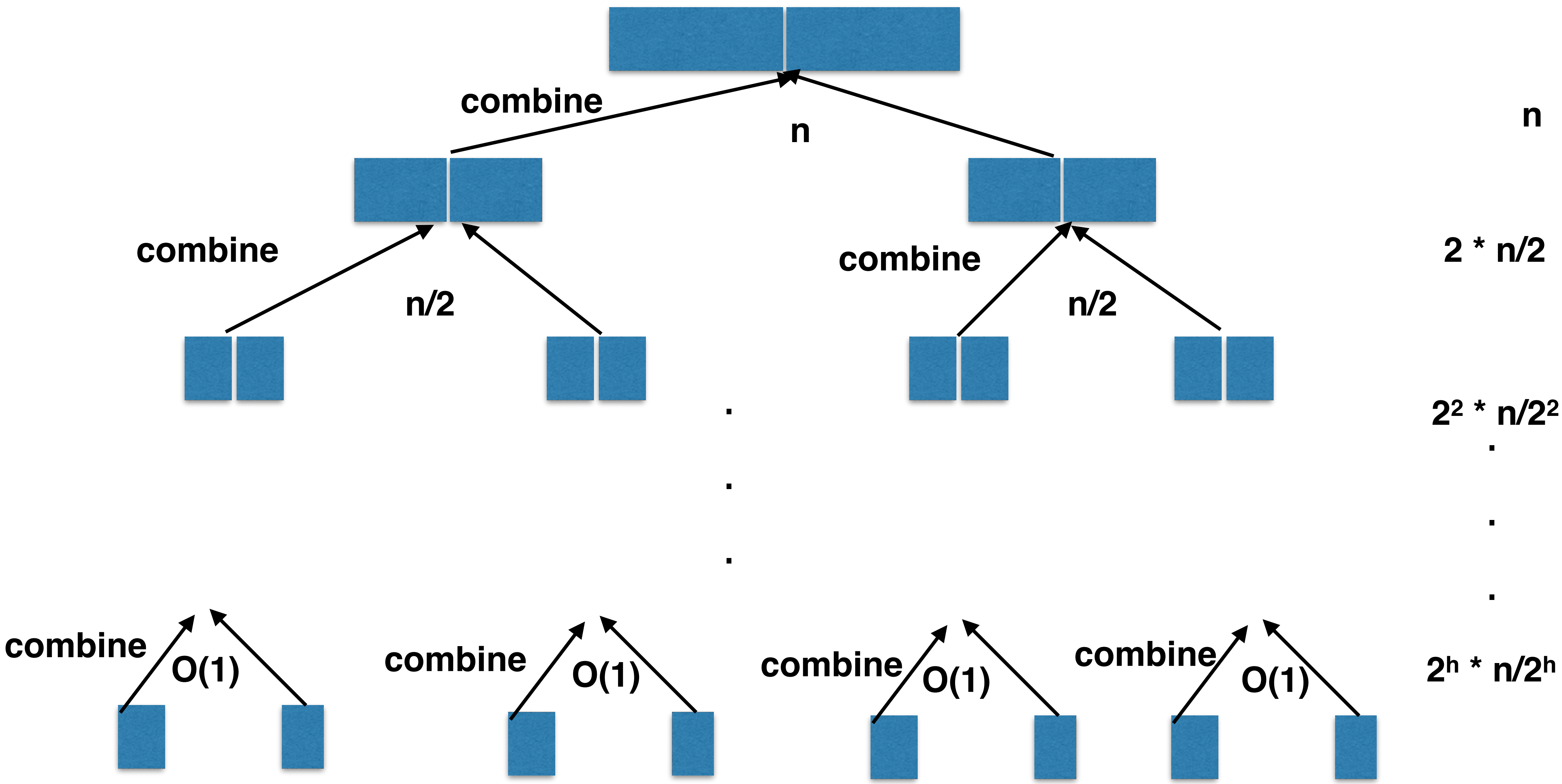
.

.

.

= Sum(T(internal combines))

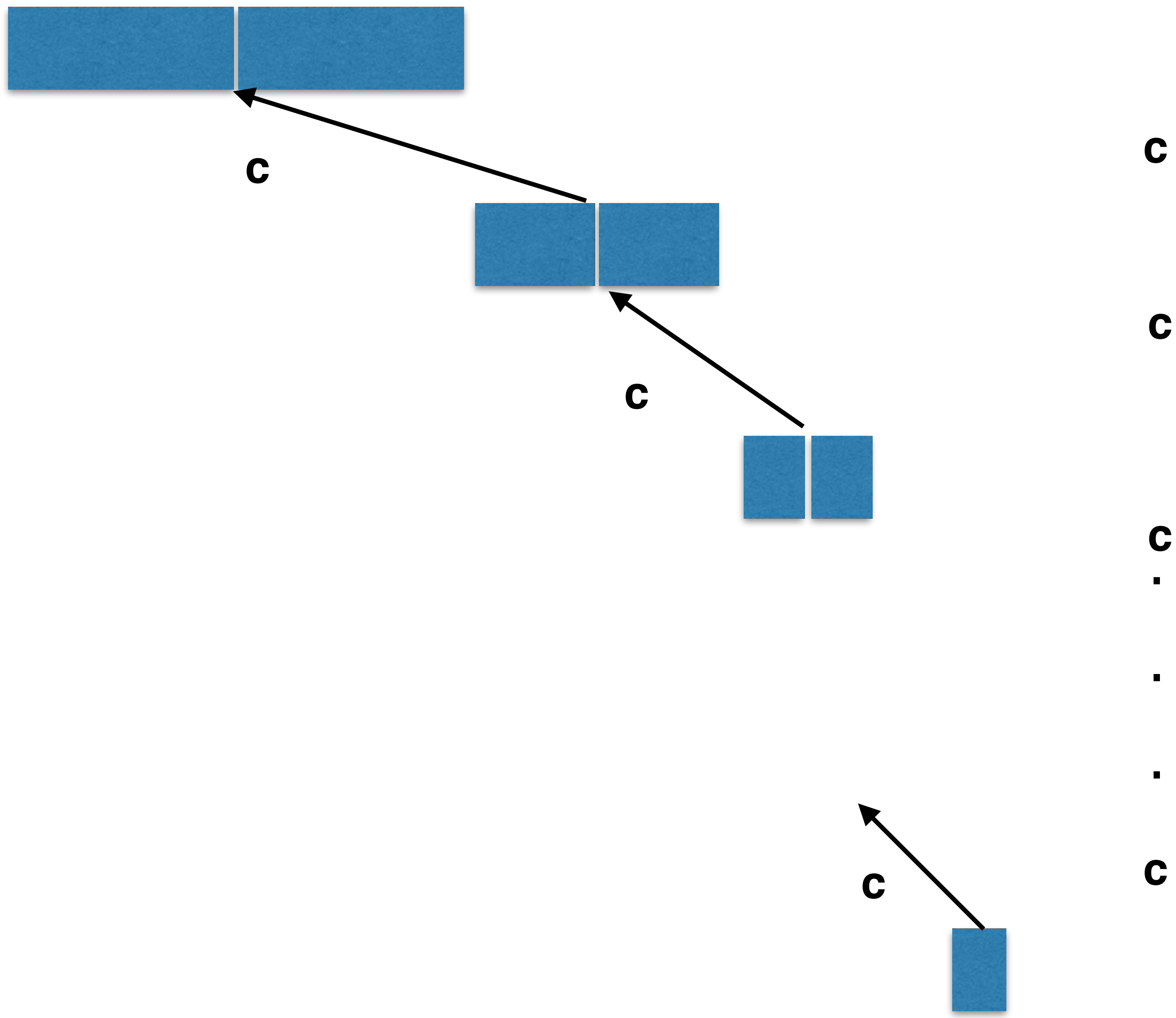
$T(n) = 2\,T(n/2) + O(n)$



$n/2^h = 1$ therefore $h = \log_2 n$

sum(internal combines) = $n + 2\ \ n/2 + 2^2\ n/2^2 + \ldots\ 2^h\ n/2^h = n\ h = n \log n$

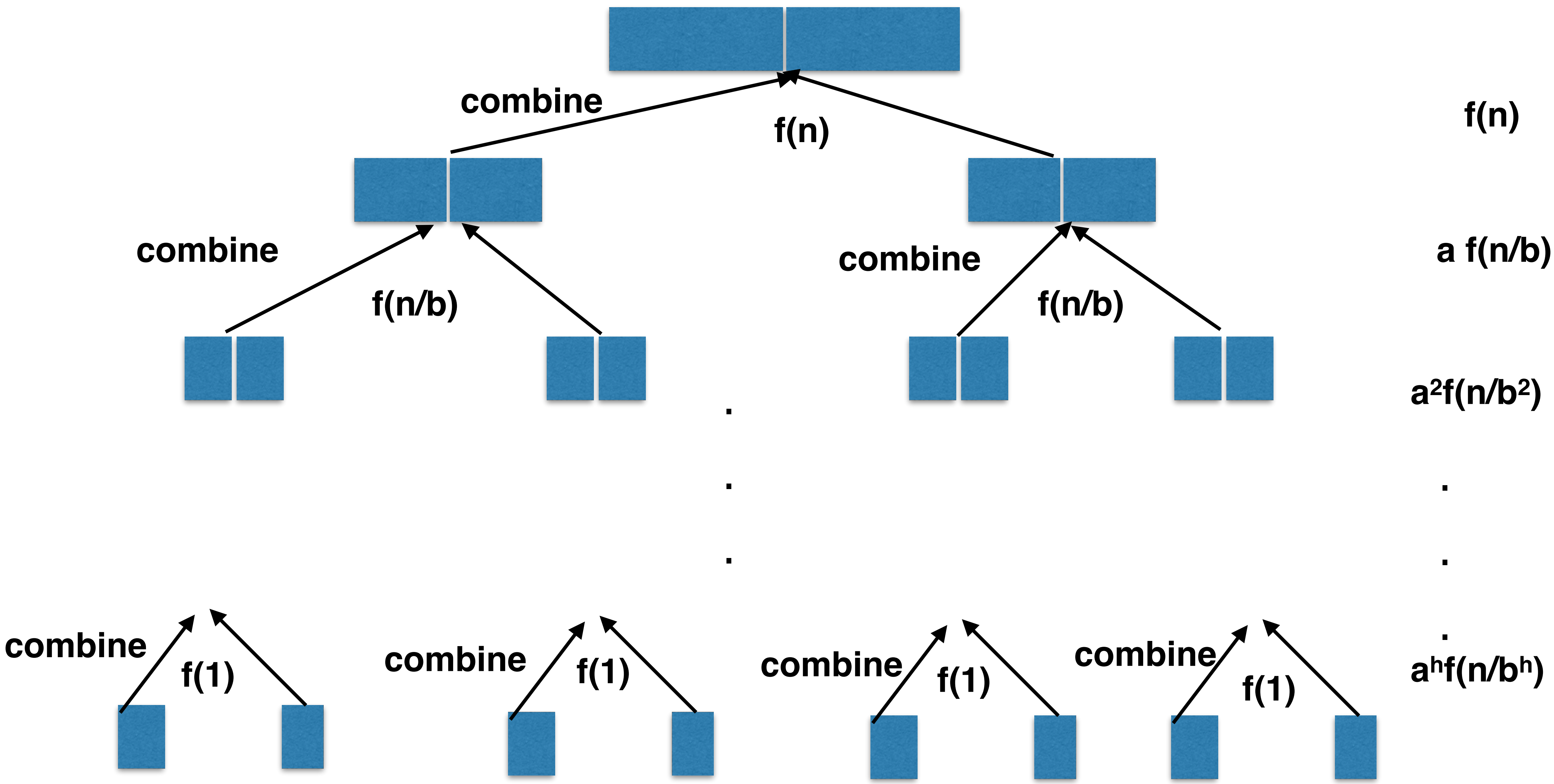
$T(n) = T(n/2) + c$



$n/2^h = 1$ therefore $h = \log_2 n$

sum(internal combines) = c + c + c + … c = c h = c log n

$$T(n) = aT(n/b) + f(n)$$



$n/b^h = 1$ therefore $h = \log_b n$

sum(internal combines) =

$$f(n) + af(n/b) + a^2f(n/b^2) + \ldots\ a^hf(n/b^h) =$$

$$f(n) + af(n/b) + a^2f(n/b^2) + \ldots\ a^{\log_b n} f(n / b^{\log_b n})$$

$$T(n) = aT(n/b) + f(n)$$

$T(n)=$

$f(n) + af(n/b) + a^2f(n/b^2) + \ldots.\ a^{\log_b n} f(n / b^{\log_b n}) =$

$f(n) + af(n/b) + a^2f(n/b^2) + \ldots.\ n^{\log_b a} f(n / n^{\log_b b}) =$

$f(n) + af(n/b) + a^2f(n/b^2) + \ldots.\ n^{\log_b a}$

$\downarrow$

$$T(n)= f(n) + af(n/b) + a^2f(n/b^2) + \ldots.\ n^{\log_b a}$$

Master Theorem

1. **Root Dominant**: f(n) is dominant then T(n) = f(n)
2. **Leave Dominant**: Term $n^{\log_b a}$ is the dominant therefore $T(n) = n^{\log_b a}$
3. **Even Terms:** $f(n) = \text{theta}(n^{\log_b a})$ therefore
   T(n) = number of terms * f(n)= $\log n_b$ * f(n)

$T(n) = a\,T(n/b) + c\,n^d$

1.root term $= f(n) = c\,n^d$
2. leave term $= n^{\log_b a}$
3. number of terms $= \log_b n$

$\downarrow$

Master Theorem

1. **Root Dominant**: $O(c\,n^d)$ if leave = O(root) which is $d > \log_b a$
2. **Leave Dominant**: $O(n^{\log_b a})$ if root = O(leave) which if $d < \log_b a$
3. **Even Terms:** if $d > \log_b a$ therefore
   $T(n)$ = number of terms * $f(n) = \log_b n * f(n)$