# N x N sorted matrix search

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 3 | 8 | 10 | 13 | 15 | 17 | 19 | 22 | 25 |
| **2** | 3 | 4 | 12 | 17 | 19 | 22 | 23 | 25 | 27 | 30 |
| **3** | 9 | 11 | 15 | 18 | 23 | 25 | 30 | 31 | 33 | 37 |
| **4** | 13 | 11 | 17 | 20 | 25 | 27 | 32 | 34 | 35 | 40 |
| **5** | 20 | 24 | 27 | 30 | 31 | 32 | 37 | 40 | 46 | 49 |
| **6** | 21 | 28 | 35 | 40 | 42 | 43 | 45 | 48 | 50 | 55 |
| **7** | 25 | 30 | 41 | 47 | 49 | 50 | 53 | 55 | 57 | 59 |
| **8** | 27 | 31 | 45 | 50 | 56 | 59 | 60 | 63 | 65 | 72 |
| **9** | 28 | 33 | 49 | 55 | 57 | 63 | 65 | 69 | 70 | 75 |
| **10** | 30 | 35 | 51 | 58 | 60 | 65 | 69 | 73 | 75 | 80 |

# Algorithm

- Let *t = target*

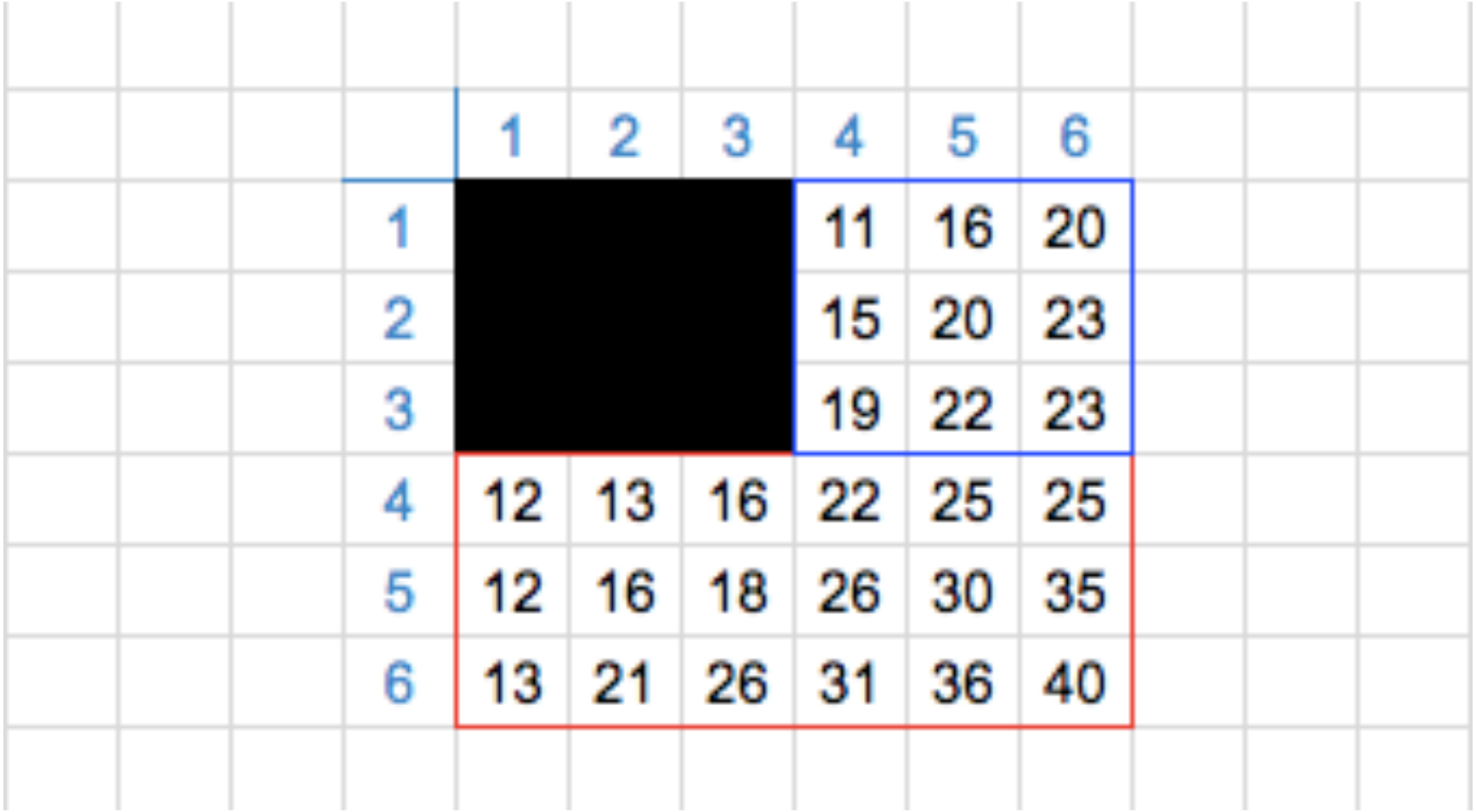
1. Let *mid = A[ceiling(n/2), ceiling(n/2)]*
2. If *mid = target*: return true
3. Else if *mid < target*: Eliminate upper left quad
4. Else if *mid > target*: Eliminate lower right quad
5. Repeat 1-4 on 2 remaining sub-matrices (may not be square)

Base case: 1 X 1 matrix, stop and return false

# Example 6 X 6
# Target = 26
# Comparisons = 1

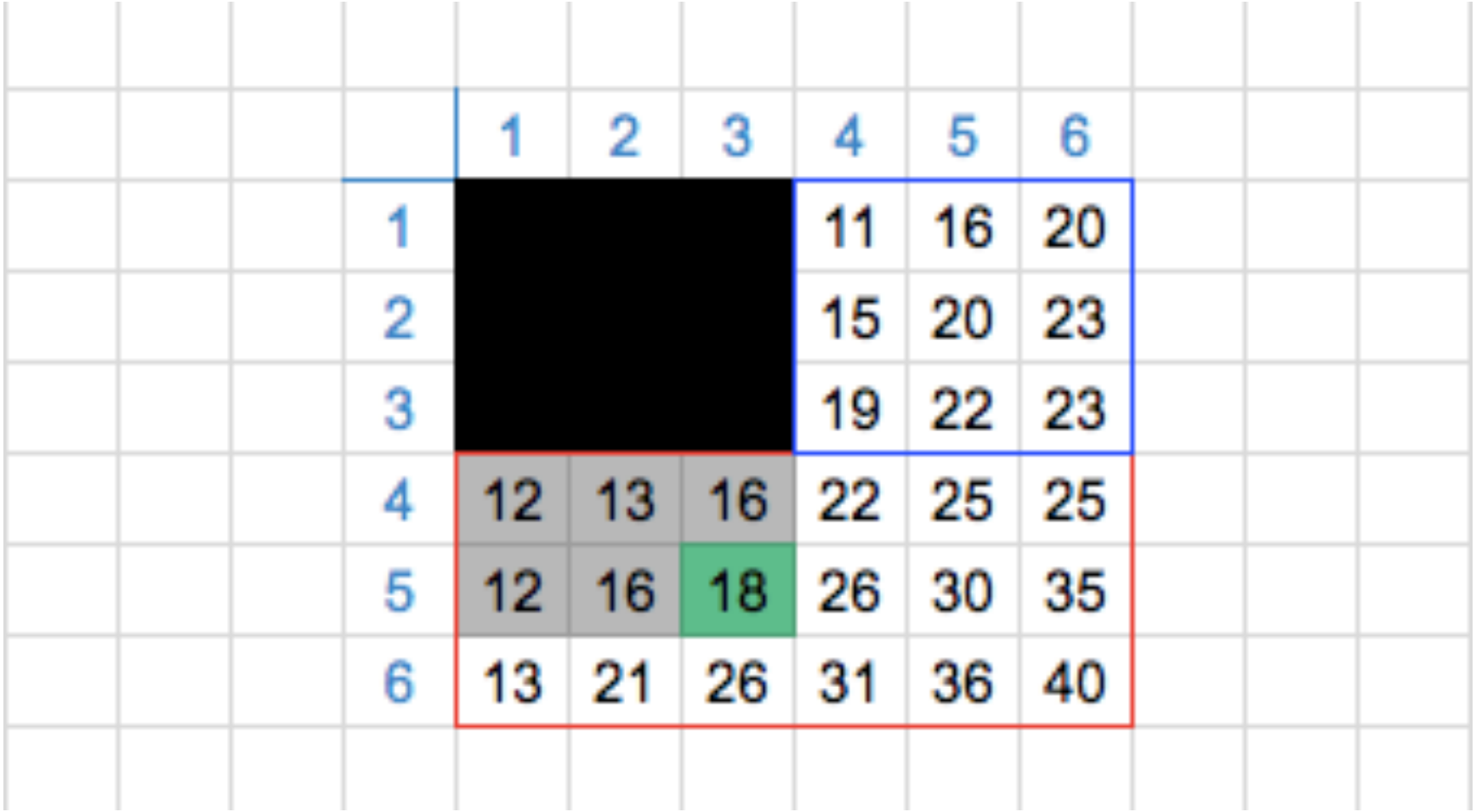
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 5 | 6 | 11 | 16 | 20 |
| 2 | 4 | 8 | 10 | 15 | 20 | 23 |
| 3 | 7 | 9 | 10 | 19 | 22 | 23 |
| 4 | 12 | 13 | 16 | 22 | 25 | 25 |
| 5 | 12 | 16 | 18 | 26 | 30 | 35 |
| 6 | 13 | 21 | 26 | 31 | 36 | 40 |

# Example 6 X 6
# Target = 26



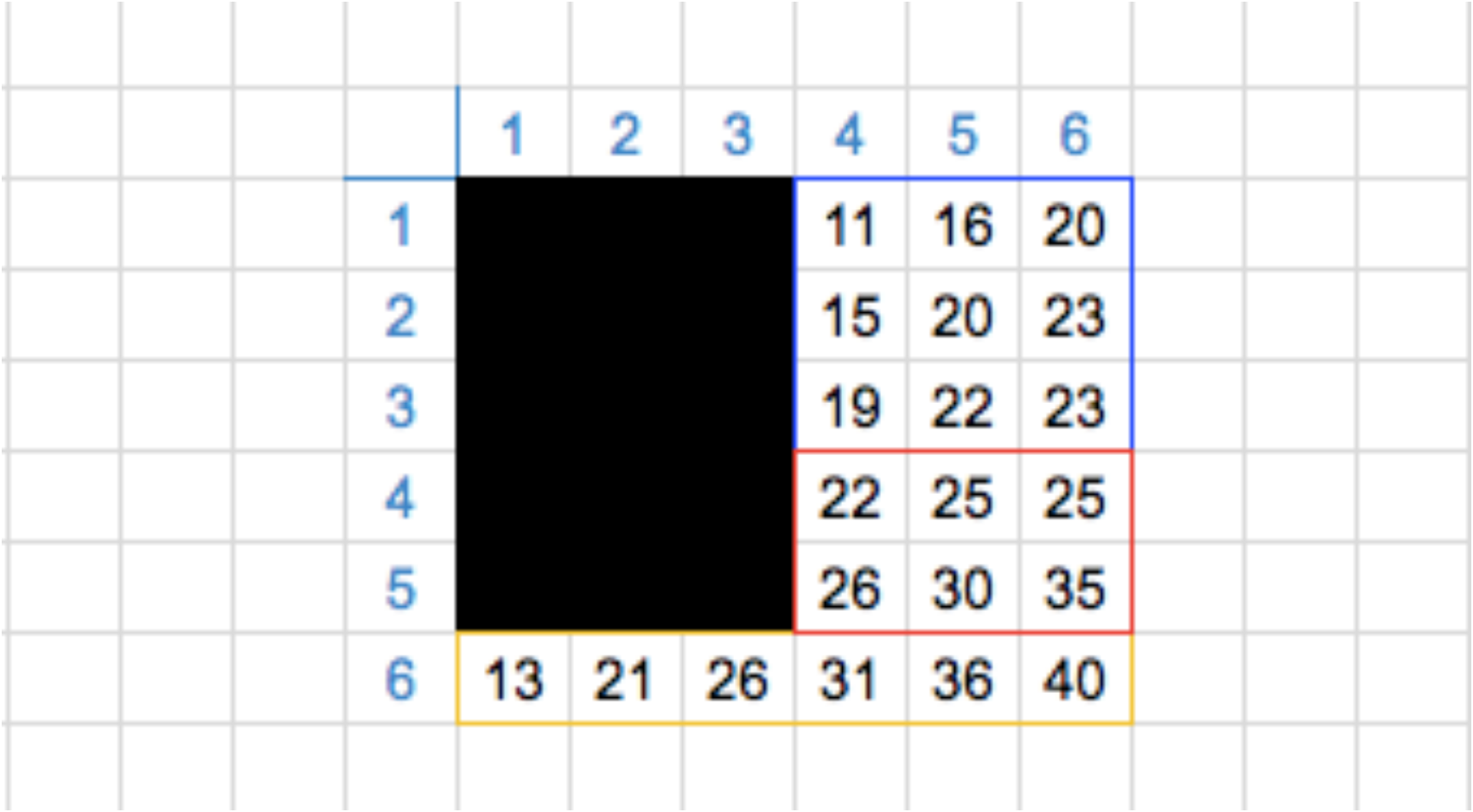
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | | | 11 | 16 | 20 |
| 2 | | | | 15 | 20 | 23 |
| 3 | | | | 19 | 22 | 23 |
| 4 | 12 | 13 | 16 | 22 | 25 | 25 |
| 5 | 12 | 16 | 18 | 26 | 30 | 35 |
| 6 | 13 | 21 | 26 | 31 | 36 | 40 |

# Example 6 X 6
# Target = 26
# Comparisons = 2



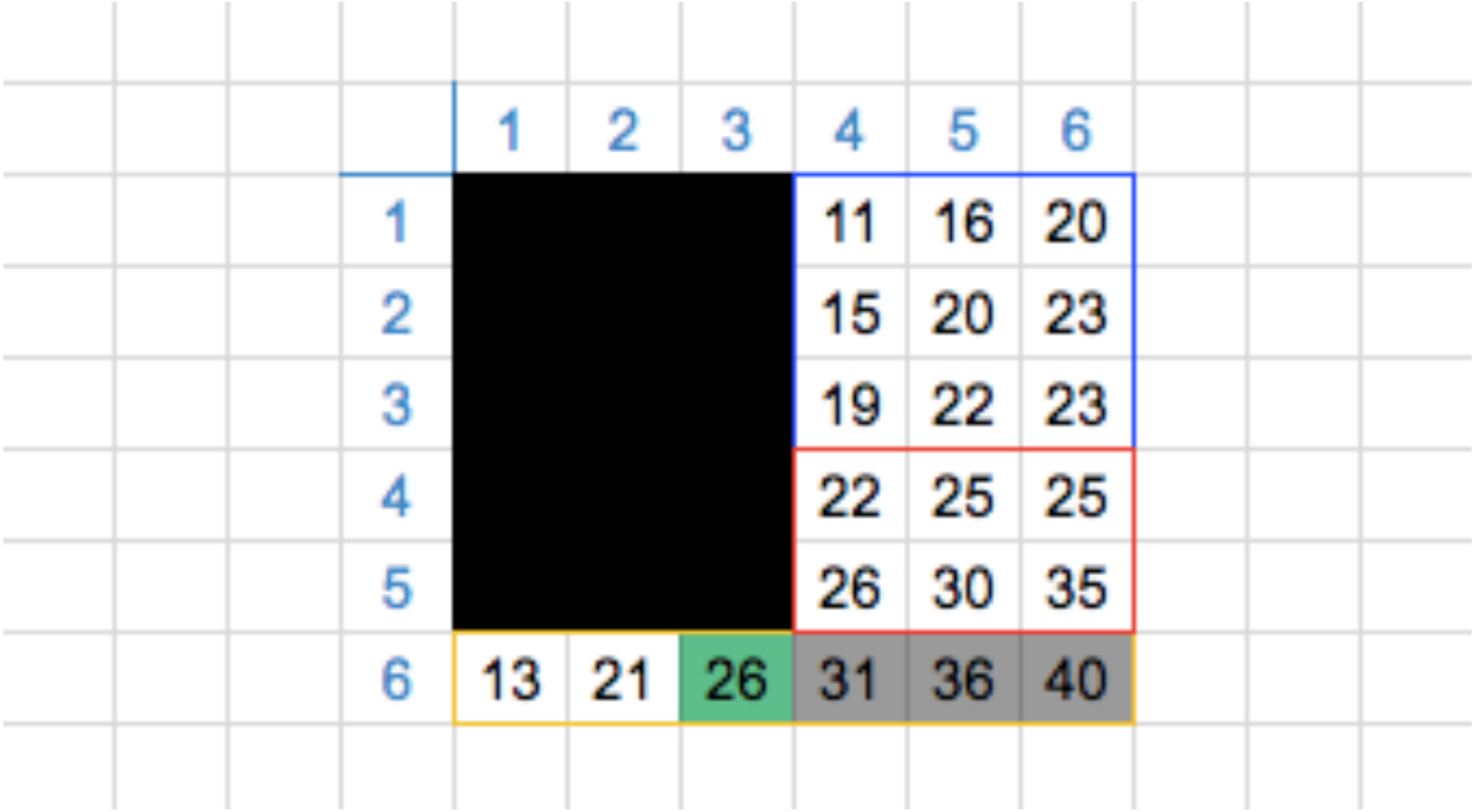
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | | | 11 | 16 | 20 |
| 2 | | | | 15 | 20 | 23 |
| 3 | | | | 19 | 22 | 23 |
| 4 | 12 | 13 | 16 | 22 | 25 | 25 |
| 5 | 12 | 16 | 18 | 26 | 30 | 35 |
| 6 | 13 | 21 | 26 | 31 | 36 | 40 |

# Example 6 X 6
# Target = 26



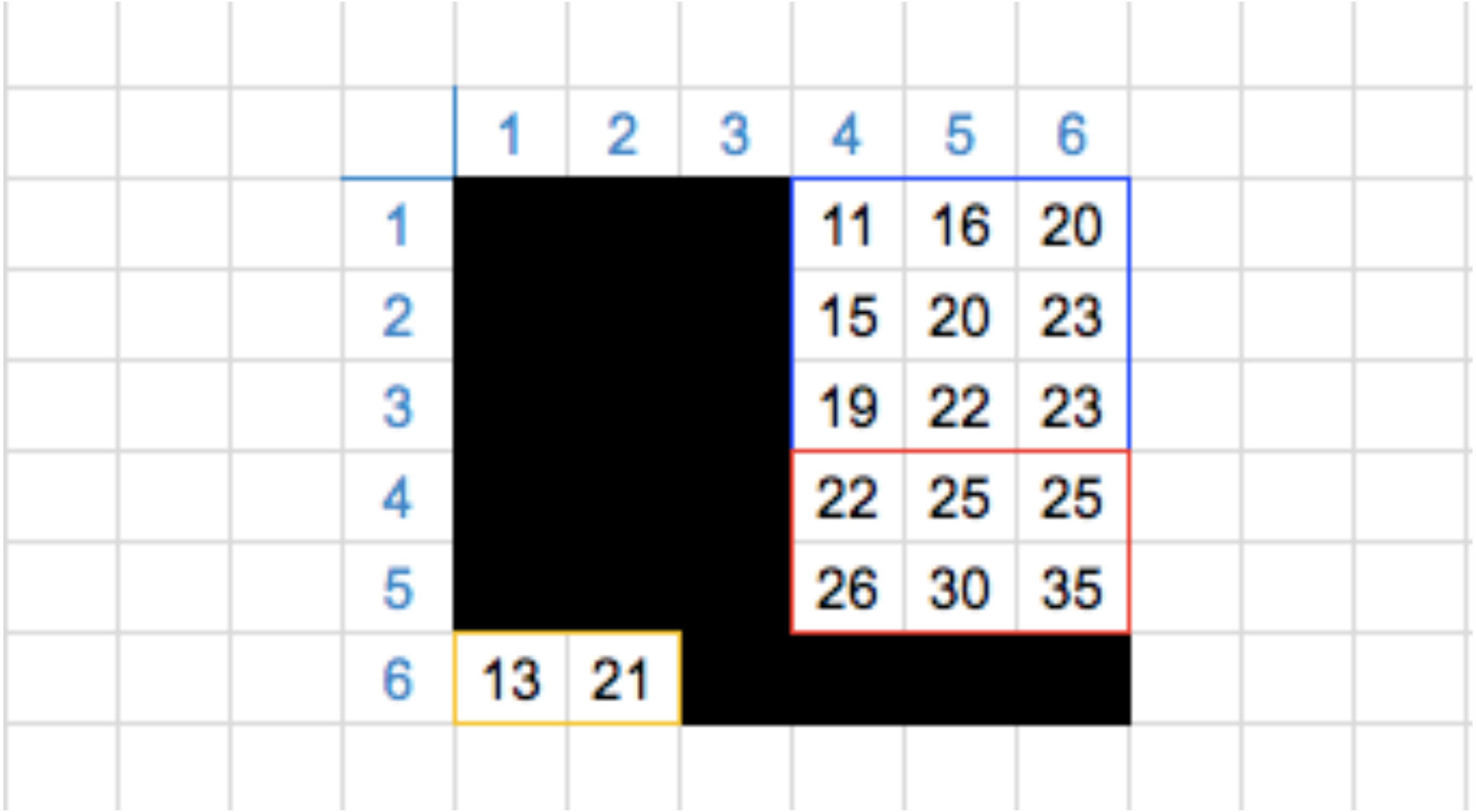
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | | | 11 | 16 | 20 |
| 2 | | | | 15 | 20 | 23 |
| 3 | | | | 19 | 22 | 23 |
| 4 | | | | 22 | 25 | 25 |
| 5 | | | | 26 | 30 | 35 |
| 6 | 13 | 21 | 26 | 31 | 36 | 40 |

# Example 6 X 6
# Target = 26
# Comparisons = 3

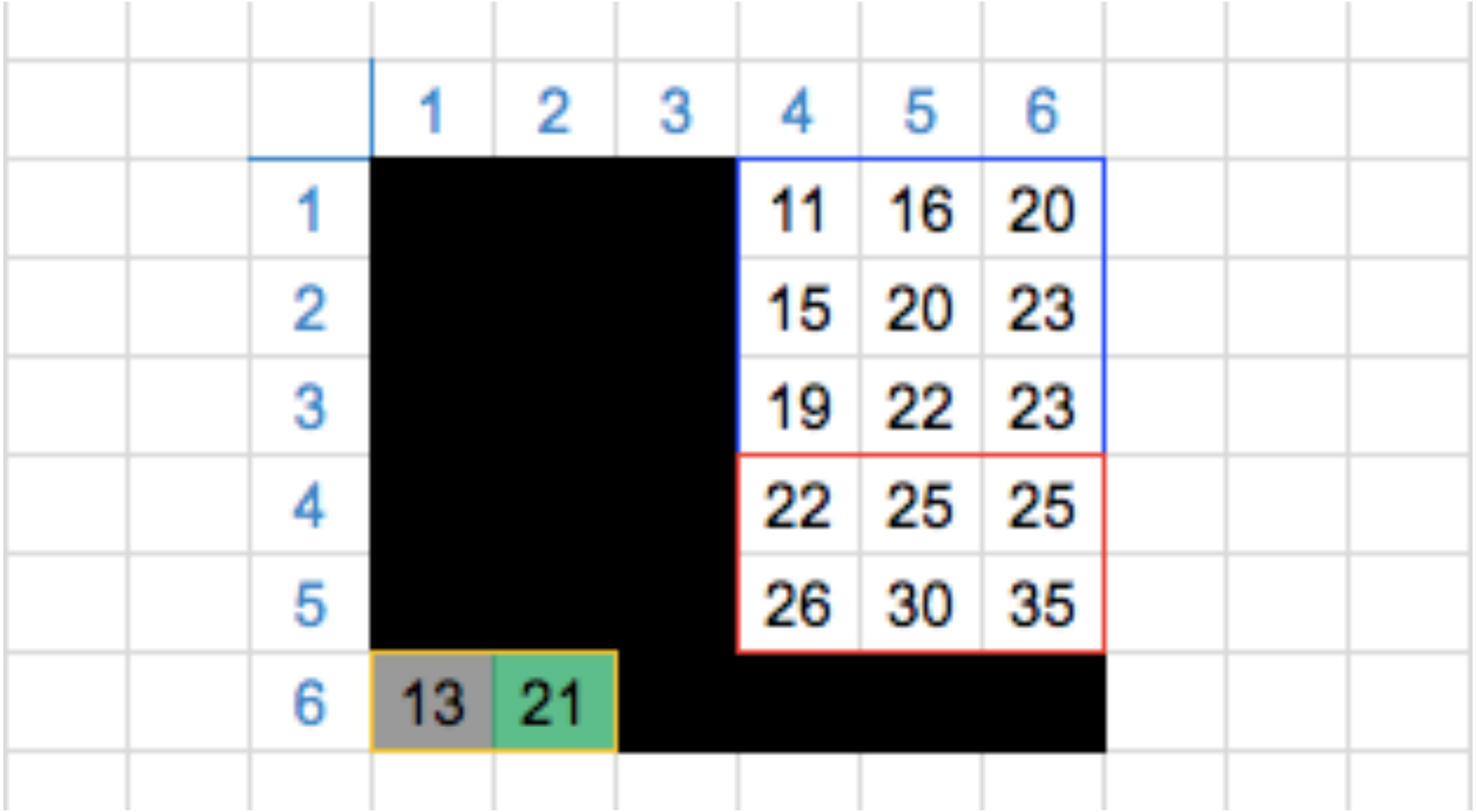


| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | | | 11 | 16 | 20 |
| 2 | | | | 15 | 20 | 23 |
| 3 | | | | 19 | 22 | 23 |
| 4 | | | | 22 | 25 | 25 |
| 5 | | | | 26 | 30 | 35 |
| 6 | 13 | 21 | 26 | 31 | 36 | 40 |

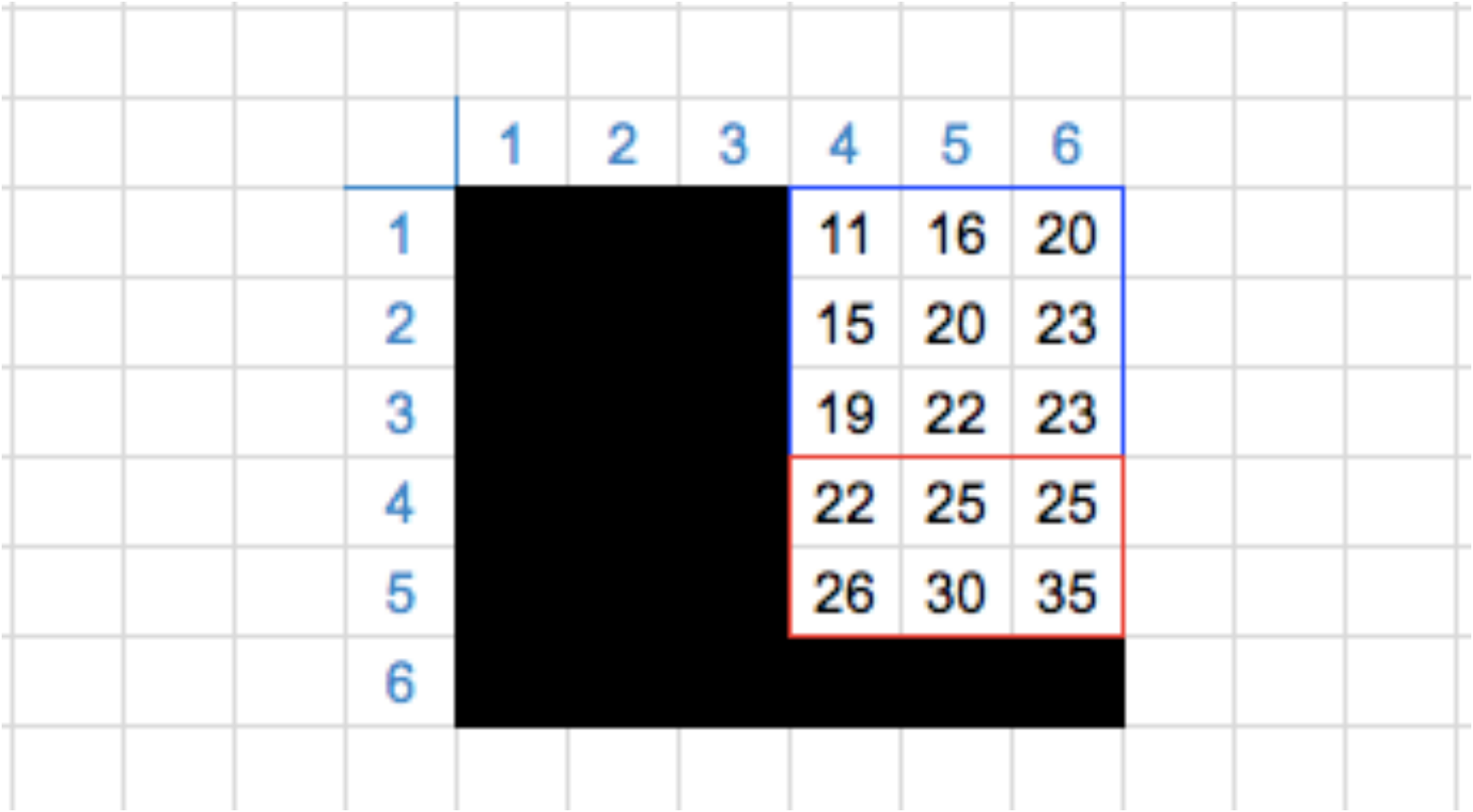
# Example 6 X 6
# Target = 24 (DNE)



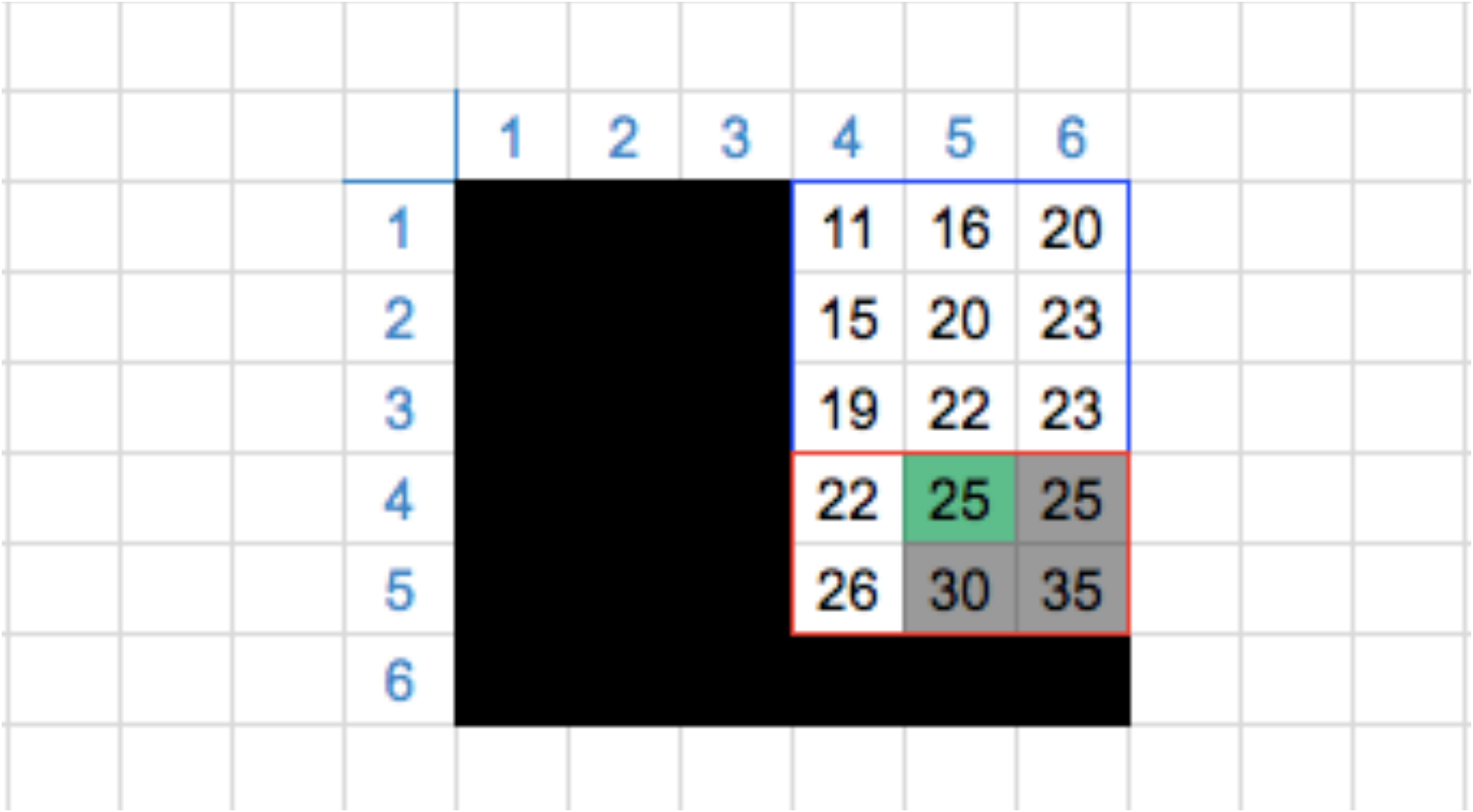
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 4



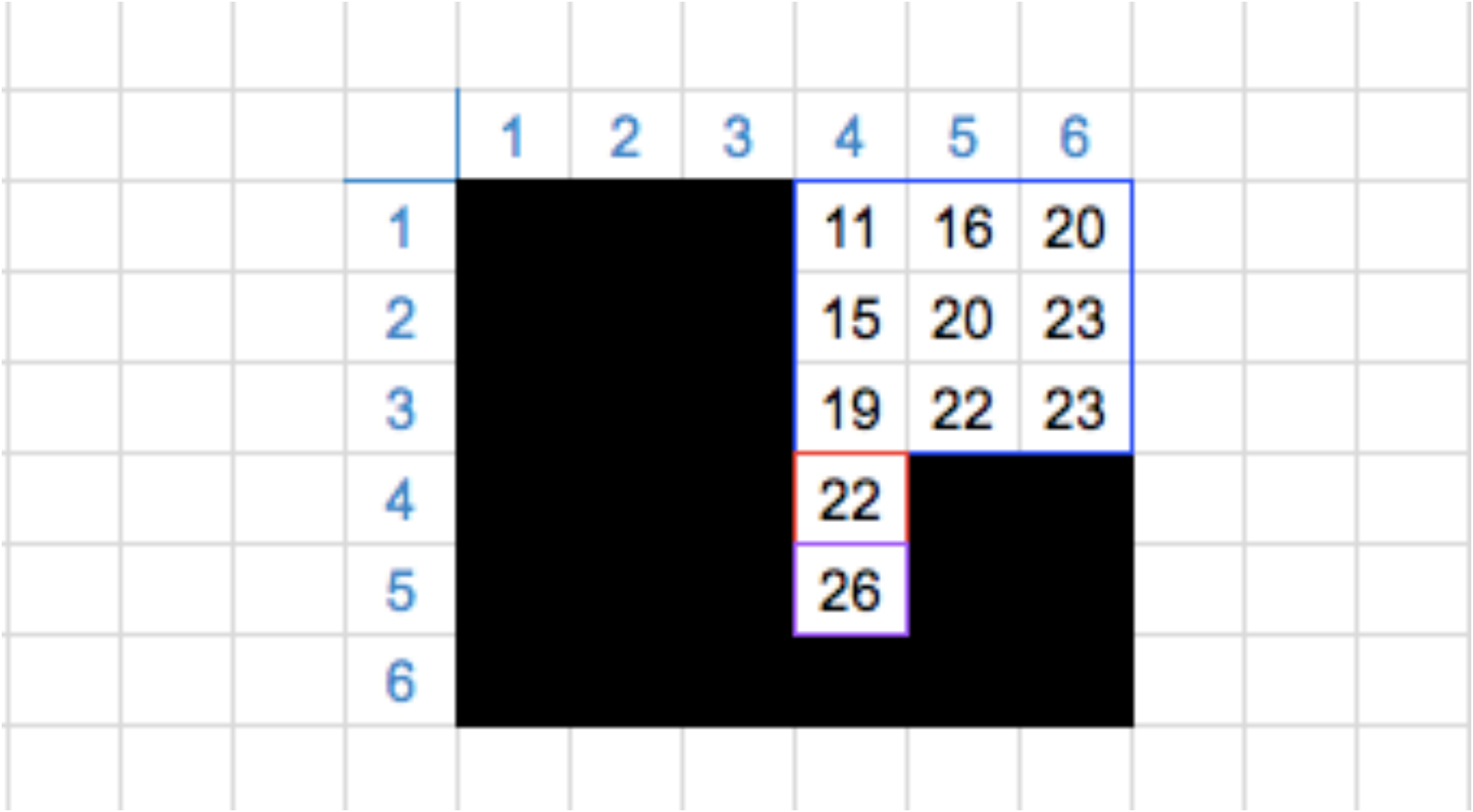
# Example 6 X 6
# Target = 24 (DNE)



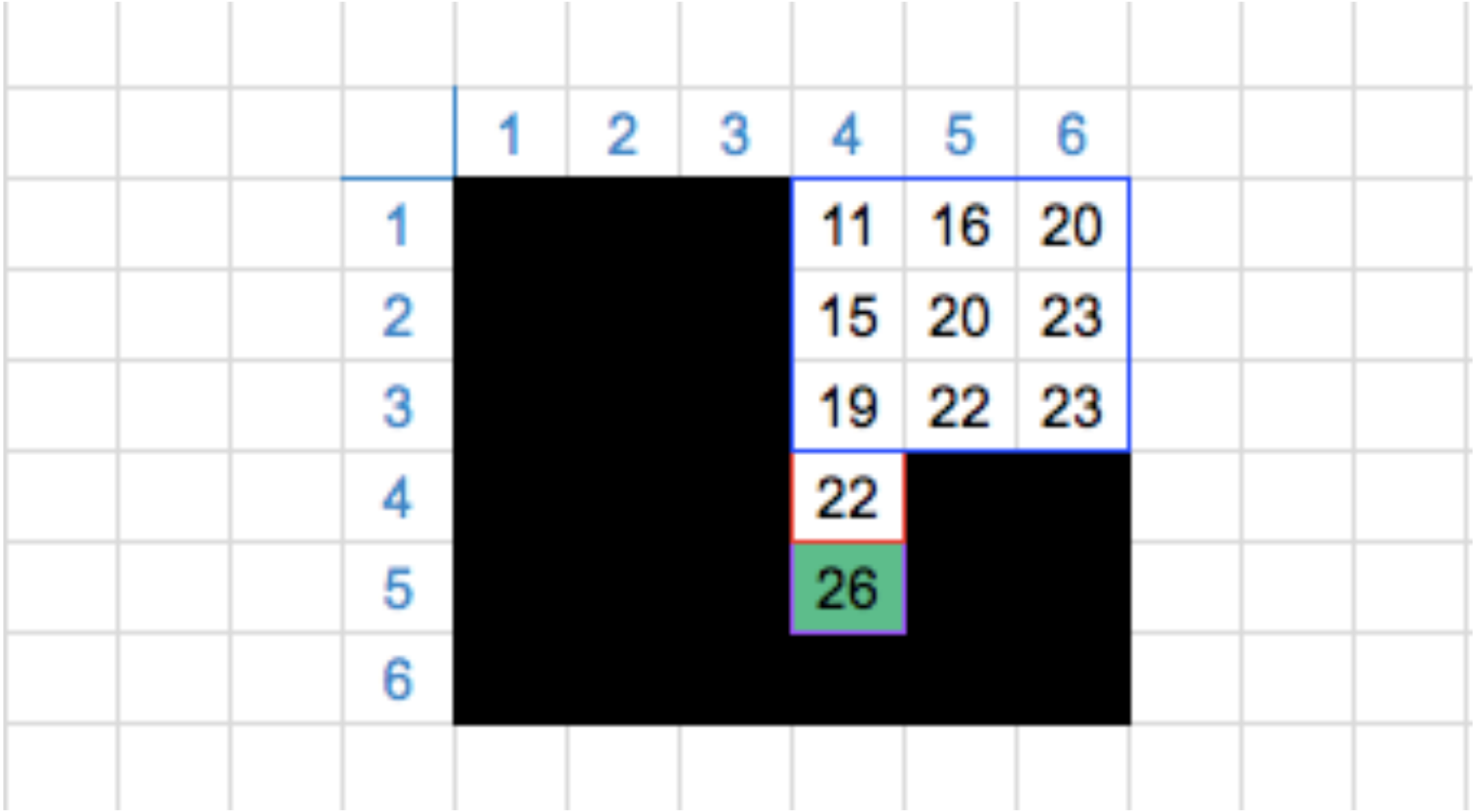
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 5



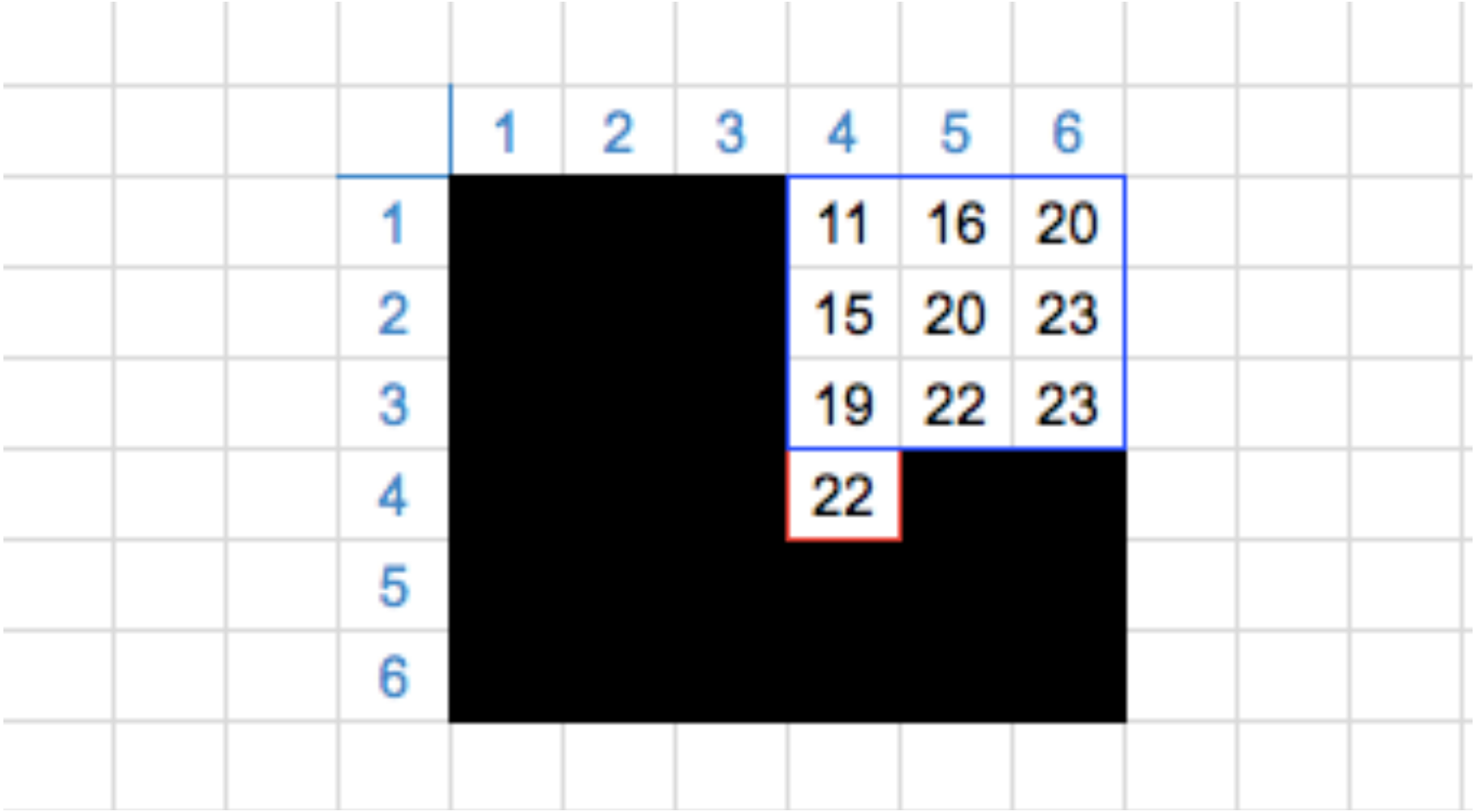
# Example 6 X 6
# Target = 24 (DNE)



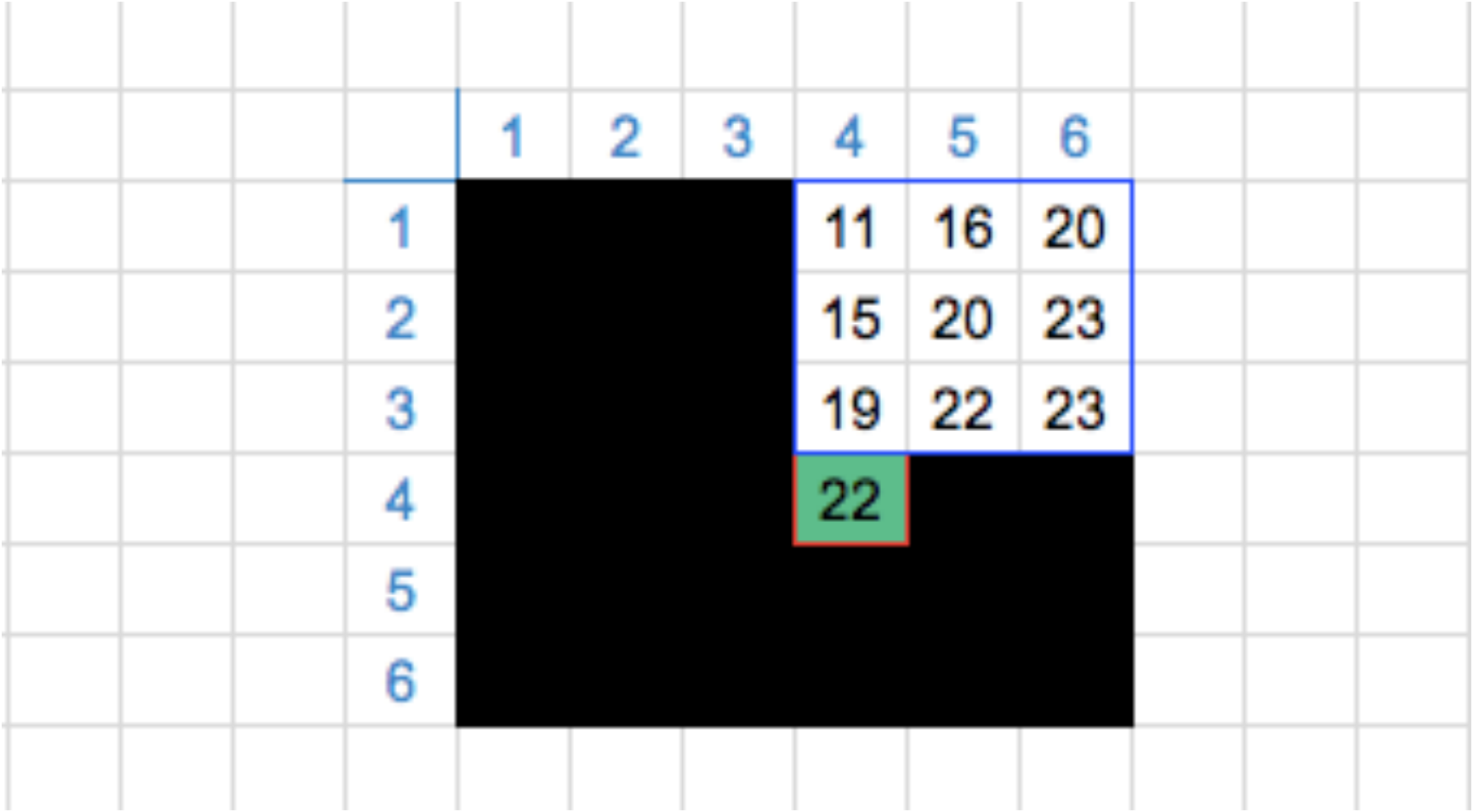
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 6



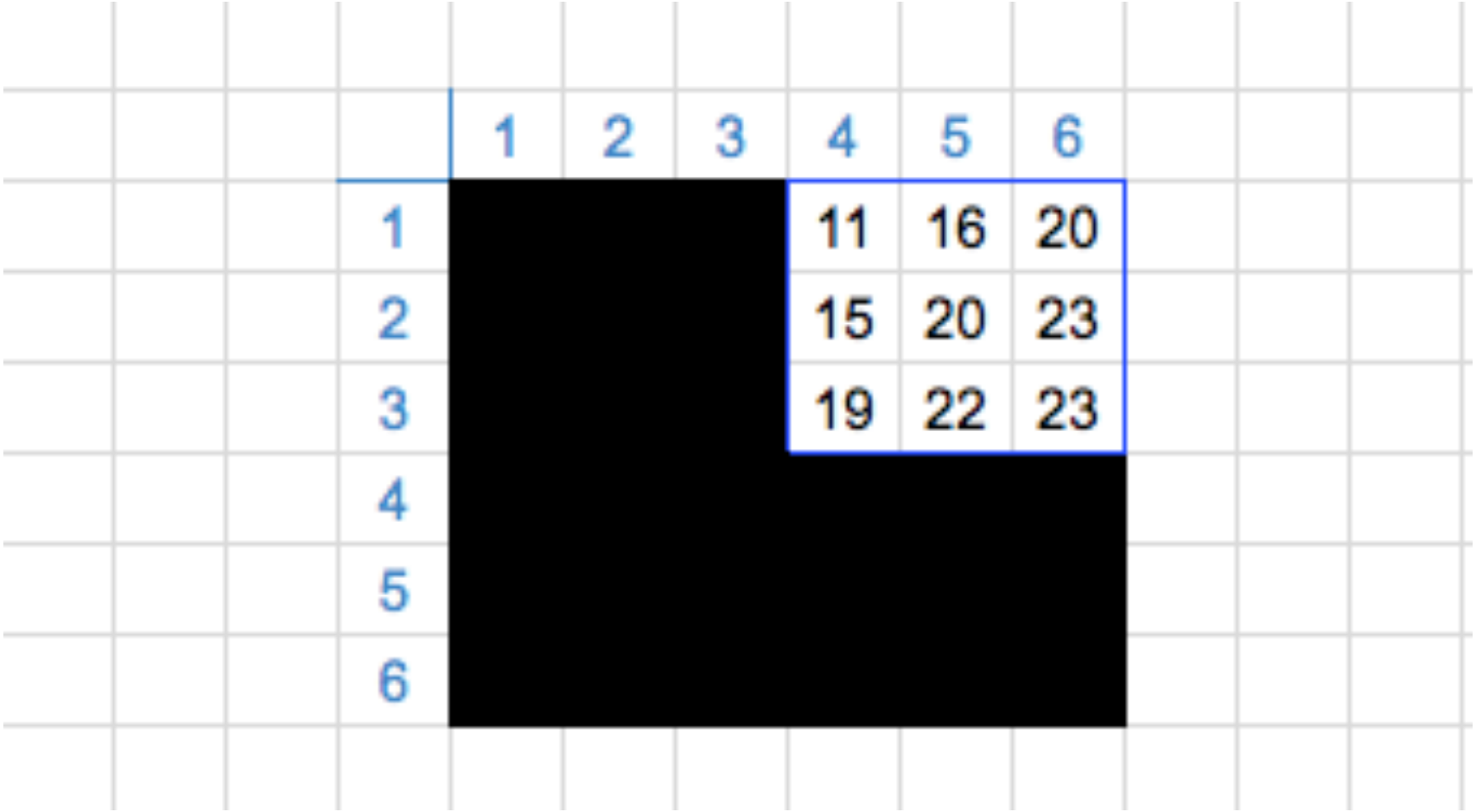
# Example 6 X 6
# Target = 24 (DNE)



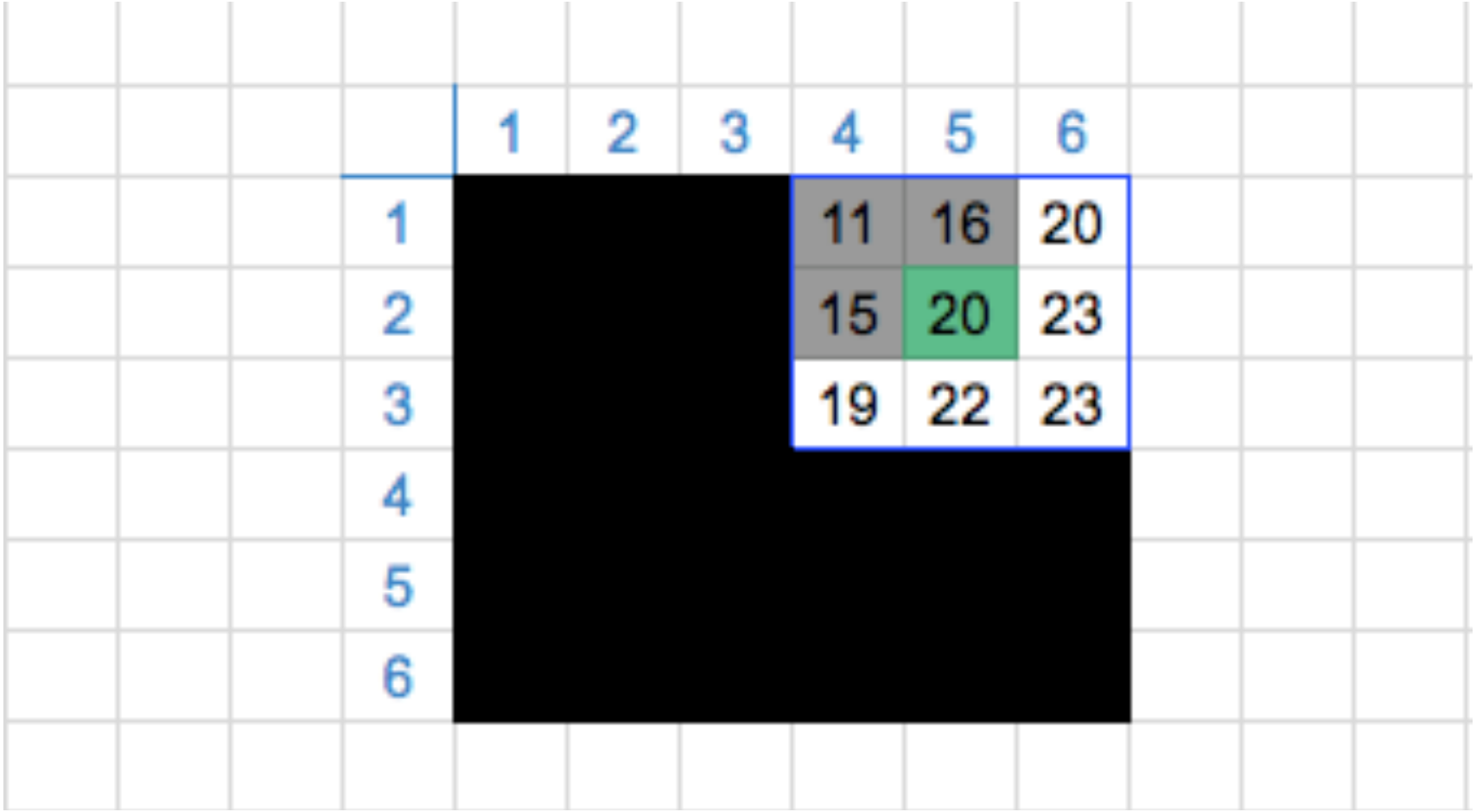
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 7



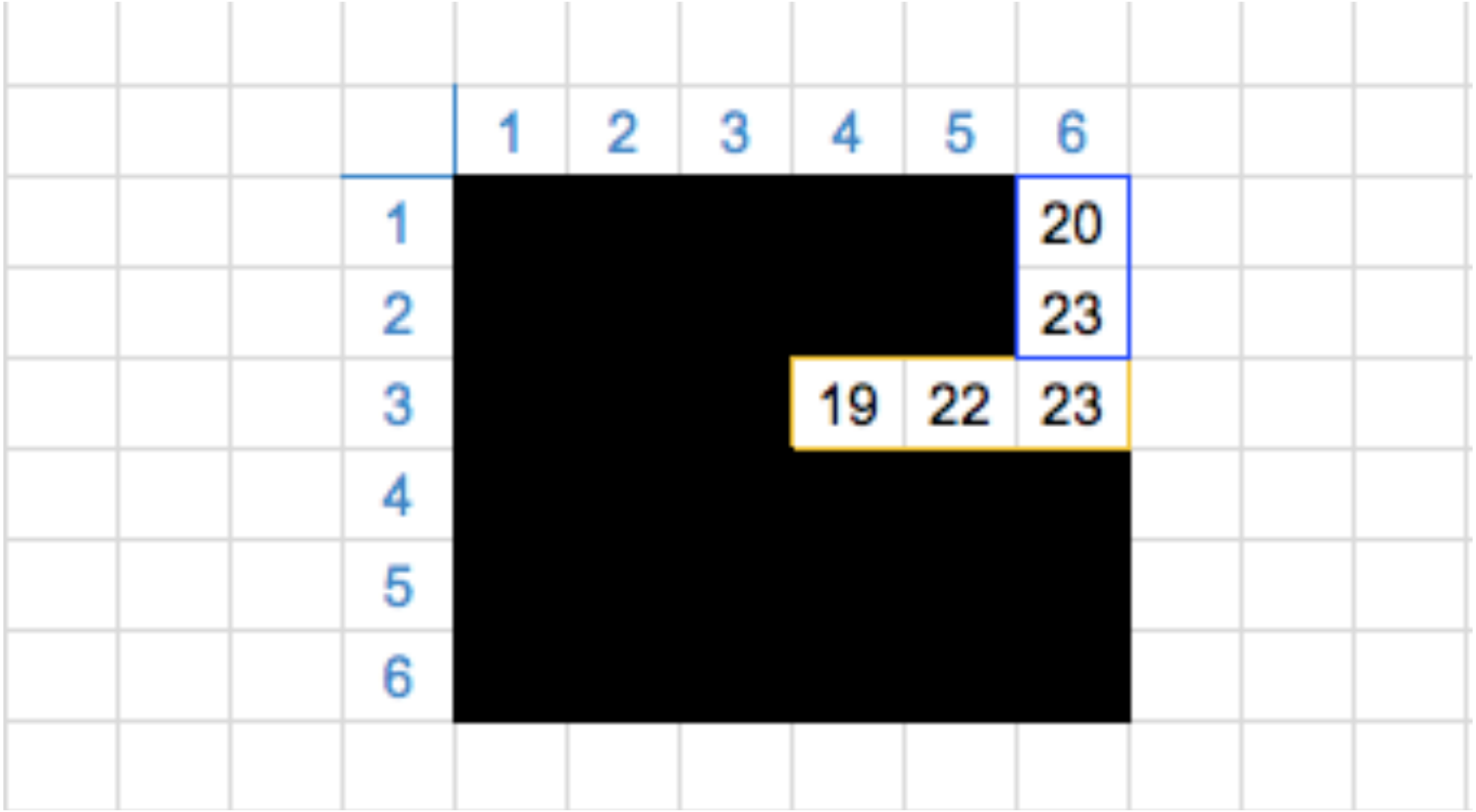
# Example 6 X 6
# Target = 24 (DNE)



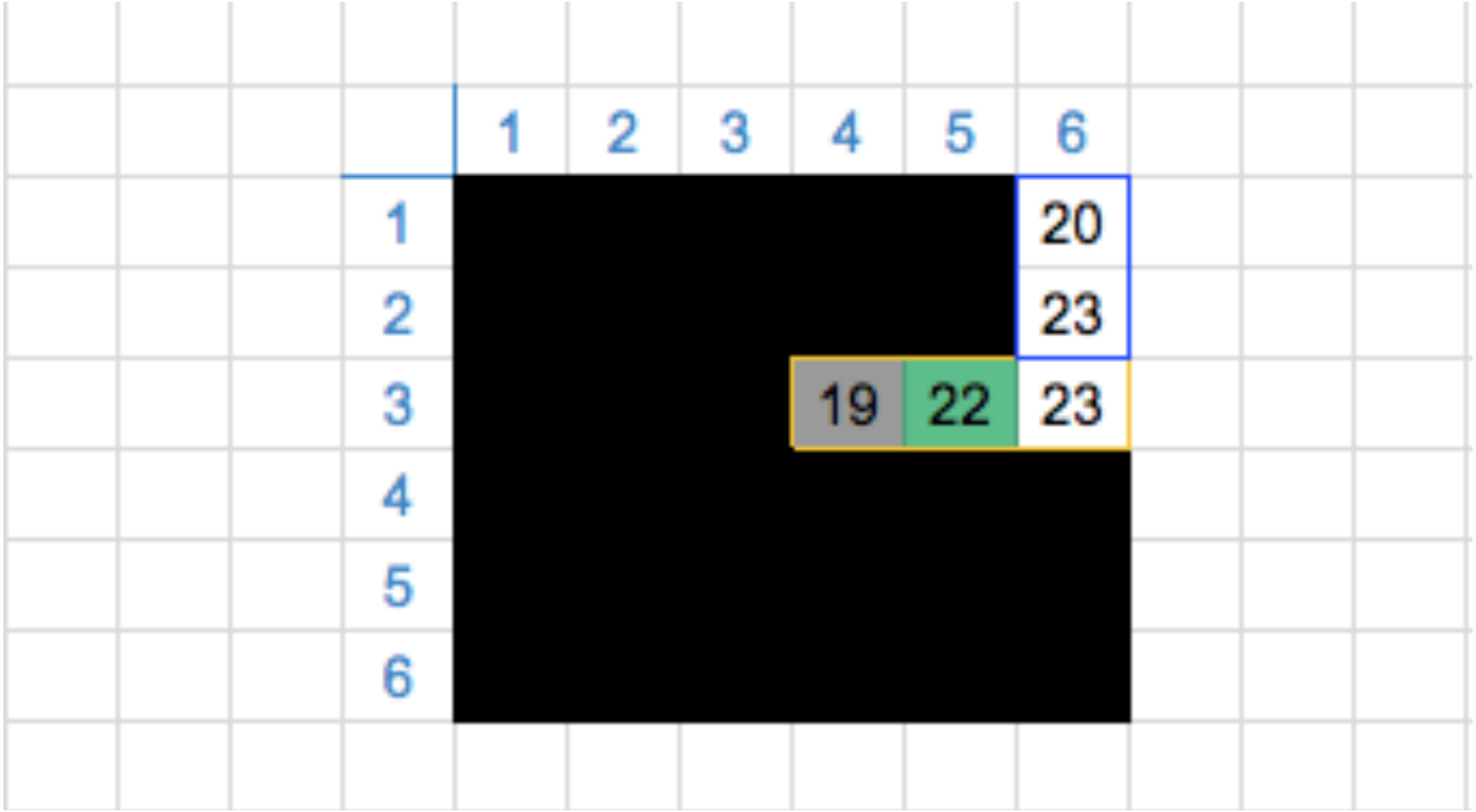
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 8



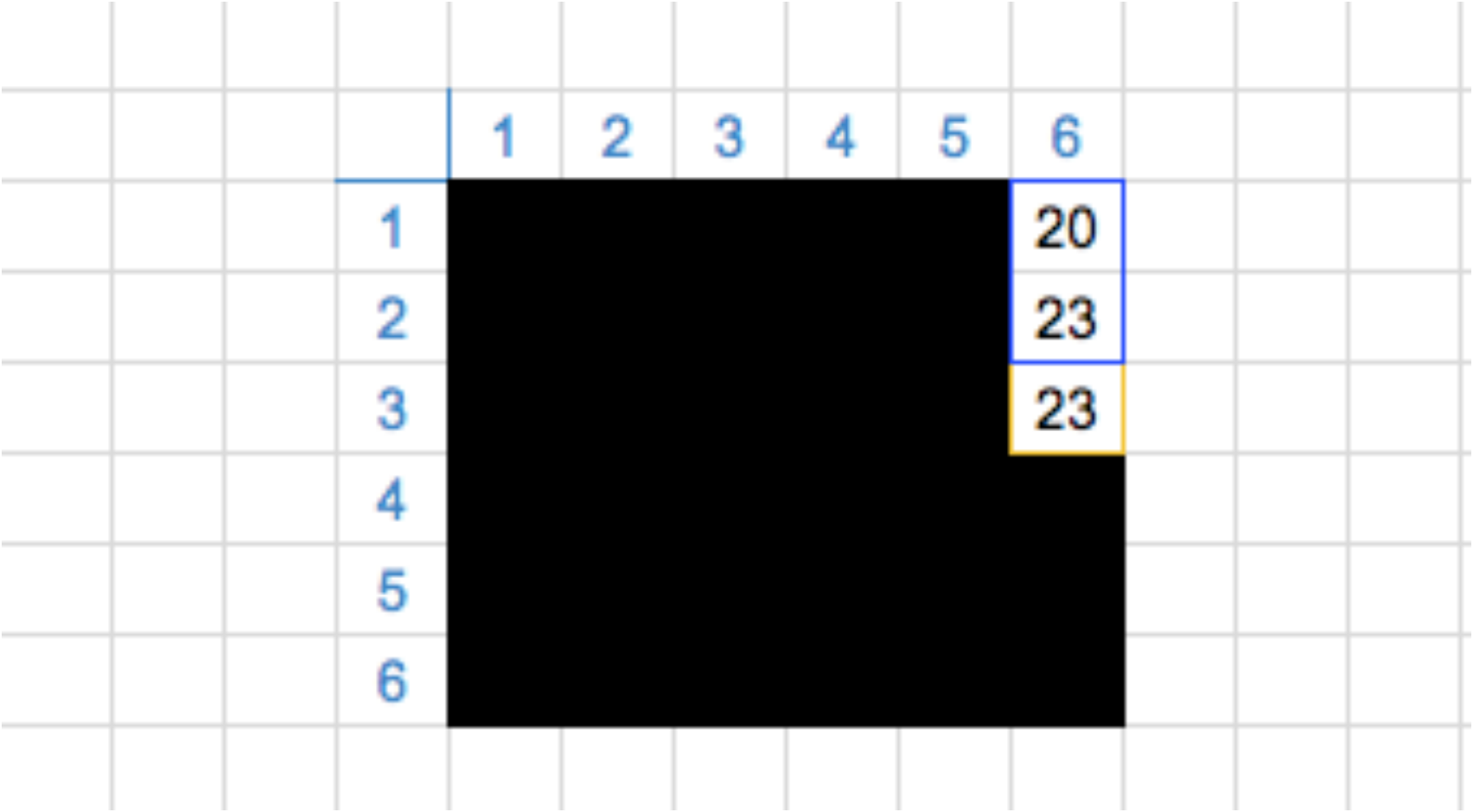
# Example 6 X 6
# Target = 24 (DNE)



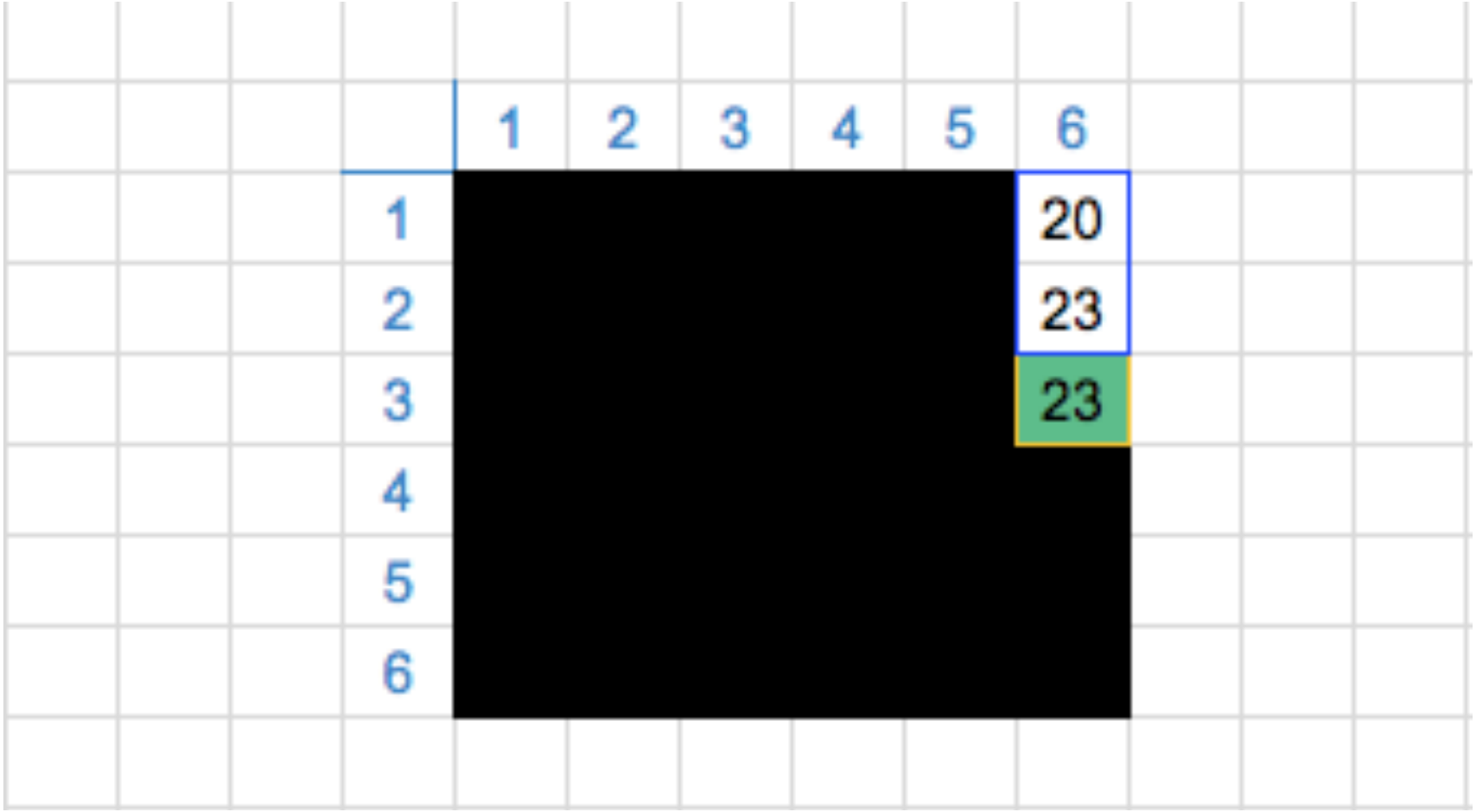
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 9



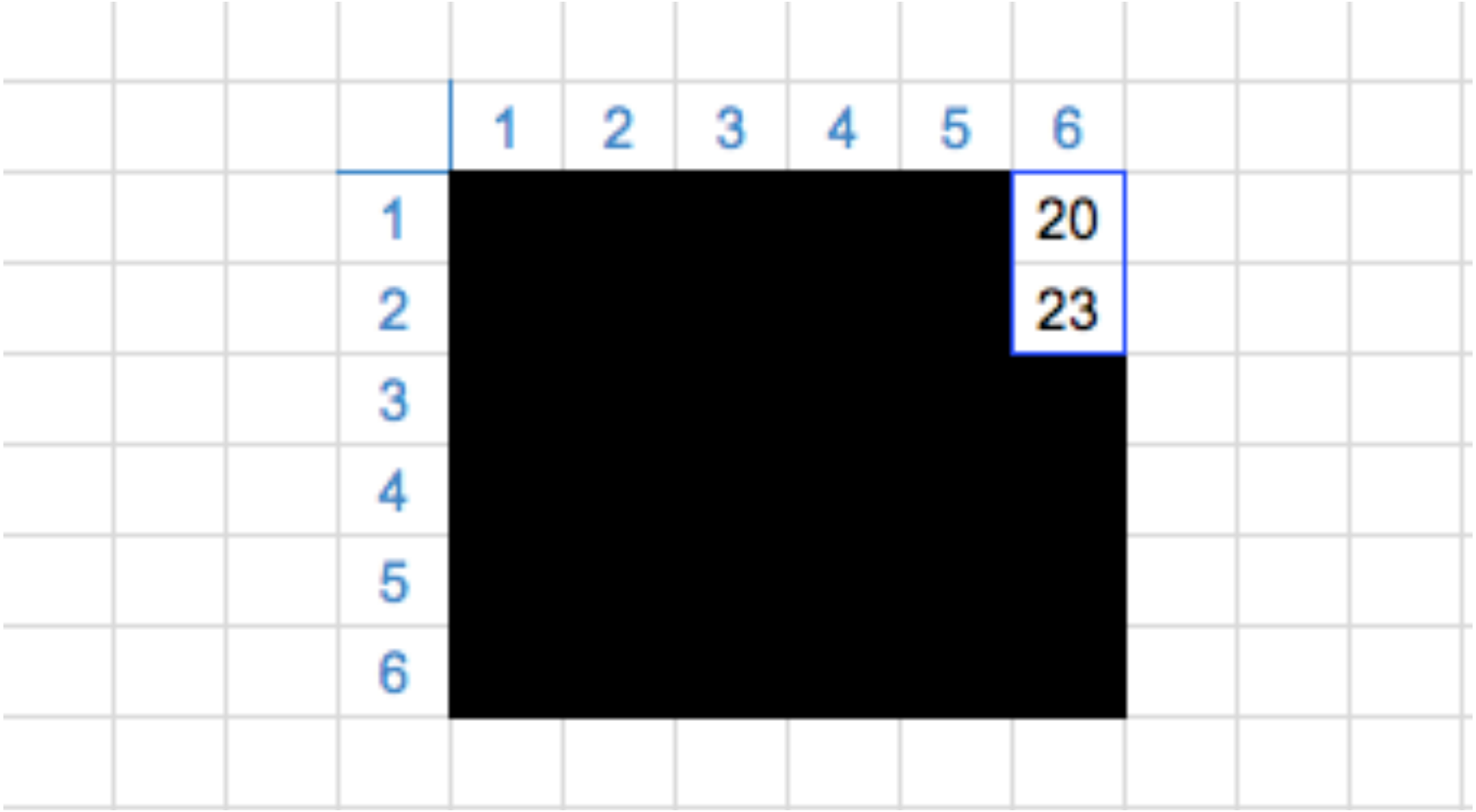
# Example 6 X 6
# Target = 24 (DNE)



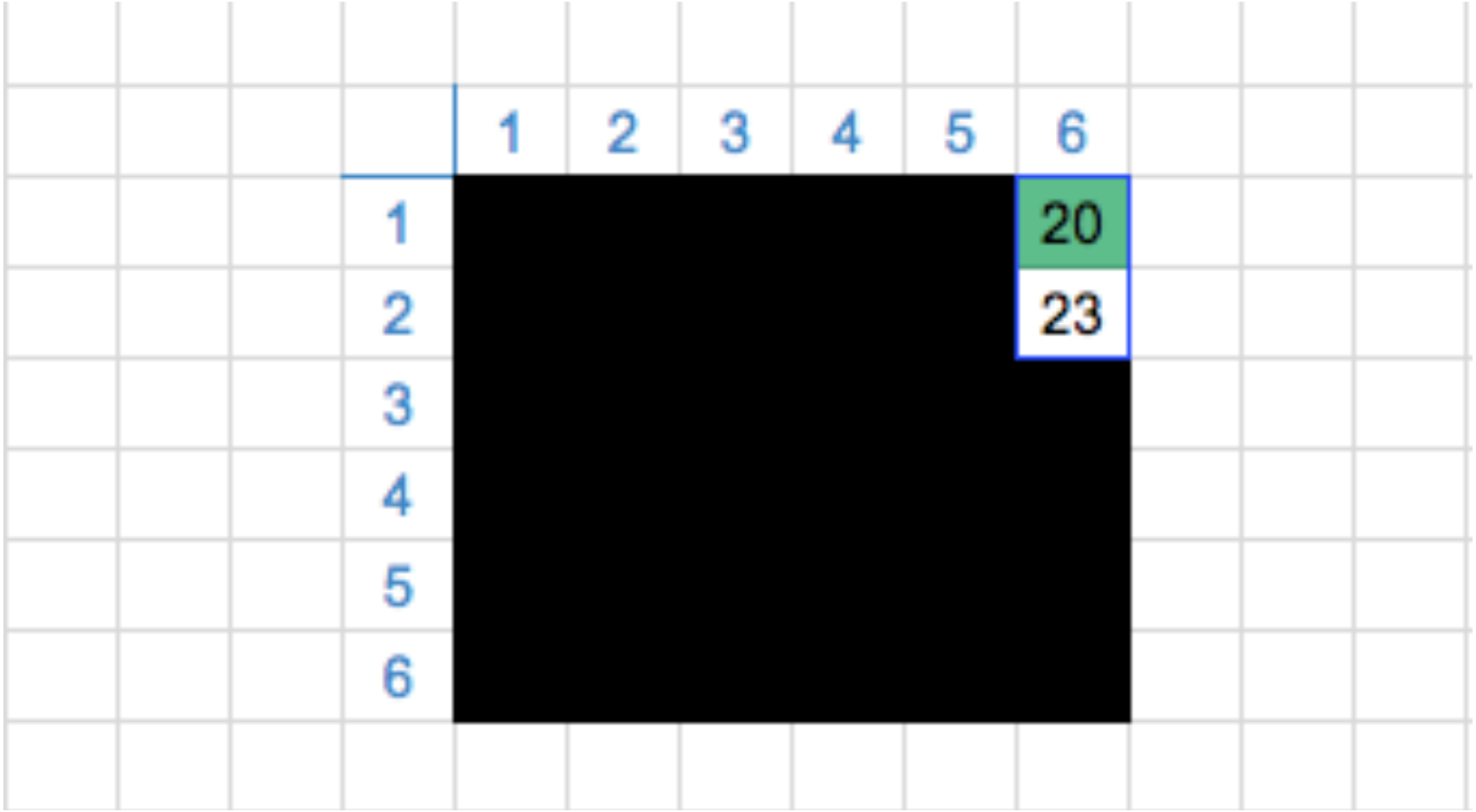
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 10



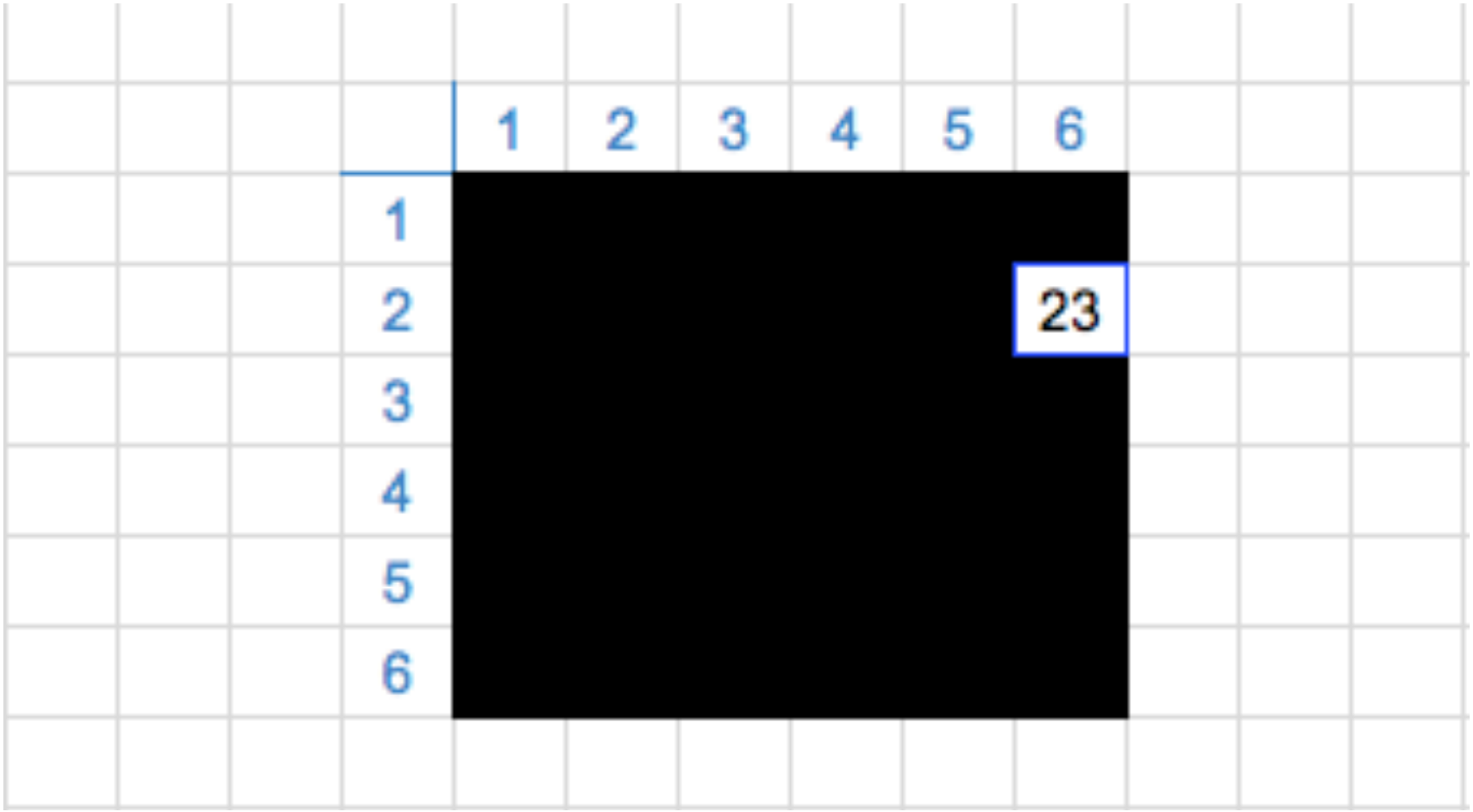
# Example 6 X 6
# Target = 24 (DNE)



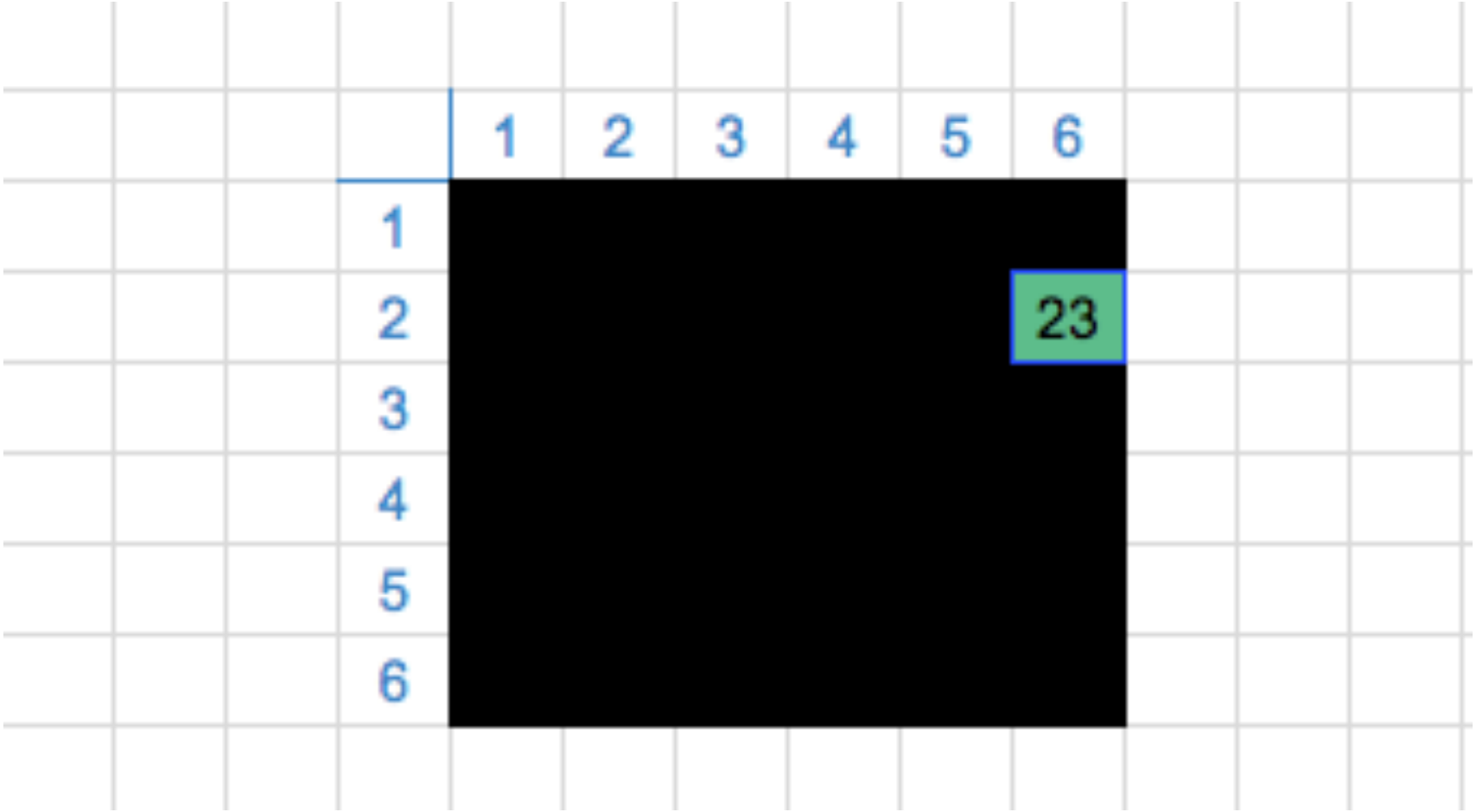
# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 11



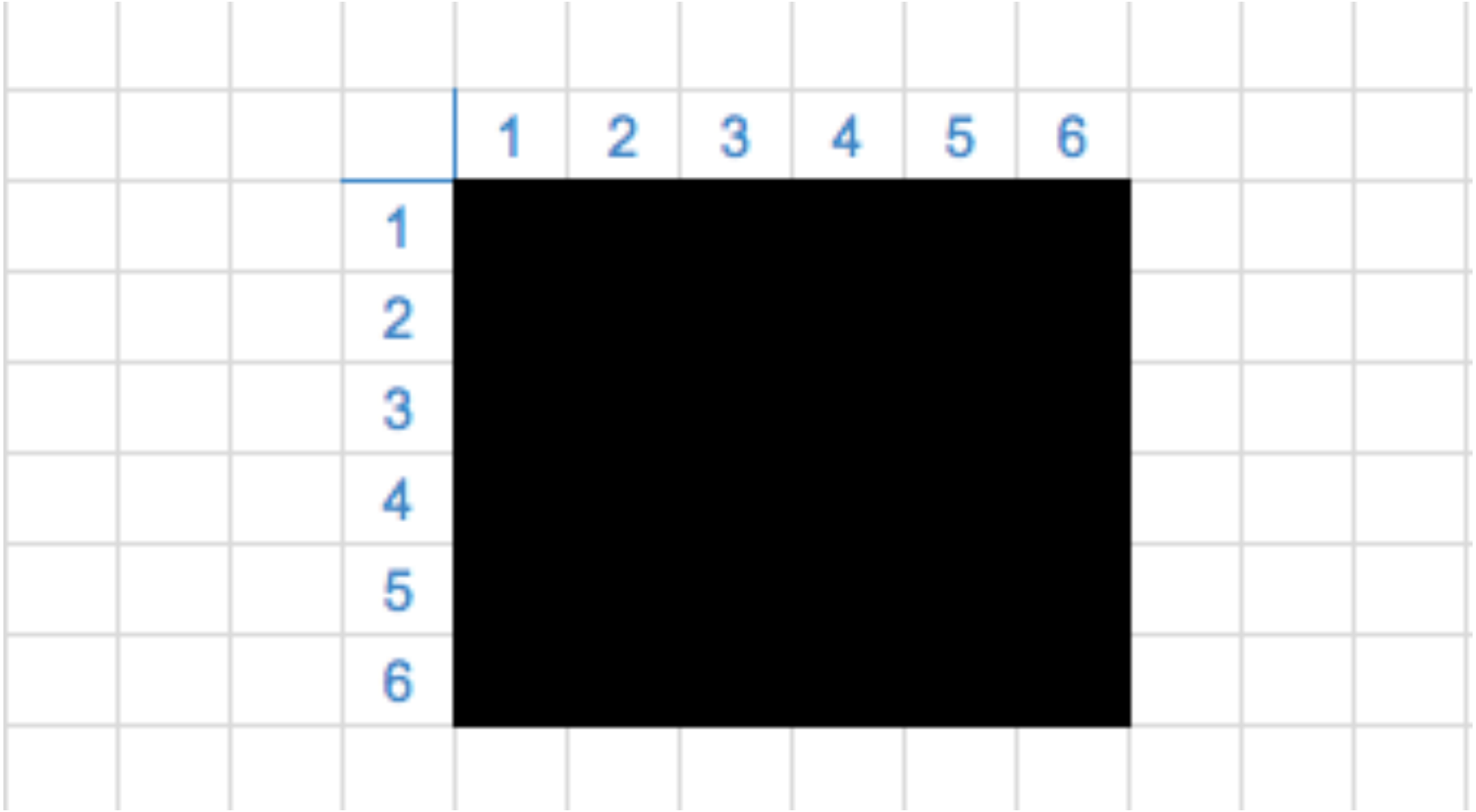
# Example 6 X 6
# Target = 24 (DNE)



# Example 6 X 6
# Target = 24 (DNE)
# Comparisons = 12



# Example 6 X 6
# Target = 24 (DNE)



# Time Analysis

- At each step, dealing with an m x n matrix (m may be equal to n)
- Each comparison eliminates AT LEAST ¼ of elements of matrix
- For m x n matrix, ¼ elements = (m*n) / 4

| m | n | # elim |
|---|---|---|
| even | even | <: (m/2 + 1) * (n/2 + 1) |
| | | >: m/2 * n/2 |
| even | odd | <: (m/2 + 1) * ceil(n/2) |
| | | >: m/2 * ceil(n/2) |
| odd | even | <: ceil(m/2) * (n/2 + 1) |
| | | >: ceil(m/2) * n/2 |
| odd | odd | <: m/2 * n/2 |
| | | >: m/2 * n/2 |

Key- “<:” ---- target < mid
“>:” ---- target > mid

# Master theorem

Given recurrence relation:

$T(s) = a\ T\ (s/b) + f(s)$ where $a >= 1, b > 1$

where:

s = size of problem = $(n^2)$

a = # of sub-problems in the recursion = 2

s/b = size of each sub problem

avg size = ((2s/4 + s/4) / 2)

= 3s/8

= s / (8/3)

f(s) = cost of work done outside recursive calls = constant (< or > comparison)

# Master theorem

Since f(s) is constant

$T(s) = \Theta(s^{\log(b)a})$

$b = 8/3,\ a = 2,\ s = n^2$

$T(s) = \Theta(s^{\log(8/3)2})$

$= \Theta(s^{.7058})$