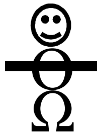
CSE 548: Analysis of Algorithms

Rezaul A. Chowdhury
Department of Computer Science
SUNY Stony Brook
Fall 2017



Asymptotic Stickman
(by Aleksandra Patrzalek, SUNY Buffalo)

Some Mostly Useless Information

- **Lecture Time:** MW 7:00 pm - 8:20 pm
- **Location:** Javits Lecture Hall 102, West Campus
- **Instructor:** Rezaul A. Chowdhury
- **Office Hours:** MW 5:00 pm - 6:30 pm
  239 Computer Science
- **Email:** rezaul@cs.stonybrook.edu
- **TA:** TBA
- **Class Webpage:**
  http://www3.cs.stonybrook.edu/~rezaul/CSE548-F17.html

Prerequisites

- **Required:** Some background (undergrad level) in the design and analysis of algorithms and data structures
  - fundamental data structures (e.g., lists, stacks, queues and arrays)
  - discrete mathematical structures (e.g., graphs, trees, and their adjacency lists & adjacency matrix representations)
  - fundamental programming techniques (e.g., recursion, divide-and-conquer, and dynamic programming)
  - basic sorting and searching algorithms
  - fundamentals of asymptotic analysis (e.g., $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$ notations)
- **Required:** Some background in programming languages (C / C++)

Topics to be Covered

The following topics will be covered (hopefully)

- recurrence relations and divide-and-conquer algorithms
- dynamic programming
- graph algorithms (e.g., network flow)
- amortized analysis
- advanced data structures (e.g., Fibonacci heaps)
- cache-efficient and external-memory algorithms
- high probability bounds and randomized algorithms
- parallel algorithms and multithreaded computations
- NP-completeness and approximation algorithms
- the alpha technique (e.g., disjoint sets, partial sums)
- FFT (Fast Fourier Transforms)

Grading Policy

- Four Homework Problem Sets
  (highest score 15%, lowest score 5%, and others 10% each): 40%
- Two Exams (higher one 30%, lower one 15%): 45%
  - Midterm (in-class): Oct 11
  - Final (in-class): Nov 29
- Scribe note (one lecture): 10%
- Class participation & attendance: 5%

Textbooks

Required

- Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein.
  Introduction to Algorithms (3rd Edition), MIT Press, 2009.

Recommended

- Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani.
  Algorithms (1st Edition), McGraw-Hill, 2006.
- Jon Kleinberg and Éva Tardos.
  Algorithm Design (1st Edition), Addison Wesley, 2005.
- Rajeev Motwani and Prabhakar Raghavan.
  Randomized Algorithms (1st Edition), Cambridge University Press, 1995.
- Vijay Vazirani.
  Approximation Algorithms, Springer, 2010.
- Joseph JáJá.
  An Introduction to Parallel Algorithms (1st Edition), Addison Wesley, 1992.

What is an Algorithm?

An algorithm is a ***well-defined computational procedure*** that solves a well-specified computational problem.

It accepts a value or set of values as ***input***, and produces a value or set of values as ***output***

Example: ***mergesort*** solves the ***sorting problem*** specified as a relationship between the input and the output as follows.

Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$.

Output: A permutation $\langle a'_1, a'_2, \dots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Desirable Properties of an Algorithm

√ Correctness

- Designing an incorrect algorithm is straightforward

√ Efficiency

- Efficiency is easily achievable if we give up on correctness

Surprisingly, sometimes incorrect algorithms can also be useful!

- If you can control the error rate
- Tradeoff between correctness and efficiency:
  Randomized algorithms
  (Monte Carlo: always efficient but sometimes incorrect,
  Las Vegas: always correct but sometimes inefficient)
  Approximation algorithms
  (always incorrect!)

How Do You Measure Efficiency?

We often want algorithms that can use the available resources efficiently.

Some measures of efficiency

- time complexity
- space complexity
- cache complexity
- I/O complexity
- energy usage
- number of processors/cores used
- network bandwidth

Goal of Algorithm Analysis

Goal is to predict the behavior of an algorithm without implementing it on a real machine.

But predicting the exact behavior is not always possible as there are too many influencing factors.

Runtime on a serial machine is the most commonly used measure.

We need to model the machine first in order to analyze runtimes.

But an exact model will make the analysis too complicated!
So we use an approximate model (e.g., assume unit-cost Random Access Machine model or RAM model).

We may need to approximate even further: e.g., for a sorting algorithm we may count the comparison operations only.

So the predicted running time will only be an approximation!

Performance Bounds

- **worst-case complexity:** maximum complexity over all inputs of a given size
- **average complexity:** average complexity over all inputs of a given size
- **amortized complexity:** worst-case bound on a sequence of operations
- **expected complexity:** for algorithms that make random choices during execution (randomized algorithms)
- **high-probability bound:** when the probability that the complexity holds is $\geq 1 - \frac{c}{n^{\alpha}}$ for input size n, positive constant c and some constant $\alpha \geq 1$

Searching in a Sorted Grid

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

You are given an $n \times n$ grid $A[1{:}n, 1{:}n]$, where $n = 2^m - 1$ for some integer $m > 0$.

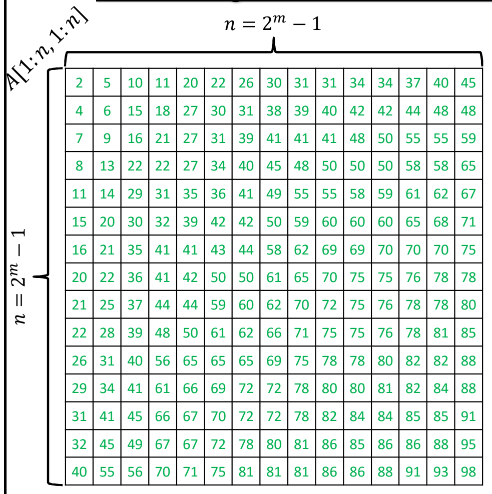
Each grid cell contains a number.

The numbers in each row are sorted in non-decreasing order from left to right.

The numbers in each column are sorted in non-decreasing order from top to bottom.

(Lecture 1) Introduction

Searching in a Sorted Grid (Algorithm 1)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

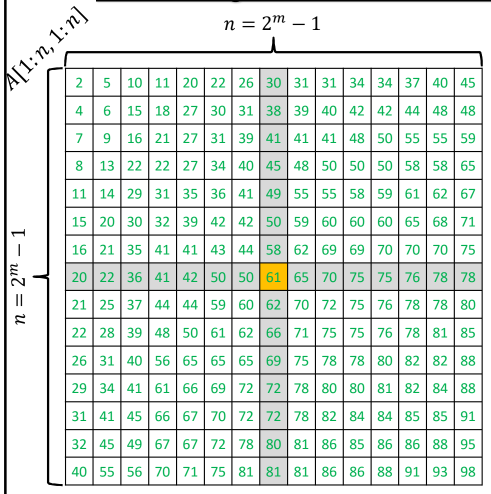
ALGORITHM 1 (SEARCH FOR x):

Scan the entire grid row by row until either x is found, or you are done scanning the entire grid.

Let $Q_1(n)$ = number of comparisons performed on an $n \times n$ grid.

Then $Q_1(n) \le n^2$

Searching in a Sorted Grid (Algorithm 2)



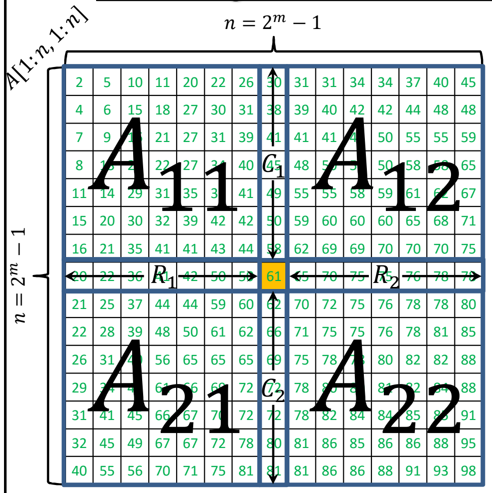
2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

ALGORITHM 2 (SEARCH FOR x):

Let y be the number at the center of the grid (i.e., at the intersection of the mid row and mid column).

- $y = x$: you found the item

Searching in a Sorted Grid (Algorithm 2)

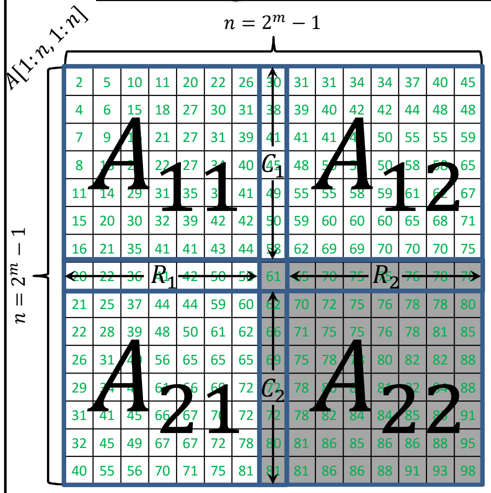


ALGORITHM 2 (SEARCH FOR x):

Let y be the number at the center of the grid (i.e., at the intersection of the mid row and mid column).

- $y = x$: you found the item

Searching in a Sorted Grid (Algorithm 2)



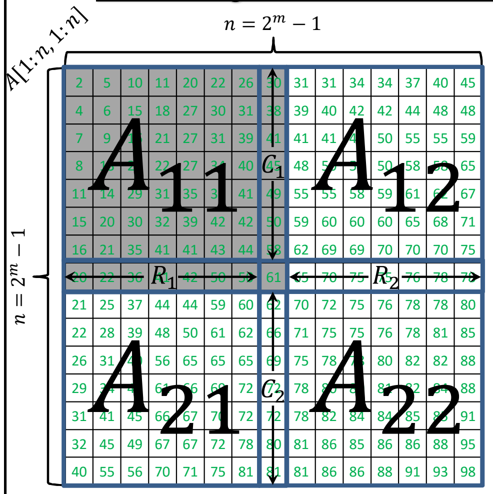
ALGORITHM 2 (SEARCH FOR x):

Let y be the number at the center of the grid (i.e., at the intersection of the mid row and mid column).

- $y = x$: you found the item
- $y > x$: the item cannot be in A_{22}, R_2 and C_2.

  Search for x in R_1 and C_1, and recursively in A_{11}, A_{12} & A_{21}.

Searching in a Sorted Grid (Algorithm 2)



ALGORITHM 2 (SEARCH FOR x):

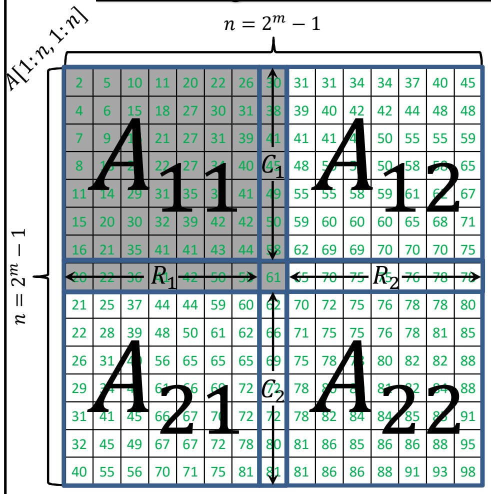
Let y be the number at the center of the grid (i.e., at the intersection of the mid row and mid column).

- $y = x$: you found the item
- $y > x$: the item cannot be in A_{22}, R_2 and C_2.

  Search for x in R_1 and C_1, and recursively in A_{11}, A_{12} & A_{21}.
- $y < x$: the item cannot be in A_{11}, R_1 and C_1.

  Search for x in R_2 and C_2, and recursively in A_{12}, A_{21} & A_{22}.

Searching in a Sorted Grid (Algorithm 2)



Let $Q_2(n)$ = number of comparisons performed on an $n \times n$ grid.

Then
$$Q_2(n) \le 1 + 2\left(\frac{n+1}{2} - 1\right) + 3Q_2\left(\frac{n+1}{2} - 1\right)$$

$$\Rightarrow Q_2(n) \le 3Q_2\left(\frac{n+1}{2} - 1\right) + n$$

Solving:
$$Q_2(n) \le 2(n+1)^{\log_2 3} \le 2(n+1)^{1.6}$$

Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

$n = 2^m - 1$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
left ← 1
right ← n
while left ≤ right do
    mid ← $\frac{left+right}{2}$
    if $A[i, mid] = x$ then
        return "item found"
    else if $A[i, mid] < x$ then
        left ← mid + 1
    else right ← mid − 1
end while
return "item not found"
```

Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

2	5	10	11	20	22	26	**30**	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

$n = 2^m - 1$

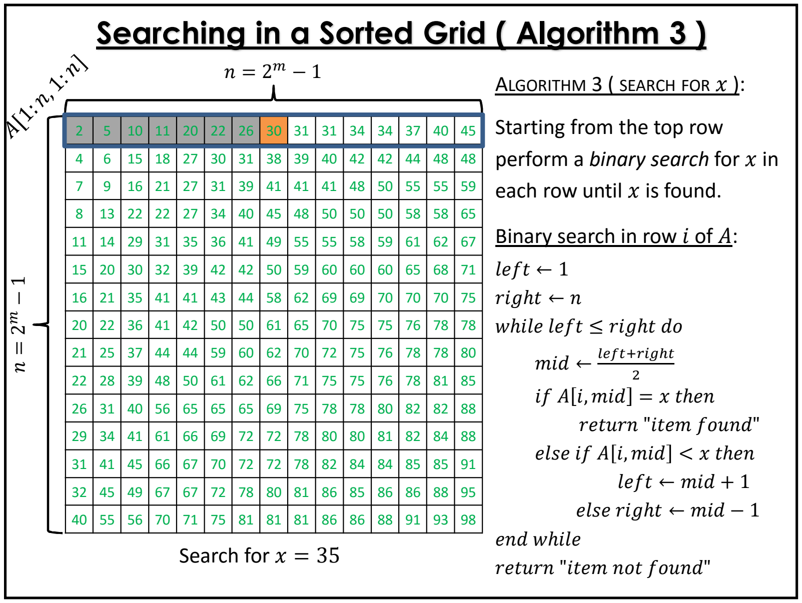
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
left ← 1
right ← n
while left ≤ right do
    mid ← $\frac{left+right}{2}$
    if $A[i, mid] = x$ then
        return "item found"
    else if $A[i, mid] < x$ then
        left ← mid + 1
    else right ← mid − 1
end while
return "item not found"
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

$n = 2^m - 1$

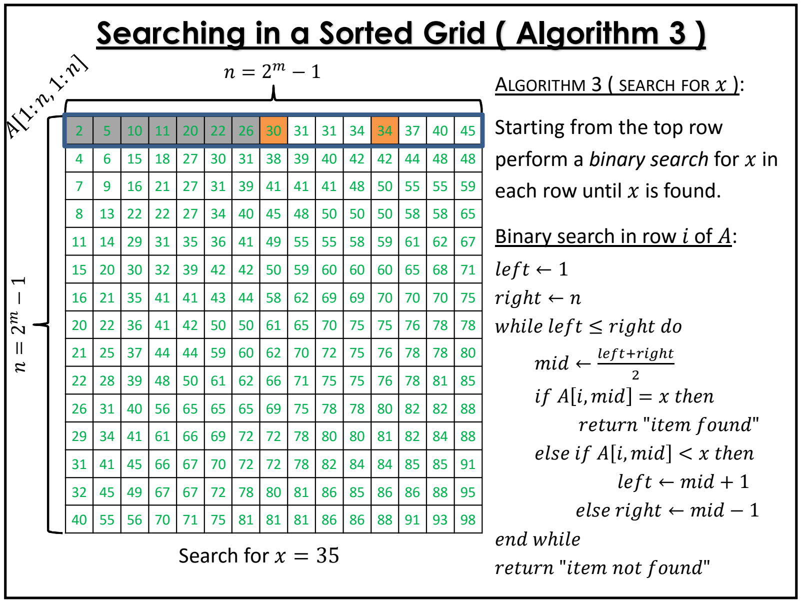
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return$ "item found"
    $else\ if\ A[i, mid] < x\ then$
        $left \leftarrow mid + 1$
    $else\ right \leftarrow mid - 1$
$end\ while$
$return$ "item not found"
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

$n = 2^m - 1$

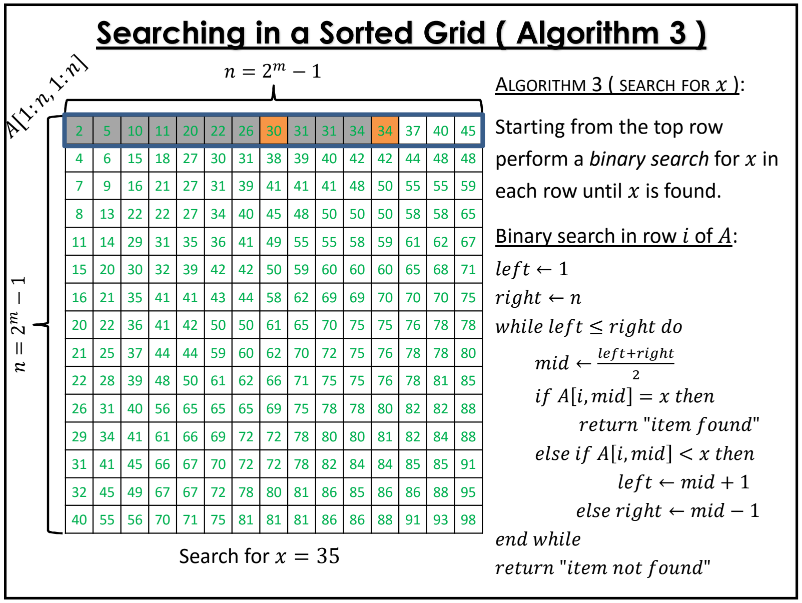
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return$ "item found"
    $else\ if\ A[i, mid] < x\ then$
        $left \leftarrow mid + 1$
    $else\ right \leftarrow mid - 1$
$end\ while$
$return$ "item not found"
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

$n = 2^m - 1$

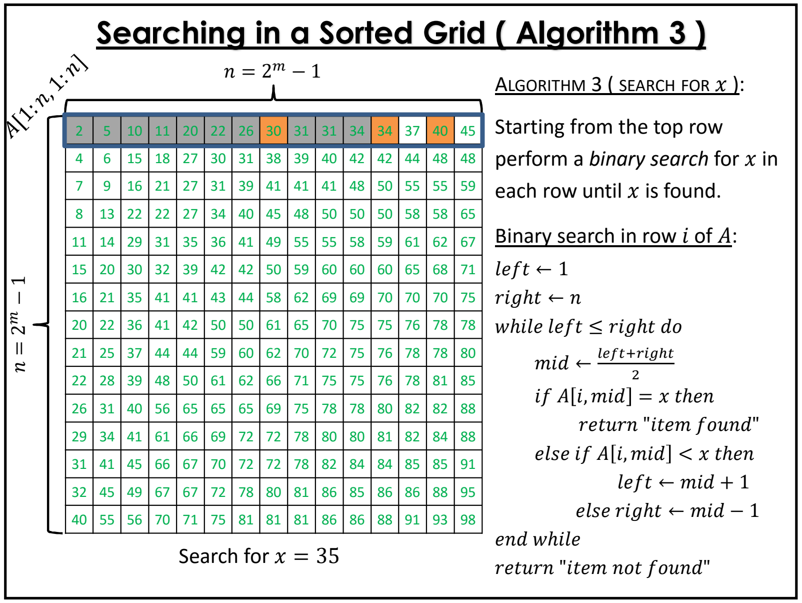
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return$ "item found"
    $else\ if\ A[i, mid] < x\ then$
        $left \leftarrow mid + 1$
    $else\ right \leftarrow mid - 1$
$end\ while$
$return$ "item not found"
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

$n = 2^m - 1$

Search for $x = 35$

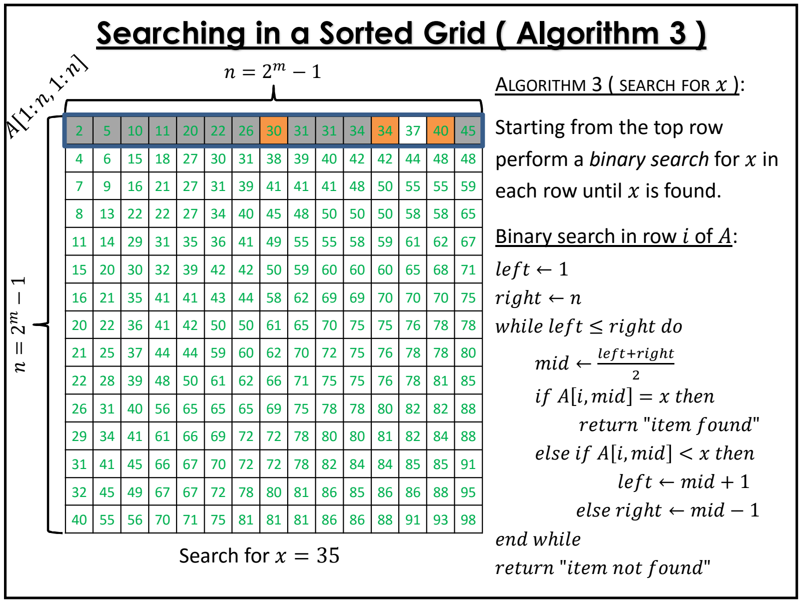
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return$ "item found"
    $else\ if\ A[i, mid] < x\ then$
        $left \leftarrow mid + 1$
    $else\ right \leftarrow mid - 1$
$end\ while$
$return$ "item not found"
```

(Lecture 1) Introduction



Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

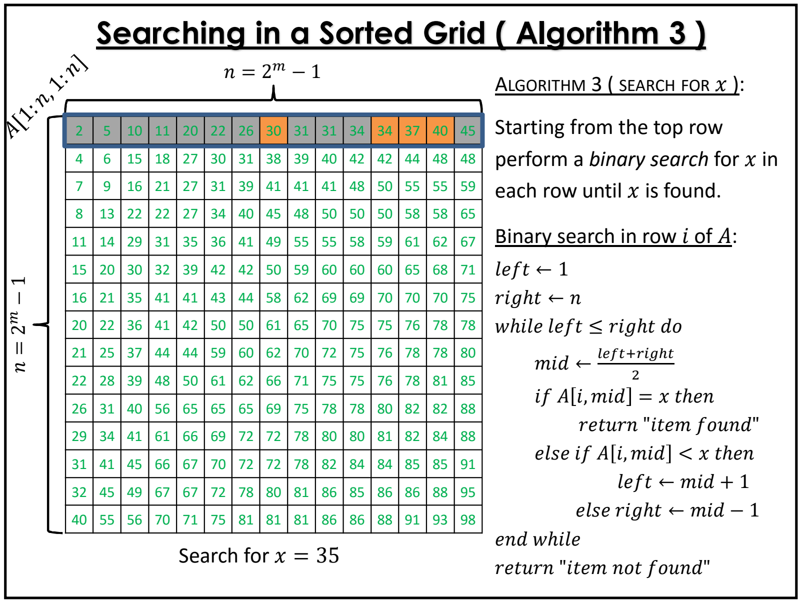
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \le right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```



Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

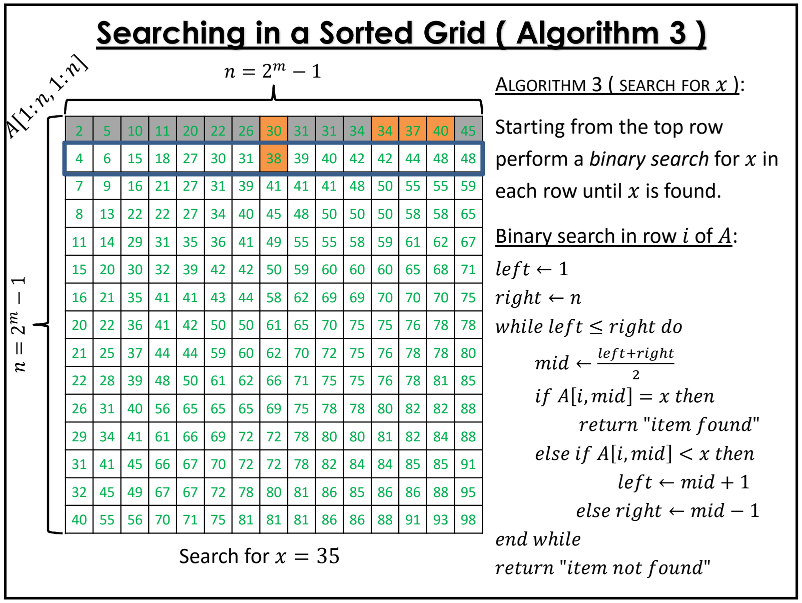
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \le right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```



Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

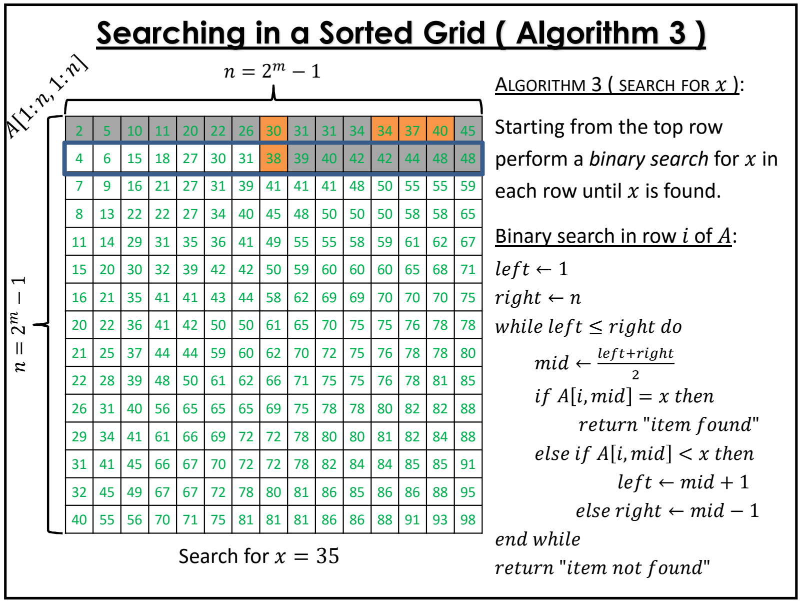
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \le right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```



Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

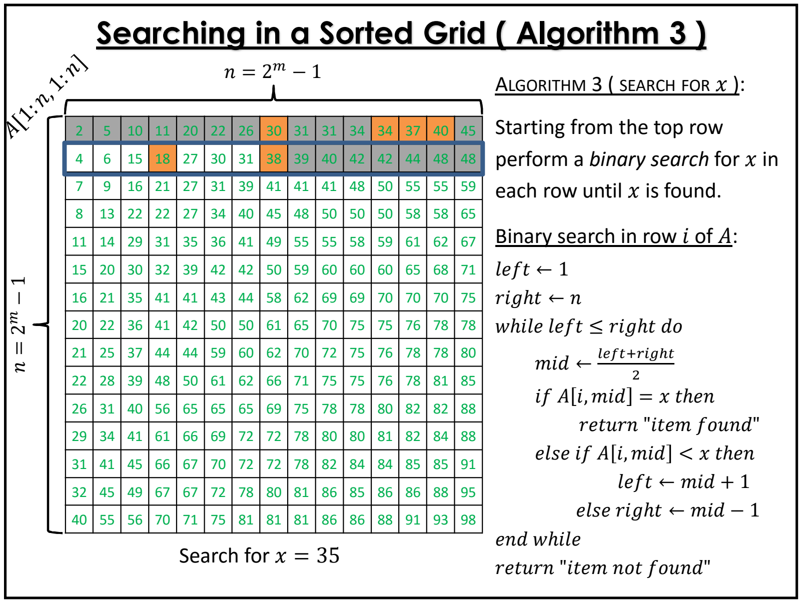
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \le right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

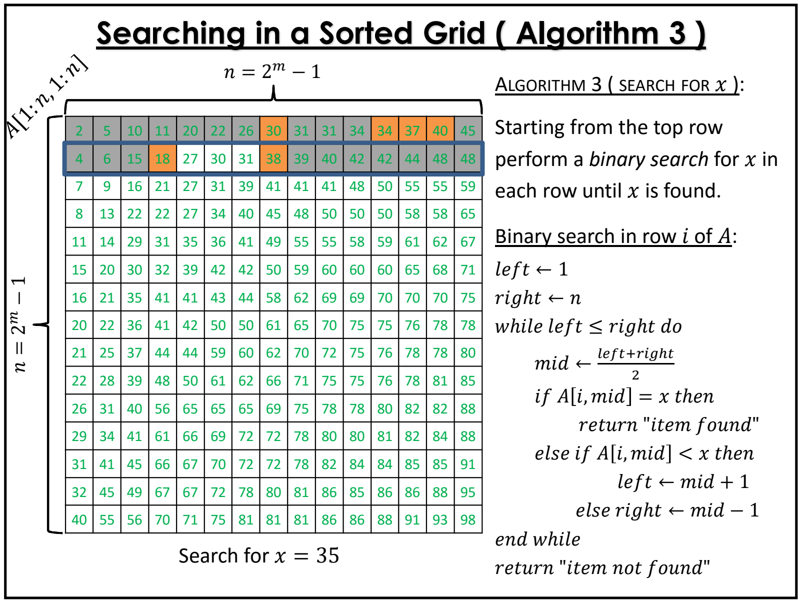
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

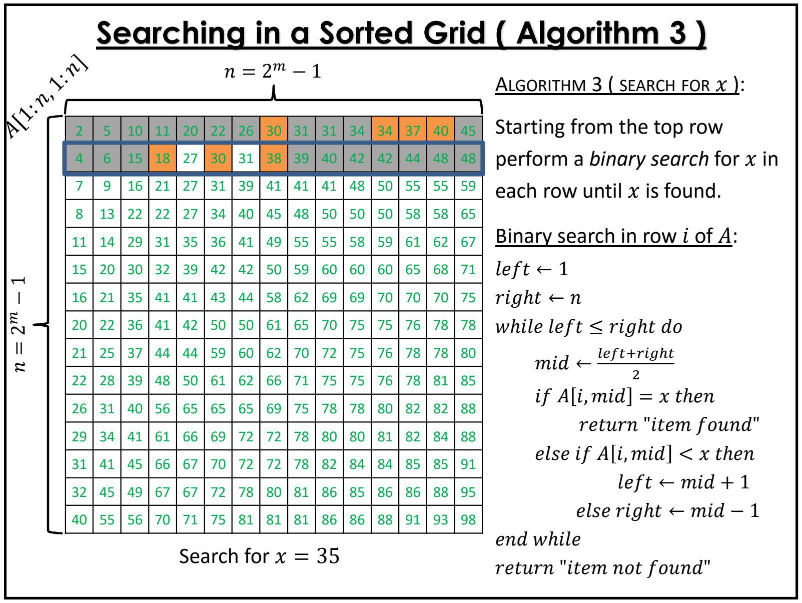
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

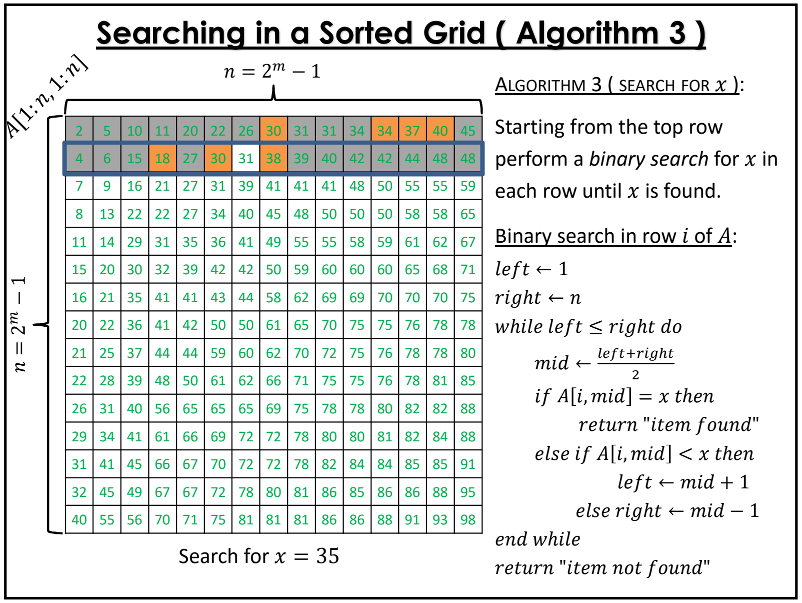
Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```



Searching in a Sorted Grid (Algorithm 3)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```

Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```

Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```

Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```

Searching in a Sorted Grid (Algorithm 3)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

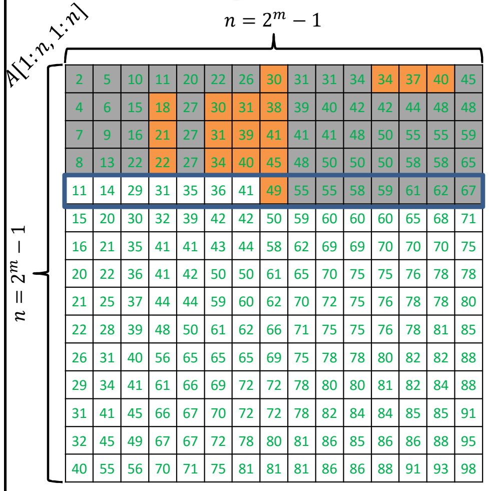
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return\ "item\ found"$
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return\ "item\ not\ found"$
```

Searching in a Sorted Grid (Algorithm 3)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

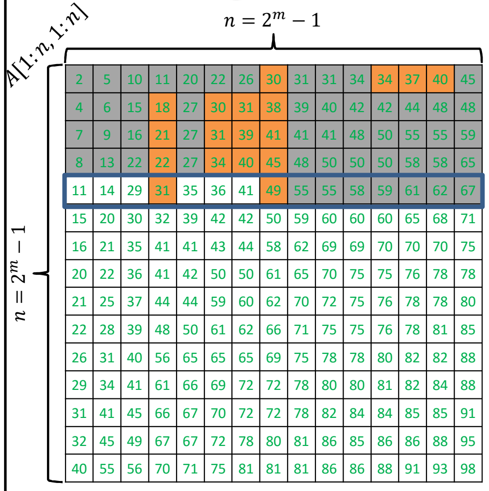
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
left ← 1
right ← n
while left ≤ right do
    mid ← $\frac{left+right}{2}$
    if $A[i, mid] = x$ then
        return "item found"
    else if $A[i, mid] < x$ then
        left ← mid + 1
    else right ← mid − 1
end while
return "item not found"
```

Searching in a Sorted Grid (Algorithm 3)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

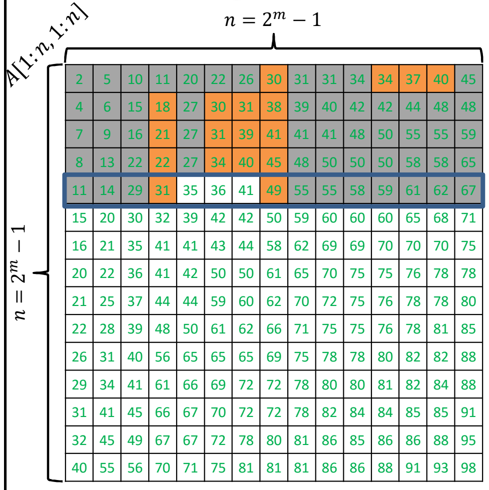
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
left ← 1
right ← n
while left ≤ right do
    mid ← $\frac{left+right}{2}$
    if $A[i, mid] = x$ then
        return "item found"
    else if $A[i, mid] < x$ then
        left ← mid + 1
    else right ← mid − 1
end while
return "item not found"
```

Searching in a Sorted Grid (Algorithm 3)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

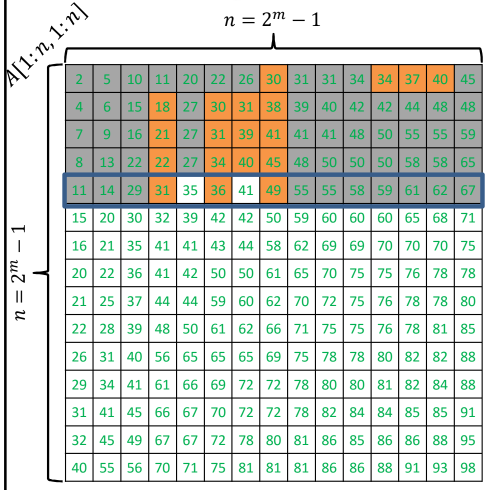
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
left ← 1
right ← n
while left ≤ right do
    mid ← $\frac{left+right}{2}$
    if $A[i, mid] = x$ then
        return "item found"
    else if $A[i, mid] < x$ then
        left ← mid + 1
    else right ← mid − 1
end while
return "item not found"
```

Searching in a Sorted Grid (Algorithm 3)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

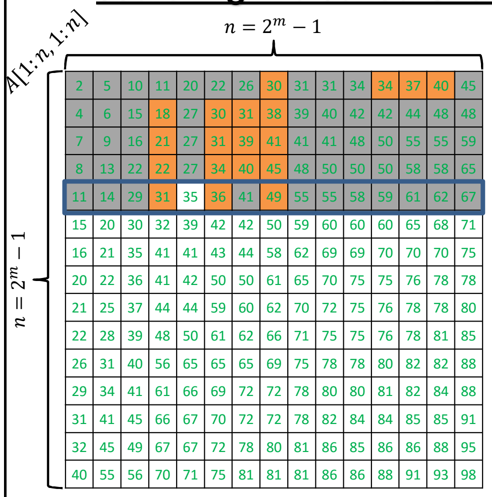
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
left ← 1
right ← n
while left ≤ right do
    mid ← $\frac{left+right}{2}$
    if $A[i, mid] = x$ then
        return "item found"
    else if $A[i, mid] < x$ then
        left ← mid + 1
    else right ← mid − 1
end while
return "item not found"
```

Searching in a Sorted Grid (Algorithm 3)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

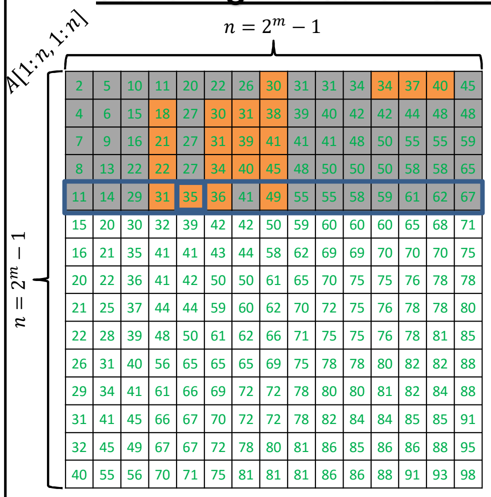
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return$ "$item\ found$"
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return$ "$item\ not\ found$"
```

Searching in a Sorted Grid (Algorithm 3)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 35$

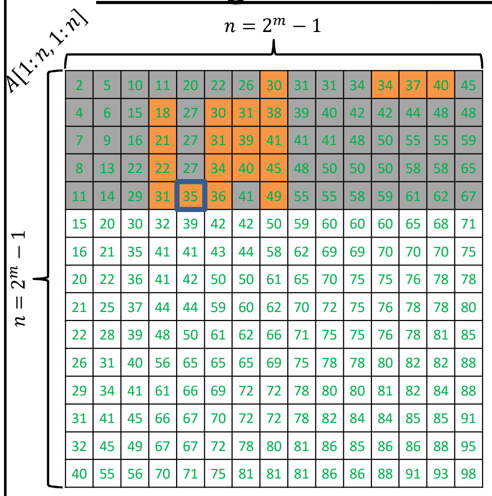
ALGORITHM 3 (SEARCH FOR x):

Starting from the top row perform a *binary search* for x in each row until x is found.

Binary search in row i of A:

```
$left \leftarrow 1$
$right \leftarrow n$
$while\ left \leq right\ do$
    $mid \leftarrow \frac{left+right}{2}$
    $if\ A[i, mid] = x\ then$
        $return$ "$item\ found$"
    $else\ if\ A[i, mid] < x\ then$
            $left \leftarrow mid + 1$
        $else\ right \leftarrow mid - 1$
$end\ while$
$return$ "$item\ not\ found$"
```

Searching in a Sorted Grid (Algorithm 3)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

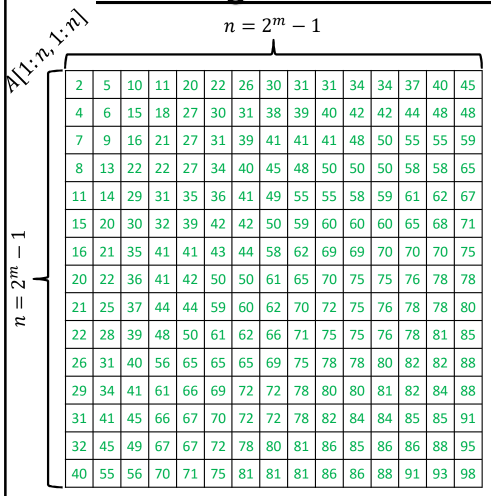
Search for $x = 35$

Let $Q_3(n)$ = number of comparisons performed on an $n \times n$ grid.

Binary search on each row performs m comparisons.

So, $Q_3(n) \leq nm = n \log_2(n + 1)$

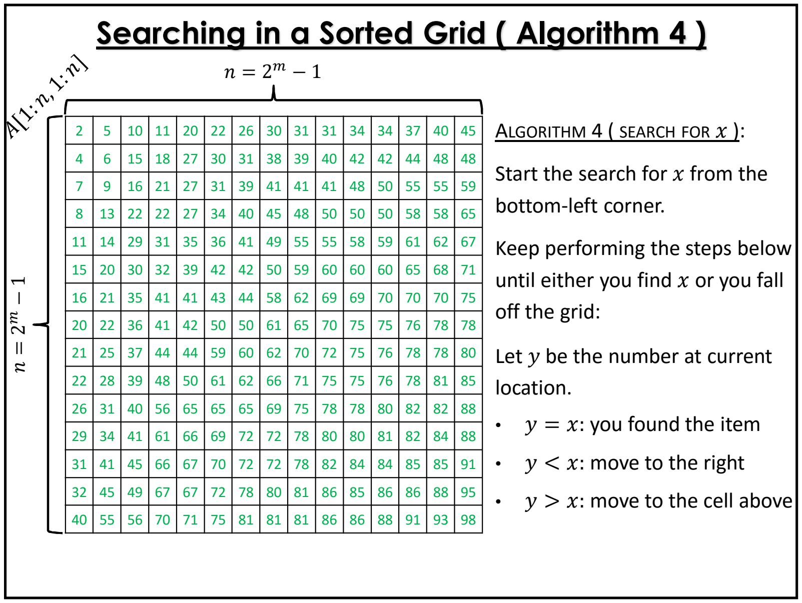
Searching in a Sorted Grid (Algorithm 4)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.



Searching in a Sorted Grid (Algorithm 4)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

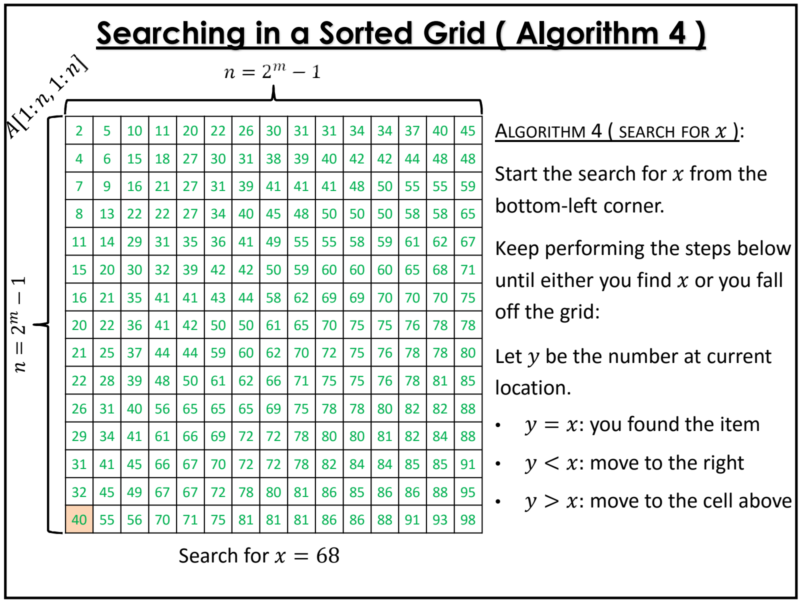
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

ALGORITHM 4 (SEARCH FOR x):

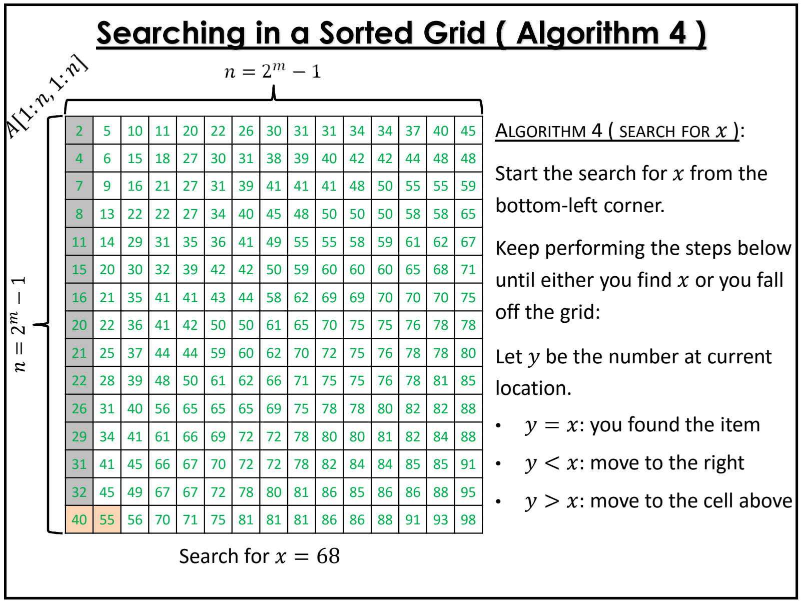
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Search for $x = 68$



Searching in a Sorted Grid (Algorithm 4)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

ALGORITHM 4 (SEARCH FOR x):

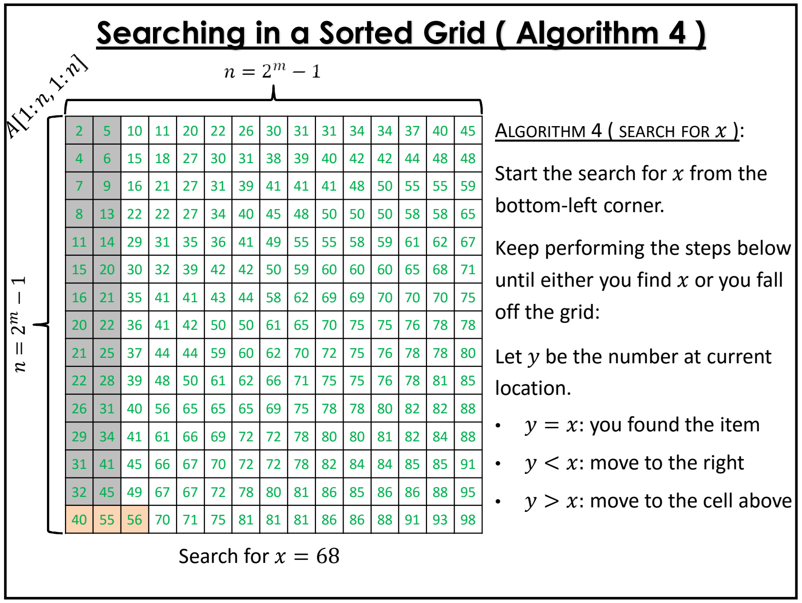
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Search for $x = 68$



Searching in a Sorted Grid (Algorithm 4)

$A[1:n, 1:n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

ALGORITHM 4 (SEARCH FOR x):

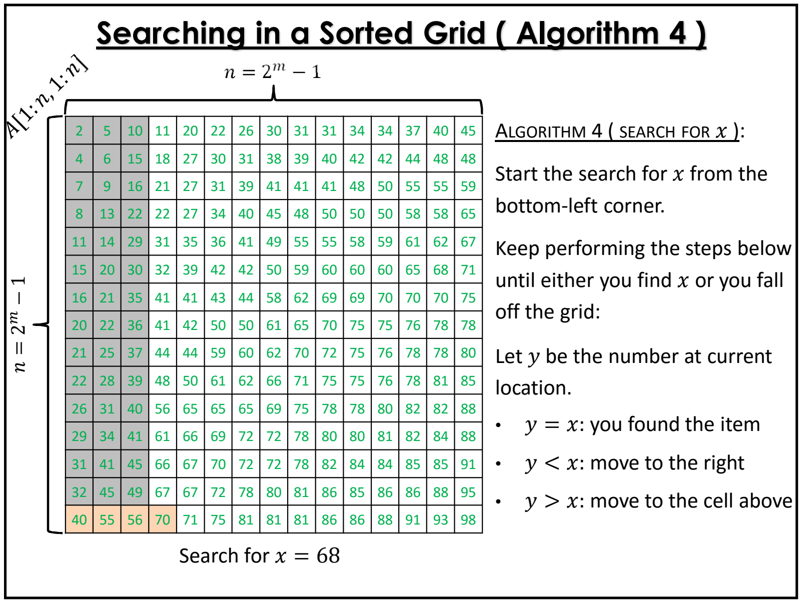
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Search for $x = 68$



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

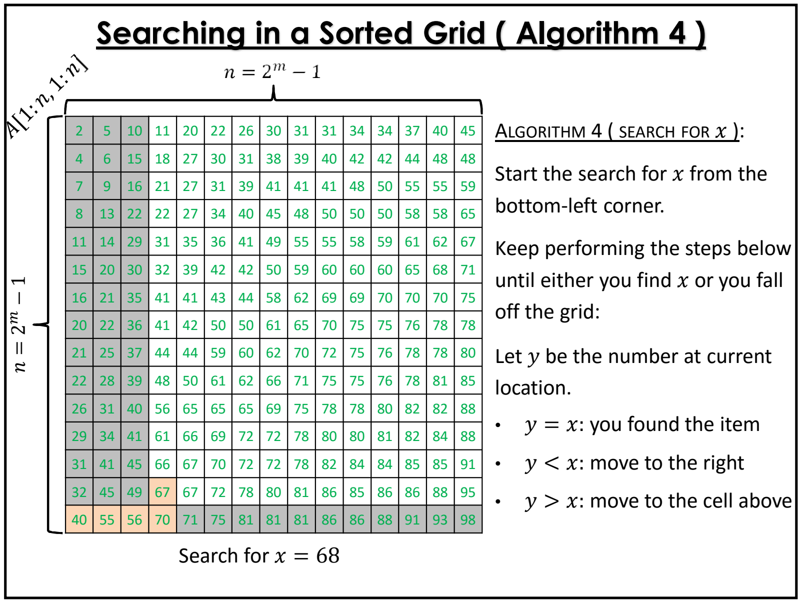
Algorithm 4 (search for x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

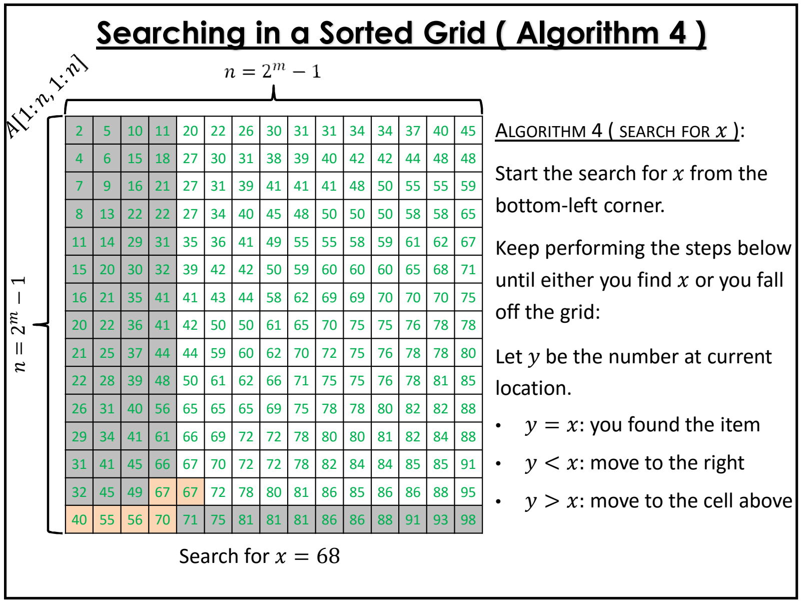
Algorithm 4 (search for x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

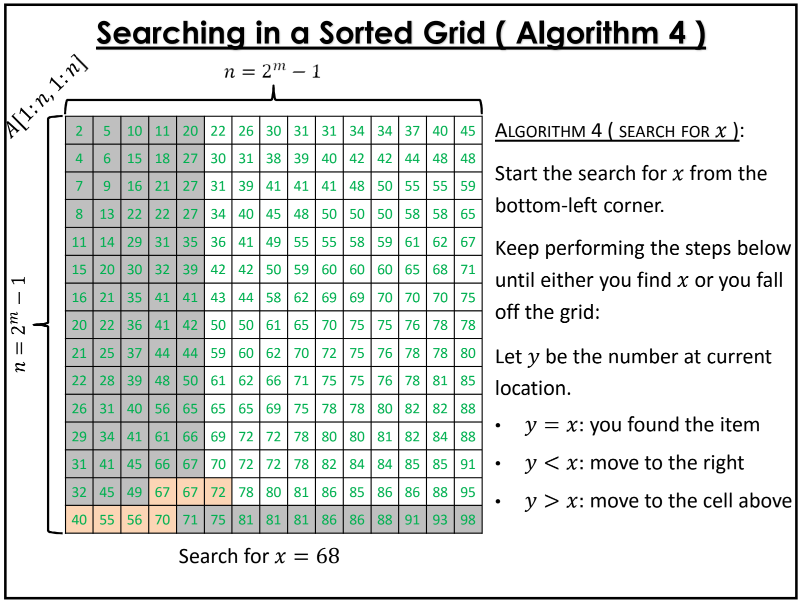
Algorithm 4 (search for x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

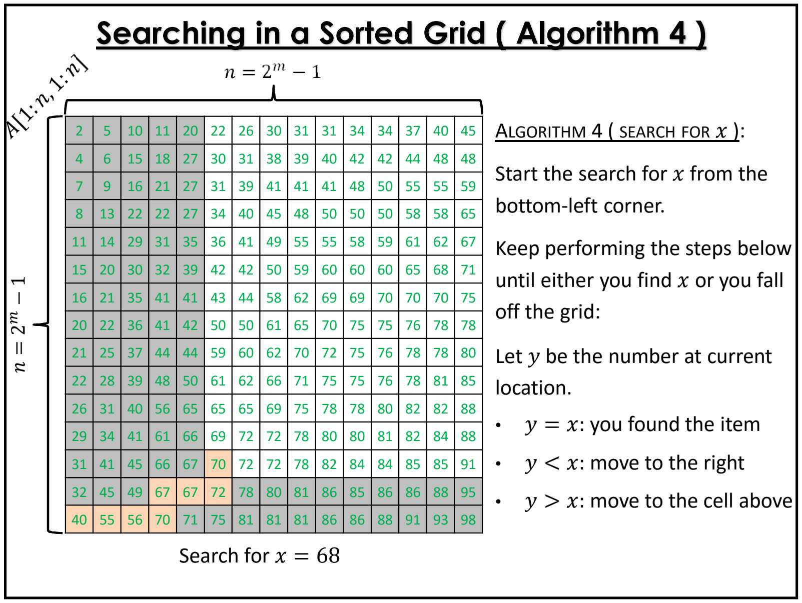
Algorithm 4 (search for x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

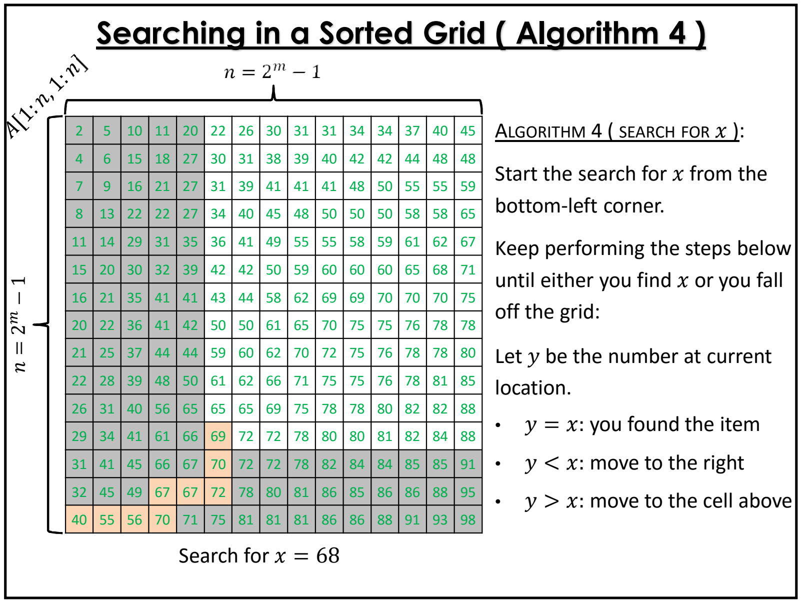
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

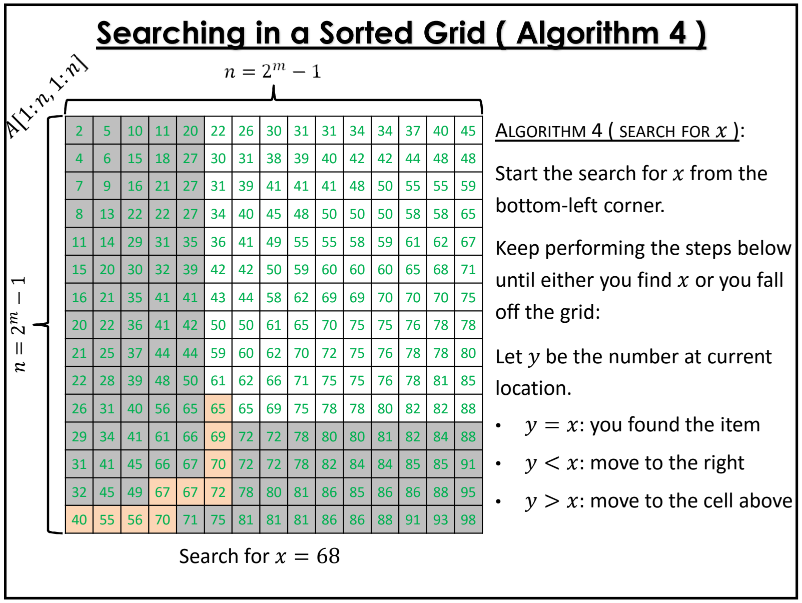
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

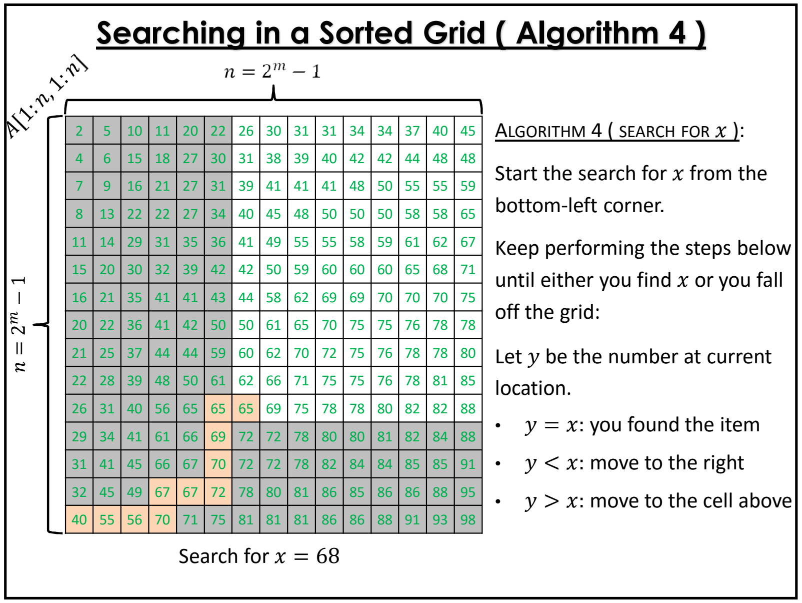
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

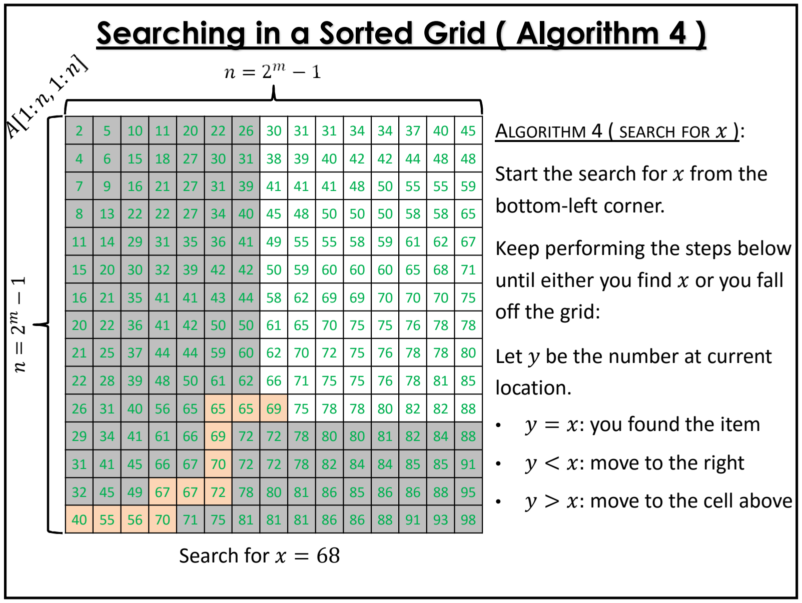
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

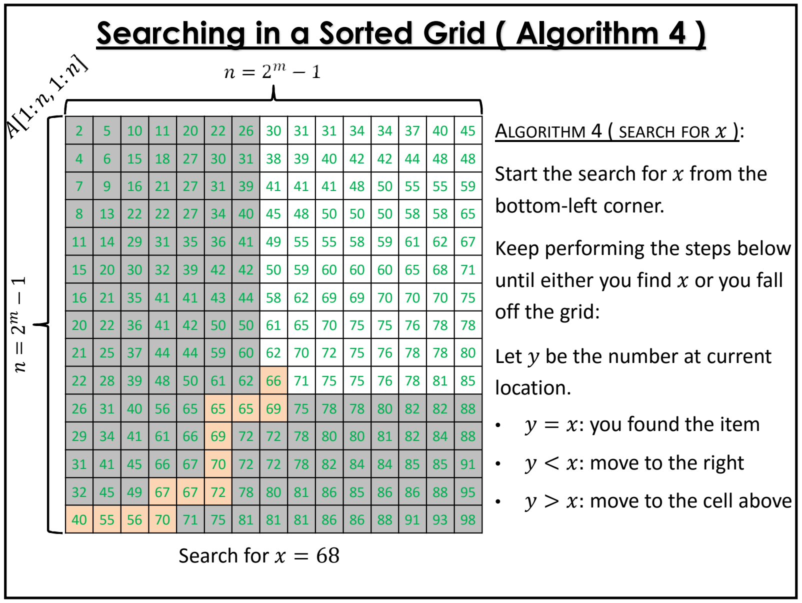
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

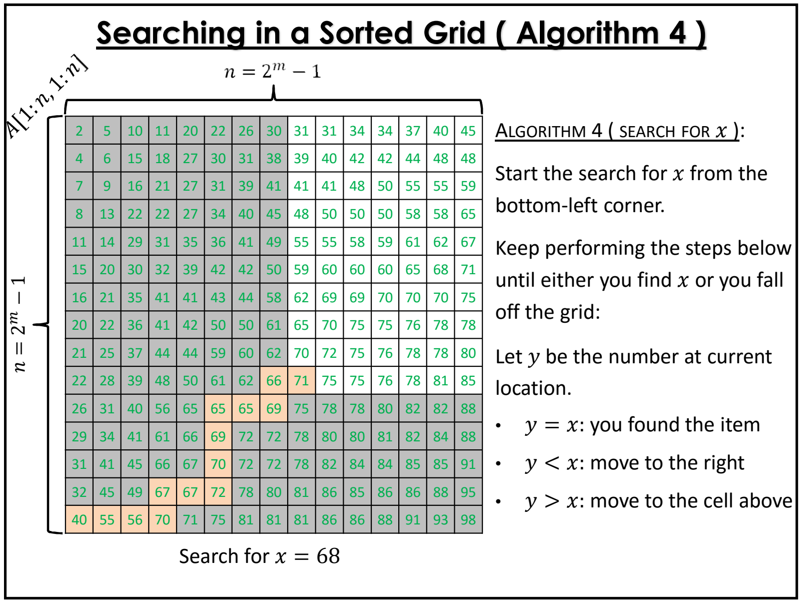
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

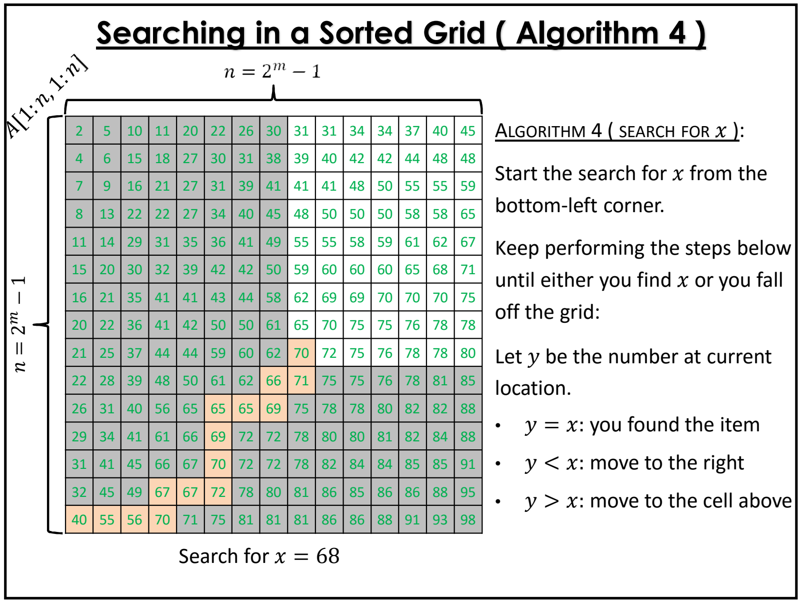
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

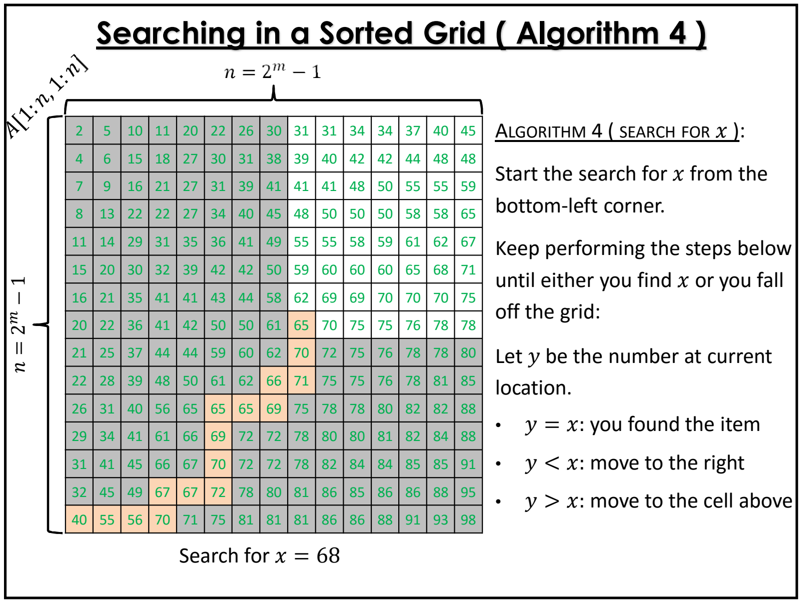
ALGORITHM 4 (SEARCH FOR x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

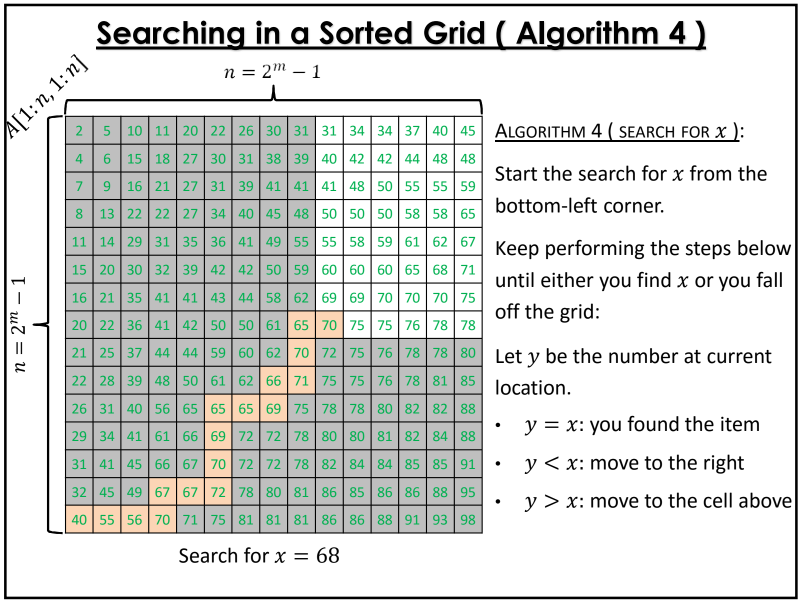
Algorithm 4 (search for x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

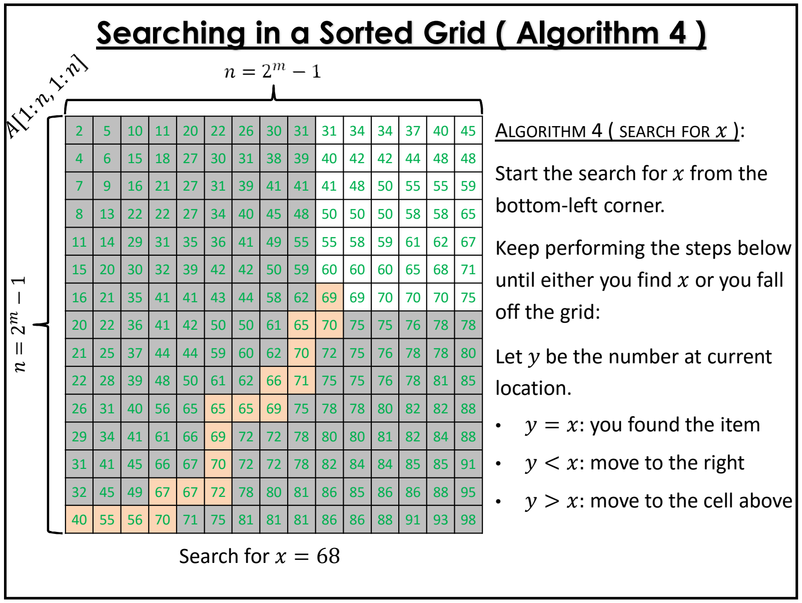
Algorithm 4 (search for x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

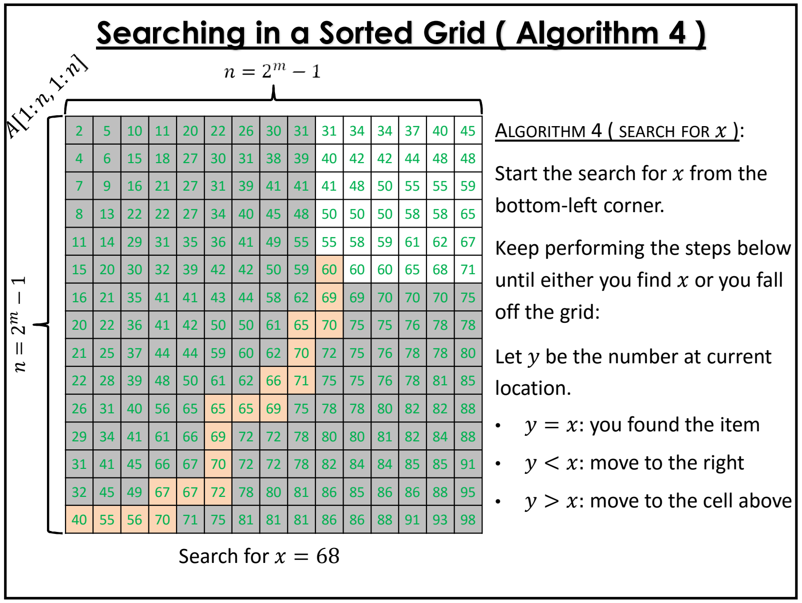
Algorithm 4 (search for x):

Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above



Searching in a Sorted Grid (Algorithm 4)

$A[1{:}n, 1{:}n]$

$n = 2^m - 1$

$n = 2^m - 1$

2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

Algorithm 4 (search for x):

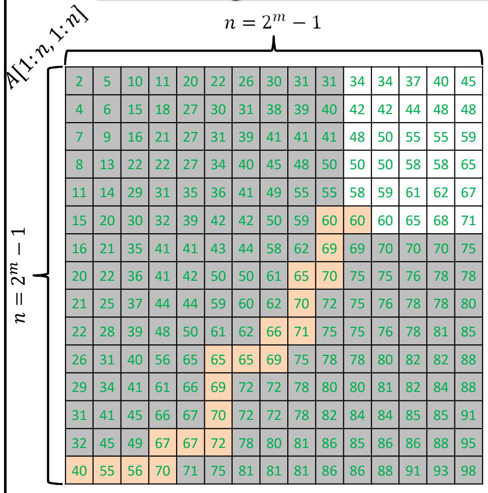
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Searching in a Sorted Grid (Algorithm 4)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

ALGORITHM 4 (SEARCH FOR x):

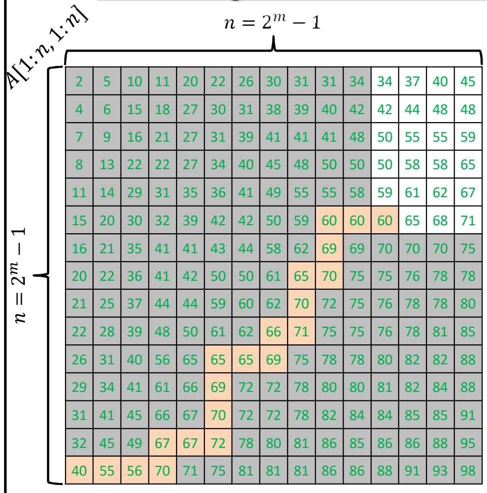
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Searching in a Sorted Grid (Algorithm 4)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

ALGORITHM 4 (SEARCH FOR x):

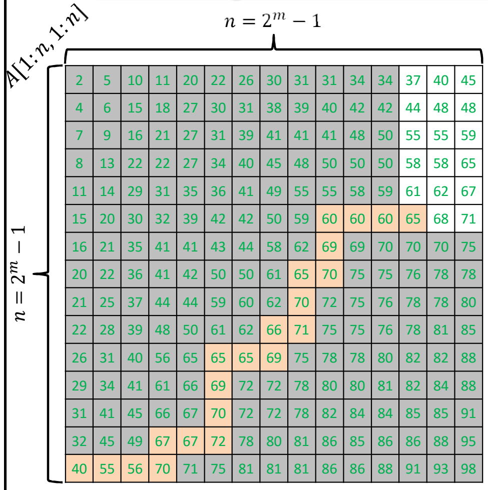
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Searching in a Sorted Grid (Algorithm 4)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

ALGORITHM 4 (SEARCH FOR x):

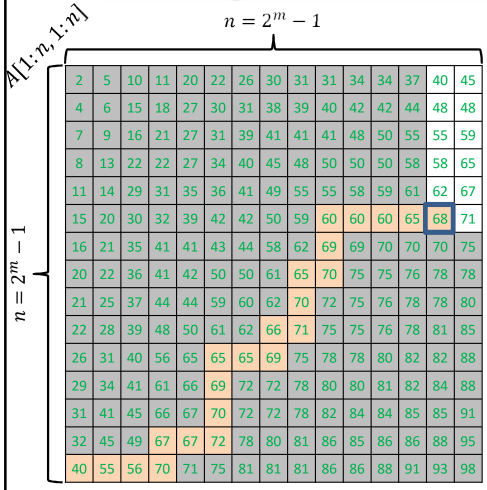
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Searching in a Sorted Grid (Algorithm 4)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

ALGORITHM 4 (SEARCH FOR x):

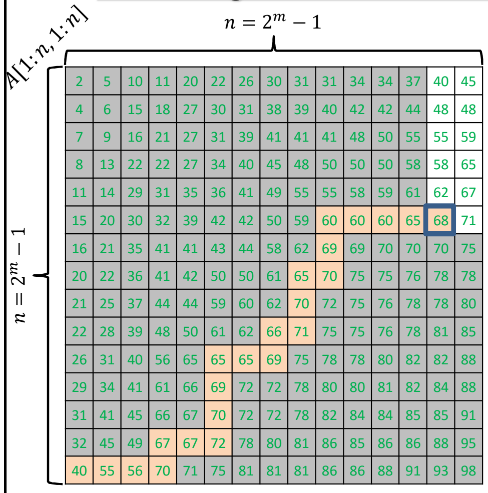
Start the search for x from the bottom-left corner.

Keep performing the steps below until either you find x or you fall off the grid:

Let y be the number at current location.

- $y = x$: you found the item
- $y < x$: move to the right
- $y > x$: move to the cell above

Searching in a Sorted Grid (Algorithm 4)



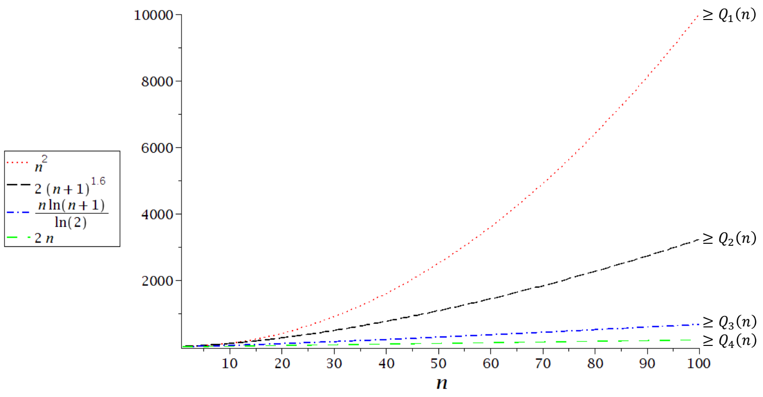
2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

Search for $x = 68$

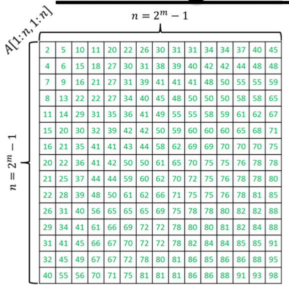
Let $Q_4(n)$ = number of comparisons performed on an $n \times n$ grid.

Then $Q_4(n) \le 2n - 1 < 2n$

Comparing the Four (4) Grid Search Algorithms



Searching in a Sorted Grid (Algorithm 1)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

ALGORITHM 1 (SEARCH FOR x):

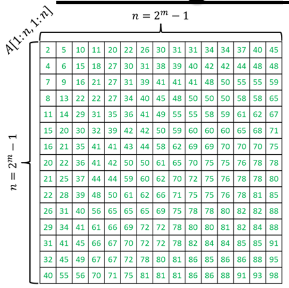
```
1. for $i = 1$ to $n$ do
2.     for $j = 1$ to $n$ do
3.         if $A[i,j] = x$ then return "item found“
4.     end for
5. end for
6. return "item not found"
```

Let $T_1(n)$ = running time of algorithm 1 on an $n \times n$ grid.

Though we were able to compute an exact worst-case bound for $Q_1(n)$, the same cannot be done for $T_1(n)$ because it depends on many other external factors such as CPU speed, programming style, compiler and optimization level used, etc.

But for large values of n, $T_1(n)$'s worst-case value will be within a constant factor of that of $Q_1(n)$. That constant is generally unknown, and depends on the specific hardware and compiler used, expertise of the programmer, etc.

Searching in a Sorted Grid (Algorithm 1)



2	5	10	11	20	22	26	30	31	31	34	34	37	40	45
4	6	15	18	27	30	31	38	39	40	42	42	44	48	48
7	9	16	21	27	31	39	41	41	41	48	50	55	55	59
8	13	22	22	27	34	40	45	48	50	50	50	58	58	65
11	14	29	31	35	36	41	49	55	55	58	59	61	62	67
15	20	30	32	39	42	42	50	59	60	60	60	65	68	71
16	21	35	41	41	43	44	58	62	69	69	70	70	70	75
20	22	36	41	42	50	50	61	65	70	75	75	76	78	78
21	25	37	44	44	59	60	62	70	72	75	76	78	78	80
22	28	39	48	50	61	62	66	71	75	75	76	78	81	85
26	31	40	56	65	65	65	69	75	78	78	80	82	82	88
29	34	41	61	66	69	72	72	78	80	80	81	82	84	88
31	41	45	66	67	70	72	72	78	82	84	84	85	85	91
32	45	49	67	67	72	78	80	81	86	85	86	86	88	95
40	55	56	70	71	75	81	81	81	86	86	88	91	93	98

ALGORITHM 1 (SEARCH FOR x):

```
1. for $i = 1$ to $n$ do
2.     for $j = 1$ to $n$ do
3.         if $A[i,j] = x$ then return "item found“
4.     end for
5. end for
6. return "item not found"
```

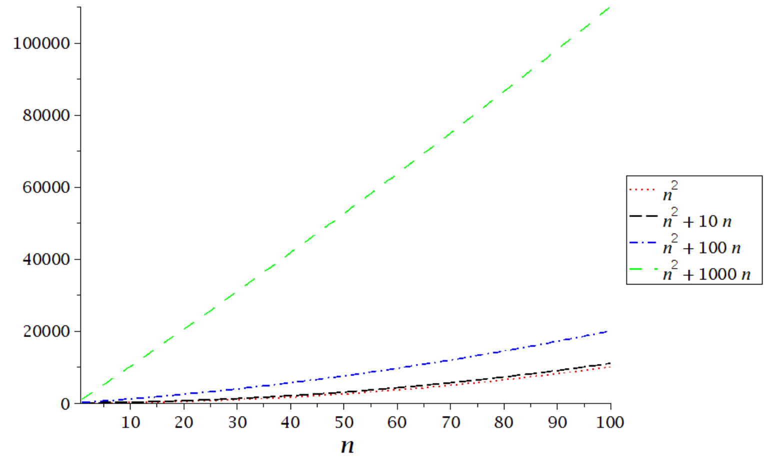
In the worst case,

- line 3 will be executed n^2 times,
- variable j in line 2 will be updated n^2 times,
- variable i in line 1 will be updated n times, and
- line 6 will be executed will be executed 1 time.

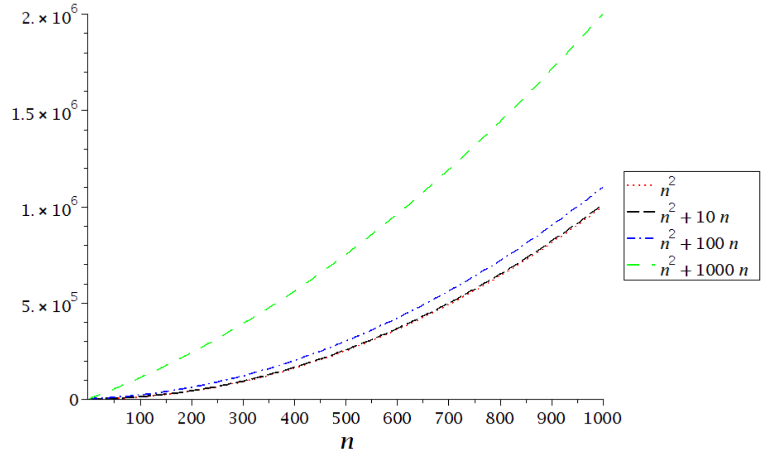
Hence, $T_1(n) \le a_1 n^2 + a_2 n + a_3$, where a_1, a_2 and a_3 are constants.

Clearly, $T_1(n) \le (a_1 + 1)n^2 = (a_1 + 1)Q_1(n)$, when $n \ge a_2 + a_3$.

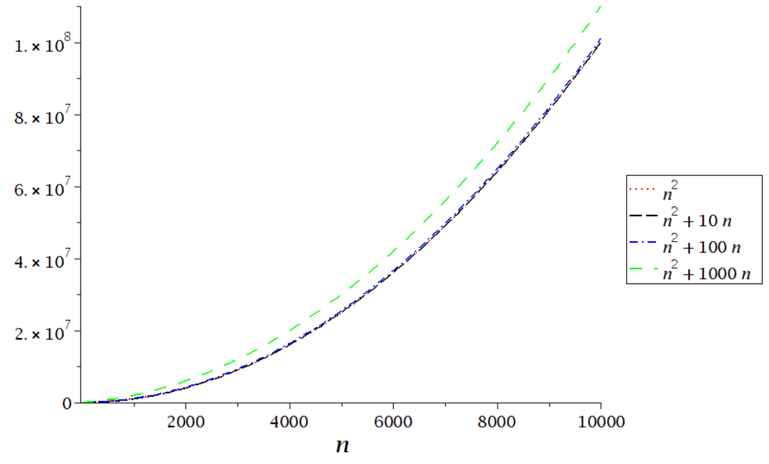
Why Lower Order Terms Can be Dropped



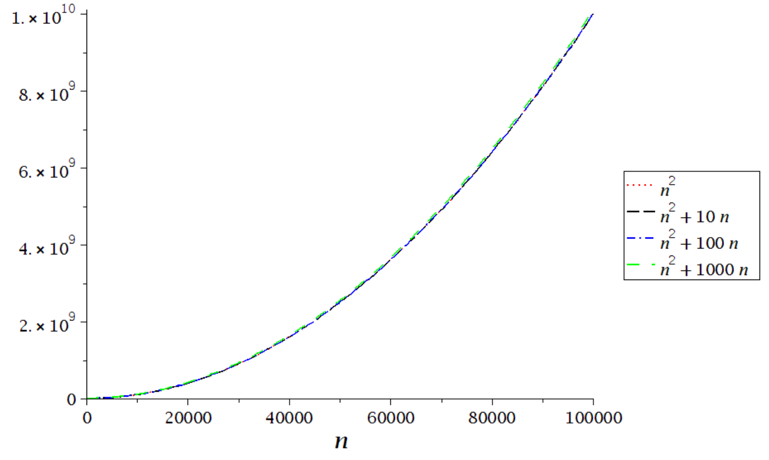
Why Lower Order Terms Can be Dropped



Why Lower Order Terms Can be Dropped



Why Lower Order Terms Can be Dropped



Running Times of the Four (4) Algorithms for Large *n*

Grid Searching Algorithm	Worst-Case Bound on #Comparisons	Worst-Case Bound on Running Times
Algorithm 1	$Q_1(n) \le n^2$	$T_1(n) \le c_1 n^2$
Algorithm 2	$Q_2(n) \le 2(n+1)^{1.6}$	$T_2(n) \le c_2(n+1)^{1.6}$
Algorithm 3	$Q_3(n) \le n \log_2(n+1)$	$T_3(n) \le c_3 n \log_2(n+1)$
Algorithm 4	$Q_4(n) \le 2n$	$T_4(n) \le c_4 n$

c_1, c_2, c_3 *and* c_4 *are constants*

Why Faster Algorithms?

As the input gets large a faster algorithm run on a slow computer will eventually beat a slower algorithm run on a fast computer!

Suppose we run Algorithm 4 on computer A that can execute only 1 million instructions per second. The algorithm was implemented by an inexperienced programmer, and so $c_4 = 10$.

Suppose we run Algorithm 1 on computer B that is 1000 times faster than A, and the algorithm was implemented by an expert programmer, and so $c_1 = 1$.

Let's run both algorithm on a large grid with $n = 100{,}000$.

Then Algorithm 1 will require up to $\frac{1 \times (100000)^2}{1000000000} = 10$ seconds,

while Algorithm 4 will terminate in only $\frac{10 \times 100000}{1000000} = 1$ second!

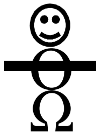
Asymptotic Bounds

We compute performance bounds as functions of input size n.

Asymptotic bounds are obtained when $n \to \infty$.

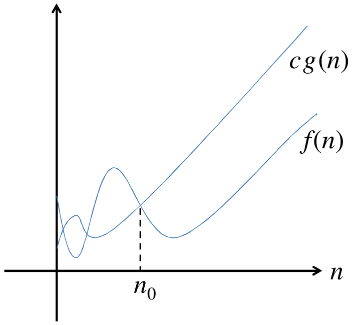
Several types of asymptotic bounds

- upper bound (O-notation)
- strict upper bound (o-notation)
- lower bound (Ω-notation)
- strict lower bound (ω-notation)
- tight bound (Θ-notation)



Asymptotic Stickman
(by Aleksandra Patrzalek)

Asymptotic Upper Bound (O-notation)



$$O(g(n)) = \left\{ \begin{array}{c} f(n)\text{: there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

$$O(g(n)) = \left\{ \begin{array}{c} f(n)\text{: there exists a positive constant } c \text{ such that} \\ \lim_{n \to \infty} \left(\frac{f(n)}{g(n)} \right) \le c \end{array} \right\}$$

Asymptotic Upper Bound (O-notation)

Recall that for Algorithm 1 we had, $T_1(n) \le f(n)$, where,

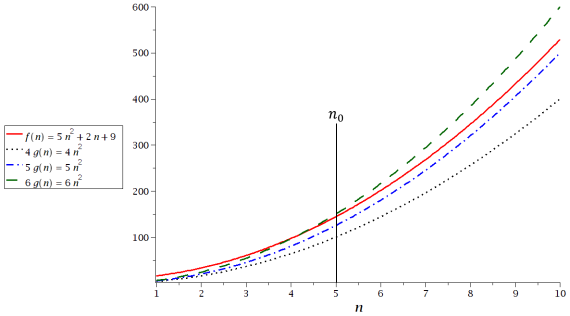
$f(n) = a_1n^2 + a_2n + a_3$, for constants a_1, a_2 and a_3.

Suppose, $a_1 = 5$, $a_2 = 2$ and $a_3 = 9$.

Then $f(n) = 5n^2 + 2n + 9$.

We will now derive asymptotic bounds for $f(n)$.

Asymptotic Upper Bound (O-notation)

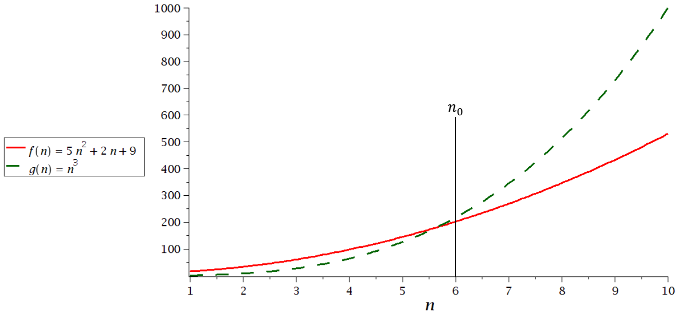


Let $g(n) = n^2$.

Then $f(n) = O\left(n^2\right)$ because:

$0 \le f(n) \le cg(n)$ for $c = 6$ and $n \ge 5$.

Asymptotic Upper Bound (O-notation)

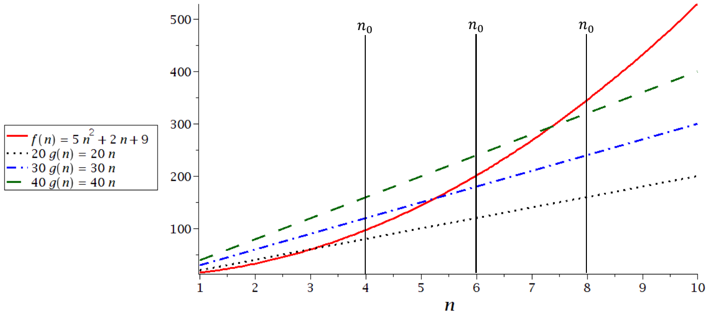


Let $g(n) = n^3$.

Then $f(n) = O\left(n^3\right)$ because:

$0 \le f(n) \le cg(n)$ for $c = 1$ and $n \ge 6$.

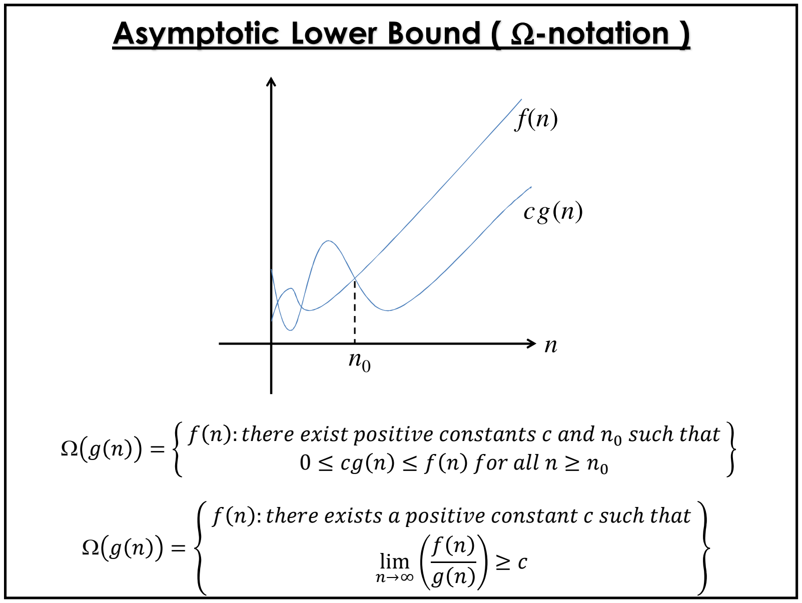
Asymptotic Upper Bound (O-notation)



Let $g(n) = n$.

Then $f(n) \ne O(n)$ because:

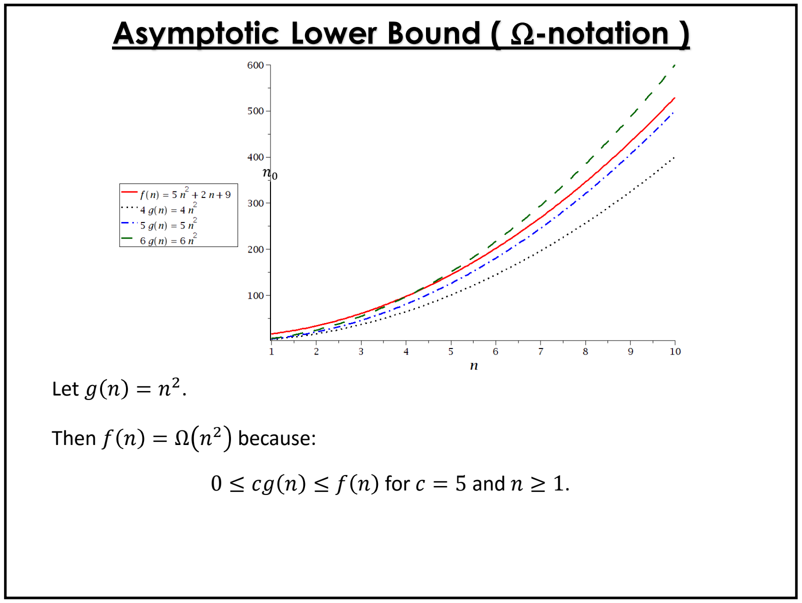
$f(n) > cg(n)$ for any c and $n \ge \frac{c}{5}$.



Asymptotic Lower Bound (Ω-notation)

$$\Omega(g(n)) = \left\{ \begin{array}{c} f(n)\text{: there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

$$\Omega(g(n)) = \left\{ \begin{array}{c} f(n)\text{: there exists a positive constant } c \text{ such that} \\ \lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) \ge c \end{array} \right\}$$

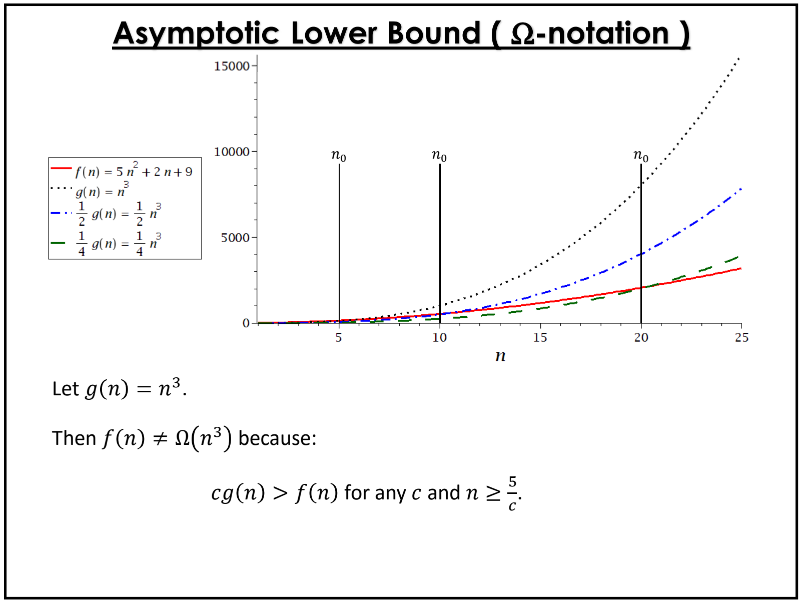


Asymptotic Lower Bound (Ω-notation)

Let $g(n) = n^2$.

Then $f(n) = \Omega(n^2)$ because:

$0 \le cg(n) \le f(n)$ for $c = 5$ and $n \ge 1$.

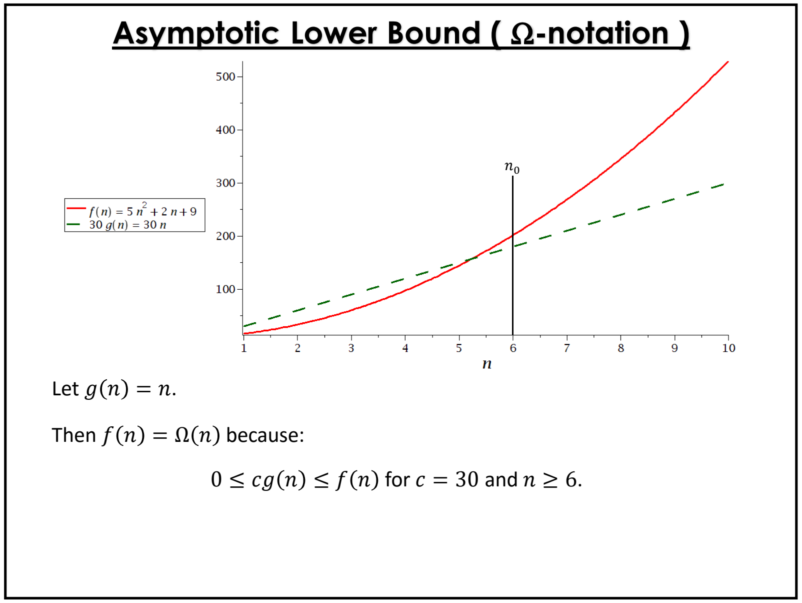


Asymptotic Lower Bound (Ω-notation)

Let $g(n) = n^3$.

Then $f(n) \ne \Omega(n^3)$ because:

$cg(n) > f(n)$ for any c and $n \ge \frac{5}{c}$.



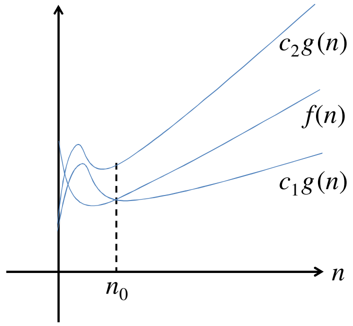
Asymptotic Lower Bound (Ω-notation)

Let $g(n) = n$.

Then $f(n) = \Omega(n)$ because:

$0 \le cg(n) \le f(n)$ for $c = 30$ and $n \ge 6$.

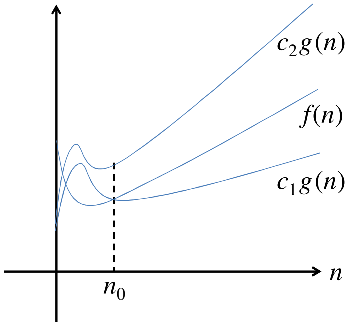
Asymptotic Tight Bound (Θ-notation)



$$\Theta(g(n)) = \left\{ \begin{matrix} f(n)\text{: there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that} \\ 0 \le c_1g(n) \le f(n) \le c_2g(n) \text{ for all } n \ge n_0 \end{matrix} \right\}$$

$$\Theta(g(n)) = \left\{ \begin{matrix} f(n)\text{: there exist positive constants } c_1 \text{ and } c_2 \text{ such that} \\ c_1 \le \lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) \le c_2 \end{matrix} \right\}$$

Asymptotic Tight Bound (Θ-notation)



$$\big(f(n) = O(g(n))\big) \wedge \big(f(n) = \Omega(g(n))\big) \quad \Leftrightarrow \quad \big(f(n) = \Theta(g(n))\big)$$

Asymptotic Tight Bound (Θ-notation)

$f(n) = 5n^2 + 2n + 9$

$f(n) = \Theta(n^2)$ because both $f(n) = O(n^2)$ and $f(n) = \Omega(n^2)$ hold.

$f(n) \ne \Theta(n^3)$ because though $f(n) = O(n^3)$ holds, $f(n) \ne \Omega(n^3)$.

$f(n) \ne \Theta(n)$ because though $f(n) = \Omega(n)$ holds, $f(n) \ne O(n)$.

Asymptotic Strict Upper Bound (o-notation)

$$O(g(n)) = \left\{ \begin{matrix} f(n)\text{: there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \end{matrix} \right\}$$

$$O(g(n)) = \left\{ \begin{matrix} f(n)\text{: there exists a positive constant } c \text{ such that} \\ \lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) \le c \end{matrix} \right\}$$

$$o(g(n)) = \left\{ f(n)\text{: } \lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) = 0 \right\}$$

Asymptotic Strict Lower Bound (ω-notation)

$$\Omega\big(g(n)\big) = \left\{\begin{matrix} f(n)\text{: there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0 \end{matrix}\right\}$$

$$\Omega\big(g(n)\big) = \left\{\begin{matrix} f(n)\text{: there exists a positive constant } c \text{ such that} \\ \lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) \ge c \end{matrix}\right\}$$

$$\omega\big(g(n)\big) = \left\{f(n)\text{: } \lim_{n\to\infty}\left(\frac{g(n)}{f(n)}\right) = 0\right\}$$

Comparing Functions: Transitivity

$f(n) = \mathrm{O}\big(g(n)\big)$ and $g(n) = \mathrm{O}\big(h(n)\big) \quad \Rightarrow \quad f(n) = \mathrm{O}\big(h(n)\big)$

$f(n) = \Omega\big(g(n)\big)$ and $g(n) = \Omega\big(h(n)\big) \quad \Rightarrow \quad f(n) = \Omega\big(h(n)\big)$

$f(n) = \Theta\big(g(n)\big)$ and $g(n) = \Theta\big(h(n)\big) \quad \Rightarrow \quad f(n) = \Theta\big(h(n)\big)$

$f(n) = \mathrm{o}\big(g(n)\big)$ and $g(n) = \mathrm{o}\big(h(n)\big) \quad \Rightarrow \quad f(n) = \mathrm{o}\big(h(n)\big)$

$f(n) = \omega\big(g(n)\big)$ and $g(n) = \omega\big(h(n)\big) \quad \Rightarrow \quad f(n) = \omega\big(h(n)\big)$

Comparing Functions: Reflexivity

$f(n) = \mathrm{O}\big(f(n)\big)$

$f(n) = \Omega\big(f(n)\big)$

$f(n) = \Theta\big(f(n)\big)$

Comparing Functions: Symmetry

$f(n) = \Theta\big(g(n)\big)$ if and only if $g(n) = \Theta\big(f(n)\big)$

Comparing Functions: Transpose Symmetry

$f(n) = \mathrm{O}(g(n))$ if and only if $g(n) = \Omega(f(n))$

$f(n) = \Omega(g(n))$ if and only if $g(n) = \mathrm{O}(f(n))$

Adding Functions

$$\mathrm{O}(f(n)) + \mathrm{O}(g(n)) = \mathrm{O}(f(n) + g(n))$$

$$\Omega(f(n)) + \Omega(g(n)) = \Omega(f(n) + g(n))$$

$$\Theta(f(n)) + \Theta(g(n)) = \Theta(f(n) + g(n))$$

Multiplying Functions by Constants

$$\mathrm{O}(c\, f(n)) = \mathrm{O}(f(n))$$

$$\Omega(cf(n)) = \Omega(f(n))$$

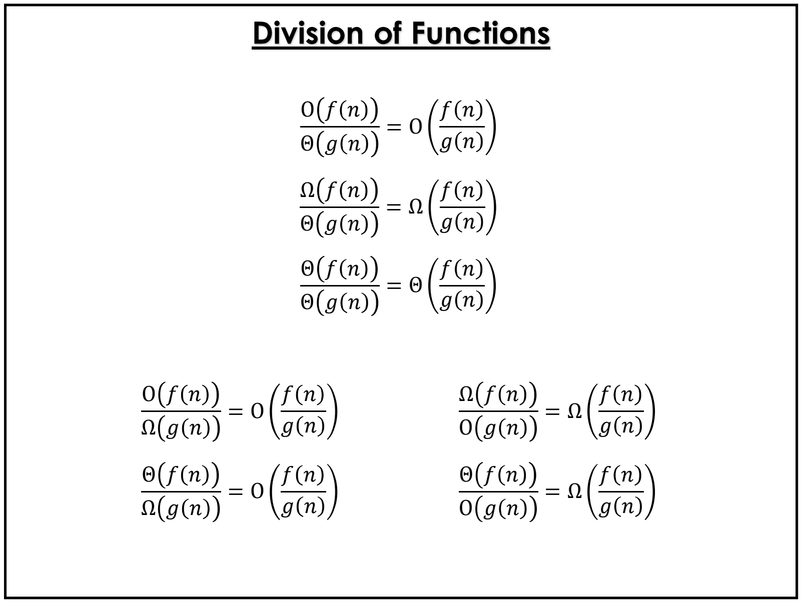
$$\Theta(cf(n)) = \Theta(f(n))$$

Multiplying Two Functions

$$\mathrm{O}(f(n)) \times \mathrm{O}(g(n)) = \mathrm{O}(f(n) \times g(n))$$

$$\Omega(f(n)) \times \Omega(g(n)) = \Omega(f(n) \times g(n))$$

$$\Theta(f(n)) \times \Theta(g(n)) = \Theta(f(n) \times g(n))$$



Division of Functions

$$\frac{O(f(n))}{\Theta(g(n))} = O\left(\frac{f(n)}{g(n)}\right)$$

$$\frac{\Omega(f(n))}{\Theta(g(n))} = \Omega\left(\frac{f(n)}{g(n)}\right)$$

$$\frac{\Theta(f(n))}{\Theta(g(n))} = \Theta\left(\frac{f(n)}{g(n)}\right)$$

$$\frac{O(f(n))}{\Omega(g(n))} = O\left(\frac{f(n)}{g(n)}\right) \qquad \frac{\Omega(f(n))}{O(g(n))} = \Omega\left(\frac{f(n)}{g(n)}\right)$$

$$\frac{\Theta(f(n))}{\Omega(g(n))} = O\left(\frac{f(n)}{g(n)}\right) \qquad \frac{\Theta(f(n))}{O(g(n))} = \Omega\left(\frac{f(n)}{g(n)}\right)$$