“Theory is when you know everything, but nothing works. Practice is when everything works but no one knows why. In our lab, theory and practice are combined: nothing works and no one knows why.”

— A practical theoretician
Basic Logistics: Who/Where/When

- **Lecture Time:** TuTh 7:00 pm - 8:20 pm
- **Location:** Frey Hall Room 102, West Campus
  Lectures will be live streamed via Echo360
- **Instructor:** Rezaul A. Chowdhury
- **Office Hours:** Tue/Thu 12:00 pm - 1:30 pm
  Zoom link available on Brightspace
- **Email:** rezaul@cs.stonybrook.edu
- **TA:** TBA
- **Class Webpage:**
- **Piazza:**
  [https://piazza.com/stonybrook/fall2023/cse548ams542](https://piazza.com/stonybrook/fall2023/cse548ams542)
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
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%
  - Form groups of up to three for problem solving.
  - Each group will submit only one copy of their solutions through Brightspace.
  - Each group must report approximate % contribution of each member in solving each problem set.

- Two Exams (higher one 30%, lower one 15%): 45%
  - Midterm 1 (in-class): Oct 12
  - Midterm 2 (in-class): Nov 30
  - No final exam.

- Scribe note (one lecture): 10%

- Class participation & attendance: 5%
Textbooks

Recommended


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: Selection Sort solves the sorting problem specified as a relationship between the input and the output as follows.

**Input:** A sequence of \( n \) numbers \( \langle a_1, a_2, ..., a_n \rangle \).

**Output:** A permutation \( \langle a'_1, a'_2, ..., a'_n \rangle \) of the input sequence such that \( a'_1 \leq a'_2 \leq \cdots \leq a'_n \).
**Selection Sort**

**State 0**

Input array

```
1 2 3 4 5 6 7 8 9 10
```

Swap $A[1]$ with the smallest number in $A[1..10]$

**State 1**

$A[1]$ is now sorted

```
1 2 3 4 5 6 7 8 9 10
```


**State 2**

$A[1..2]$ is now sorted

```
1 2 3 4 5 6 7 8 9 10
```


**State 3**

$A[1..3]$ is now sorted

```
1 2 3 4 5 6 7 8 9 10
```


**State 4**

$A[1..4]$ is now sorted

```
1 2 3 4 5 6 7 8 9 10
```


**State 5**

$A[1..5]$ is now sorted

```
1 2 3 4 5 6 7 8 9 10
```
Selection Sort

State 5
A[1..5] is now sorted

1 2 3 4 5 6 7 8 9 10

A 1 3 4 5 7 15 12 9 18 11


State 6
A[1..6] is now sorted

1 2 3 4 5 6 7 8 9 10

A 1 3 4 5 7 9 12 15 18 11


State 7
A[1..7] is now sorted

1 2 3 4 5 6 7 8 9 10

A 1 3 4 5 7 9 11 15 18 12


State 8
A[1..8] is now sorted

1 2 3 4 5 6 7 8 9 10

A 1 3 4 5 7 9 11 12 18 15


State 9
A[1..9] is now sorted

1 2 3 4 5 6 7 8 9 10

A 1 3 4 5 7 9 11 12 15 18

Do nothing!

State 10
A[1..10] is now sorted

1 2 3 4 5 6 7 8 9 10

A 1 3 4 5 7 9 11 12 15 18
Selection Sort

**Input:** An array $A[1:n]$ of $n$ numbers.


```
SELECTION-SORT ( A )
1.  for $j = 1$ to $A$.length - 1
2.    // find the index of an entry with the smallest value in $A[j..A.length]$
3.    $min = j$
4.    for $i = j + 1$ to $A.length$
5.      if $A[i] < A[min]$
6.        $min = i$
7.    // swap $A[j]$ and $A[min]$
```
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
       ( usually efficient but 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

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.
**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 $Q_1(n) \leq n^2$
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
Searching in a Sorted Grid (Algorithm 2):

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 \( A_{11}, A_{12} \) & \( A_{21} \).
**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 \( A_{12}, A_{21} \) & \( A_{22} \).
Let $Q_2(n) = \text{number of comparisons performed on an } n \times n \text{ grid.}$

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

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

Solving: $Q_2(n) \leq \left( \frac{3}{2} \right) (n + 1)^{\log_2 3}$

$\leq \left( \frac{3}{2} \right) (n + 1)^{1.6}$
### Searching in a Sorted Grid (Algorithm 3)

\[ n = 2^m - 1 \]

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 2 | 5 | 10 | 11 | 20 | 22 | 26 | 30 | 31 | 34 | 37 | 40 | 45 |
| 4 | 6 | 15 | 18 | 27 | 30 | 31 | 38 | 39 | 40 | 42 | 44 | 48 | 48 |
| 7 | 9 | 16 | 21 | 27 | 31 | 39 | 41 | 41 | 48 | 50 | 55 | 55 | 59 |
| 8 | 13 | 22 | 22 | 27 | 34 | 40 | 45 | 48 | 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 | 65 | 68 | 71 |
| 16 | 21 | 35 | 41 | 41 | 43 | 44 | 58 | 62 | 69 | 69 | 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 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 \( A \):**

- \( \text{left} \leftarrow 1 \)
- \( \text{right} \leftarrow n \)
- while \( \text{left} \leq \text{right} \) do
  - \( \text{mid} \leftarrow \frac{\text{left} + \text{right}}{2} \)
  - if \( A[i, \text{mid}] = x \) then
    - return "item found"
  - else if \( A[i, \text{mid}] < x \) then
    - \( \text{left} \leftarrow \text{mid} + 1 \)
  - else \( \text{right} \leftarrow \text{mid} - 1 \)
- end while
- return "item not found"
### Searching in a Sorted Grid (Algorithm 3)

**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 ← \( \frac{\text{left} + \text{right}}{2} \)
    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"
```

**Example:**

Search for $x = 35$

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
</tr>
<tr>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
</tr>
<tr>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
</tr>
<tr>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
</tr>
<tr>
<td>79</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
</tr>
<tr>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>90</td>
</tr>
<tr>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
</tr>
<tr>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
<td>101</td>
<td>102</td>
</tr>
</tbody>
</table>

$n = 2^m - 1$
Searching in a Sorted Grid (Algorithm 3):

Starting from the top row, perform a binary search for \( x \) in each row until \( x \) is found.

Binary search in row \( i \) of \( A \):

\[
\text{left} \leftarrow 1 \\
\text{right} \leftarrow n \\
\text{while } \text{left} \leq \text{right} \text{ do} \\
\quad \text{mid} \leftarrow \frac{\text{left} + \text{right}}{2} \\
\quad \text{if } A[i, \text{mid}] = x \text{ then} \\
\quad \quad \text{return "item found"} \\
\quad \text{else if } A[i, \text{mid}] < x \text{ then} \\
\quad \quad \text{left} \leftarrow \text{mid} + 1 \\
\quad \text{else right} \leftarrow \text{mid} - 1 \\
\text{end while} \\
\text{return "item not found"}
\]
# Searching in a Sorted Grid (Algorithm 3)

$n = 2^m - 1$

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

**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)

\[ n = 2^m - 1 \]

\[ n = 2^m - 1 \]

**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 \( A \):**

\[ \text{left} \leftarrow 1 \]
\[ \text{right} \leftarrow n \]

**Algorithm**:

while \( \text{left} \leq \text{right} \) do

\[ \text{mid} \leftarrow \frac{\text{left} + \text{right}}{2} \]

if \( A[i, \text{mid}] = x \) then

return "item found"

else if \( A[i, \text{mid}] < x \) then

\[ \text{left} \leftarrow \text{mid} + 1 \]

else right \leftarrow \text{mid} - 1

end while

return "item not found"
Searching in a Sorted Grid (Algorithm 3)

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)

**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 \( A \):**

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

**Table:**

<table>
<thead>
<tr>
<th>( n = 2^m - 1 )</th>
<th>( n = 2^m - 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n = 2^m - 1 )</td>
<td>( n = 2^m - 1 )</td>
</tr>
<tr>
<td>( 2 ) 5 10 11 20 22 26 30 31 31 34 34 37 40 45</td>
<td>( 2 ) 5 10 11 20 22 26 30 31 31 34 34 37 40 45</td>
</tr>
<tr>
<td>( 4 ) 6 15 18 27 30 31 38 39 40 42 42 44 48 48</td>
<td>( 4 ) 6 15 18 27 30 31 38 39 40 42 42 44 48 48</td>
</tr>
<tr>
<td>( 7 ) 9 16 21 27 31 39 41 41 41 48 50 50 50 59</td>
<td>( 7 ) 9 16 21 27 31 39 41 41 41 48 50 50 50 59</td>
</tr>
<tr>
<td>( 8 ) 13 22 22 27 34 40 45 48 50 50 58 58 65 65</td>
<td>( 8 ) 13 22 22 27 34 40 45 48 50 50 58 58 65 65</td>
</tr>
<tr>
<td>( 11 ) 14 29 31 35 36 41 49 55 55 58 59 61 62 67</td>
<td>( 11 ) 14 29 31 35 36 41 49 55 55 58 59 61 62 67</td>
</tr>
<tr>
<td>( 15 ) 20 30 32 39 42 42 50 59 60 60 65 68 71 71</td>
<td>( 15 ) 20 30 32 39 42 42 50 59 60 60 65 68 71 71</td>
</tr>
<tr>
<td>( 16 ) 21 35 41 43 44 45 58 62 69 70 70 70 75 75</td>
<td>( 16 ) 21 35 41 43 44 45 58 62 69 70 70 70 75 75</td>
</tr>
<tr>
<td>( 20 ) 22 36 41 42 50 50 61 65 70 75 75 76 78 78</td>
<td>( 20 ) 22 36 41 42 50 50 61 65 70 75 75 76 78 78</td>
</tr>
<tr>
<td>( 21 ) 25 37 44 44 59 60 62 70 72 75 76 78 81 85</td>
<td>( 21 ) 25 37 44 44 59 60 62 70 72 75 76 78 81 85</td>
</tr>
<tr>
<td>( 22 ) 28 39 48 50 61 62 66 71 75 75 76 81 85 85</td>
<td>( 22 ) 28 39 48 50 61 62 66 71 75 75 76 81 85 85</td>
</tr>
<tr>
<td>( 26 ) 31 40 56 65 65 65 69 75 78 80 82 82 88 95</td>
<td>( 26 ) 31 40 56 65 65 65 69 75 78 80 82 82 88 95</td>
</tr>
<tr>
<td>( 29 ) 34 41 61 66 69 72 72 78 80 80 81 82 84 88</td>
<td>( 29 ) 34 41 61 66 69 72 72 78 80 80 81 82 84 88</td>
</tr>
<tr>
<td>( 31 ) 41 45 66 67 70 72 72 78 82 84 84 85 85 91</td>
<td>( 31 ) 41 45 66 67 70 72 72 78 82 84 84 85 85 91</td>
</tr>
<tr>
<td>( 32 ) 45 49 67 67 72 78 80 81 86 85 86 86 88 95</td>
<td>( 32 ) 45 49 67 67 72 78 80 81 86 85 86 86 88 95</td>
</tr>
<tr>
<td>( 40 ) 55 56 70 71 75 81 81 81 86 86 88 91 93 98</td>
<td>( 40 ) 55 56 70 71 75 81 81 81 86 86 88 91 93 98</td>
</tr>
</tbody>
</table>

**Search for \( x = 35 \):**

- Start at the top row.
- Perform a binary search in the row until \( x = 35 \) is found.
Searching in a Sorted Grid (Algorithm 3)

**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:**

1. Set `left ← 1`.
2. Set `right ← n`.
3. While `left ≤ right` do:
   - Set `mid ← \frac{left+right}{2}`.
   - If `A[i, mid] = x` then
     - Return "item found".
   - Else if `A[i, mid] < x` then
     - Set `left ← mid + 1`.
   - Else
     - Set `right ← mid - 1`.
4. End while.
5. Return "item not found".
### Searching in a Sorted Grid (Algorithm 3)

#### 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 $A$:

1. $\text{left} \leftarrow 1$
2. $\text{right} \leftarrow n$
3. While $\text{left} \leq \text{right}$ do
   - $\text{mid} \leftarrow \frac{\text{left} + \text{right}}{2}$
   - If $A[i, \text{mid}] = x$ then
     - Return "item found"
   - Else if $A[i, \text{mid}] < x$ then
     - $\text{left} \leftarrow \text{mid} + 1$
   - Else $\text{right} \leftarrow \text{mid} - 1$
4. End while
5. Return "item not found"
### Searching in a Sorted Grid (Algorithm 3)

#### 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 \( A \):

1. \( left \leftarrow 1 \)
2. \( right \leftarrow n \)
3. While \( left \leq right \) do
   1. \( mid \leftarrow \frac{left + right}{2} \)
   2. If \( A[i, mid] = x \) then
      - Return "item found"
   3. Else if \( A[i, mid] < x \) then
      - \( left \leftarrow mid + 1 \)
   4. Else
      - \( right \leftarrow mid - 1 \)
4. End while
5. Return "item not found"
### Searching in a Sorted Grid (Algorithm 3)

#### 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 $A$:**

1. Set `left ← 1`
2. Set `right ← n`
3. While `left ≤ right` do
   1. Set `mid ← (left + right) / 2`
   2. If $A[i, mid] = x$ then
      * Return "item found"
   3. Else if $A[i, mid] < x$ then
      * Set `left ← mid + 1`
   4. Else
      * Set `right ← mid - 1`
5. End while

Return "item not found"
### 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 $A$:**

1. $left \leftarrow 1$
2. $right \leftarrow n$
3. while $left \leq right$ do
   1. $mid \leftarrow \frac{left + right}{2}$
   2. if $A[i, mid] = x$ then
      1. return "item found"
   3. else if $A[i, mid] < x$ then
      1. $left \leftarrow mid + 1$
   4. else $right \leftarrow mid - 1$
5. end while
6. return "item not found"
Searching in a Sorted Grid (Algorithm 3):

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

Binary search in row \( i \) of \( A \):

\[
\text{left} \leftarrow 1 \\
\text{right} \leftarrow n \\
\text{while } \text{left} \leq \text{right} \text{ do} \\
\quad \text{mid} \leftarrow \frac{\text{left} + \text{right}}{2} \\
\quad \text{if } A[i, \text{mid}] = x \text{ then} \\
\quad\quad \text{return "item found"} \\
\quad \text{else if } A[i, \text{mid}] < x \text{ then} \\
\quad\quad \quad \text{left} \leftarrow \text{mid} + 1 \\
\quad \text{else right} \leftarrow \text{mid} - 1 \\
\text{end while} \\
\text{return "item not found"}
\]
### Searching in a Sorted Grid (Algorithm 3)

**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 \( A \):**

- \( \text{left} \leftarrow 1 \)
- \( \text{right} \leftarrow n \)
- while \( \text{left} \leq \text{right} \) do
  - \( \text{mid} \leftarrow \frac{\text{left} + \text{right}}{2} \)
  - if \( A[i, \text{mid}] = x \) then
    - return "item found"
  - else if \( A[i, \text{mid}] < x \) then
    - \( \text{left} \leftarrow \text{mid} + 1 \)
  - else \( \text{right} \leftarrow \text{mid} - 1 \)

end while

return "item not found"
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 ← \(\frac{\text{left} + \text{right}}{2}\)
    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)**

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:**

1. Set `left ← 1`
2. Set `right ← n`
3. While `left ≤ right` do
   - Compute `mid ← (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`
4. End while
5. Return "item not found"
Searching in a Sorted Grid (Algorithm 3)

**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 \( A \):**

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

1. \( \text{left} \leftarrow 1 \)
2. \( \text{right} \leftarrow n \)
3. While \( \text{left} \leq \text{right} \) do
   1. \( \mid \text{mid} \mid \leftarrow \frac{\text{left} + \text{right}}{2} \)
   2. If \( A[i, \text{mid}] = x \) then
      1. Return "item found"
   3. Else if \( A[i, \text{mid}] < x \) then
      1. \( \text{left} \leftarrow \text{mid} + 1 \)
   4. Else right \( \leftarrow \text{mid} - 1 \)
5. End while
6. Return "item not found"
### Searching in a Sorted Grid (Algorithm 3)

**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 $A$:**

1. $left \leftarrow 1$
2. $right \leftarrow n$
3. while $left \leq right$ do
   4. $mid \leftarrow \frac{left + right}{2}$
   5. if $A[i, mid] = x$ then
      6. return "item found"
   7. else if $A[i, mid] < x$ then
      8. $left \leftarrow mid + 1$
   9. else $right \leftarrow mid - 1$
10. end while
11. return "item not found"

---

**Table:**

<table>
<thead>
<tr>
<th>$A[1:n, 1:n]$</th>
<th>$n = 2^m - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 5 10 11 20 22 26 30 31 34 37 40 45</td>
<td>$n = 2^m - 1$</td>
</tr>
<tr>
<td>4 6 15 18 27 30 31 38 40 42 42 44 48 48</td>
<td></td>
</tr>
<tr>
<td>7 9 16 21 27 31 39 41 41 41 48 50 55 59</td>
<td></td>
</tr>
<tr>
<td>8 13 22 22 27 34 40 45 48 50 50 50 58 65</td>
<td></td>
</tr>
<tr>
<td>11 14 29 31 35 36 41 49 55 55 58 59 61 67</td>
<td></td>
</tr>
<tr>
<td>15 20 30 32 39 42 42 50 59 60 60 65 68 71</td>
<td></td>
</tr>
<tr>
<td>16 21 35 41 41 43 44 58 62 69 70 70 70 75</td>
<td></td>
</tr>
<tr>
<td>20 22 36 41 42 50 50 61 65 70 75 75 76 78 78</td>
<td></td>
</tr>
<tr>
<td>21 25 37 44 44 59 60 62 70 72 75 76 78 80 80</td>
<td></td>
</tr>
<tr>
<td>22 28 39 48 50 61 62 66 71 75 75 76 78 81 85</td>
<td></td>
</tr>
<tr>
<td>26 31 40 56 65 65 65 69 75 78 78 80 82 82 88</td>
<td></td>
</tr>
<tr>
<td>29 34 41 61 66 69 72 72 78 80 80 81 82 84 88</td>
<td></td>
</tr>
<tr>
<td>31 41 45 66 67 70 72 72 78 82 84 84 85 85 91</td>
<td></td>
</tr>
<tr>
<td>32 45 49 67 67 72 78 80 81 86 85 86 86 88 95</td>
<td></td>
</tr>
<tr>
<td>40 55 56 70 71 75 81 81 81 86 86 88 91 93 98</td>
<td></td>
</tr>
</tbody>
</table>
**Searching in a Sorted Grid (Algorithm 3)**

### 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)

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

$\begin{array}{c}
n = 2^m - 1 \\
\end{array}$

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 $A$:

$\begin{array}{c}
left \leftarrow 1 \\
right \leftarrow n \\
while left \leq right do \\
\quad mid \leftarrow \frac{left + right}{2} \\
\quad if A[i, mid] = x then \\
\quad \quad return "item found" \\
\quad else if A[i, mid] < x then \\
\quad \quad left \leftarrow mid + 1 \\
\quad else right \leftarrow mid - 1 \\
\quad end while \\
\quad return "item not found"
\end{array}$
**Searching in a Sorted Grid (Algorithm 3)**

\[ n = 2^m - 1 \]

<table>
<thead>
<tr>
<th>( A[1:n, 1:n] )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**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 \( A \):**

1. \( left \leftarrow 1 \)
2. \( right \leftarrow n \)
3. 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 \( right \leftarrow mid - 1 \)
4. Return "item not found"
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 $A$:

$\text{left} \leftarrow 1$

$\text{right} \leftarrow n$

while $\text{left} \leq \text{right}$ do

(mid $\leftarrow \frac{\text{left} + \text{right}}{2}$)

if $A[i, \text{mid}] = x$ then

return "item found"

else if $A[i, \text{mid}] < x$ then

$\text{left} \leftarrow \text{mid} + 1$

else $\text{right} \leftarrow \text{mid} - 1$

end while

return "item not found"
**Searching in a Sorted Grid (Algorithm 3)**

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 $A$:**

1. $left \leftarrow 1$
2. $right \leftarrow n$
3. 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 $right \leftarrow mid - 1$
4. End while
5. Return "item not found"
# Searching in a Sorted Grid (Algorithm 3)

\[ n = 2^m - 1 \]

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

Binary search on each row performs \( m \) comparisons.

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

Search for \( x = 35 \)
## Algorithm 4 (Search for $x$):

Start the search for $x$ from the bottom-left corner.
Searching in a Sorted Grid (Algorithm 4)

\[ n = 2^m - 1 \]

```
A[1..n, 1..n]
```

**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
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)

The grid is sorted in increasing order both row-wise and column-wise. The grid size is \(n = 2^m - 1\).

#### 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

#### Example:

Search for \(x = 68\):

The search process would involve comparing the current location's value with 68 and moving accordingly until either 68 is found or the search hits the grid boundaries.

---

**Grid Example**

\[
\begin{array}{cccccccccccccccc}
2 & 5 & 10 & 11 & 20 & 22 & 26 & 30 & 31 & 34 & 37 & 40 & 45 \\
4 & 6 & 15 & 18 & 27 & 30 & 31 & 38 & 39 & 40 & 42 & 44 & 48 & 48 \\
7 & 9 & 16 & 21 & 27 & 31 & 39 & 41 & 41 & 48 & 50 & 55 & 55 & 59 \\
8 & 13 & 22 & 22 & 27 & 34 & 40 & 45 & 48 & 50 & 50 & 58 & 58 & 65 \\
15 & 20 & 30 & 32 & 39 & 42 & 50 & 59 & 60 & 60 & 60 & 65 & 68 & 71 \\
16 & 21 & 35 & 41 & 41 & 43 & 44 & 58 & 62 & 69 & 69 & 70 & 70 & 70 & 75 \\
20 & 22 & 36 & 41 & 42 & 50 & 50 & 61 & 65 & 70 & 75 & 76 & 78 & 78 & 78 \\
21 & 25 & 37 & 44 & 44 & 59 & 60 & 62 & 70 & 72 & 75 & 76 & 78 & 78 & 80 \\
22 & 28 & 39 & 48 & 50 & 50 & 61 & 66 & 71 & 75 & 75 & 76 & 78 & 81 & 85 \\
26 & 31 & 40 & 56 & 65 & 65 & 65 & 69 & 75 & 78 & 78 & 80 & 82 & 82 & 88 \\
29 & 34 & 41 & 61 & 66 & 69 & 72 & 72 & 78 & 80 & 80 & 81 & 82 & 84 & 88 \\
31 & 41 & 45 & 66 & 67 & 70 & 72 & 72 & 78 & 82 & 84 & 84 & 85 & 85 & 91 \\
32 & 45 & 49 & 67 & 67 & 72 & 78 & 80 & 81 & 86 & 85 & 86 & 86 & 88 & 95 \\
40 & 55 & 56 & 70 & 71 & 75 & 81 & 81 & 81 & 86 & 86 & 88 & 91 & 93 & 98 \\
\end{array}
\]
Searching in a Sorted Grid (Algorithm 4)

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)

\[ n = 2^m - 1 \]

**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)

**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

---

<table>
<thead>
<tr>
<th>A[1:n, 1:n]</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>11</th>
<th>20</th>
<th>22</th>
<th>26</th>
<th>30</th>
<th>31</th>
<th>34</th>
<th>37</th>
<th>40</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>15</td>
<td>18</td>
<td>27</td>
<td>30</td>
<td>31</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>42</td>
<td>42</td>
<td>44</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>16</td>
<td>21</td>
<td>27</td>
<td>31</td>
<td>39</td>
<td>41</td>
<td>41</td>
<td>41</td>
<td>48</td>
<td>50</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>22</td>
<td>22</td>
<td>27</td>
<td>34</td>
<td>40</td>
<td>45</td>
<td>48</td>
<td>50</td>
<td>50</td>
<td>58</td>
<td>58</td>
<td>65</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>29</td>
<td>31</td>
<td>35</td>
<td>36</td>
<td>41</td>
<td>49</td>
<td>55</td>
<td>55</td>
<td>58</td>
<td>59</td>
<td>61</td>
<td>62</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>30</td>
<td>32</td>
<td>39</td>
<td>42</td>
<td>42</td>
<td>50</td>
<td>59</td>
<td>60</td>
<td>60</td>
<td>65</td>
<td>68</td>
<td>71</td>
</tr>
<tr>
<td>16</td>
<td>21</td>
<td>35</td>
<td>41</td>
<td>41</td>
<td>43</td>
<td>44</td>
<td>58</td>
<td>62</td>
<td>69</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>75</td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>36</td>
<td>41</td>
<td>42</td>
<td>50</td>
<td>50</td>
<td>61</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>75</td>
<td>76</td>
<td>78</td>
</tr>
<tr>
<td>21</td>
<td>25</td>
<td>37</td>
<td>44</td>
<td>44</td>
<td>59</td>
<td>60</td>
<td>62</td>
<td>70</td>
<td>72</td>
<td>75</td>
<td>76</td>
<td>78</td>
<td>80</td>
</tr>
<tr>
<td>22</td>
<td>28</td>
<td>39</td>
<td>48</td>
<td>50</td>
<td>61</td>
<td>62</td>
<td>66</td>
<td>71</td>
<td>75</td>
<td>75</td>
<td>76</td>
<td>78</td>
<td>81</td>
</tr>
<tr>
<td>26</td>
<td>31</td>
<td>40</td>
<td>56</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>75</td>
<td>78</td>
<td>80</td>
<td>80</td>
<td>82</td>
<td>82</td>
<td>88</td>
</tr>
<tr>
<td>29</td>
<td>34</td>
<td>41</td>
<td>61</td>
<td>66</td>
<td>69</td>
<td>72</td>
<td>72</td>
<td>78</td>
<td>80</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>84</td>
</tr>
<tr>
<td>31</td>
<td>41</td>
<td>45</td>
<td>66</td>
<td>67</td>
<td>70</td>
<td>72</td>
<td>72</td>
<td>78</td>
<td>82</td>
<td>84</td>
<td>84</td>
<td>85</td>
<td>91</td>
</tr>
<tr>
<td>32</td>
<td>45</td>
<td>49</td>
<td>67</td>
<td>67</td>
<td>72</td>
<td>78</td>
<td>80</td>
<td>81</td>
<td>86</td>
<td>86</td>
<td>86</td>
<td>88</td>
<td>95</td>
</tr>
<tr>
<td>40</td>
<td>55</td>
<td>56</td>
<td>70</td>
<td>71</td>
<td>75</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>86</td>
<td>86</td>
<td>88</td>
<td>91</td>
<td>93</td>
</tr>
</tbody>
</table>

**Search for $x = 68$**
### Searching in a Sorted Grid (Algorithm 4)

**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:

1. Let \(y\) be the number at current location.
2. **\(y = x\):** you found the item
3. **\(y < x\):** move to the right
4. **\(y > x\):** move to the cell above

Search for \(x = 68\)

---

### Grid

\[ n = 2^m - 1 \]

<table>
<thead>
<tr>
<th>A[1:n, 1:n]</th>
<th>n = 2^m - 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 5 10 11</td>
<td>20 22 26 30 31 31 34 34 37 40 45</td>
</tr>
<tr>
<td>4 6 15 18</td>
<td>27 30 31 38 39 40 42 42 44 48 48</td>
</tr>
<tr>
<td>7 9 16 21</td>
<td>27 31 39 41 41 48 50 50 50 58 58</td>
</tr>
<tr>
<td>8 13 22 22</td>
<td>27 34 40 45 48 50 50 58 58 65 65</td>
</tr>
<tr>
<td>11 14 29 31</td>
<td>35 36 41 49 55 55 58 59 61 62 67</td>
</tr>
<tr>
<td>15 20 30 32</td>
<td>39 42 42 50 59 60 60 60 65 68 71</td>
</tr>
<tr>
<td>16 21 35 41</td>
<td>41 43 44 58 62 69 69 70 70 70 75</td>
</tr>
<tr>
<td>20 22 36 41</td>
<td>42 50 50 61 65 70 75 75 76 78 78</td>
</tr>
<tr>
<td>21 25 37 44</td>
<td>44 59 60 62 70 72 75 76 78 78 80</td>
</tr>
<tr>
<td>22 28 39 48</td>
<td>50 61 62 66 71 75 75 76 78 81 85</td>
</tr>
<tr>
<td>26 31 40 56</td>
<td>65 65 65 69 75 78 78 80 82 82 88</td>
</tr>
<tr>
<td>29 34 41 61</td>
<td>66 69 72 72 78 80 80 81 82 84 88</td>
</tr>
<tr>
<td>31 41 45 66</td>
<td>67 70 72 72 78 82 84 84 85 85 91</td>
</tr>
<tr>
<td>32 45 49 67</td>
<td>67 72 78 80 81 86 85 86 86 88 93</td>
</tr>
<tr>
<td>40 55 56 70</td>
<td>71 75 81 81 81 86 86 88 91 93 98</td>
</tr>
</tbody>
</table>
## Searching in a Sorted Grid (Algorithm 4)

\[ n = 2^m - 1 \]

### 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 \( x = 68 \)

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

\[ A[1:n, 1:n] \]
### Searching in a Sorted Grid (Algorithm 4)

![Sorted Grid](image)

**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):**

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

**Algorithm 4 (Search for $x$):**

Search for $x = 68$
**Searching in a Sorted Grid (Algorithm 4)**

\[ n = 2^m - 1 \]

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

**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)**

![Matrix grid](image)

### 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 $x = 68$
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)**

$\text{Search for } x = 68$


### 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:

1. 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)

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

$n = 2^m - 1$

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

**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)

#### 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):  

![Sorted Grid Diagram](image)

\[ n = 2^m - 1 \]

**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)

**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 $x = 68$**

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

$n = 2^m - 1$
Searching in a Sorted Grid (Algorithm 4):

Let $n = 2^m - 1$.

**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:

1. Let $y$ be the number at current location.
2. If $y = x$: you found the item
3. If $y < x$: move to the right
4. If $y > x$: move to the cell above

Search for $x = 68$
### Searching in a Sorted Grid (Algorithm 4)

Let $n = 2^m - 1$. The grid is of size $A[1:n, 1:n]$. We search for $x$ from the bottom-left corner. Keep performing the steps below until either you find $x$ or you fall off the grid:

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

**Algorithm 4 (Search for $x$):**

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

Let $y$ be the number at the current location.

<table>
<thead>
<tr>
<th>A[1:n, 1:n]</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>11</th>
<th>20</th>
<th>22</th>
<th>26</th>
<th>30</th>
<th>31</th>
<th>33</th>
<th>34</th>
<th>37</th>
<th>40</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>15</td>
<td>18</td>
<td>27</td>
<td>30</td>
<td>31</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>42</td>
<td>42</td>
<td>44</td>
<td>48</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>16</td>
<td>21</td>
<td>27</td>
<td>31</td>
<td>39</td>
<td>41</td>
<td>41</td>
<td>41</td>
<td>48</td>
<td>50</td>
<td>55</td>
<td>55</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>22</td>
<td>22</td>
<td>27</td>
<td>34</td>
<td>40</td>
<td>45</td>
<td>48</td>
<td>50</td>
<td>50</td>
<td>58</td>
<td>58</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>29</td>
<td>31</td>
<td>35</td>
<td>36</td>
<td>41</td>
<td>49</td>
<td>55</td>
<td>55</td>
<td>58</td>
<td>59</td>
<td>61</td>
<td>62</td>
<td>67</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>30</td>
<td>32</td>
<td>39</td>
<td>42</td>
<td>42</td>
<td>50</td>
<td>59</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>65</td>
<td>68</td>
<td>71</td>
</tr>
<tr>
<td>16</td>
<td>21</td>
<td>35</td>
<td>41</td>
<td>41</td>
<td>43</td>
<td>44</td>
<td>58</td>
<td>62</td>
<td>69</td>
<td>69</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>75</td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>36</td>
<td>41</td>
<td>42</td>
<td>50</td>
<td>50</td>
<td>61</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>75</td>
<td>76</td>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>21</td>
<td>25</td>
<td>37</td>
<td>44</td>
<td>44</td>
<td>59</td>
<td>60</td>
<td>62</td>
<td>70</td>
<td>72</td>
<td>75</td>
<td>76</td>
<td>78</td>
<td>78</td>
<td>80</td>
</tr>
<tr>
<td>22</td>
<td>28</td>
<td>39</td>
<td>48</td>
<td>50</td>
<td>61</td>
<td>62</td>
<td>66</td>
<td>71</td>
<td>75</td>
<td>75</td>
<td>76</td>
<td>78</td>
<td>81</td>
<td>85</td>
</tr>
<tr>
<td>26</td>
<td>31</td>
<td>40</td>
<td>56</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>69</td>
<td>75</td>
<td>78</td>
<td>80</td>
<td>82</td>
<td>82</td>
<td>82</td>
<td>88</td>
</tr>
<tr>
<td>29</td>
<td>34</td>
<td>41</td>
<td>61</td>
<td>66</td>
<td>69</td>
<td>72</td>
<td>72</td>
<td>78</td>
<td>80</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>84</td>
<td>88</td>
</tr>
<tr>
<td>31</td>
<td>41</td>
<td>45</td>
<td>66</td>
<td>67</td>
<td>70</td>
<td>72</td>
<td>72</td>
<td>78</td>
<td>82</td>
<td>84</td>
<td>84</td>
<td>85</td>
<td>85</td>
<td>91</td>
</tr>
<tr>
<td>32</td>
<td>45</td>
<td>49</td>
<td>67</td>
<td>67</td>
<td>72</td>
<td>78</td>
<td>80</td>
<td>81</td>
<td>86</td>
<td>85</td>
<td>86</td>
<td>86</td>
<td>88</td>
<td>95</td>
</tr>
<tr>
<td>40</td>
<td>55</td>
<td>56</td>
<td>70</td>
<td>71</td>
<td>75</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>86</td>
<td>86</td>
<td>88</td>
<td>91</td>
<td>93</td>
<td>98</td>
</tr>
</tbody>
</table>

Search for $x = 68$
### Searching in a Sorted Grid (Algorithm 4)

#### 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:

1. Let $y$ be the number at current location.
2. If $y = x$: you found the item
3. If $y < x$: move to the right
4. If $y > x$: move to the cell above

---

**Search for $x = 68$**

<table>
<thead>
<tr>
<th>A[1:n, 1:n]</th>
<th>$n = 2^m - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 5 10 11 20 22 26 30 31 31 34 34 37 40 45</td>
<td></td>
</tr>
<tr>
<td>4 6 15 18 27 30 31 38 39 40 42 42 44 48 48</td>
<td></td>
</tr>
<tr>
<td>7 9 16 21 27 31 39 41 41 41 48 50 55 55 59</td>
<td></td>
</tr>
<tr>
<td>8 13 22 22 27 34 40 45 48 50 50 50 58 58 65</td>
<td></td>
</tr>
<tr>
<td>11 14 29 31 35 36 41 49 55 55 58 59 61 62 67</td>
<td></td>
</tr>
<tr>
<td>15 20 30 32 39 42 42 50 59 60 60 60 65 68 71</td>
<td></td>
</tr>
<tr>
<td>16 21 35 41 41 43 44 58 62 69 69 70 70 70 75</td>
<td></td>
</tr>
<tr>
<td>20 22 36 41 42 50 50 61 65 70 75 75 76 78 78</td>
<td></td>
</tr>
<tr>
<td>21 25 37 44 44 59 60 62 70 72 75 76 78 78 80</td>
<td></td>
</tr>
<tr>
<td>22 28 39 48 50 61 62 66 71 75 75 76 78 81 85</td>
<td></td>
</tr>
<tr>
<td>26 31 40 56 65 65 65 69 75 78 78 80 82 82 88</td>
<td></td>
</tr>
<tr>
<td>29 34 41 61 66 69 72 72 78 80 80 81 82 84 88</td>
<td></td>
</tr>
<tr>
<td>31 41 45 66 67 70 72 72 78 82 84 84 85 85 91</td>
<td></td>
</tr>
<tr>
<td>32 45 49 67 67 72 78 80 81 86 85 86 86 88 95</td>
<td></td>
</tr>
<tr>
<td>40 55 56 70 71 75 81 81 81 86 86 88 91 93 98</td>
<td></td>
</tr>
</tbody>
</table>
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):**

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):

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)

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

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

Search for $x = 68$
Comparing the Four (4) Grid Search Algorithms

\[ \geq Q_1(n) \]
\[ \geq Q_2(n) \]
\[ \geq Q_3(n) \]
\[ \geq Q_4(n) \]
Searching in a Sorted Grid (Algorithm 1)

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

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.

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"
Searching in a Sorted Grid (Algorithm 1):

<table>
<thead>
<tr>
<th></th>
<th>A[1,n,1:n]</th>
<th>n = 2^m − 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>29</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>16</td>
<td>21</td>
<td>35</td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>36</td>
</tr>
<tr>
<td>21</td>
<td>25</td>
<td>37</td>
</tr>
<tr>
<td>22</td>
<td>28</td>
<td>39</td>
</tr>
<tr>
<td>26</td>
<td>31</td>
<td>40</td>
</tr>
<tr>
<td>29</td>
<td>34</td>
<td>41</td>
</tr>
<tr>
<td>31</td>
<td>41</td>
<td>45</td>
</tr>
<tr>
<td>32</td>
<td>45</td>
<td>49</td>
</tr>
<tr>
<td>40</td>
<td>55</td>
<td>56</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Grid Searching Algorithm</th>
<th>Worst-Case Bound on #Comparisons</th>
<th>Worst-Case Bound on Running Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1</td>
<td>$Q_1(n) \leq n^2$</td>
<td>$T_1(n) \leq c_1 n^2$</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>$Q_2(n) \leq 1.5(n + 1)^{1.6}$</td>
<td>$T_2(n) \leq c_2(n + 1)^{1.6}$</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>$Q_3(n) \leq n \log_2(n + 1)$</td>
<td>$T_3(n) \leq c_3 n \log_2(n + 1)$</td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>$Q_4(n) \leq 2n$</td>
<td>$T_4(n) \leq c_4 n$</td>
</tr>
</tbody>
</table>

$c_1, c_2, c_3$ 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 $\frac{1 \times (100000)^2}{100000000000} = 10$ seconds,

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

Source: Hennessey & Patterson [2018]
Why Faster Algorithms?

48 Years of Microprocessor Trend Data


Source: https://semiwiki.com/ip/risc-v/312695-white-paper-scaling-is-falling/
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 ( $\Omega$-notation )
- strict lower bound ( $\omega$-notation )
- tight bound ( $\Theta$-notation )
Asymptotic Upper Bound (O-notation)

\[ O(g(n)) = \left\{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that} \right\] 
\[ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \]

\[ \lim_{n \to \infty} \left( \frac{f(n)}{g(n)} \right) \neq \infty \Rightarrow f(n) = O(g(n)) \]
Recall that for Algorithm 1 we had, \( T_1(n) \leq f(n) \), where,
\[
f(n) = a_1 n^2 + a_2 n + a_3,
\]
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 \( f(n) \).
Let $g(n) = n^2$.

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

$$0 \leq f(n) \leq cg(n) \text{ for } c = 6 \text{ and } n \geq 5.$$
Let $g(n) = n^3$.

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

$$0 \leq f(n) \leq cg(n) \text{ for } c = 1 \text{ and } n \geq 6.$$
Asymptotic Upper Bound (O-notation)

Let $g(n) = n$.

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

$$f(n) > cg(n) \text{ for any } c \text{ and } n \geq \frac{c}{5}.$$
Asymptotic Lower Bound ($\Omega$-notation)

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

$$\lim_{n \to \infty} \left( \frac{f(n)}{g(n)} \right) \neq 0 \Rightarrow f(n) = \Omega(g(n))$$
Asymptotic Lower Bound (\( \Omega \)-notation)

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

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

\[
0 \leq cg(n) \leq f(n) \text{ for } c = 5 \text{ and } n \geq 1.
\]
Let \( g(n) = n^3 \).

Then \( f(n) \neq \Omega(n^3) \) because:

\[ cg(n) > f(n) \text{ for any } c \text{ and } n \geq \frac{5}{c}.\]
Asymptotic Lower Bound (Ω-notation)

Let \( g(n) = n \).

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

\[
0 \leq cg(n) \leq f(n) \quad \text{for } c = 30 \text{ and } n \geq 6.
\]
Asymptotic Tight Bound (Θ-notation)

\[ \Theta(g(n)) = \left\{ f(n) : \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that} \right. \\
\left. 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \right\} \\

\left( \lim_{n \to \infty} \left( \frac{f(n)}{g(n)} \right) \neq \infty \right) \wedge \left( \lim_{n \to \infty} \left( \frac{f(n)}{g(n)} \right) \neq 0 \right) \Rightarrow f(n) = \Theta(g(n)) \]
Asymptotic Tight Bound (Θ-notation)

\[
(f(n) = O(g(n))) \land (f(n) = \Omega(g(n))) \iff (f(n) = \Theta(g(n)))
\]
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.

\[ f(n) \neq \Theta(n^3) \] because though \( f(n) = O(n^3) \) holds, \( f(n) \neq \Omega(n^3) \).

\[ f(n) \neq \Theta(n) \] because though \( f(n) = \Omega(n) \) holds, \( f(n) \neq O(n) \).
Asymptotic Strict Upper Bound (o-notation)

\[ O(g(n)) = \begin{cases} f(n): \text{there exist positive constants } c \text{ and } n_0 \text{ such that} \\ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \end{cases} \]

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

\[ \lim_{n \to \infty} \left(\frac{f(n)}{g(n)}\right) = 0 \Rightarrow f(n) = o(g(n)) \]
**Asymptotic Strict Lower Bound (ω-notation)**

\[
\Omega(g(n)) = \left\{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \right\}
\]

\[
\Omega(g(n)) = \left\{ f(n) : \text{there exists a positive constant } c \text{ such that } \lim_{n \to \infty} \left( \frac{f(n)}{g(n)} \right) \geq c \right\}
\]

\[
\lim_{n \to \infty} \left( \frac{f(n)}{g(n)} \right) = \infty \Rightarrow f(n) = \omega(g(n))
\]
Comparing Functions: Transitivity

\[ f(n) = O(g(n)) \quad \text{and} \quad g(n) = O(h(n)) \quad \Rightarrow \quad f(n) = O(h(n)) \]

\[ f(n) = \Omega(g(n)) \quad \text{and} \quad g(n) = \Omega(h(n)) \quad \Rightarrow \quad f(n) = \Omega(h(n)) \]

\[ f(n) = \Theta(g(n)) \quad \text{and} \quad g(n) = \Theta(h(n)) \quad \Rightarrow \quad f(n) = \Theta(h(n)) \]

\[ f(n) = o(g(n)) \quad \text{and} \quad g(n) = o(h(n)) \quad \Rightarrow \quad f(n) = o(h(n)) \]

\[ f(n) = \omega(g(n)) \quad \text{and} \quad g(n) = \omega(h(n)) \quad \Rightarrow \quad f(n) = \omega(h(n)) \]
Comparing Functions: Reflexivity

\[ f(n) = \Theta(f(n)) \]

\[ f(n) = \Omega(f(n)) \]

\[ f(n) = \Theta(f(n)) \]
Comparing Functions: Symmetry

\[ f(n) = \Theta(g(n)) \quad \text{if and only if} \quad g(n) = \Theta(f(n)) \]
Comparing Functions: Transpose Symmetry

\[ f(n) = O(g(n)) \quad \text{if and only if} \quad g(n) = \Omega(f(n)) \]

\[ f(n) = \Omega(g(n)) \quad \text{if and only if} \quad g(n) = O(f(n)) \]
Adding Functions

\[ O(f(n)) + O(g(n)) = O(f(n) + g(n)) \]
\[ \Omega(f(n)) + \Omega(g(n)) = \Omega(f(n) + g(n)) \]
\[ \Theta(f(n)) + \Theta(g(n)) = \Theta(f(n) + g(n)) \]
Multiplying Functions by Constants

\[ O(cf(n)) = O(f(n)) \]

\[ \Omega(cf(n)) = \Omega(f(n)) \]

\[ \Theta(cf(n)) = \Theta(f(n)) \]
Multiplying Two Functions

\[ O(f(n)) \times O(g(n)) = O(f(n) \times g(n)) \]

\[ \Omega(f(n)) \times \Omega(g(n)) = \Omega(f(n) \times g(n)) \]

\[ \Theta(f(n)) \times \Theta(g(n)) = \Theta(f(n) \times g(n)) \]
Division of Functions

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

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

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

\[ \frac{O(f(n))}{\Omega(g(n))} = O \left( \frac{f(n)}{g(n)} \right) \]

\[ \frac{\Omega(f(n))}{0(g(n))} = \Omega \left( \frac{f(n)}{g(n)} \right) \]

\[ \frac{\Theta(f(n))}{0(g(n))} = \Omega \left( \frac{f(n)}{g(n)} \right) \]
Growth Rates of Common Functions

The following table shows how much time an algorithm that performs $f(n)$ operations on an input of size $n$ takes assuming each operation takes one nanosecond ($10^{-9}$ seconds) to execute.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$f(n)$</th>
<th>$\lg n$</th>
<th>$n$</th>
<th>$n \lg n$</th>
<th>$n^2$</th>
<th>$2^n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td>0.003 µs</td>
<td>0.01 µs</td>
<td>0.033 µs</td>
<td>0.1 µs</td>
<td>1 µs</td>
<td>3.63 ms</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>0.004 µs</td>
<td>0.02 µs</td>
<td>0.086 µs</td>
<td>0.4 µs</td>
<td>1 ms</td>
<td>77.1 years</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>0.005 µs</td>
<td>0.03 µs</td>
<td>0.147 µs</td>
<td>0.9 µs</td>
<td>1 sec</td>
<td>$8.4 \times 10^{15}$ yrs</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>0.005 µs</td>
<td>0.04 µs</td>
<td>0.213 µs</td>
<td>1.6 µs</td>
<td>18.3 min</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>0.006 µs</td>
<td>0.05 µs</td>
<td>0.282 µs</td>
<td>2.5 µs</td>
<td>13 days</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>0.007 µs</td>
<td>0.1 µs</td>
<td>0.644 µs</td>
<td>10 µs</td>
<td>1 ms</td>
<td>$4 \times 10^{13}$ yrs</td>
</tr>
<tr>
<td>1,000</td>
<td></td>
<td>0.010 µs</td>
<td>1.00 µs</td>
<td>9.966 µs</td>
<td>1 ms</td>
<td>100 ms</td>
<td></td>
</tr>
<tr>
<td>10,000</td>
<td></td>
<td>0.013 µs</td>
<td>10 µs</td>
<td>130 µs</td>
<td>10 sec</td>
<td>16.7 min</td>
<td></td>
</tr>
<tr>
<td>100,000</td>
<td></td>
<td>0.017 µs</td>
<td>0.10 ms</td>
<td>1.67 ms</td>
<td>19.93 ms</td>
<td>1.16 days</td>
<td></td>
</tr>
<tr>
<td>1,000,000</td>
<td></td>
<td>0.020 µs</td>
<td>1 ms</td>
<td>2.66 sec</td>
<td>115.7 days</td>
<td>31.7 years</td>
<td></td>
</tr>
<tr>
<td>10,000,000</td>
<td></td>
<td>0.023 µs</td>
<td>0.01 sec</td>
<td>0.23 sec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100,000,000</td>
<td></td>
<td>0.027 µs</td>
<td>0.10 sec</td>
<td>29.90 sec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000,000,000</td>
<td></td>
<td>0.030 µs</td>
<td>1 sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>