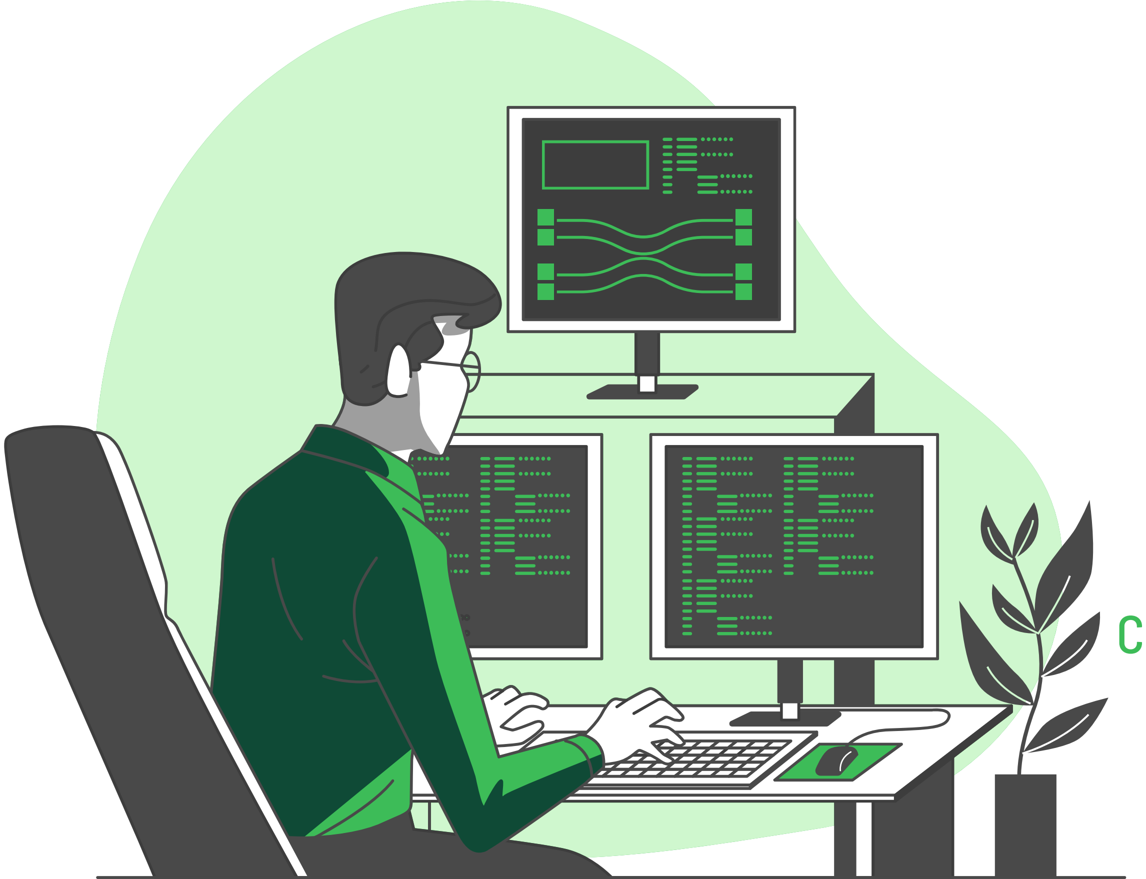
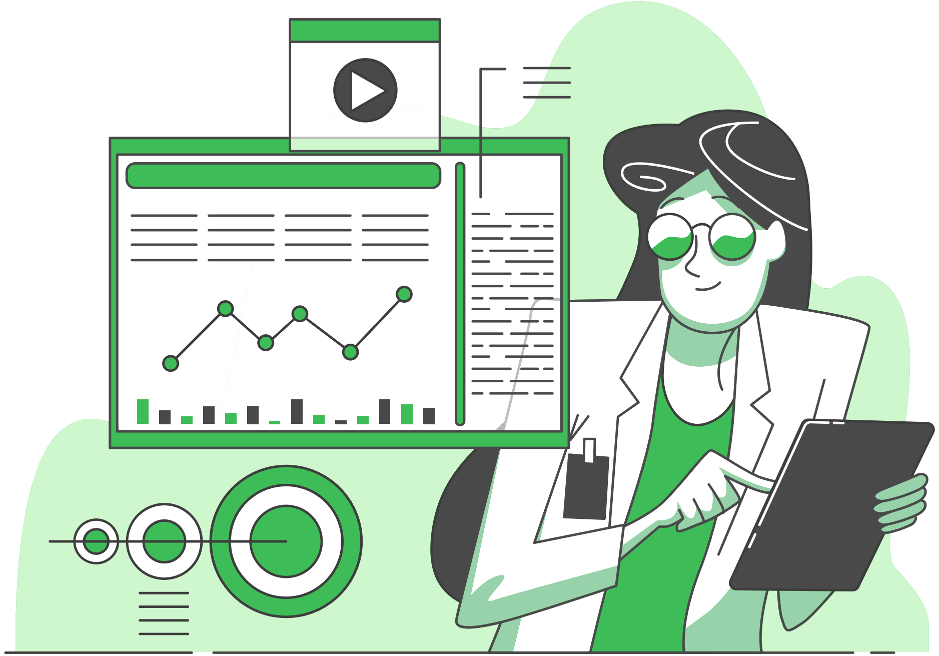


Array

CS 251 - Data Structures and Algorithms

Note:
Slides complement the discussion in class

Table of Contents

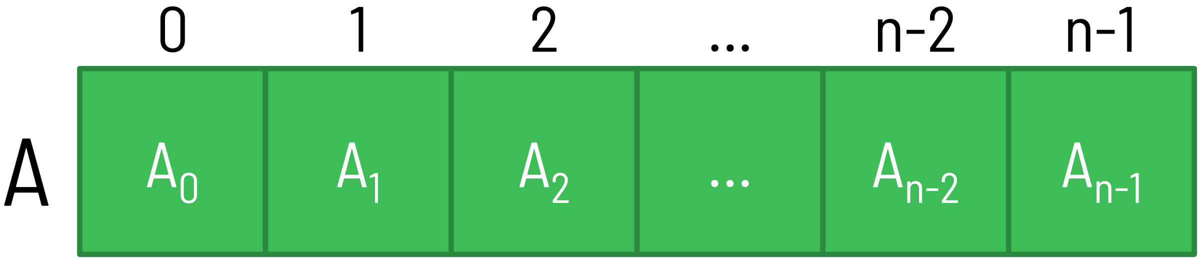


01

Array

Static linear data structure

Array



	0	1	2	…	n-2	n-1
A	A_0	A_1	A_2	…	A_{n-2}	A_{n-1}

An array stores an item per index. An item could be as simple as a single value, or as complex as another data structure. Each item is accessed by its index in the array.

The size of an array is the number of items it stores. The capacity of an array is the amount of space reserved to store items. For example, an array could be of capacity 10 and size 3 (i.e., there are 3 items in it).

Insertion at the End

```
algorithm InsertEnd(A:array, x:item) → array
   let n be the size of A
   let m be the capacity of A
   if n = m then
      A ← Resize(A, 2 * m)
   end if
   A[n] ← x
   increase the size of A by 1
   return A
end algorithm


algorithm Resize(A:array, m:$\mathbb{Z}^+$) → array
   let B be an array of capacity m
   let n be the size of A
   for i from 0 to n-1 do
      B[i] ← A[i]
   end for
   set the size of B to n
   return B
end algorithm
```

Remember, size is the number of items, while capacity is the amount of space reserved for storing the items.

Insertion at Some Index

```
algorithm InsertAt(A:array, x:item, i:$\mathbb{Z}_{\geq 0}$) → array
   let n be the size of A
   let m be the capacity of A
   if n = m then
      A ← Resize(A, 2 * m)
   end if
   if i < n then
      A ← RightShift(A, i)
   end if
   A[i] ← x
   increase the size of A by 1
   return A
end algorithm


algorithm RightShift(A:array, i:$\mathbb{Z}_{\geq 0}$) → array
   let n be the size of A
   for j from n to i+1 by -1 do
      A[j] ← A[j-1]
   end for
   return A
end algorithm
```

Remember, size is the number of items, while capacity is the amount of space reserved for storing the items.

Linear Search

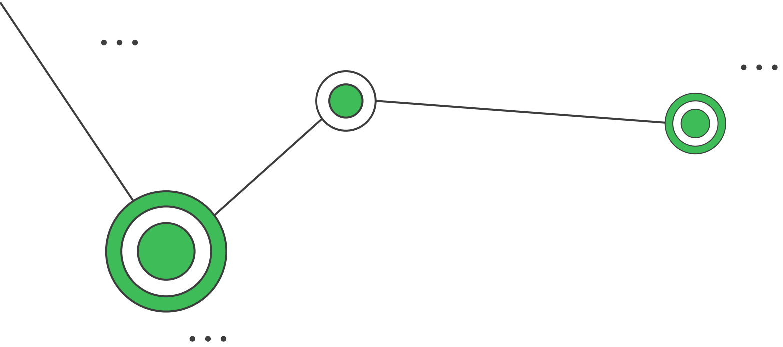
```
algorithm LinearSearch(A:array, x:item) → ℤ
   let n be the size of A
   for i from 0 to n-1 do
      if A[i] = x then
         return i
      end if
   return -1
end algorithm
```

Recursive Linear Search

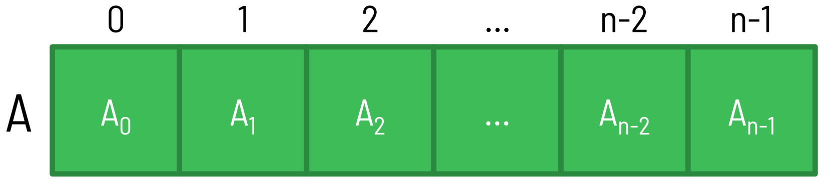
```
algorithm RLinearSearch(A:array, x:item, i:ℤ) → ℤ
   if i < 0 then
      return -1
   end if
   if A[i] = x then
      return i
   end if
   return RLinearSearch(A, x, i-1)
end algorithm
```

First call:

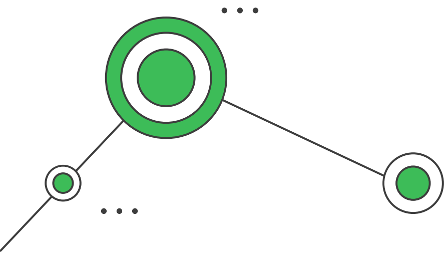
```
let n be the size of A
index ← RLinearSearch(A, x, n-1)
```



Array



	0	1	2	…	n-2	n-1
A	A_0	A_1	A_2	…	A_{n-2}	A_{n-1}



Static Data Structure (i.e., fixed capacity)

Array access: $A[i] \in \Theta(1)$

Q: How many operations are required to **insert** an item into an unsorted array?
Keep track of next available cell?: $\Theta(1)$
Sorry, no tracking: $\Theta(n)$

Q: How much **space** is required to keep track of the next available cell?
One variable: $\Theta(1)$

Q: How many operations are required to **insert** an item into a sorted array?
Find location + Right shifting: $\Theta(n)$

Resize an Array

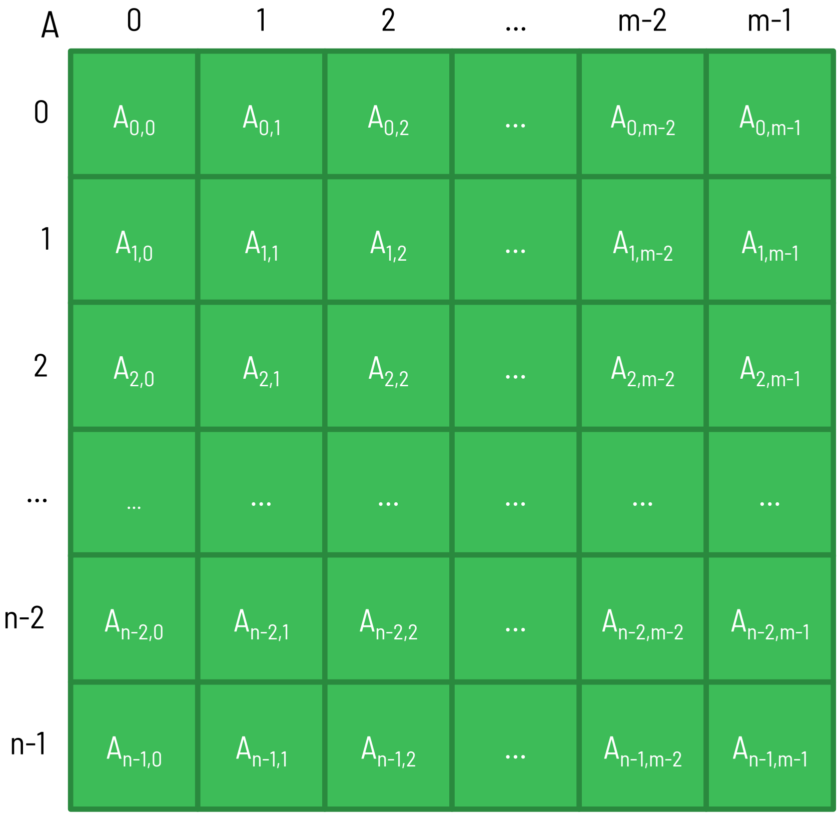
```
algorithm resize(A:array, m:$\mathbb{Z}^+$) → array
  let n be the size of A
  let B be an array of capacity m
  copy/move the elements from A to B
  return B or let A point to B?
  perhaps delete A?
end algorithm
```

Time? $T(n) \approx n \in \Theta(n)$
Space? $T(n) \approx n \in \Theta(n)$

Usual strategy:

Full? Increase its size (double the current size)
Half empty? reduce its size (half the current size)

2D Array (AKA Grid or Matrix)



A	0	1	2	…	m-2	m-1
0	$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	…	$A_{0,m-2}$	$A_{0,m-1}$
1	$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	…	$A_{1,m-2}$	$A_{1,m-1}$
2	$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	…	$A_{2,m-2}$	$A_{2,m-1}$
…	…	…	…	…	…	…
n-2	$A_{n-2,0}$	$A_{n-2,1}$	$A_{n-2,2}$	…	$A_{n-2,m-2}$	$A_{n-2,m-1}$
n-1	$A_{n-1,0}$	$A_{n-1,1}$	$A_{n-1,2}$	…	$A_{n-1,m-2}$	$A_{n-1,m-1}$

Static Data Structure (i.e., fixed size)

Array access: $A[i][j] \in \Theta(1)$

Q: How many operations are required to traverse a grid?
General case: $\Theta(nm)$
Squared matrix: $\Theta(n^2)$

Q: How much space is required to store a grid?
General case: $\Theta(nm)$
Squared matrix: $\Theta(n^2)$

EOF

Do you have any questions?

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik and illustrations by Stories