Lists and Arrays

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 3

Sequences

- Two basic ways of storing a sequence of values
  - Lists
  - Arrays
- What's the difference?

Sequences

- Two basic ways of storing a sequence of values
  - Lists
  - Arrays
- What's the difference?

- Lists
  - Flexible length
  - Easy to modify the structure
  - Values are scattered in memory

Sequences

- Two basic ways of storing a sequence of values
  - Lists
  - Arrays
- What's the difference?

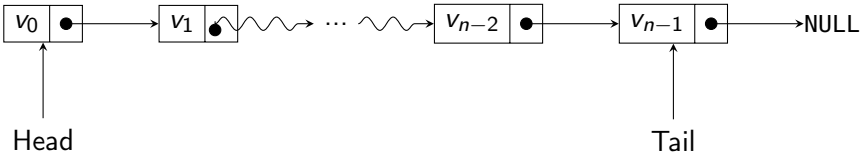
- Lists
  - Flexible length
  - Easy to modify the structure
  - Values are scattered in memory
- Arrays
  - Fixed size
  - Allocate a contiguous block of memory
  - Supports random access

Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list

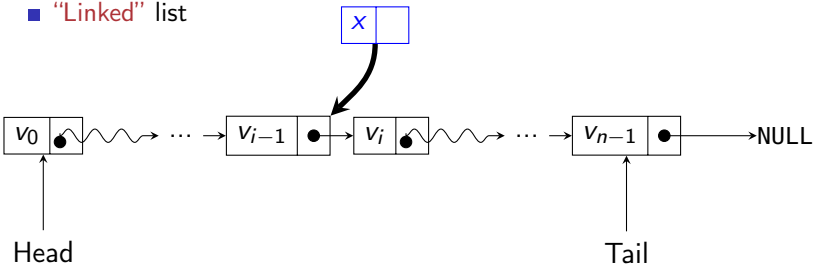
Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list



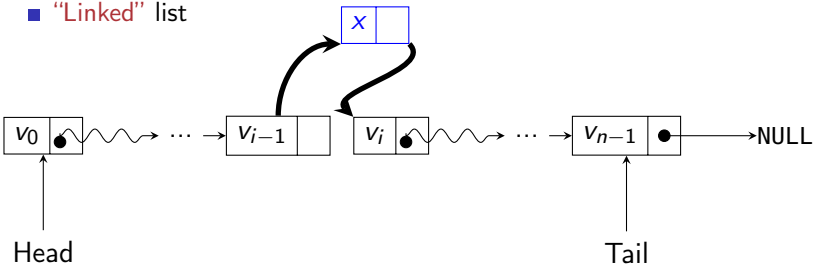
Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list
- Easy to modify
  - Inserting and deletion is easy via local "plumbing"
  - Flexible size



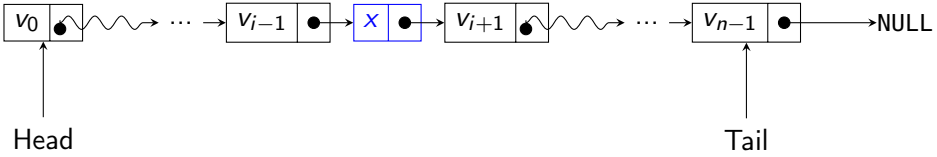
Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list
- Easy to modify
  - Inserting and deletion is easy via local "plumbing"
  - Flexible size



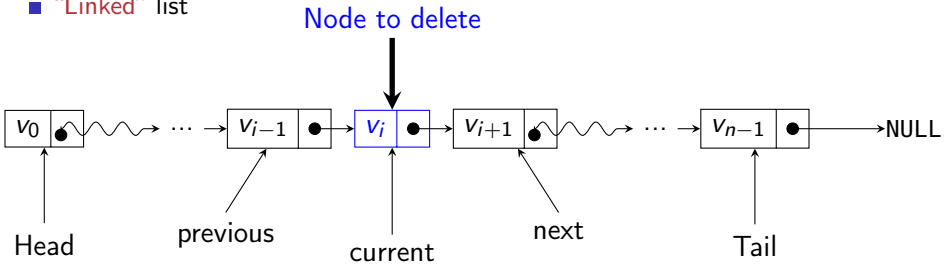
Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list
- Easy to modify
  - Inserting and deletion is easy via local "plumbing"
  - Flexible size



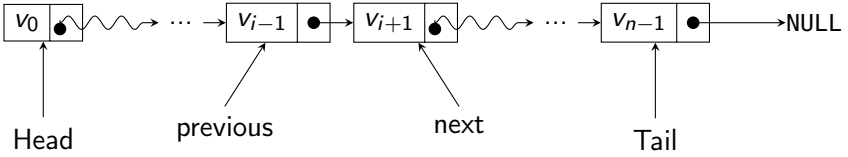
Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list
- Easy to modify
  - Inserting and deletion is easy via local "plumbing"
  - Flexible size



Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list
- Easy to modify
  - Inserting and deletion is easy via local "plumbing"
  - Flexible size



Lists

- Typically a sequence of nodes
- Each node contains a value and points to the next node in the sequence
  - "Linked" list
- Easy to modify
  - Inserting and deletion is easy via local "plumbing"
  - Flexible size
- Need to follow links to access `A[i]`
  - Takes time $O(i)$

Arrays

- Fixed size, declared in advance
- Allocate a contiguous block of memory
  - n times the storage for a single value

Index	Value
`A[0]`	v_0
`A[1]`	v_1
⋮	⋮
`A[i]`	v_i
⋮	⋮
`A[n-1]`	v_{n-1}

Arrays

- Fixed size, declared in advance
- Allocate a contiguous block of memory
  - n times the storage for a single value
- "Random" access
  - Compute offset to `A[i]` from `A[0]`
  - Accessing `A[i]` takes constant time, independent of `i`

Index	Value
`A[0]`	v_0
`A[1]`	v_1
⋮	⋮
`A[i]`	v_i
⋮	⋮
`A[n-1]`	v_{n-1}

Arrays

- Fixed size, declared in advance
- Allocate a contiguous block of memory
  - n times the storage for a single value
- "Random" access
  - Compute offset to `A[i]` from `A[0]`
  - Accessing `A[i]` takes constant time, independent of `i`
- Inserting and deleting elements is expensive
  - Expanding and contracting requires moving $O(n)$ elements in the worst case

Index	Value
`A[0]`	v_0
`A[1]`	v_1
⋮	⋮
`A[i]`	v_i
⋮	⋮
`A[n-1]`	v_{n-1}

Operations

- Exchange `A[i]` and `A[j]`
  - Constant time for arrays
  - $O(n)$ for lists
- Delete `A[i]`, insert `v` after `A[i]`
  - Constant time for lists if we are already at `A[i]`
  - $O(n)$ for arrays
- Need to keep implementation in mind when analyzing data structures
  - For instance, can we use binary search to insert in a sorted sequence?
  - Either search is slow, or insertion is slow, still $O(n)$

Summary

- Sequences can be stored as lists or arrays

Summary

- Sequences can be stored as lists or arrays
- Lists are flexible but accessing an element is $O(n)$

Summary

- Sequences can be stored as lists or arrays
- Lists are flexible but accessing an element is $O(n)$
- Arrays support random access but are difficult to expand, contract

Summary

- Sequences can be stored as lists or arrays
- Lists are flexible but accessing an element is $O(n)$
- Arrays support random access but are difficult to expand, contract
- Algorithm analysis needs to take into account the underlying implementation

Summary

- Sequences can be stored as lists or arrays
- Lists are flexible but accessing an element is $O(n)$
- Arrays support random access but are difficult to expand, contract
- Algorithm analysis needs to take into account the underlying implementation
- How does it work in Python?
  - Is the built-in list type in Python really a "linked" list?
  - Numpy library provides arrays — are these faster than lists?