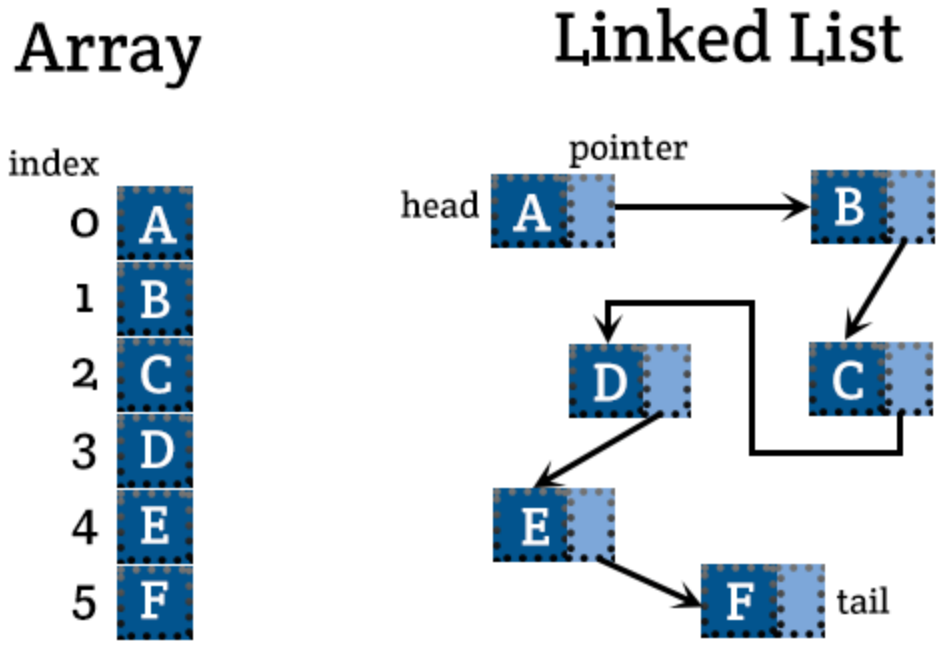
UNIVERSITY OF CALGARY

CPSC 319
Data Structures, Algorithms, and Their Applications

Winter 2024

Array vs LinkedList



Array vs LinkedList

- Arrays:
  - Fixed-size data structure.
  - Contiguous memory allocation.
  - Efficient random access.
- Linked Lists:
  - Dynamic-size data structure.
  - Non-contiguous memory allocation.
  - Efficient insertions and deletions.

Memory Allocation

- Arrays:
  - Contiguous memory block.
  - Pre-allocated size.

- Linked Lists:
  - Non-contiguous nodes with pointers.
  - Dynamic memory allocation.

Access Time

- Arrays:
  - Constant time access (O(1)).
  - Direct index-based access.

- Linked Lists:
  - Linear time access (O(n)).
  - Sequential traversal required.

Insertions and Deletions

- Arrays:
  - Costly for insertions and deletions (O(n)).
  - Shifting elements may be required.

- Linked Lists:
  - Efficient for insertions and deletions (O(1) if node is known).
  - No need to shift elements.

Use Cases

- Arrays:
  - When size is fixed and known in advance.
  - Random access is crucial.

- Linked Lists:
  - Frequent insertions and deletions.
  - Dynamic size is required.

Trade-Offs

- Arrays:
  - Fast access, limited flexibility.
  - Memory may be wasted.

- Linked Lists:
  - Flexible size, slower access.
  - Efficient memory utilization.

Sample Question 2:

- Write a Java code that reads a series of positive numbers from the "input.txt" file, each line contains one number. Group these numbers according to their smallest prime factor and then record each group on a separate line in the "output.txt" file, each line should start with the smallest common prime factor. It is guaranteed that the biggest smallest common prime factor of those numbers is less than 1000.

- Test case:

  Input:
  5, 15, 2, 3, 4, 9, 10 (each number is in a separate line)

  Output:
  2 4 10
  15 3 9
  5