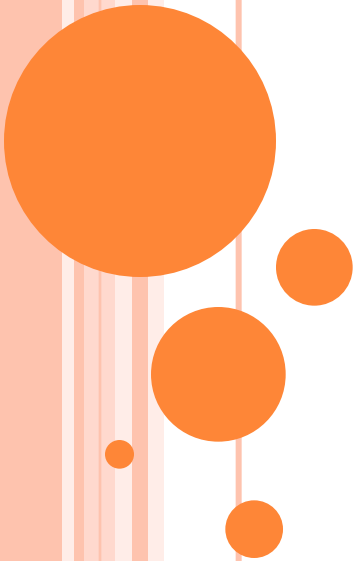


Uses of Linked Lists

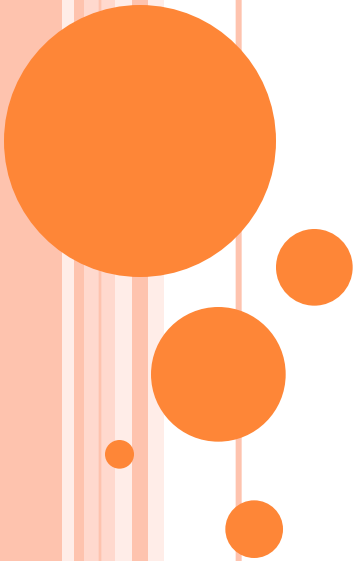
Omieno K.K

Advantages of linked lists

- No limit to number of items (up to memory limits)
- Insert and remove operations are very efficient
  - There is no shuffling of the remaining list items
- Storage required is proportional to the number of elements currently in the list
  - An empty list is very small
- Access via pointer is faster than accessing an array component
  - No computation

Disadvantages of linked lists

- SetPos, prev, and length are very inefficient
  - length could be easily made efficient by adding one data member
  - SetPos is inherently inefficient since a linked list is not a random access structure
- Linked lists are designed to be processed in forward order
  - prev cannot be improved in a simple linked list
- Each Node contains a pointer which requires additional storage



Uses of Stack

- Temporary storage
- Function calls and recursion
- Reversing order
- Verifying legal nesting
- Evaluating postfix expressions
- Parsing infix expressions
- Depth first search

QUEUES

- The most well known operation of the queue is the First-In-First-Out (FIFO) queue process — In a FIFO queue.
- The first element in the queue will be the first one out; this is equivalent to the requirement that whenever an element is added, all elements that were added before have to be removed before the new element can be invoked

Queues operations

- There are two basic operations associated with a queue: enqueue and dequeue.
- Enqueue means adding a new item to the rear of the queue while;
- dequeue refers to removing the front item from queue and returns it in item

Queues

- enqueue
  - insert item at back of queue
- dequeue
  - remove item from front of queue
- front
- is_empty
- is_full

- Array and Linked implementations are typical
- These share the same advantages and disadvantages as the stack implementations

Queue Applications

- Temporary storage
- Print queues, execution queues
- Simulations
- Buffering
- Breadth first search