CZ1106 Problem Solving and Computation II

Introduction to Data Structures

Linked List, Stack and Queue

Dr Tay Seng Chuan

Faculty of Science

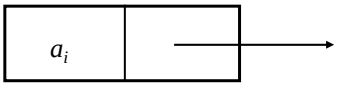
Data Structures

Data structure is an organizational scheme, such as a record, array, or pointer that can be applied to data to facilitate interpreting the data or performing operations on it.

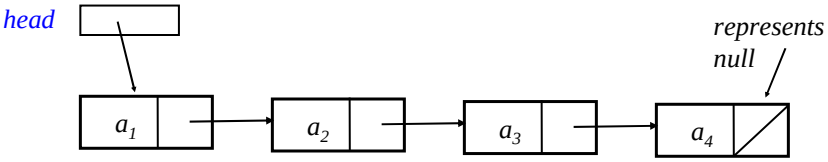
Linked List

A linked list is a sequence of items.

A node in linked list: *element* *next*

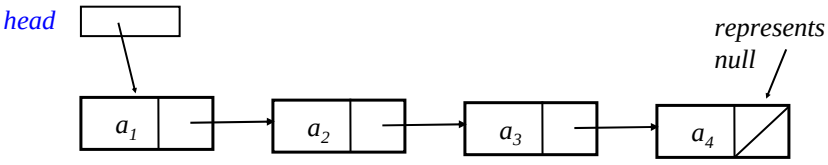


The linked list:

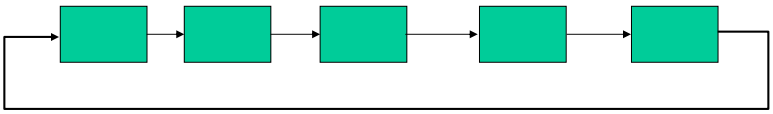


One-Way Linked List Representation

- O(n) as opposed to an array O(1) access time
- In a linked list we have to start at the first position.

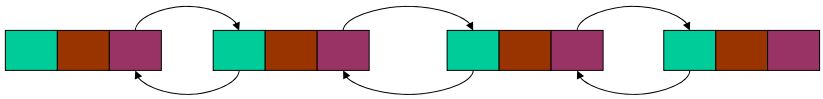


Circular Linked List



Given a pointer to an arbitrary node on a circular Linked List, we can follow links from a node to access any other node.

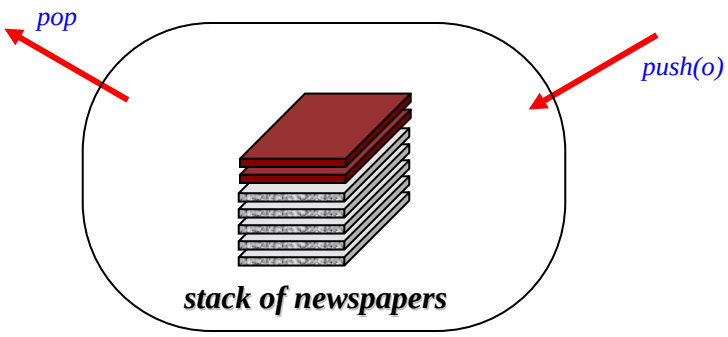
Two Way Linked List



Point to both their left and right neighbours, you can follow links in either direction to access other nodes.

What is a Stack?

- Stacks can be implemented efficiently and are very useful in computing.
- Stacks exhibit the LIFO behaviour.



Applications

Many application areas use stacks:

- *line editing*
- *bracket matching*
- *postfix calculation*
- *function call stack*

Line Editing

A line editor would place the characters read into a buffer but may use a backspace symbol (denoted by ←) to do error correction.

Refined Task
- read in a line
- correct the errors via backspace
- print the corrected line in reverse

Example:

Input : abc_defgh←←2klp←←←wxyz

Corrected Input : abc_defg2klpwxyz

Reversed Output : zyxwplk2gfed_cba

Informal Procedure

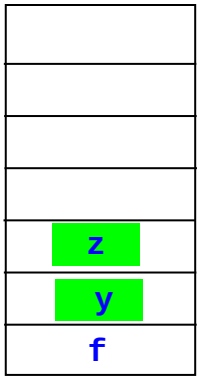
Line Editing

- Initialise a new stack.
- For each character read:
  - if it is a backspace, *pop out last char entered*
  - if not a backspace, *push the char into stack*
- To print in reverse, pop out each char for output.

Input : fgh←r←←yz

Corrected Input : fyz

Reversed Output : zyf

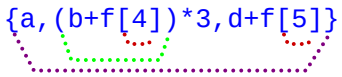


Stack

Bracket Matching Problem

Ensures that pairs of brackets are properly matched.

- An Example: `{a,(b+f[4])*3,d+f[5]}`



- Bad Examples:

```
(..)..)          // too many closing brackets

(..(..)          // too many open brackets

[..(..]..)       // mismatched brackets
```

Informal Procedure

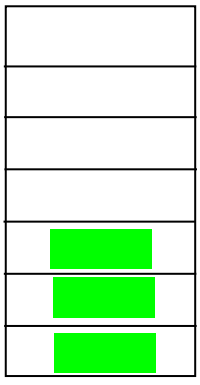
Initialise the stack to empty.

For every char read.
- if open bracket then *push onto stack*
- if close bracket, then
  - *topAndPop from the stack*
  - if doesn't match then *flag error*
- if non-bracket, *skip the char read*

Example

`{a,(b+f[4])*3,d+f[5]}`

Bracket Matching

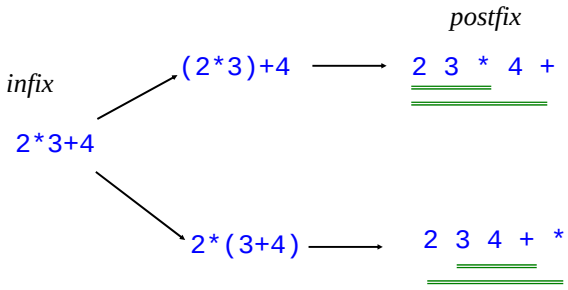


Stack

Postfix Calculator

Computation of arithmetic expressions can be efficiently carried out in Postfix notation with the help of a stack.

Infix - `arg1 op arg2`
Prefix - `op arg1 arg2`
Postfix - `arg1 arg2 op`



Informal Procedure Postfix Calculator

Initialise stack
For each item read.
    If it is an operand,
        push on the stack
    If it is an operator,
        pop arguments from stack;
        perform operation;
        push result onto the stack

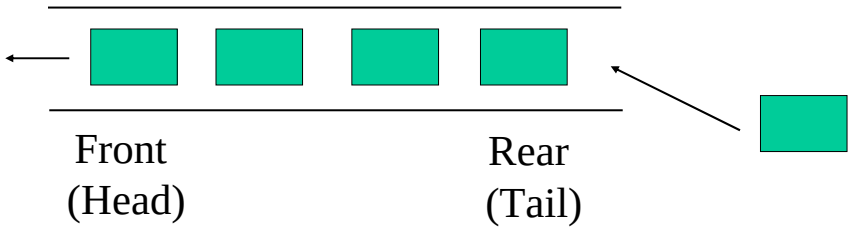
Expr	
2	`s.push(2)`
3	`s.push(3)`
4	`s.push(4)`

+	`arg2=s.topAndPop()`
	`arg1=s.topAndPop()`
	`s.push(arg1+arg2)`

*	`arg2=s.topAndPop()`
	`arg1=s.topAndPop()`
	`s.push(arg1*arg2)`

Stack

Queue



Queue has the disciple of First In First Out (FIFO).

15

Example: Compute the Average on List

```
double average (listptr head)
{
  double sum = 0;
  int n = 0;

  if (head == NULL)
  {
    printf ("\n empty list");
    exit (1);
  }

  do
  {
    n++;
    sum += head->value;
    head = head->next;
  }
  while (head != NULL);

  return sum/n;
}
```

Example: Concatenate 2 lists

```
listptr concatlists (listptr first, listptr second)
{
  listptr temp;

  if (first == NULL)
    return second;

  if (second != NULL)
  {
    temp = first;
    while (temp->next != NULL) temp = temp->next;

    temp->next = second;
  }

  return first;
}
```