CS 32 Week 4
Discussion 2C

UCLA CS

Yiyou Chen

I Was Wrong! :(

We'll be back in person starting from next week!

Discussion2C will be held at Rolfe 3126.

A building in the north campus.

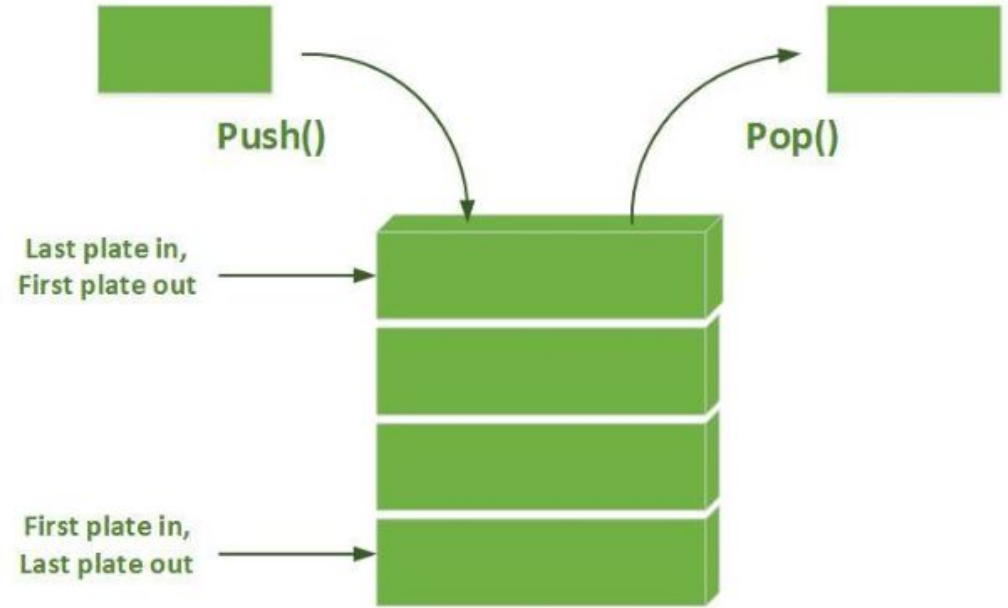
Topics

- Stacks and Queues:
  1. Definitions and Operations
  2. Infix->Postfix conversion and evaluation of postfix expressions using stacks
  3. Depth-first Search and Breadth-first Search

Stack

A sequential data structure.

First in last out (last in first out).



https://appdividend.com

C++ Stack

```
1 #include <stack>
2 using namespace std;
3
4 stack<TYPENAME> s;
5
6 s.push(item);  //item type: TYPENAME
7 s.pop(); //undefined behavior if empty
8 s.empty() //returns a boolean:
9           true if stack is empty, false otherwise.
0 s.top() //returns the top element of stack
1           undefined behavior if empty
2 s.size() //returns an integer: size of the stack
```

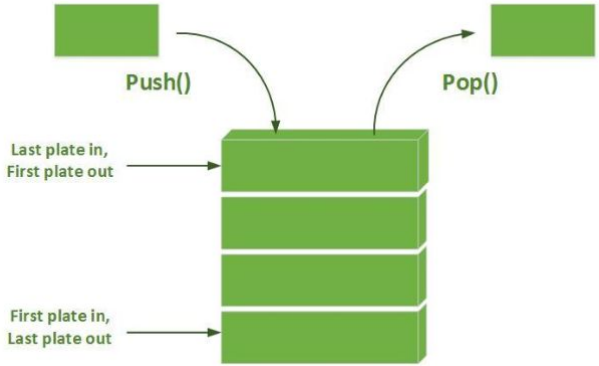


Image: https://appdividend.com

Queue

A sequential data structure.

First in First out (FIFO).

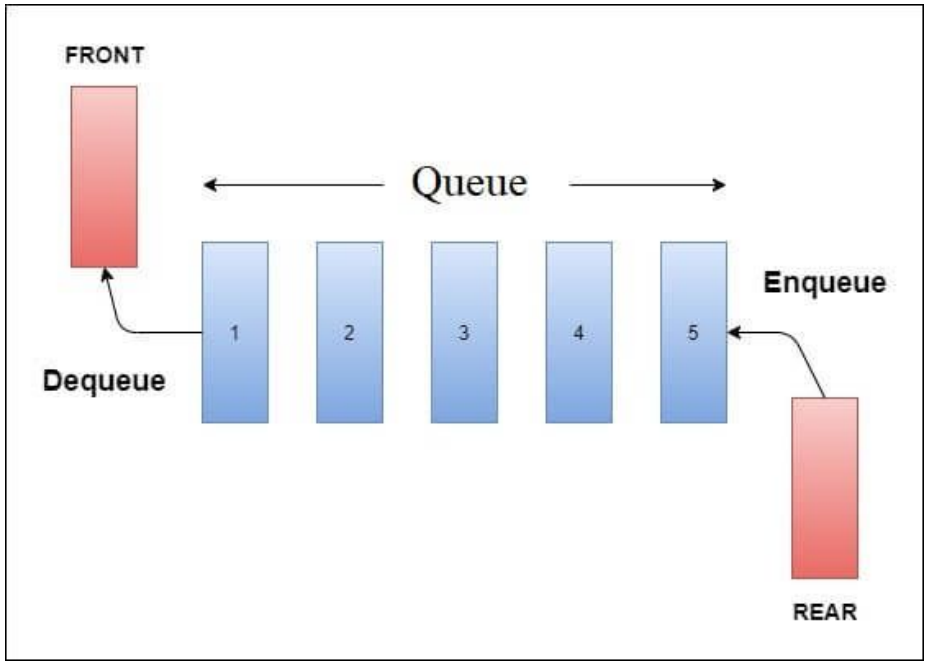


Image: https://www.tutorialandexample.com

C++ Queue

```
15 #include <queue>
16 using namespace std;
17
18 queue<TYPENAME> q;
19
20 q.push(item); //enqueue
21 q.pop(); //dequeue: undefined behavior if empty
22 q.empty() //returns a boolean:
23           true if queue is empty, false otherwise.
24 q.front() //returns the front element of queue
25           undefined behavior if empty
26 q.back() //returns the back element of queue
27           undefined behavior if empty
28 q.size() //returns an integer: size of the queue
```

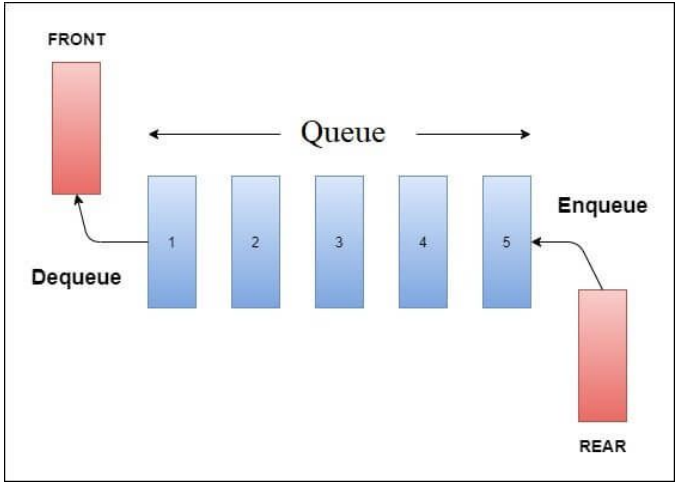


Image: https://www.tutorialandexample.com

Prefix, Infix, Postfix expressions

Pre-, in-, and post- tell us the location of **operators**.

E.g.

Infix: (3+2)*7-6/2

Postfix:

Prefix:

Prefix, Infix, Postfix expressions

Pre-, in-, and post- tell us the location of **operators**.

E.g.

Infix: (3+2)*7-6/2

Postfix: 3 2 + 7 * 6 2 / -

Prefix: - * + 3 2 7 / 6 2

Infix -> postfix using stack

Infix: 3+(2*7-6)/2

Postfix: 3 2 7 * 6 - 2 / +

Observations: Let O1, O2, O3, O4, … be the operators. Then we add On if O(n-1) < On >= O(n+1).

1. The order of operands doesn't change, so we just need to add the operators in correct order.
2. Use a stack to keep track of the operators of the highest precedence so far. No operators with equal or higher precedence should occur in the stack before the current (non '(' operator. If so, pop them out.
3. '(' has lowest precedence, but once paired with ')' they make the operators in between them highest precedence.

3 2	+ (
3 2 7	+ (* -
3 2 7 * 6	+ (-)
3 2 7 * 6 -	+ /
3 2 7 * 6 - 2 / +	

Evaluate postfix using stack

Postfix: 3 2 7 * 6 - 2 / +

Each operator always acts on the last two operands before it.

Stack:

3 2 7 *

3 14 6 -

3 8 2 /

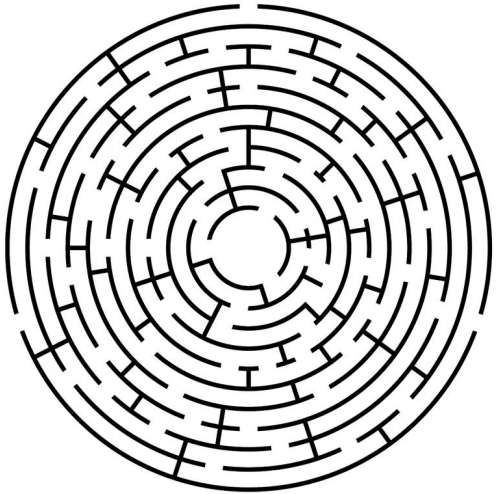
3 4 +

7

Searching Algorithms

Depth-first search (DFS): I'll stick to this path till I reach a dead end!

Breadth-first search (BFS): I'll explore all paths at the same time!

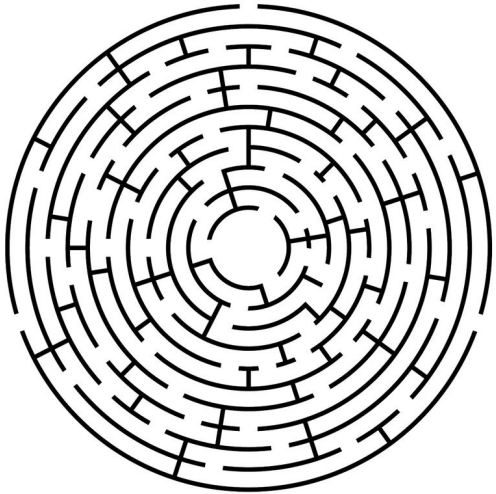


https://www.popsci.com

Searching Algorithms

Depth-first search (DFS): I'll stick to this path till I reach a dead end!

Breadth-first search (BFS): I'll explore all paths at the same time!



https://www.popsci.com

Q: Is there any case when DFS will be inefficient?

Depth-First Search (DFS) using a stack

```
30 bool DFS(TYPE start, TYPE target) {
31   stack<TYPE> s;
32   s.push(start);
33   visisted[all nodes] = false;
34   while(!s.empty()) {
35     TYPE u = s.top();
36     if (ending_condition(start, target))
37        return true;
38     s.pop();
39     visited[u] = true;
40     for (t unvisited neighbor of u) {
41       s.push(t);
42     }
43   }
44   return false;
45 }
```

Breadth-First Search (BFS) using a queue

```
47 bool BFS(TYPE start, TYPE target) {
48   queue<TYPE> q;
49   q.push(start);
50   visisted[all nodes] = false;
51   while(!q.empty()) {
52     TYPE u = q.front();
53     if (ending_condition(start, target))
54        return true;
55     q.pop();
56     visited[u] = true;
57     for (t unvisited neighbor of u) {
58       q.push(t);
59     }
60   }
61   return false;
62 }
```