GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

3. Stack & Queue

大纲

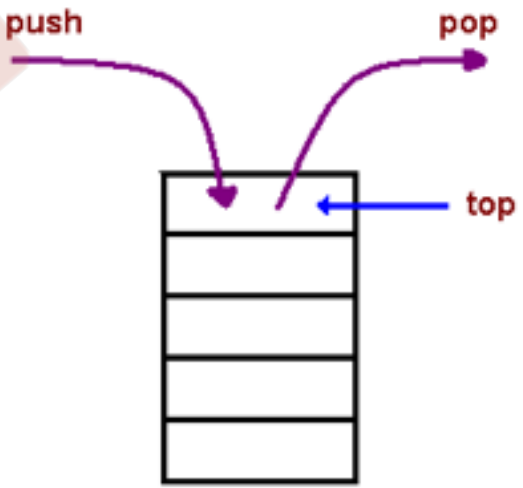
1. Stack介绍
2. Queue介绍
3. 例题分析

Stack

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle

Operations:

- Push O(1)
- Pop O(1)
- Top O(1)



栈的用途

可以用Stack作为辅助，实现深度优先算法（Depth first search, DFS），或者将递归转为while循环

递归本身就是相当于把函数本身一层一层加到操作系统的内存栈上

入栈操作相当于递归调用自身，出栈操作相当于递归返回。

工具箱：C++

stack and queue:
bool empty() const; // Returns whether the stack is empty: i.e. whether its size is *zero*.
void push (const value_type& val); // Inserts a new element at the top of the stack. The content of this new element is initialized to a copy of *val*.
void pop(); // Removes the element on top of the stack, effectively reducing its size by one.
value_type& top(); // Returns a reference to the *top element* in the stack

Example:

```
stack<int> myStack;
myStack.push(10);
myStack.push(20);
int value = myStack.top();    // value equals to 20

queue<int> myQueue;
myQueue.push(10);
myQueue.push(20);     // queue now has two elements, the value of which is 10, 20
int value = myQueue.front(); // value equals to 10
myQueue.pop();  // queue now has one element, the value of which is 20
```

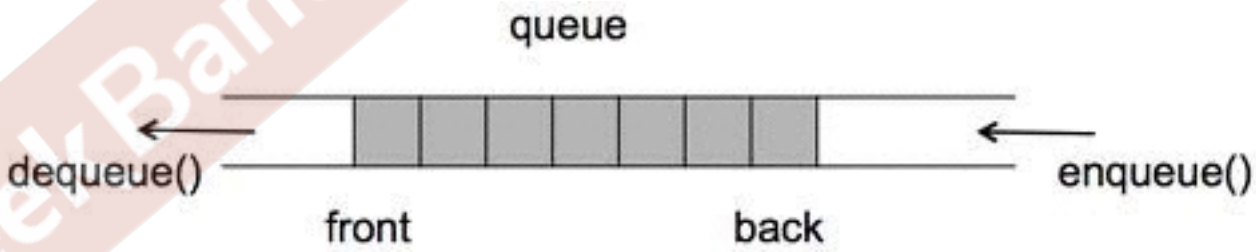
Queue

A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle

Operations:

- O(1) Push
- O(1) Pop
- O(1) Top

Always used for BFS



用途

我们可以用Queue作为辅助，实现广度优先算法（Breadth first search, BFS）

Queue还可以作为buffer，构建一个生产者- 消费者模型：生产者把新的元素加到队尾，消费者从队头读取元素。在有两个线程同时读取同一个queue时，需要考虑同步（synchronization）

stack 与 queue 可以视作封装好的Linked list，只是限制了访问和插入的自由。适用stack或queue的情境，也可以考虑使用更为强大的list。

模式识别

1. 通过stack实现特殊顺序的读取
由于stack具有LIFO的特性，如需实现任何特定顺序的读取操作，往往可以借助两个stack互相”倾倒"来实现特定顺序。

另一个stack作为辅助。

Get Max Stack

Implement a stack, enable O(1) Push, Pop Top, Max. Where Max() will return the value of maximum number in the stack.

Get Max Stack 解答

Using two stacks.
The first one is the regular stack.
The second one only store maximum numbers if a larger number comes.

复杂度分析：时间复杂度符合题目要求为O(1)。空间复杂度最坏情况附加的stack中需要储存每个元素，故额外使用O(n)空间。

Queue using Stack

Implement a queue with stack structure.

Queue using Stack 伪代码

```
Q.Push(x):
   S1-Push(x)
Q.Pop():
   if S2.empty -> S1->S2
   S2.pop()
Q.Top():
    Similar with Q.Pop()
```

Sort Stack

How to sort a stack in ascending order (i.e. pop in ascending order) with another stack?

复杂度分析：由于调整一个元素的顺序可能要求将之前的*n*个元素来回倾倒，故时间复杂度*O(n^2)*。

“save or later”问题

有一类问题有这样的特性：当前节点的解依赖后驱节点。

对于某个当前节点，如果不能获知后驱节点，就无法得到有意义的解。这类问题可以通过stack（或等同于stack的若干个临时变量）

解决：先将当前节点入栈，然后看其后继节点的值，直到其依赖的所有节点都完备时，再从栈中弹出该节点求解。某些时候，甚至需要反复这个过程：将当前节点的计算结果再次入栈，直到其依赖的后继节点完备。

Validate Parenthesis

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is a valid parentheses string. For example, “(([]))” is valid, but “[]” or “((” is not.

模式匹配

3. 用stack解决Top-Down结构的问题
所谓的Top-Down结构，从逻辑理解的角度来看，实际上就是一种树形结构，从顶层出发，逐渐向下扩散，例如二叉树的周游问题。 在实际运算的时候，我们先解决子问题，再利用子问题的结果解决当前问题。

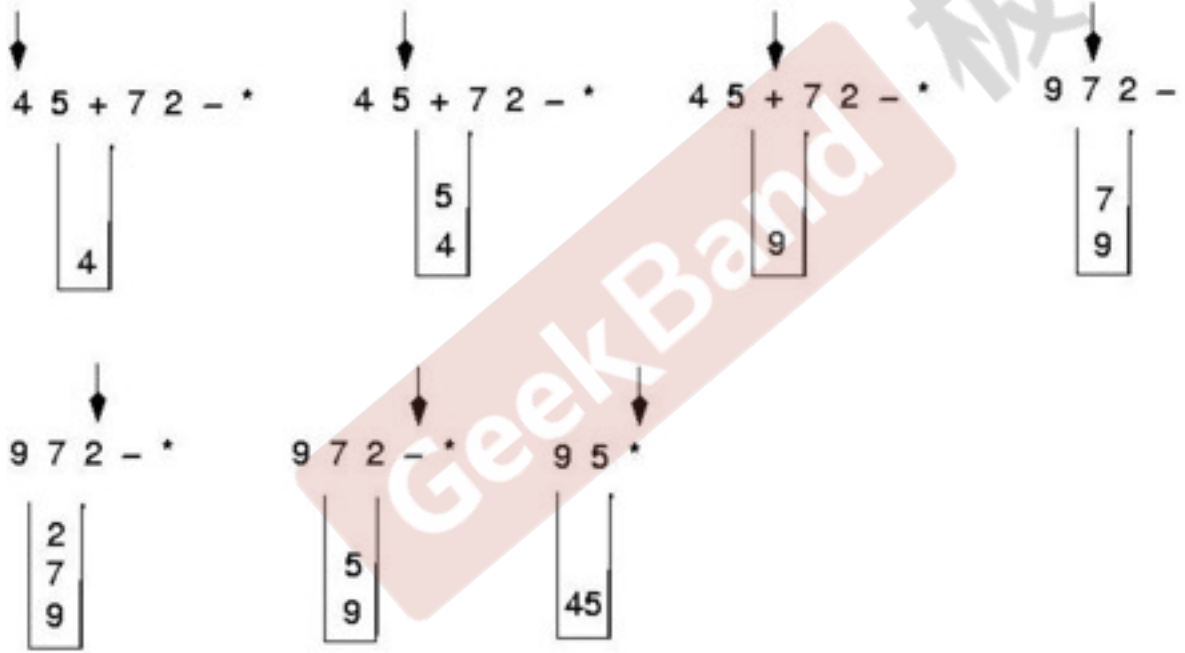
由于Stack的LIFO特性，可以利用Stack数据结构消除递归。Recursion通常用函数调用自身实现，在调用的时候系统会分配额外的空间，并且需要用指针记录返回位置，故overhead比较大。

In-order Traversal

Given a binary tree, implement the In-Order Traversal using a stack

Evaluate Expression

Inorder fix (4 + 5) * (7 -2)
Postfix Expression: 4 5 + 7 2 - *



Extension

How to calculate

3 + 14*5

(3+4) * 5

(12+3) * (3+2)^2

Queue扩展

Circles Queue
Queue with Max
PriorityQueue
Blocking Queue (Multi-thread)

Homework

1. Queue with Max
2. Evaluate (3+4) * 14