算法分析与设计 II

2022-2023-2

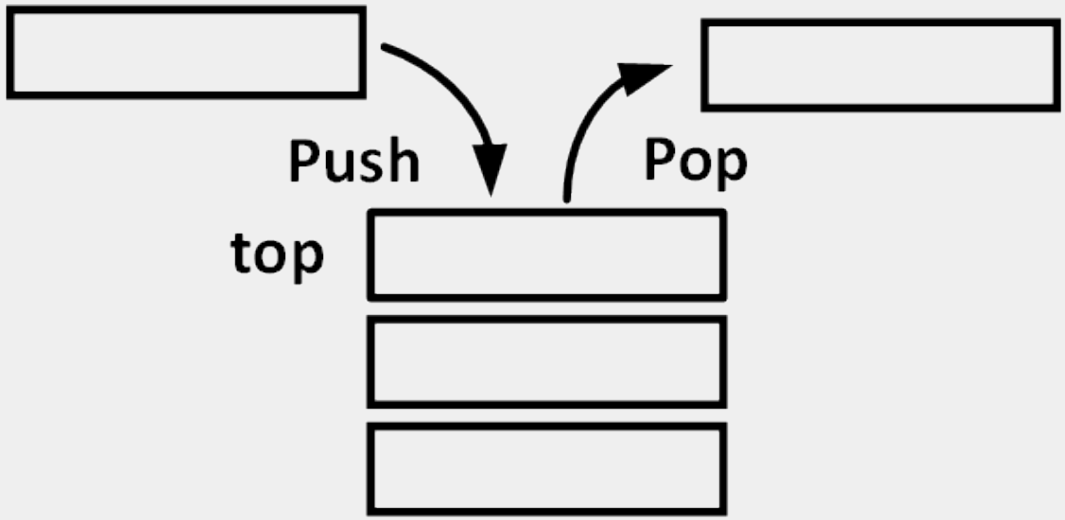
数学与计算机学院
数据科学与大数据技术

Last Modified: 2023.1.16

3. 基础数据结构

堆栈

堆栈(stack) 是基础的数据结构，是一种抽象数据类型(ADT, Abstract Data Type)，其特点是后进先出 (LIFO, Last In First Out)



堆栈

- 在 C++ STL 中提供了专门的 stack[1]实现堆栈的基本功能
- 手动实现栈的操作也很容易，如果空间允许，可以使用数组，预先为堆栈分配足够的空间；如果空间紧张，也可以使用链表，动态的改变堆栈的空间
- 函数的调用和返回，在计算机系统中就是通过堆栈来实现的，系统堆栈空间并不是无限的，所以限制了函数调用的层次和规模，递归算法是函数的反复调用，所以更需要注意

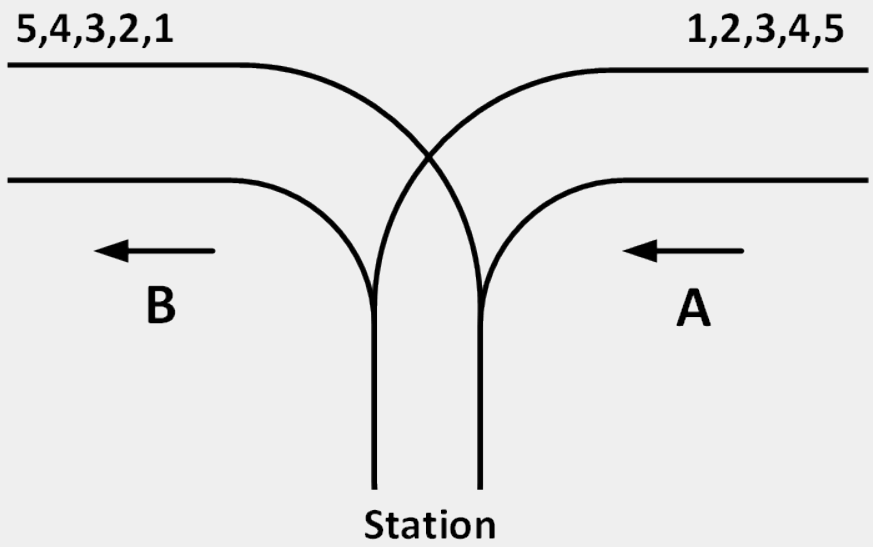
[1] https://cplusplus.com/reference/stack/stack/

基本操作

```
1   // 将元素x加入堆栈S，S.top为栈顶位置
2   // 元素入栈，栈顶位置加1
3   PUSH(S,x)
4       S.top = S.top + 1
5       S[S.top] = x
6
7   // 栈顶元素出栈，栈顶位置减1，返回栈顶元素的值
8   POP(S)
9       if  S.top>0
10          S.top = S.top - 1
11      return S[S.top + 1]
```

1363 – Rails (poj.org)

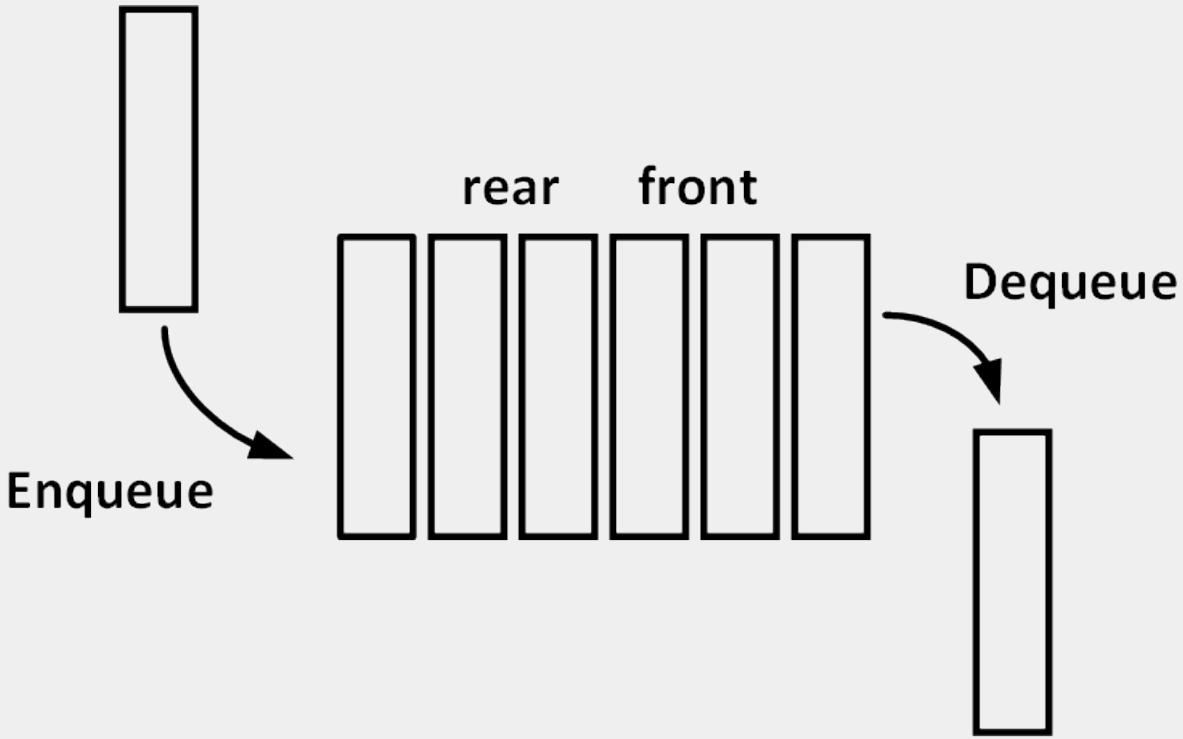
- 如图的车站，可以通过操作将 A 序列转变成 B 序列。现在 A 为 1 到 n 的顺序序列，问给出一个序列 B，能否通过 A 生成



3.2 队列

队列

队列(queue) 也是一种抽象数据类型，其特点是先进先出 (FIFO, First-In-First-Out)
通常情况下，只允许在队列后端 (rear) 入队列 (Enqueue)，在队列前端 (front) 出队列 (Dequeue)



队列

- 在 C++ STL 中提供了专门的 queue[2]实现队列的基本功能
- 在队列的操作的过程中，每次进行 1 个元素的入队或者出队，不适合元素数量过多，反复进行操作
- 在计算机系统中，缓冲区管理就是采用队列实现，当队列空间满而没有有效检查时，会出现缓冲区溢出(Buffer overflow)，利用系统中的缓冲区溢出是黑客攻击系统的常用手段
- 队列通常采用链表或者数组来实现，使用数组的时候会出现“伪溢出”的现象，就是数组进行一定量的入队出队操作之后，队列头还有空间，而队列尾已满而无法入队，为了解决这个问题，常常采用循环队列的方式
- 数组实现队列时，队列空间不方便扩展，可以采用链表

[2]https://cplusplus.com/reference/queue/

基本操作

```
1   // 将元素x加入队列Q，队列尾位置加1
2   Enqueue(Q,x)
3       Q[Q.rear] = x
4       Q.rear = Q.rear + 1
5
6   // 出队列，返回队列头元素，队列头位置加1
7   Dequeue(Q)
8       x = Q[Q.front]
9       Q.front = Q.front + 1
10      return x
```

2823 – Sliding Window (poj.org)

- An array of size $n \leqslant 10^6$ is given to you. There is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves rightwards by one position.

- Following is an example:
  - The array is [1 3 -1 -3 5 3 6 7], and k is 3

- Your task is to determine the maximum and minimum values in the sliding window at each position.

Example

Window position	Minimum value	Maximum value
[1 3 -1] -3 5 3 6 7	-1	3
1 [3 -1 -3] 5 3 6 7	-3	3
1 3 [-1 -3 5] 3 6 7	-3	5
1 3 -1 [-3 5 3] 6 7	-3	5
1 3 -1 -3 [5 3 6] 7	3	6
1 3 -1 -3 5 [3 6 7]	3	7

堆

堆(heap) 是一种特别的完全二叉树，如果完全二叉树中每个节点的值都小于等于其子节点的值，称为最小堆(min heap)；反之，如果每个节点的值都大于等于其子节点的值，称为最大堆(max heap)

- 在 C++ STL 中的优先队列 priority_queue[3]就是堆的实现，对于堆的应用可以直接使用优先队列
- 使用堆实现的堆排序(Heapsort)，平均时间复杂度为 $O(nlogn)$
- 优先队列常用于计算机操作系统中的任务调度，例如基于优先级的进程调度
- 字符串中求最优无前缀码即哈夫曼编码，使用最小堆来实现
- 图论中，使用优先队列解决基于贪心思想的最小生成树等问题

[3]https://cplusplus.com/reference/queue/priority_queue/

3253 – Fence Repair (poj.org)

- 将一个长木板进行 n-1 次切割，切成 n 个小木板，切木板的花费等于木板的长度。给出最后切割后的长度，求费用最小的切割方案

例

- 将 21 切割成 8,5,8，可以先切成 13 和 8，总费用 21+13=34
- 如果先切成 16 和 5，则总费用 21+16=37，超过 34

- 分析
  - 假定最后切割完的所有木板在一个集合里，每次从集合中选出两个最短的，它们的和为一次切割费用，将它们合并后放回集合，重复以上操作，最后累计切割费用即可
  - 题目的主要操作是从集合中每次取出最小的两个值，每次都重新排序的话算法复杂度会提高，而采用堆能很好的解决这个问题