一、堆

1、堆的定义

【例题】序列16 14 10 8 7 9 3 2 4 1的说法下面哪一个正确（）

大顶堆

小顶堆

不是堆

二叉排序树

把序列画成完全二叉树形式后可见是大顶堆。

堆实际上是一棵完全二叉树，分为大顶堆和小顶堆，满足Key[i]>=Key[2i+1] && key[i]>=key[2i+2]称为大顶堆，满足 Key[i]<=key[2i+1] && Key[i]<=key[2i+2]称为小顶堆，二叉排序树需要满足左孩子<根<右孩子，本题中，首位16为根，满足大顶堆的概念。

二叉排序树需要满足，左孩子<根<右孩子，而大顶堆需要满足根>左孩子且根>右孩子

【例题】下列关键字序列为堆的是：（）

100,60,70,50,32,65

60,70,65,50,32,100

65,100,70,32,50,60

70,65,100,32,50,60

32,50,100,70,65,60

50,100,70,65,60,32

2、堆的调整

【例题】已知序列25， 13， 10， 12， 9 是大根堆，在序列尾部插入新元素 18，将其再调整为大根堆，调整过程中元素之间进行的比较次数是（ ）。

1

2

4

5

把大根堆画成完全二叉树后，堆插入元素相当于是从最后面添加叶子节点18，先是18和10比较，18>10，将18与10对调，继续判断18与25的大小。所以总共比较2次。

【例题】最小堆[0,3,2,5,7,4,6,8],在删除堆顶元素0之后，其结果是（）

[3，2，5，7，4，6，8]

[2，3，5，7，4，6，8]

[2，3，4，5，7，8，6]应该是

[2，3，4，5，6，7，8]

插入元素是在最后插入，然后进行调整堆；删除元素是把最后的元素放到删除元素的地方，然后进行调整堆。 删除0后用8填充0的位置，为[8,3,2,5,7,4,6]，然后再继续调整堆。直到满足最小堆为止。

【例题】已知关键字序列5,8,12,19,28,20,15,22是最小堆，插入关键字3，调整后得到的最小堆是()

3,8,12,5,20,15,22,28,19

3,5,12,19,20,15,22,8,28

3,12,5,8,28,20,15,22,19

3,5,12,8,28,20,15,22,19

插入元素是在最后插入，然后进行调整堆；删除元素是把最后的元素放到删除元素的地方，然后进行调整堆。

加入3 之后的置换顺序 3--19 3--8 3--5

3、堆的建立

【例题】一组记录排序码为(5 11 7 2 3 17),则利用堆排序方法建立的初始堆为

(11 5 7 2 3 17)

(11 5 7 2 13 3)

(17 11 7 2 3 5)

(17 11 7 5 3 2)

(17 7 11 3 5 2)

(17 7 11 3 2 5)

如果堆的有序状态因为某个节点变得比它的父节点更大而打破，那么就需要通过交换它和它的父节点来修复堆。从最后一个非叶结点逐渐往上浮，直到有序。

4、堆的复杂度

【例题】对于根元素为最小值的二叉堆，下面说法正确的是

删除最小元素的复杂度是o(logn)

插入新元素的复杂度是o(1)

合并两个堆的复杂度是o(logn)

查询最小元素的复杂度是o(1)

根元素为最小值的二叉堆：

插入节点时间复杂度为O(log n)

删除节点时间复杂度为O(log n)

查询最小元素的复杂度是o(1)

合并两个堆的复杂度是o(nlogn)

【例题】对n个记录的文件进行堆排序,最坏情况下的执行时间是多少?()

O(logn)

O(n)

O(nlogn)

O(n\*n)

时间复杂度分析，第一步首先建堆需要用时o(n)，第二步对大小为n的堆，取出元素放入数组尾部用时o(1)，重新进行保持堆特性为o(lgn)，因此o(n)+o(nlgn)，总体时间时间复杂度为o(nlgn)

5、堆和栈的区别

【例题】下列关于堆和栈的区别描述错误的有？

申请方式的不同，堆是系统自动分配，栈是自己申请

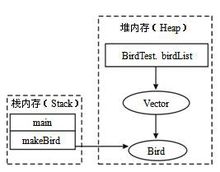
栈的大小是固定的，堆的大小受限于系统中有效的虚拟内存

栈的空间由系统决定何时释放，堆需要自己决定何时去释放

堆的使用容易产生碎片，但是用起来最方便

1.栈内存操作系统来分配，堆内存由程序员自己来分配。

2.栈有系统自动分配，只要栈 剩余空间大于所申请空间，系统将为程序提供内存，否则将报异常提示栈溢出。



两个术语虽然只有一字之差，但是所表达的意义还是有差别的，堆内存和栈内存的区别可以用如下的比喻来看出：使用堆内存就象是自己动手做喜欢吃的菜肴，比较麻烦，但是比较符合自己的口味，而且 自由度 大。使用栈内存就象我们去饭馆里吃饭，只管点菜（发出申请）、付钱和吃（使用），吃饱了就走，不必理会切菜、洗菜等准备工作和洗碗、刷锅等扫尾工作，他的好处是快捷，但是自由度小。操作系统中所说的堆内存和 栈内存 ，在操作上有上述的特点，这里的堆内存实际上指的就是（满足堆内存性质的） 优先队列 的一种数据结构，第1个元素有最高的 优先权 ；栈内存实际上就是满足先进后出的性质的数学或数据结构。

二、栈

1、数据结构的比较

【例题】当很频繁地对序列中部进行插入和删除操作时，应该选择使用的容器是（）B

vector

list

deque

stack

补充：

C++ STL 的实现：

1.vector 底层数据结构为数组 ，支持快速随机访问

2.list 底层数据结构为双向链表，支持快速增删

3.deque 底层数据结构为一个中央控制器和多个缓冲区，详细见STL源码剖析P146，支持首尾（中间不能）快速增删，也支持随机访问

4.stack 底层一般用23实现，封闭头部即可，不用vector的原因应该是容量大小有限制，扩容耗时

5.queue 底层一般用23实现，封闭头部即可，不用vector的原因应该是容量大小有限制，扩容耗时

6.45是适配器,而不叫容器，因为是对容器的再封装

7.priority\_queue 的底层数据结构一般为vector为底层容器，堆heap为处理规则来管理底层容器实现

8.set 底层数据结构为红黑树，有序，不重复

9.multiset 底层数据结构为红黑树，有序，可重复

10.map ﻿﻿﻿﻿底层数据结构为红黑树，有序，不重复

11.multimap 底层数据结构为红黑树，有序，可重复

12.hash\_set ﻿﻿﻿﻿底层数据结构为hash表，无序，不重复

13.hash\_multiset 底层数据结构为hash表，无序，可重复

14.hash\_map ﻿﻿﻿﻿底层数据结构为hash表，无序，不重复

15.hash\_multimap 底层数据结构为hash表，无序，可重复

2、顺序栈和链式栈的区别

【例题】和顺序栈相比,链栈有一个比较明显的优势是()

通常不会出现栈满的情况

通常不会出现栈空的情况

插入操作更容易实现

删除操作更容易实现

这题其实考的是链表和顺序表的优缺点。

对于链表，插入/删除结点是比较困难的，因为每个结点只保存与相邻结点间的关系，如果要插入/删除中间某结点，必须从头遍历到该节点

但链表采用的是非连续内存存储的模式，意味着可以从未使用的内存中开辟一个空间，很容易的添加一个结点，这样就不会出现栈满的情况，而对于连续存储的顺序栈而言，是很难增加大小的

【例题】下列说法错误的是 ()

利用一组地址连续的存储单元依次存放自栈底到栈顶的数据元素，这种形式的栈也称为顺序栈

top=0 时为空栈,元素进栈时指针 top 不断地减 1

当top等于数组的最大下标值时则栈满

栈不能对输入序列部分或全局起求逆作用

A正确，虽然链栈也可以实现地址连续，但无法做到依次存放数据元素，因为除了元素域还需要存放指针域。因此满足条件的只有顺序栈。

B错误，top= -1时为空栈，top=0只能说明栈中只有一个元素，并且元素进栈时top应该自增

C正确，top所指向的对象永远是栈顶元素

D错误，栈的特性（先进后出）就决定了可以作为求逆元素的辅助空间。所谓求逆运算就是ABC依次入栈，输入CBA。

3、出入栈顺序

【例题】一个栈的入栈序列是a,b,c,d,e,f,则栈的不可能的输出序列是（）

fedcba

defbca

defcba

abcdef

x最先出栈，说明下面压着比它小的字母序列，这个序列必须以倒序出栈，但不要求该序列要连续无间隔。

如：f最先出栈，则abcde必然以倒序出栈

d最先出栈，则adc必然以倒序出栈

【例题】元素a，b，c，d，e 依次进入初始为空的栈中，若元素进栈后可停留、可出栈，直到所有的元素都出栈，则在所有可能的出栈序列中，以元素 d 开头的序列个数是（ ）。

3

4

5

6

入栈的顺序已经确定，即a、b、c、d、e，如果d在出栈的第一位，则说明abc在栈中，出栈的顺序必须是cba（顺序确定），在cba序列前中后共有4个可插入的位置即c前，c和b中间，b和a中间，a后。所以，以b开头的出栈序列共有4种。

【例题】一个栈的入栈序列为1,2,3,...,n ，其出栈序列是 p 1 ,p 2 ,p 3 ,...p n 。若p 2 = 3，则 p 3 可能取值的个数是（）

n-3

n-2

n-1

无法确定

看了别人的一些解析，觉得不够完善，下面给出自己的见解。

首先，栈的先进后出原则大家应该是知道的。

根据题意 p 2 = 3，可以知道 p 1 的可能情况有三种：1，2 或 4 。（看到有些人只想到了 1，2）

为啥这样想呢？这里估计还有一个关键是要考虑到 n 的大小。

当 n = 3 时， p 2 = 3 的话，那么 p 1 有两种情况 1 和 2 。

- 如果 p 1 = 1 ， 那么 p 3 = 2 ；

- 如果 p 1 = 2 ，那么 p 3 = 1 ；

此时的话我们就可以看到 p 3 只有两种可能 1 或者 2 （n - 1）个。

当 n > 3 时： p 2 = 3 的话，那么 p 1 有三种情况 1 ， 2 和 4 。

- 如果 p 1 = 1 ， 那么 p 3 = 2，4，5，... n （n - 2）个

- 如果 p 1 = 2 ，那么 p 3 = 1，4，5，... n （n - 2）个

- 如果 p 1 = 4 ，那么 p 3 = 2，5，6，... n （n - 3）个

此时的话我们就可以看到 p 3 的情况有 1，2，4，5，... n （n - 1）个。

综上所述就是 p 3 可能取值的个数是 （n - 1）个。

4、栈的应用

【例题】解析XML时，需要校验节点是否闭合，如必须有与之对应，用（）数据结构实现比较好

链表

树

队列

栈

栈的常见应用：浏览器历史纪录，Android中的最近任务，Activity的启动模式，CPU中栈的实现，Word自动保存，解析计算式，解析xml/json

节点闭合的话，有头尾符号相对应，遇到头符号将其放入栈中，遇到尾符号时，弹出栈的内容，看是否有与之对应的头符号，栈的特性刚好符合符号匹配的就近原则。